Air cargo load and route planning in pickup and delivery operations

A.C.P. Mesquita, C.A.A. Sanches*

Instituto Tecnológico de Aeronáutica - DCTA/ITA/IEC
Praça Mal. Eduardo Gomes, 50
São José dos Campos - SP - 12.228-900 - Brazil

Abstract

In the scenario of aerial pickup and delivery of goods in a distribution network, transport aviation faces risks of cargo unbalancing due to the urgency required for loading for rapid take-off and mission accomplishment, especially in times of crisis, public calamity, client contract short lead times, or any external pressure for immediate take-off. Also, there is no technological assistance to help the load and trip planners with the huge number of demands for transport in each hub. This enables other risks such as improper delivery, excessive fuel burn, and longer than necessary turn-around time. We contributed by modelling and solving a new problem of planning the loading and routing of an aircraft according to a utility score, weight and balance principles, and fuel consumption in a tour of simultaneous pickup and delivery at intermediate hubs. This new hard problem, named *Air Cargo Load Planning with Routing, Pickup, and Delivery Problem (ACLP+RPDP)*, is mathematically modelled using standardised pallets in fixed positions, obeying the centre of gravity constraints, delivering each item to its destination, and minimising fuel consumption costs. We also contributed by carrying out multiple experiments with a commercial solver and four well-known meta-heuristics on synthetic data based on real data from the *Brazilian Air Force* transportation history. These challenging benchmark instances are made publicly available. We also created a heuristic that quickly finds good solutions for a wide range of problem sizes, an essential contribution as it was the unique method that managed to solve all testing scenarios.

Keywords: Air Cargo, Load Planning, Air Palletization, Weight and Balance, Pickup and Delivery, Vehicle Routing

1. Introduction

Air cargo transport involves several sub-problems that are difficult to solve. Recently, (**?**, p. 401) defined the *Air Cargo Load Planning Problem* (ACLPP) as four sub-problems: *Aircraft Configuration Problem* (ACP), *Build-up Scheduling Problem* (BSP), *Air Cargo Palletization Problem* (APP) and *Weight and Balance Problem* (WBP). Several aspects were considered in this modelling: characteristics of the items to be transported (dimensions, scores, dangerousness, etc.); types and quantities of *unit load devices* (ULDs), commonly called pallets; when these pallets are assembled; how items are allocated to pallets; in which positions these pallets are to be placed; how the total cargo weight is balanced; etc. They also presented a comprehensive bibliographic survey of solving methods that had been developed in different situations.

However, there are still other important challenges in air cargo transport that go beyond the definition of the ACLPP, especially with regard to the flight itinerary and the loading and unloading at each destination (or node) of this travel plan. In this context, at least two more important sub-problems can be considered: pickup and delivery operations at each node, called *Pickup and Delivery Problem* (PDP), and the search for the lowest cost route, which is the well-known *Travelling Salesman Problem* (TSP).

Considering air cargo transport, Table 1 lists the main works in the literature and the corresponding sub-problems addressed. We also indicate whether the dimensions of the items were taken into account (**3D** or **2D**) and which solution method was used: heuristics (**Heu**), integers (**Int**), or linear programming (**Lin**).

*Corresponding author.
Email addresses: celio@ita.br (A.C.P. Mesquita), alonso@ita.br (C.A.A. Sanches)

Table 1: Air cargo transport: literature, problems and features

	APP	WBP	PDP	TSP	2D	3D	Heu	Int	Lin
?	.	★	.	.	.	.	★	.	.
?	.	★	.	.	.	.	.	.	★
?	.	★	.	.	.	.	.	★	.
?	.	★	.	.	★	.	★	.	.
?	★	★	.	.	.	.	.	★	.
?	.	★	.	.	.	.	.	★	.
?	★	.	.	.	.	★	★	.	.
?	.	★	.	.	★	.	.	★	.
?	.	★	.	.	.	.	.	★	.
?	.	.	★	.	.	.	★	.	.
?	.	★	.	.	.	.	.	★	.
?	★	★	.	.	.	★	.	★	.
?	★	★	.	.	.	.	.	★	.
?	.	★	★	.	.	.	.	★	.
?	.	★	.	.	.	.	★	.	.
??	★	★	.	.	.	★	★	★	.
?	.	★	.	.	★	.	★	.	.
?	★	★	.	.	.	.	.	★	.
?	★	★	.	.	.	.	.	★	.
?	.	★	.	.	.	.	.	★	.
This work	★	★	★	★	.	.	★	★	.

As can be seen, so far **?** is the only work that simultaneously addresses an air cargo (WBP) and a flight itinerary (PDP) sub-problem. 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 restricted to two legs. It is important to note that these authors consider an aircraft with two doors, and the minimization of loading and unloading costs at the intermediate node was modelled through a container sequencing problem. Referring directly to this work, (**?**, p. 409) comment: *However, not even these sub-problems are acceptably solved for real-world problem sizes or the models omit some practically relevant constraints.*

There are real situations that are much more complex. In this work, we consider a practical case in Brazil, which is the largest economy in Latin America. Due to its dimensions, this country has the largest air market on the continent with $2,499$ registered airports, of which $1,911$ are private and 588 are public. Although it is an immense distribution network, airlift missions consider 3 to 5 nodes per flight plan. Throughout this work, we address routes with up to 7 nodes, as can be seen in Table 2 and Figure 1.

Table 2: Brazilian airports distances (km)

Node IATA*	l_0 GRU	l_1 GIG	l_2 SSA	l_3 CNF	l_4 CWB	l_5 BSB	l_6 REC
GRU	0	343	1,439	504	358	866	2,114
GIG	343	0	1,218	371	677	935	1,876
SSA	1,439	1,218	0	938	1,788	1,062	676
CNF	504	371	938	0	851	606	1,613
CWB	358	677	1,788	851	0	1,084	2,462
BSB	866	935	1,062	606	1,084	0	1,658
REC	2,114	1,876	676	1,613	2,462	1,658	0

*International Air Transport Association
Source: www.airportdistancecalculator.com

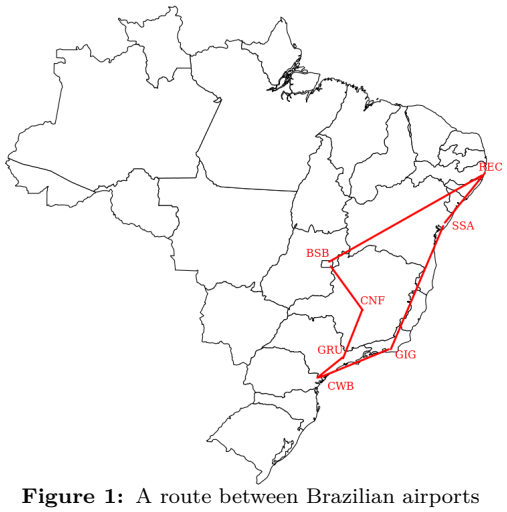

Figure 1: A route between Brazilian airports

In the *Brazilian Air Force* missions, hundreds of items can be carried at each node, where the objectives are to prioritise the transport of the most important items and minimise the cost of fuel along the route. As standardised pallets are used with predefined positions on the aircraft, it is possible to carry out loading and unloading at each node in around two hours. However, there is no technological assistance that guarantees the achievement of these objectives.

This work proposes a method that attends these objectives: a pallet building and arrangement plan with a

routing option that maximises the benefit-cost ratio for the smooth execution of pickup and delivery transport missions. We develop a heuristic that can be executed on a simple handheld computer (like a laptop or a tablet) and that provides a solution quickly enough to keep this cargo handling time under an hour. This heuristic will 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 and TSP has been addressed. This new problem is named *Air Cargo Load Planning with Routing, Pickup and Delivery Problem* (ACLP+RPDP).

This article is organised into six more sections. In Section 2, we give a brief review of the literature. In Section 3, we present the problem context and assumptions and, in Section 4, the mathematical model and how we dealt with its issues. In Section 5, we describe the elaborate algorithms, whose results are presented in Section 6. Finally, our conclusions are in Section 7.

2. Related literature

In this section, we briefly describe the characteristics of the main works related to air cargo transport, following the chronological order of Table 1.

? developed an interactive procedure for loading 14 types of Boeing 747 into a two-leg flight plan. Seven types of items were considered to be allocated in 17 to 42 positions. With non-linear programming and heuristics, they present a solution that minimises positioning changes in the intermediate node, optimising the load balancing in the aircraft.

? addressed the problem of planning the allocation of cargo on an aircraft. Considering volume, weight and structural constraints, the author finds the optimal load layout through a fractional programming problem.

? developed a multi-criteria optimization approach to load the *C-130* aircraft of the *Canadian Air Force*. Based on integer programming, this model provides timely planning and improves airlift support for combat operations, solving WBP with pallets in fixed positions, and considering 20 different items.

? developed a heuristic for 2D packing in air loading, comparing it with methods for solving the *Bin Packing Problem*. Authors conclude that the classical algorithms are inadequate in this context, because they ignore the aircraft balancing constraints.

? presented a method based on linear integer programming to solve the problem of choosing and positioning containers on the *Airbus 340-300*. Safety and stability restrictions were considered, with the objective of minimising fuel consumption.

? developed a web-based application to make efficient use of space and load balancing for an air cargo company. Based on an analysis of historical data, an operational load planning with mathematical optimization is obtained. This container load planning is usually done roughly 2 hours before departure, when all cargo details are in place.

? carried out a case study with heterogeneous pallets. In order to minimise the total cost of shipping, they developed a 3D packing heuristic, with a loading plan for each pallet. Although the authors do not consider load balancing or positioning of pallets in the cargo hold, this method is relevant in commercial and industrial applications, where cargo items tend to be less dense.

? developed a mixed integer linear programming model to arrange a set of items in a military context that optimises the load balance.

? solved WBP by selecting the most profitable subset of containers to be loaded onto an aircraft using mixed-integer programming. Experimental results on real-life data showed significant improvements compared to those obtained manually by an experienced planner.

? presented a heuristic for a real problem of the *Brazilian Air Force*, which consists of defining transport routes with simultaneous collection and delivery from a central distribution terminal.

? developed a mixed-integer program for optimally rearranging a set of pallets into a compartmentalised cargo aircraft, specifically the *Boeing 747*.

? solved APP and WBP as an integer programming problem, which also allows items to be loaded into pallets according to a specific orientation (e.g., this side up).

? presented a mixed integer linear programming model that selects the most profitable pallets, satisfying safety and load balancing constraints on the *Boeing 747-400*. Using a solver, authors solved real problems in less than an hour.

As already mentioned, **?** was the first work that simultaneously modelled WBP and PDP in air cargo transport. The authors demonstrated that this problem is NP-hard and performed some experiments with real data, noting that their model offers better results than those obtained manually.

? proposed a heuristic to solve the *Dynamic Airlift Loading Problem* (DALP). Given a set of palletized cargo items that require transport between two nodes in a time frame, the objective of this problem is to select an efficient subset of aircraft, partition the pallets into aircraft loads and assign them to allowable positions on those aircraft.

? presented a mathematical modelling to optimise the loading of heterogeneous 3D boxes on pallets with a truncated parallelepipeds format. Its objective is to maximise the volume used in containers, considering load balancing restrictions, the presence of fragile items and the possibility of rotating these boxes. **?** developed some heuristics to solve this problem.

? modelled the air transport problem as a 2D packing problem, and presented a heuristic for its optimization in several aircraft, considering load balancing in order to minimise fuel consumption.

?A developed a mathematical model and a tool based on mixed integer programming for optimising cargo in aircraft with different pallet configurations. Balance restrictions and the presence of dangerous items were considered. **?** integrated this tool to a digital simulation model, with a visualisation and validation system, based on sensors that alert about load deviations.

? proposed a new modelling for WBP based on mixed integer programming. Instead of focusing on the centre of gravity (CG) deviation, the authors consider the original CG envelope of the aircraft, with a linearization method for its non-linear constraints.

As can be seen, none of these works address air cargo palletization and load balancing with route optimization in a multi-leg flight plan. This is the objective of our work: to model and elaborate heuristics for a real problem that simultaneously involves 4 intractable sub-problems: APP, WBP, PDP and TSP.

3. Problem context and assumptions

In this section, we describe the context of the problem addressed in this work, as well as the assumptions considered.

3.1. Operational premises

As we are dealing with an extremely complex and diverse problem, we decided to establish some simplifying characteristics:

- At each node of the flight plan, the items to be allocated are characterised by weight, volume, scores, and previously known destinations, but do not have dimensions. We leave the consideration of 2D or 3D items for future work.
- We also disregarded *hazardous* items, which eventually could be treated as high score items and other specific constraints.
- We considered a unique pallet type: the *463L Master Pallet*, a common size platform for bundling and moving air cargo. It is the primary air cargo pallet for more than 70 Air Forces and many air transport companies. This pallet has a capacity of $4500kg$ and $13.7m^3$, is equipped for locking 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`.
- All items allocated on a pallet must have the same destination. A pallet which has not yet reached its destination may receive more items, although it is known that these operations of removing restraining nets increase handling time and the risk of improper delivery. We do not consider oversized cargo in this work, but only cargoes that fit on these pallets.
- Finally, as we are interested in minimising fuel costs, we disregarded some lower ones as *handling operation costs*.

Throughout this text, we call a *consolidated item* a set of items of the same destination stacked on a pallet and covered with a restraining net. It is considered unique, having the same attributes of its components, whose values are the sum of individual scores, weights and volumes. See Figure 2. To ensure accuracy in pickup and delivery operations, consolidated items must remain on board until their destination.

Figure 2: Consolidated items on pallets inside a *Boeing C-17*
Source: https://www.sacprogram.org/en/Pages/Boeing-C-17-Globemaster-III.aspx

3.2. Aircraft and load balancing

We consider real scenarios with a smaller or a larger aircraft with payloads of $26,000kg$ or $75,000kg$ respectively. Both layouts are represented in Figures 3 and 4, where the pallets are identified by p_i.

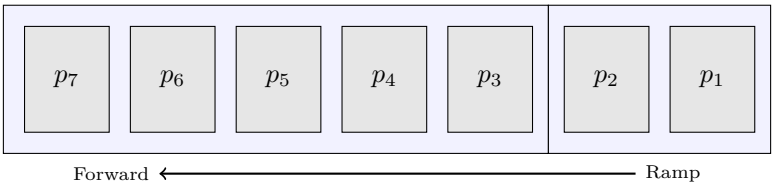

Figure 3: Smaller aircraft layout

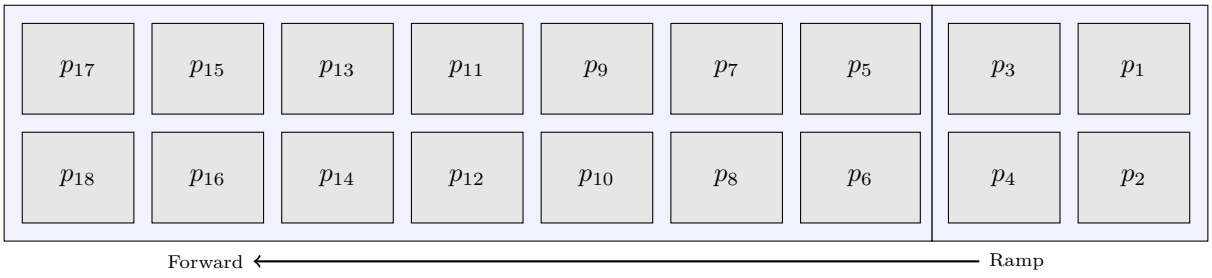

Figure 4: Larger aircraft layout

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* [1]: $0.556m$ in the smaller aircraft and $1.17m$ in the larger one. See Figure 5.

[1] Chord is the distance between the leading and trailing edges of the wing, measured parallel to the normal airflow over the wing. The average length of the chord is known as the *Mean Aerodynamic Chord* (MAC).

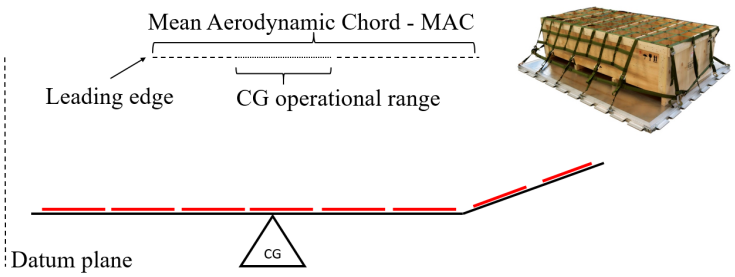

Figure 5: Aircraft longitudinal cut showing red lines as pallets

Tables 3 and 4 show the parameters in both cases. *CGx* and *CGy* refer to the relative distances of pallet centroids (in metres) in relation to the CG of aircraft along both axes. In both aircraft, as the ramps have an inclination of 25 degrees, we made the necessary corrections in *CGx*, *Weight* and *Volume limits* of the corresponding pallets. The monetary costs of both aircraft are also indicated: per unit of distance in flights between legs (c_d) and per deviation in the CG (c_g). It is important to consider that c_g tends to zero as the aircraft attitude tends to level.

Table 3: Smaller aircraft parameters

Limits	*Payload*: $26,000kg$				$limit^{CG}_{long}$: $0.556m$		
Pallets	p_7	p_6	p_5	p_4	p_3	p_2	p_1
CGx (m)	-5.10	-2.70	-0.30	2.10	4.50	6.25	8.39
Weight limits (kg)	4,500	4,500	4,500	4,500	4,500	4,000	3,500
Volume limits (m^3)	13.7	13.7	13.7	13.7	13.7	8.9	6.9
Costs	c_d: US$ 1.100/km				$c_g = 0.05$		

Table 4: Larger aircraft parameters

Limits	*Payload*: $75,000kg$			$limit^{CG}_{long}$: $1.170m$			$limit^{CG}_{lat}$: $0.19m$		
Pallets	p_{17}	p_{15}	p_{13}	p_{11}	p_9	p_7	p_5	p_3	p_1
	p_{18}	p_{16}	p_{14}	p_{12}	p_{10}	p_8	p_6	p_4	p_2
CGx (m)	-17.57	-13.17	-8.77	-4.40	0	4.40	8.77	11.47	14.89
	-17.57	-13.17	-8.77	-4.40	0	4.40	8.77	11.47	14.89
CGy (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
Weight limits (kg)	4,500	4,500	4,500	4,500	4,500	4,500	4,500	4,000	3,000
Volume limits (m^3)	14.8	14.8	14.8	14.8	14.8	14.8	14.8	10.0	7.0
Costs	c_d: US$ 4.900/km						$c_g = 0.05$		

We also make the following assumptions:

- on each pallet, the items are distributed in such a way that their CG coincides with the centroid of the pallet;
- the CG of the payload must be at a maximum longitudinal distance of $limit^{CG}_{long}$ from the CG of the aircraft;
- in the larger aircraft, the CG of the payload must be at a maximum lateral distance of $limit^{CG}_{lat}$ from the CG of the aircraft;
- in the larger aircraft, pallets are distributed in two identical rows (with odd and even indices, respectively), and their centroids are at a distance d^{CG}_{pallet} from the centre-line of the aircraft.

3.3. Problem Summary

Informally, ACLP+RPDP can be summarised as follows:

max	(items score sum) / (tour cost) of picked up and delivered items at each node on a tour
s.t.	Along a tour, the set of unvisited nodes is updated.
	In each node, an item may be included in at most one pallet.
	In each node, consolidated items are composed of items with the same destination.
	Weight, volume and score of a consolidated item are the corresponding sum of their components.
	Consolidated items remain on board until their destinations.
	Consolidated items can only be included in the same pallet if their destinations are the same.
	Only items destined for the remaining nodes can be loaded.
	The lateral and longitudinal torques must be within the operational range of the aircraft.
	Weight and volume limitations of pallets must be respected.
	The total weight must be less than the aircraft payload or the total pallet capacity, whichever is the lowest.

4. The mathematical modelling

Given the assumptions, scenarios and parameters described in the previous section, we are ready to present the mathematical modelling of ACLP+RPDP, which is one of the contributions of our work.

Let $L = \{l_0, l_1, \ldots, l_K\}$ be the set of $K+1$ nodes (or destinations), where l_0 is the origin and end of a flight plan. Let $d(l_i, l_j)$ be the distance from l_i to l_j, where $0 \leq i, j \leq K$. By definition, $d(l_i, l_i) = 0$. Let L_k be the set of remaining nodes when the aircraft is in l_k, $0 \leq k \leq K$. Therefore, $L_0 = L$ and $L_K = \{l_0\}$.

Let $C = \{c_{ij}\}$ be the cost matrix of flights, where $c_{ij} = c_d * d(l_i, l_j), 0 \leq i, j \leq K$.

Let $S_K = \{s : \{1, \ldots, K\} \rightarrow \{1, \ldots, K\}\}$ be the set of $K!$ permutations, which correspond to all possible tours (or itineraries) that have l_0 as origin and end, passing through the others K nodes.

Let $M = \{p_1, p_2, \ldots, p_m\}$ the set of m pallets. Each pallet p_i, $1 \leq i \leq m$, has weight capacity $p_i.w$, volume capacity $p_i.v$, pallet destinations $p_i.to[k]$, $0 \leq k \leq K$, and distance to the CG of aircraft $p_i.d$. $p_i.to[k]$ denotes that pallet p_i may assume a different destination in each node k.

Let $N_k = \{t_1^k, t_2^k, \ldots, t_{n_k}^k\}$ be the set of n_k items to be loaded in node l_k, $0 \leq k \leq K$. Each item t_j^k, $1 \leq j \leq n_k$, has score $t_j^k.s$, weight $t_j^k.w$, volume $t_j^k.v$, and destination $t_j^k.to \in L_k$. Let $N = \bigcup_{0 \leq k \leq K} N_k$ be the set of items of all nodes along a tour.

Let $Q_k = \{a_1^k, a_2^k, \ldots, a_{m_k}^k\}$ be the set of consolidated items loaded in $m_k \leq m$ pallets when the aircraft arrives at node l_k, with $0 \leq k \leq K$. a_i^k, $1 \leq i \leq m_k$, is the group of picked-up items that were allocated on pallet p_i in some of the previous nodes. a_i^k has total weight $a_i^k.w$, total volume $a_i^k.v$, and destination $a_i^k.to \in L_k \cup \{l_k\}$. If $a_i^k.to = l_k$, then a_i^k is unloaded, and p_i will be available for reloading; otherwise, a_i^k remains on the aircraft, eventually in another pallet, and with items of N_k having the same destination.

Let X_{ij}^k and Y_{iq}^k be binary variables, where $0 \leq k \leq K$, $1 \leq j \leq n_k$, $1 \leq i \leq m$ and $1 \leq q \leq m_k$. $X_{ij}^k = 1$ if t_j^k is assigned to p_i in node l_k, and 0 otherwise. $Y_{iq}^k = 1$ if a_q^k is assigned to p_i in node l_k, and 0 otherwise. By definition, $Y_{iq}^0 = 0$. Allocations of items or consolidated items to pallets in node l_k can be seen as a bipartite graph $G_k(V_k, E_k)$, where $V_k = M \cup N_k \cup Q_k$, $E_k = E_k^N \cup E_k^Q$, $(p_i, t_j^k) \in E_k^N$ if $X_{ij}^k = 1$, and $(p_i, a_q^k) \in E_k^Q$ if $Y_{iq}^k = 1$.

The mathematical modelling of this problem is described in the equations below.

$$\max_{\pi \in S_K} f_\pi(\tilde{s}, \tilde{c}) \tag{1}$$

$$\tilde{s} = \sum_{k=0}^{K} \sum_{i=1}^{m} \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.s \tag{2}$$

$$\tilde{c} = c_{0,\pi(1)} \times (1 + c_g \times |\epsilon_0|) + \sum_{k=1}^{K-1} [c_{\pi(k),\pi(k+1)} \times (1 + c_g \times |\epsilon_k|)] + c_{\pi(K),0} \times (1 + c_g \times |\epsilon_K|) \tag{3}$$

$$maxW = \min(Payload, \sum_{i=1}^{m} p_i.w) \tag{4}$$

$$\tau_k = \sum_{i=1}^{m}[p_i.d \times (\sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m} Y_{iq}^k \times a_q^k.w)];\ k \in \{0, 1, \ldots, K\} \tag{5}$$

$$\epsilon_k = \frac{\tau_k}{maxW \times limit_{long}^{CG}};\ k \in \{0, 1, \ldots, K\} \tag{6}$$

$$L_0 = L; L_k = L_{k-1} - \{l_{\pi(k)}\};\ k \in \{1, 2, \ldots, K\} \tag{7}$$

$$Y_{iq}^0 = 0; a_i^0.w = 0; a_i^0.v = 0; a_i^0.to = -1;\ i, q \in \{1, 2, \ldots, m\} \tag{8}$$

$$X_{ij}^k = 0 \text{ if } t_j^k.to \notin L_k;\ i \in \{1, 2, \ldots, m\};\ j \in \{1, 2, \ldots, n_k\};\ k \in \{1, 2, \ldots, K\} \tag{9}$$

$$Y_{iq}^k = 0 \text{ if } a_i^k.to \notin L_k;\ i \in \{1, 2, \ldots, m\};\ q \in \{1, 2, \ldots, m_k\};\ k \in \{1, 2, \ldots, K\} \tag{10}$$

$$a_i^{\pi(k+1)}.w = \sum_{j=1}^{n_k} X_{ij}^{\pi(k)} \times t_j^{\pi(k)}.w + \sum_{q=1}^{m_k} Y_{iq}^{\pi(k)} \times a_q^{\pi(k)}.w;\ i \in \{1, 2, \ldots, m_k\}; k \in \{0, 1, \ldots, K-1\} \tag{11}$$

$$a_i^{\pi(k+1)}.v = \sum_{j=1}^{n_k} X_{ij}^{\pi(k)} \times t_j^{\pi(k)}.v + \sum_{q=1}^{m_k} Y_{iq}^{\pi(k)} \times a_q^{\pi(k)}.v;\ i \in \{1, 2, \ldots, m_k\}; k \in \{0, 1, \ldots, K-1\} \tag{12}$$

$$a_i^{\pi(k+1)}.to = t_j^{\pi(k)}.to \text{ if } X_{ij}^{\pi(k)} = 1 \text{ and } t_j^{\pi(k)}.to \in L_{k+1};\ i \in \{1, 2, \ldots, m_k\};\ j \in \{1, 2, \ldots, n_k\};\ k \in \{0, 1, \ldots, K-1\} \tag{13}$$

$$LatIt_k = \sum_{i=1}^{m}\sum_{j=1}^{n_k}(X_{ij}^k \times t_j^k.w \times (i\%2) - X_{ij}^k \times t_j^k.w \times (i+1)\%2);\ k \in \{0, 1, \ldots, K\} \tag{14}$$

$$LatCons_k = \sum_{i=1}^{m}\sum_{q=1}^{m_k}(Y_{iq}^k \times a_q^k.w \times (i\%2) - Y_{iq}^k \times a_q^k.w \times (i+1)\%2);\ k \in \{0, 1, \ldots, K\} \tag{15}$$

$$s.t. : d_{pallet}^{CG} \times |LatIt_k + LatCons_k| \leq \sum_{i=1}^{m} p_i.w \times limit_{lat}^{CG};\ k \in \{0, 1, \ldots, K\} \tag{16}$$

$$s.t. : |\tau_k| \leq maxW \times limit_{long}^{CG};\ k \in \{0, 1, \ldots, K\} \tag{17}$$

$$s.t. : \sum_{i=1}^{m}(\sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.w) \leq maxW;\ k \in \{0, 1, \ldots, K\} \tag{18}$$

$$s.t. : \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.w \leq p_i.w;\ i \in \{1, 2, \ldots, m_k\};\ k \in \{0, 1, \ldots, K\} \tag{19}$$

$$s.t. : \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.v + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.v \leq\ p_i.v;\ i \in \{1, 2, \ldots, m_k\};\ k \in \{0, 1, \ldots, K\} \tag{20}$$

$$s.t. : \sum_{i=1}^{m} X_{ij}^k \leq 1;\ j \in \{1, 2, \ldots, n_k\};\ k \in \{0, 1, \ldots, K\} \tag{21}$$

$$s.t.: Y_{iq}^k = 1 \text{ if } a_q^k.to \in L_k;\ q \in \{1, 2, \ldots, m_k\};\ k \in \{0, 1, \ldots, K\} \tag{22}$$

$$s.t.: p_i.to[k] = t_j^k.to \text{ if } X_{ij}^k = 1;\ i \in \{1, 2, \ldots, m\};\ j \in \{1, 2, \ldots, n_k\};\ k \in \{1, 2, \ldots, K\} \tag{23}$$

$$s.t.: p_i.to[k] = a_q^k.to \text{ if } Y_{iq}^k = 1;\ i \in \{1, 2, \ldots, m\};\ q \in \{1, 2, \ldots, m_k\};\ k \in \{1, 2, \ldots, K\} \tag{24}$$

The objective of this problem is to find a permutation $\pi \in S_K$ that maximises the function $f_\pi(\tilde{s}, \tilde{c})$ 1. In this way, the flight plan will be $l_0, l_{\pi(1)}, \ldots, l_{\pi(K)}, l_0$. $\tilde{s}$ is the total score of transported items 2 and $\tilde{c}$ is the total cost of fuel consumed 3. As can be seen, $\tilde{c}$ corresponds to the fuel consumption due to the flights carried out and the CG deviation of the transported cargo. Throughout this work, for simplicity, we use $f = \tilde{s}/\tilde{c}$.

The maximum load will be the minimum between the payload and the capacity supported by the pallets 4. Considering the maximum longitudinal distance allowed for the CG, all torques 5 and deviations 6 are calculated.

For each step of the flight plan, the set of unvisited nodes is updated 7. Although there are no items consolidated at the beginning of the flight plan, we defined these variables for ease of notation 8. Items destined outside the rest of the flight plan will not be loaded (9 and 10).

Consolidated items appear when there are items on the pallets that will not be unloaded on the next node. Its weights 11 and volumes 12 correspond to all the items that were on the pallet, since all these items have the same destination. On subsequent nodes, consolidated items can be allocated with other items of same destination 13.

Equations 14 and 15 respectively, are applied only to the larger aircraft, and calculate the lateral torques of items and consolidated items loaded in both rows of pallets, whose constraint is described in 16. Similarly, 17 is the longitudinal torque constraint, which is applied to both aircraft sizes.

The weight limitation of the aircraft must be respected 18. The sum of weights 19 and volumes 20 in each pallet must not exceed its capacity. Each item is associated with a pallet at most 21.

Consolidated items remain on board 22 until their destinations. At each node, an item (23) and a consolidated item (24) must only be allocated to a pallet if the destinations are the same.

5. Resolution strategy

Once the assumptions of this work and the mathematical modelling of the problem are presented, it is easy to see that ACLP+RPDP is NP-hard. In a similar way to (**?**, p. 6), consider the simple case where $K = 1$ (one leg), $m = 2$ (two pallets around the aircraft CG), $2n$ sufficiently light items with same scores in l_0, and no items in l_1. Under these conditions, through polynomial reductions for the *Set-Partition Problem*, it is possible to demonstrate that the decision problem associated with ACLP+RPDP is NP-complete.

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 modelling presented in the previous section, we verify that *Mixed-Integer Programming* (MIP) 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 maximise the objective function f.

Our strategy is based on the fact that, in real cases, K is usually small. Specifically, we will consider $K \leq 6$ throughout this work, which is a higher value than usual in *Brazilian Air Force* missions. As a result, if we have 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 value for the f function.

On the other hand, our tactic will be, at each shipping node, to predefine the destinations of the pallets at that node. In this way, we will reserve a number of pallets proportional to the volume demanded by each destination at the shipping node. We 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, we will use heuristics and a MIP solver to find the best possible node-by-node solutions. This strategy is summarized in Algorithm 1.

Algorithm 1 $ACLP + RPDP(scenario, volume)$

1: Let L, M, C be according to *scenario*
2: $N \leftarrow ItemsGeneration(scenario, volume)$
3: **for** each *method*
4: **for** each $\pi \in S_K$
5: $f_\pi \leftarrow SolveTour(\pi, L, M, C, N, method)$
6: $answer[scenario, volume, method] \leftarrow \max f$
7: **return** *answer*

In this algorithm, there are six values for the *scenario* parameter, according to Table 5, which defines K, the sets of nodes, the aircraft, the pallets and the costs from Tables 3 or 4 that will be used (line 1).

The other parameter *volume* is a value greater than 1, which corresponds, at each node l_k, to the ratio between the sum of the volumes of the items ($\sum_{j=1}^{n_k} t_j^k.v$) and the load capacity of the pallets ($\sum_{i=1}^{m} p_i.v$). This parameter is passed to *ItemsGeneration* (line 2), responsible for creating the items to be shipped, which will be presented in the next section (Algorithm 9).

method corresponds to a MIP solver or one of the heuristics that we will present in subsection 5.2. The loop of lines 4-5 goes through all permutations π, where the node-by-node resolutions are performed by *SolveTour*, whose result is stored in f_π. The best result among all $K!$ tours will be the answer for *scenario*, *volume* and *method* (line 6).

Table 5: Testing scenarios

Scenario	K	L	**Aircraft**
1	2	$\{l_0, l_1, l_2\}$	smaller
2	2	$\{l_0, l_1, l_2\}$	larger
3	3	$\{l_0, l_1, l_2, l_3\}$	larger
4	4	$\{l_0, l_1, l_2, l_3, l_4\}$	larger
5	5	$\{l_0, l_1, l_2, l_3, l_4, l_5\}$	larger
6	6	$\{l_0, l_1, l_2, l_3, l_4, l_5, l_6\}$	larger

Next, we will present two subsections: in the first we explain how *SolveTour* is executed, while in the second we will present the heuristics developed for node-by-node resolutions.

5.1. SolveTour algorithm

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

As we mentioned in the previous section, all tours start and end at l_0 (lines 1-2). After initializing the score and cost values (lines 3-4), there is a loop for the $K+1$ flights (lines 5-19). Initially we set pallets destination as -1 (line 7). When the aircraft is at node l_0, the initial graph G_1 is empty because it has no consolidated items 10. Otherwise, the set L_k of remaining nodes is updated (line 12), and *UpdateConsolidated* (line 13) returns the set of consolidated items that have not yet reached their destination and remain on board, rearranging them on the pallets to minimize CG deviation. This allocation is stored in graph G_1 (line 14).

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. *SetPalletsDestination* (line 15) presets the destination of each pallet based on the volume demands of the current node, without changing the pallets destination with consolidated items.

Finally, *SolveNode* includes the edges corresponding to the items shipped at the current node, returning the graph G_2 (line 16). The score and the CG deviation of this graph are calculated (line 17) and accumulated (lines 18-19), allowing the final result of this tour (line 20).

Algorithm 2 $SolveTour(\pi, L, M, C, N, method)$

1: $\pi(0) \leftarrow 0$
2: $\pi(K+1) \leftarrow 0$
3: $score \leftarrow 0$
4: $cost \leftarrow 0$
5: **for** $k \leftarrow 0$ to K
6: **for** $i \leftarrow 1$ to m
7: $p_i.to[k] \leftarrow -1$
8: **if** $k = 0$
9: $L_0 \leftarrow L$
10: Let $G_1(M \cup N_0, \varnothing)$
11: **else**
12: $L_k \leftarrow L_k - \pi(k)$
13: $Q_{\pi(k)}, E^Q_{\pi(k)}, M \leftarrow UpdateConsolidated(\pi(k))$
14: Let $G_1(M \cup N_{\pi(k)} \cup Q_{\pi(k)}, E^Q_{\pi(k)})$
15: $M \leftarrow SetPalletsDestination(\pi(k))$
16: $G_2 \leftarrow SolveNode(method, \pi(k), G_1)$
17: $s, \epsilon \leftarrow ScoreAndDeviation(\pi(k), G_2)$
18: $score \leftarrow score + s$
19: $cost \leftarrow cost + c_{\pi(k),\pi(k+1)} * (1 + c_g * |\epsilon|)$
20: **return** $score/cost$

UpdateConsolidated, described in Algorithm 3, finds the best allocation for the consolidated items that remain on board. Initially, the set Q is created, with the consolidated items that did not reach their destination (lines 2-6). Then *MinCGDeviation* (line 7) is run through a MIP solver with the equations 25, 26 and 27, to relocate the consolidated items 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_k very quickly. Finally, the destination of each pallet with consolidated items is updated (lines 8-11).

Algorithm 3 $UpdateConsolidated(k)$

1: Let $Q_k = \{a^k_1, a^k_2, \ldots, a^k_{m_k}\}$
2: $Q \leftarrow \varnothing$
3: **for** $q \leftarrow 1$ to m_k
4: **if** $a^k_q.to \in L_k$
5: $Q \leftarrow Q \cup \{a^k_q\}$
6: $m_k \leftarrow |Q|$
7: $E^Q_k \leftarrow MinCGDeviation(k, Q)$
8: **for** $i \leftarrow 1$ to m
9: **for** $q \leftarrow 1$ to m_k
10: **if** $(p_i, a^k_q) \in E^Q_k$
11: $p_i.to[k] \leftarrow a^k_q.to$
12: **return** Q, E^Q_k, M

$$\text{minimize } |\sum_{i=1}^{m}\sum_{q=1}^{m_k} Y^k_{iq} \times p_i.d \times a^k_q.w|, \text{ where } Q = \{a^k_1, a^k_2, \ldots, a^k_{m_k}\} \tag{25}$$

$$s.t.: \sum_{i=1}^{m} Y^k_{iq} = 1;\ q \in \{1, 2, \ldots, m_k\} \tag{26}$$

$$s.t.: \sum_{q=1}^{m_k} Y_{iq}^k \leq 1;\ i \in \{1, 2, \ldots, m\} \tag{27}$$

SetPalletsDestination, which set pallets destination not yet defined, is described in Algorithm 4. The vectors $vol[0..K]$ and $PalCons[0..K]$ store the total volume of items and the number of pallets with consolidated items destined for each node, respectively (lines 1-20). The destinations of each pallet are defined proportionally to the volume of items, regarding the pallets with consolidated items (lines 21-30). The node with the maximum volume demand defines the destination of any remaining pallets (lines 31-33).

Algorithm 4 *SetPalletsDestination*(k)

```
1: for x ← 0 to K
2:     vol[x] ← 0
3:     PalCons[x] ← 0
4: max ← 0
5: total ← 0
6: for j ← 1 to n_k
7:     d ← t_j^k.to
8:     if d ∈ L_k
9:         vol[d] ← vol[d] + t_j^k.v
10:        total ← total + t_j^k.v
11:        if vol[d] > vol[max]
12:            max ← d
13: for q ← 1 to m_k
14:    d ← a_q^k.to
15:    PalCons[d] ← PalCons[d] + 1
16:    if d ∈ L_k
17:        vol[d] ← vol[d] + a_q^k.v
18:        total ← total + a_q^k.v
19:        if vol[d] > vol[max]
20:            max ← d
21: for x ← 0 to K
22:    if vol[x] ≠ 0
23:        quant ← max{1, ⌊m × vol[x]/total⌋} − PalCons[x]
24:        np ← 0
25:        for i ← 1 to m
26:            if np = quant
27:                break
28:            if p_i.to[k] = −1
29:                p_i.to[x] ← x
30:                np ← np + 1
31: for i ← 1 to m
32:    if p_i.to[k] = −1
33:        p_i.to[x] ← max
34: return M
```

ScoreAndDeviation is described in Algorithm 5, which 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 (lines 7-16), accumulating the scores (lines 11-15) and the torques (lines 12-16) of the shipped items, allowing the final calculation of the CG deviation (lines 17-19).

Algorithm 5 *ScoreAndDeviation*(k, G)

Let $G(V_k, E_k^N \cup E_k^Q)$
Let $X_{ij}^k = 1$ if $(p_i, t_j^k) \in E_k^N$, $1 \leq i \leq m$, $1 \leq j \leq n_k$
Let $Y_{iq}^k = 1$ if $(p_i, a_q^k) \in E_k^Q$, $1 \leq i \leq m$, $1 \leq q \leq m_k$
$s \leftarrow 0$
$\epsilon \leftarrow 0$
$weight \leftarrow 0$
for $i \leftarrow 1$ to m
    $weight \leftarrow weight + p_i.w$
    for $j \leftarrow 1$ to n_k
        if $X_{ij}^k = 1$
            $s \leftarrow s + t_{ij}^k.s$
            $\epsilon \leftarrow \epsilon + t_{ij}^k.w * p_i.d$
    for $q \leftarrow 1$ to m_k
        if $Y_{iq}^k = 1$
            $s \leftarrow s + a_{iq}^k.s$
            $\epsilon \leftarrow \epsilon + a_{iq}^k.w * p_i.d$
$maxWeight \leftarrow \min(Payload, weight)$
$maxTorque \leftarrow maxWeight \times limit_{long}^{CG}$
$\epsilon \leftarrow \epsilon / maxTorque$
return s, ϵ

5.2. Node-by-node solutions

In this subsection we present implementations of $SolveNode(method, k, G)$, where *method* corresponds to a MIP solver or a heuristic, k is the index of the current node and G is the allocation graph of the consolidated items that remain on board at node l_k.

An important issue on meta-heuristics is that they are known for the hard work in tuning parameters. For this, when we use the word *empirically* throughout the text, we want to say that we accomplished many tests with the hardest instances to fit these values. This does not guarantee best results for all instances, but, on average, overall results are acceptable. Another important observation is that our heuristics were designed based on traditional ideas from the literature **?????** and on the experiences learned during tests.

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

Data structure:	Items, pallets and edges are arrays of *python* class objects, which are initialized with the same data loaded from text files.
Common procedures:	All heuristics use the same procedures for selecting and inserting edges, and printing solutions.
Stopping criterion:	All heuristics have the same time limit of 0.7 seconds per node.

The selection of edges for E_k^N uses the *edge attractiveness* θ_{ij}^k 28, which can be understood as the tendency to allocate the item t_j^k to the pallet p_i. It is directly proportional to the score, and inversely proportional to the volume and the torque of each item.

$$\theta_{ij}^k = \frac{t_j^k.s}{t_j^k.v} \times \left(1 - \frac{t_j^k.w \times |p_i.d|}{\max_s\{t_s^k.w\} \times \max_q\{|p_q.d|\}}\right);\ i \in \{1, 2, \ldots, m\},\ j \in \{1, 2, \ldots, n_k\} \tag{28}$$

We use four types of random selections:

- $RandomReal(r_1, r_2)$: randomly selects a real number in $[r_1, r_2]$, where r_1 and r_2 are real numbers;
- $RandomInt(i_1, i_2)$: randomly selects a integer number in $[i_1, i_2]$, where i_1 and i_2 are integer numbers;
- $RandomChoice(set)$: randomly selects an element from *set*;

- *Roulette*(set) biased through ϕ: selects an element from set, where the probability of each element is proportional to the value of a given function ϕ defined on set.

We also developed Algorithm 6, which generates a greedy allocation of the items available in node l_k, according to the non-ascending order of θ_{ij}^k, and considering the consolidated items already shipped. Edges are included in this allocation as long as they respect feasibility constraints. Furthermore, the volume of each pallet p_i cannot exceed $p_i.v \times limit$, where $0 < limit \leq 1$ is a parameter.

Algorithm 6 *Greedy*$(k, G, limit)$

```
1: Let G(V_k, E_k^Q)
2: E_k^N ← ∅
3: for i ← 1 to m
4:     pv[i] ← 0
5: for q ← 1 to m_k
6:     if (p_i, a_q^k) ∈ E_k^Q
7:         pv[i] ← pv[i] + a_q^k.v
8: for each e_ij^k in non-ascending order of θ_ij^k
9:     if (E_k^N ∪ {e_ij^k} is feasible) and (pv[i] ≤ p_i.v × limit)
10:        E_k^N ← E_k^N ∪ {e_ij^k}
11:        pv[i] ← pv[i] + t_j^k.v
12: return G(V_k, E_k^N ∪ E_k^Q)
```

To compare allocation graphs generated by the heuristics, we choice the one that offer a higher associated score, since the CG deviation is limited by the restrictions. In the pseudocodes presented below, given an allocation graph G for the node l_k, $f_s(G)$ is equivalent to the score s returned by *ScoreAndDeviation*(k, G), described in Algorithm 5.

Below, we present six implementations for *SolveNode*: the first using a MIP solver, and five with developed heuristics: *Ant Colony Optimization* (ACO), *Noising Method Optimization* (NMO), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP) and *Shims*.

5.2.1. MIP solver

In order to find a viable solution for ACLP+RPDP, the strategy adopted in *SolveTour* previously defines the values of some variables. Concretely, all permutations between the nodes are tested, the set of nodes to be visited is updated, the consolidated items that remain on board are reallocated to minimize the CG deviation, and the pallets destination is determined according to the volume of items available for shipment. In this way, the mathematical modelling for $SolveNode(MIP, k, G)$ becomes simpler, which finds an allocation of available items in node l_k using previously defined values of L_k, $p_i.to[k]$, Y_{iq}^k and a_q^k.

This simplified modelling is described in the equations 29 to 41, which maximizes the score of the t_j^k items to be shipped, maintaining load balancing. The variables X_{ij}^k define the set of edges E_k^N, which will be added to the graph G.

$$\max f_s = \sum_{i=1}^{m} \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.s \tag{29}$$

$$maxW = \min(Payload, \sum_{i=1}^{m} p_i.w) \tag{30}$$

$$\epsilon_k = \sum_{i=1}^{m} [p_i.d \times (\sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m} Y_{iq}^k \times a_q^k.w)]/(maxW \times limit_{long}^{CG}) \tag{31}$$

$$X_{ij}^k = 0 \text{ if } t_j^k.to \notin L_k;\ i \in \{1, 2, \ldots, m\};\ j \in \{1, 2, \ldots, n_k\} \tag{32}$$

$$LatIt_k = \sum_{i=1}^{m}\sum_{j=1}^{n_k}(X_{ij}^k \times t_j^k.w \times (i\%2) - X_{ij}^k \times t_j^k.w \times (i+1)\%2) \tag{33}$$

$$LatCons_k = \sum_{i=1}^{m}\sum_{q=1}^{m_k}(Y_{iq}^k \times a_q^k.w \times (i\%2) - Y_{iq}^k \times a_q^k.w \times (i+1)\%2) \tag{34}$$

$$s.t. : d_{pallet}^{CG} \times |LatIt_k + LatCons_k| \leq \sum_{i=1}^{m} p_i.w \times limit_{lat}^{CG} \tag{35}$$

$$s.t. : |\tau_k| \leq maxW \times limit_{long}^{CG} \tag{36}$$

$$s.t. : \sum_{i=1}^{m}(\sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.w) \leq maxW \tag{37}$$

$$s.t. : \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.w + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.w \leq p_i.w;\ i \in \{1,2,\ldots,m_k\} \tag{38}$$

$$s.t. : \sum_{j=1}^{n_k} X_{ij}^k \times t_j^k.v + \sum_{q=1}^{m_k} Y_{iq}^k \times a_q^k.v \leq\ p_i.v;\ i \in \{1,2,\ldots,m_k\} \tag{39}$$

$$s.t. : \sum_{i=1}^{m} X_{ij}^k \leq 1;\ j \in \{1,2,\ldots,n_k\} \tag{40}$$

$$s.t. : p_i.to[k] = t_j^k.to \text{ if } X_{ij}^k = 1;\ i \in \{1,2,\ldots,m\};\ j \in \{1,2,\ldots,n_k\} \tag{41}$$

5.2.2. Ant Colony Optimization (ACO)

In ACO **?**, a population of A ants performs independent search, where each ant a finds its own allocation graph G^a. Each edge $e_{ij}^k = (p_i, t_j^k)$ has a general attractiveness $ga_{ij}^k = (\tau_{ij}^k)^\alpha + (\theta_{ij}^k)^\beta$, where τ_{ij}^k is the pheromone of this edge, and α and β are constants. If e_{ij}^k is included in G^a, ant a increases the pheromone τ_{ij}^k. Algorithm 7 presents this heuristic.

Algorithm 7 *SolveNode*(ACO, k, G)

1: Let $G(V_k, E_k^Q)$
2: $G^*(V_k, E_k^{N*} \cup E_k^Q) \leftarrow Greedy(k, G, 1)$
3: $G^+(V_k, E_k^{N+} \cup E_k^Q) \leftarrow G^*(V_k, E_k^{N*} \cup E_k^Q)$
4: $A \leftarrow 4$
5: $ga_{ij}^k \leftarrow 0$, $1 \leq i \leq m$, $1 \leq j \leq n_k$
6: $\tau_{ij}^k \leftarrow 1$, $1 \leq i \leq m$, $1 \leq j \leq n_k$
7: $\alpha \leftarrow 1.0$
8: $\beta \leftarrow 4.0$
9: $\rho \leftarrow 0.2$
10: $stagnant \leftarrow 0$
11: **while** $(stagnant < 3)$ **and** $(runtime < 0.7s)$
12: **for** $a \leftarrow 1$ to A
13: $G^a(V_k, E_k^{Na} \cup E_k^Q) \leftarrow G^+(V_k, E_k^{N+} \cup E_k^Q)$
14: $E \leftarrow \{(p_i, t_j^k) \mid (p_i, t_j^k) \notin E_k^{Na}\}$
15: **while** $|E| > 0$
16: $e \leftarrow Roulette(E)$ biased through ga_{ij}^k
17: $E \leftarrow E - \{e\}$
18: **if** $E_k^{Na} \cup \{e\}$ is feasible
19: $E_k^{Na} \leftarrow E_k^{Na} \cup \{e\}$
20: **if** $f_s(G^a) > f_s(G^+)$
21: $G^+(V_k, E_k^{N+} \cup E_k^Q) \leftarrow G^a(V_k, E_k^{Na} \cup E_k^Q)$
22: **for** each $(p_i, t_j^k) \in E_k^{N+}$
23: $\tau_{ij}^k \leftarrow \tau_{ij}^k - \Delta\tau_{ij}^k$
24: $ga_{ij}^k \leftarrow (\tau_{ij}^k)^\alpha + (\theta_{ij}^k)^\beta$
25: **if** $f_s(G^+) > f_s(G^*)$
26: $stagnant \leftarrow 0$
27: $G^*(V_k, E_k^{N*} \cup E_k^Q) \leftarrow G^+(V_k, E_k^{N+} \cup E_k^Q)$
28: **for** each $(p_i, t_j^k) \in E_k^{N*}$
29: $\tau_{ij}^k \leftarrow (1 - \rho)\tau_{ij}^k$
30: $\tau_{ij}^k \leftarrow \tau_{ij}^k + \Delta\tau_{ij}^k$
31: $ga_{ij}^k \leftarrow (\tau_{ij}^k)^\alpha + (\theta_{ij}^k)^\beta$
32: **else**
33: $stagnant \leftarrow stagnant + 1$
34: **for** all τ_{ij}^k
35: $\tau_{ij}^k \leftarrow (1 - \rho)\tau_{ij}^k$
36: **return** $G^*(V_k, E_k^{N*} \cup E_k^Q)$

A greedy solution is obtained in line 2, being initially stored in G^* and G^+. The number A of ants was empirically set to 4 (line 4), yielding solutions with a good relation quality/time. The $m \times n_k$ matrices τ_{ij}^k and ga_{ij}^k (lines 5-6) are used and updated by all ants.

In each generation, A ants are created (lines 12-24). The number of generations depends on the time limit or stagnation in improving results (line 11). Each ant a elaborates its allocation graph G^a, considering the current values of ga_{ij}^k. To enforce solution diversification, if (p_i, t_j^k) is included in G^a then pheromone τ_{ij}^k is decreased (line 23). Similarly, the values of ga_{ij}^k are also updated (line 24), where the exponents α and β were chosen empirically: α was set to 1.0 and β to 4.0 (lines 7-8). For further explanation, see **?**.

The best solution of each generation is stored in G^+ (lines 20-21), and the update of G^* and the stopping criterion are performed on lines 25-33. Each edge (p_i, t_j^k) in some solution has its corresponding pheromone updated in a cumulative way (line 30), that is, more pheromone will be given to the most relevant edges, where the increment is defined in 42. The same with ga_{ij}^k (line 31).

$$\Delta\tau_{ij}^{k} = \frac{ga_{ij}^{k} \times t_{j}^{k}.s}{A \times t_{j}^{k}.v} \tag{42}$$

Pheromone τ_{ij}^{k} evaporates (lines 29 and 35), where ρ is a evaporation rate, which can be seen as a mechanism that delays premature convergence towards a sub-optimal solution. ρ was empirically set to 0.2 (line 9).

5.2.3. Shims

Finally, we present a new heuristic designed specifically for ACLP+RPDP, which we named *Shims*. Like in mechanics, shims are collections of spacers to fill gaps, which may be composed of parts with different thicknesses (see Figure 6). This strategy is based on a practical observation: usually, subsets of smaller and lighter items are saved for later adjustments to the remaining clearances.

Figure 6: Shims of various thicknesses
Source: www.mscdirect.com/product/details/70475967

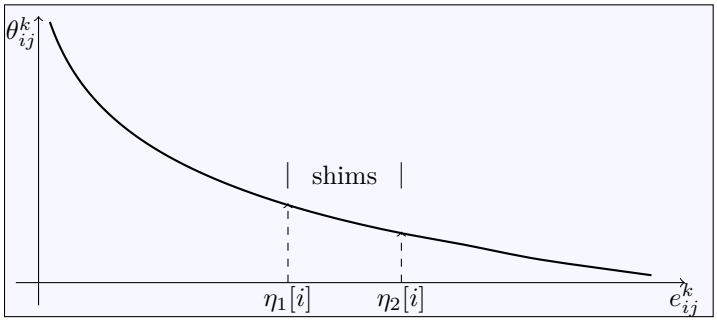

Figure 7: n_k edges e_{ij}^{k} of p_i sorted by θ_{ij}^{k} in non-ascending order

In this heuristic, described in Algorithm 8, pallets p_i are considered in non-descending order of $|p_i.d|$ (line 2). For each p_i, the n_k possible edges e_{ij}^{k} are considered in non-increasing order of θ_{ij}^{k} (line 4) in three phases: the greedy phase (lines 7-21), the composition phase of the shims (lines 22-39) and the selection phase of the best shim (lines 40-43).

Figure 7 represents the n_k possible edges e_{ij}^{k} of p_i sorted by θ_{ij}^{k}. *Shims* starts with a greedy solution, stopping at the edge with index $\eta_1[i]$ close to the local optimum (first phase). Then, considering even the edge with the index $\eta_2[i]$, it elaborates different possible complements for this pallet (second phase) and selects the best of these complements (third phase).

Algorithm 8 $SolveNode(Shims, k, G)$

1: Let $G(V_k, E_k^Q)$
2: Sort M by $|p_i.d|$ in non-descending order
3: $E_k^N \leftarrow \varnothing$
4: Let $E[1..m][1..n_k]$, where $E[i]$ is an array with n_k edges (p_i, t_j^k) sorted by θ_{ij}^k in non-ascending order, $1 \leq i \leq m$
5: $gap \leftarrow 2000/n_k * m$
6: **for** $i \leftarrow 1$ to m
7: $\quad pv[i] \leftarrow 0$
8: $\quad \eta_1[i] \leftarrow 0$
9: $\quad$ **while** $(p_i.v - pv[i] > gap * p_i.v)$ **and** $(\eta_1[i] < n_k)$
10: $\quad\quad \eta_1[i] \leftarrow \eta_1[i] + 1$
11: $\quad\quad e_{ij}^k \leftarrow E[i][\eta_1[i]]$
12: $\quad\quad$ **if** (e_{ij}^k is feasible) **and** $(t_j^k.v \leq p_i.v - pv[i])$
13: $\quad\quad\quad E_k^N \leftarrow E_k^N \cup \{e_{ij}^k\}$
14: $\quad\quad\quad pv[i] \leftarrow pv[i] + t_j^k.v$
15: $\quad slack[i] \leftarrow p_i.v - pv[i]$
16: $\quad \eta_2[i] \leftarrow \eta_1[i]$
17: $\quad$ **repeat**
18: $\quad\quad \eta_2[i] \leftarrow \eta_2[i] + 1$
19: $\quad\quad e_{ij}^k \leftarrow E[i][\eta_2[i]]$
20: $\quad\quad pv[i] \leftarrow pv[i] + t_j^k.v$
21: $\quad$ **until** $(\eta_2[i] = n_k)$ **or** $(pv[i] \geq (1 + 3 * gap) * p_i.v))$
22: $\quad Set \leftarrow \varnothing$
23: $\quad q \leftarrow 1$
24: $\quad shim[q] \leftarrow \varnothing$
25: $\quad volume \leftarrow 0$
26: $\quad$ **for** $x \leftarrow \eta_1[i]$ to $\eta_2[i]$
27: $\quad\quad e_{ij}^k \leftarrow E[i][x]$
28: $\quad\quad$ **if** $(e_{ij}^k \notin E_k^N \cup shim[q])$ **and** (e_{ij}^k is feasible)
29: $\quad\quad\quad$ **if** $t_j^k.v + volume \leq slack[i]$
30: $\quad\quad\quad\quad shim[q] \leftarrow shim[q] \cup \{e_{ij}^k\}$
31: $\quad\quad\quad\quad volume \leftarrow volume + t_j^k.v$
32: $\quad\quad\quad$ **else**
33: $\quad\quad\quad\quad Set \leftarrow Set \cup shim[q]$
34: $\quad\quad\quad\quad q \leftarrow q + 1$
35: $\quad\quad\quad\quad shim[q] \leftarrow \varnothing$
36: $\quad\quad\quad\quad volume \leftarrow 0$
37: $\quad\quad\quad\quad$ **if** $t_j^k.v \leq slack[i]$
38: $\quad\quad\quad\quad\quad shim[q] \leftarrow shim[q] \cup \{e_{ij}^k\}$
39: $\quad\quad\quad\quad\quad volume \leftarrow volume + t_j^k.v$
40: $\quad sh_w \leftarrow S$, where $S \in Set$ and $\sum_{e_{ab}^k \in S} t_b^k.w$ is maximum
41: $\quad sh_v \leftarrow S$, where $S \in Set$ and $\sum_{e_{ab}^k \in S} t_b^k.v$ is maximum
42: $\quad sh_{best} \leftarrow S$, where $S \in \{sh_w, sh_v\}$ and $\sum_{e_{ab}^k \in x} t_b^k.s$ is maximum
43: $\quad E_k^N \leftarrow E_k^N \cup sh_{best}$
44: **return** $G(V_k, E_k^N \cup E_k^Q)$

In the first phase (lines 7-21), a greedy and partial solution for p_i is constructed by edges inclusion, until a certain slack or clearance is reached. This slack was empirically defined as $gap * p_i.v$ (line 9), where $gap = 2000/n_k * m$ (line 5).

In the second phase (lines 22-39), a set of shims named Set is created, where each shim is formed by a group

of edges in the range $[\eta_1[i], \eta_2[i]]$, whose total volume is limited by $slack[i]$. In this phase, the heuristic that provided the best results, both in terms of time and quality, is based on *Next-Fit*, which is an approximation algorithm for the *Bin Packing Problem* **?**. Basically, shims are created by accumulating the following edges, taking $slack[i]$ as a limit.

In the third phase (lines 40-43), the best shim in Set is chosen. Initially, two shims are found: sh_w with larger weight and sh_v with larger volume. Among the two, the shim with the highest score will be chosen and its edges are inserted into E_k^N.

6. Implementation and results

This section is composed of two parts: the generation of test instances and the results obtained in our implementations.

6.1. Instances generation

As we are dealing with a new problem, which until now had not been modelled in the literature, we have to create our own 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. On the other hand, the average density of these items is approximately $246kg/m^3$.

Table 6: Items weight distribution (kg) in Brazil

item	*P*	*low*	*high*
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

ItemsGeneration, which generates N, is described in Algorithm 9. The parameter *scenario* defines L and M (line 1), and the parameter *volume* sets a limit on the total volume of items at each node (line 2). To avoid simply loading all items, we use $volume > 1$.

Algorithm 9 *ItemsGeneration*$(scenario, volume)$

1: Let L and M be according to $scenario$
2: $limit \leftarrow volume \times \sum_{i=1}^{m} p_i.v$
3: **for** $k \leftarrow 0$ to K
4: $\quad N_k \leftarrow \varnothing$
5: $\quad j \leftarrow 0$
6: $\quad vol \leftarrow 0$
7: $\quad$ **while** $vol < limit$
8: $\quad\quad j \leftarrow j + 1$
9: $\quad\quad$ **repeat**
10: $\quad\quad\quad t_j^k.to \leftarrow RandomInt(0, K)$
11: $\quad\quad$ **until** $t_j^k.to \neq k$
12: $\quad\quad x = Roulette(item)$ biased through P
13: $\quad\quad t_j^k.w \leftarrow RandomReal(low(x), high(x))$
14: $\quad\quad t_j^k.s \leftarrow \lfloor 100 \times (1 - \log_{10}(RandomInt(1, 9)))\rceil$
15: $\quad\quad t_j^k.v \leftarrow t_j^k.w/RandomReal(148, 344)$
16: $\quad\quad vol \leftarrow vol + t_j^k.v$
17: $\quad\quad N_k \leftarrow N_k \cup \{t_j^k\}$
18: $\quad n_k \leftarrow j$
19: $N \leftarrow \bigcup_{0 \leq k \leq K} N_k$
20: **return** N

For each generated t_j^k item, its destination is uniformly random selected (line 10), its weight has a distribution according to Table 6 (lines 12-13), 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 15).

6.2. Results obtained

In the tests performed, we used a 64-bit, 16GB, 3.6GHz, eight-core processor with *Linux Ubuntu 22.04.1 LTS 64-bit* as the operational system and *Python 3.10.4* as the programming language. We also used the well-known solver *Gurobi* (www.gurobi.com), version 9.5.2.

Table 7: xxx

Node 0: GRU solution			
Pallet	$From : To$	Kg	m^3
1	GRU:GIG	2504	10.7
2	GRU:GIG	2405	10.4
3	GRU:GIG	3710	15.2
4	GRU:SSA	3386	14.3
5	GRU:SSA	3835	15.6
6	GRU:SSA	3124	12.6
7	GRU:SSA	3518	13.8
Score: 8911.0	Weight: 0.86	Volume: 1.10	Torque: 2.24
Node 1: GIG solution			
Pallet	$From : To$	Kg	m^3
1	GIG:SSA	2585	12.2
2	GIG:SSA	3603	17.0
3	GIG:SSA	2881	12.8
4	GIG:SSA	3039	13.7
5	GIG:GRU	2075	8.9
6	GIG:GRU	3478	14.5
7	GIG:GRU	1632	7.4
Score: 3546.0	Weight: 0.74	Volume: 1.03	Torque: 3.13
Node 2: SSA solution			
Pallet	$From : To$	Kg	m^3
1	SSA:GRU	3027	13.2
2	SSA:GRU	1287	6.2
3	SSA:GRU	3044	15.0
4	SSA:GRU	1939	9.1
5	SSA:GRU	2010	8.9
6	SSA:GRU	2686	11.8
7	SSA:GRU	1410	5.8
Score: 3632.0	Weight: 0.59	Volume: 0.83	Torque: 2.50
Tour 0	S/C: 4.291	elapsed: 0.05s	method: Greedy

7. Conclusions

xxxxx

Acknowledgments

This research was partially supported by *São Paulo Research Foundation* (FAPESP, grant 2016/01860-1).

References