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 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 fast take-off, and mission accomplishment, especially in times of crisis, disaster relief missions, short deadlines for contracting customers, or any external pressure for immediate take-off. In addition, there are no commercially available systems that can assist the load and trip planners with the pallet building and aircraft loading and balancing demands for transport at each hub. This enables other risks, such as improper delivery, excessive fuel burn, and longer than necessary turn-around time. We defined and solved the problem of planning the loading and routing of a single 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 NP-hard problem, named *Air Cargo Load Planning with Routing, Pickup, and Delivery Problem (ACLP+RPDP)*, is mathematically modeled using standardized pallets in fixed positions, obeying center of gravity constraints, delivering each item to its destination, and minimizing fuel consumption costs. We performed multiple experiments with a commercial solver and four well-known meta-heuristics on data based on the transport history of the *Brazilian Air Force*. We also designed a new heuristic that quickly finds good solutions for a wide range of problem sizes: an essential contribution, which solved all test scenarios within an acceptable operational run time.

Keywords: 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, Brandt and Nickel [30, 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 modeling: 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 *Traveling 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: heuristic search methods (**Heu**), integer programming (**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

Work	APP	WBP	PDP	TSP	2D	3D	Heu	Int	Lin
Larsen and Mikkelsen [1]	.	★	.	.	.	.	★	.	.
Brosh [2]	.	★	.	.	.	.	.	.	★
Ng [6]	.	★	.	.	.	.	.	★	.
Heidelberg *et al.* [11]	.	★	.	.	★	.	★	.	.
Mongeau and Bes [13]	★	★	.	.	.	.	.	★	.
Fok and Chun [14]	.	★	.	.	.	.	.	★	.
Chan *et al.* [16]	★	.	.	.	.	★	★	.	.
Kaluzny and Shaw [17]	.	★	.	.	★	.	.	★	.
Verstichel *et al.* [19]	.	★	.	.	.	.	.	★	.
Mesquita and Cunha [18]	.	.	★	.	.	.	★	.	.
Limbourg *et al.* [20]	.	★	.	.	.	.	.	★	.
Roesener and Hall [21]	★	★	.	.	.	★	.	★	.
Vancroonenburg *et al.* [22]	★	★	.	.	.	.	.	★	.
Lurkin and Schyns [23]	.	★	★	.	.	.	.	★	.
Roesener and Barnes [24]	.	★	.	.	.	.	★	.	.
Paquay *et al* [25, 26]	★	★	.	.	.	★	★	★	.
Chenguang *el al.* [28]	.	★	.	.	★	.	★	.	.
Wong and Ling [31]	★	★	.	.	.	.	.	★	.
Wong *et al.* [33]	★	★	.	.	.	.	.	★	.
Zhao *et al.* [34]	.	★	.	.	.	.	.	★	.
This article	★	★	★	★	.	.	★	★	.

As can be seen, so far Lurkin and Schyns [23] 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 predefined tour 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 modeled through a container sequencing problem. Referring directly to this work, Brandt and Nickel [30, 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, the *Brazilian Air Force* missions have always considered 3 to 5 nodes per flight plan.

It is important to emphasize that this data is not an imposed limitation, but a historical fact that we will explore in our solution method. Throughout this work we approach routes with up to 7 nodes, as can be seen in Table 2 and in Figure 1. Although there are many other airports of interest to *Brazilian Air Force*, these 7 nodes were chosen due to their high demand and short transport times. The other Brazilian airports tend to have a lower demand, which is usually met, in a less expensive way, by cabotage, rail or road transport.

Table 2: Brazilian airports distances (km)

Node	l_0	l_1	l_2	l_3	l_4	l_5	l_6
IATA*	GRU	GIG	SSA	CNF	CWB	BSB	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

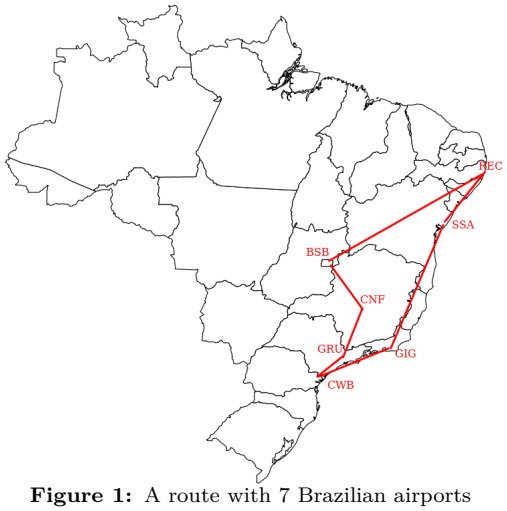

Figure 1: A route with 7 Brazilian airports

Cargo handling at modern airports usually involves powerful equipment such as dollies, motorized conveyors, elevator transfer vehicles, caster decking, scissor lifts, electrical controls, hydraulically adjustable heights, forklift trucks, and plenty of room in the airport cargo terminal (see Figure 2). These facilities, together with the use of standardized pallets with predefined positions on the aircraft, allow loading and unloading at each node to be carried out in about 30 minutes.

Figure 2: Cargo handling equipment
Source: NARA & DVIDS Public Domain Archive

However, as reported by Fok and Chun, *load planning (...) is usually done in roughly 2 hours before departure, when all the details of the cargo are present.* [14]. We may infer that this is because there is no commercial software to solve this problem. In addition, there is still the necessary time to plan the route, in order to minimize fuel consumption and ensure the transport of priority items. In these circumstances, it is of great importance to find a method that speeds up the load planning and the definition of the flight itinerary.

Our work proposes a method that prioritizes the transport of the most relevant items at each node and the fuel economy along the tour. We developed a heuristic that can be run on a simple handheld computer (such as a laptop or a tablet) that previously defines a flight plan, and provides a quick solution for cargo handling. Consequently, this method reduces 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 (regarding their positions), and a pick up and delivery plan to 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 organized 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, we describe 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.

Larsen and Mikkelsen [1] developed an interactive procedure for loading 14 types of Boeing 747 into a two-leg transportation 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 minimizes positioning changes in the intermediate node, optimizing the load balancing in the aircraft.

Brosh [2] 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.

Ng [6] 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.

Heidelberg *et al.* [11] developed a heuristic for 2D packing in air loading, comparing it with methods for solving the *Bin Packing Problem*. These authors conclude that the classical algorithms are inadequate in this context, because they ignore the aircraft balancing constraints.

Mongeau and Bes [13] presented a method based on linear integer programming to solve the problem of choosing and positioning containers in the *Airbus 340-300*. Safety and stability constraints were considered, with the objective of minimizing fuel consumption.

Fok and Chun [14] 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 in roughly 2 hours before departure, when all cargo details are in place.

Chan *et al.* [16] carried out a case study with heterogeneous pallets. To minimize 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.

Kaluzny and Shaw [17] developed a mixed integer linear programming model to arrange a set of items in a military context that optimizes the load balance. The objective function can be chosen to minimize the deviation of the center of gravity (CG) from the target position or to maximize the function of the items loaded. This approach does not palletize items, which are arranged in the cargo bay by a 2D packing procedure.

Verstichel *et al.* [19] solved WBP by selecting the most profitable subset of containers to be loaded into an aircraft by the use of mixed-integer programming. Experimental results on real-life data showed significant improvements compared to those obtained manually by an experienced planner.

Mesquita and Cunha [18] 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.

Limbourg *et al.* [20] developed a mixed-integer program for optimally rearranging a set of pallets into a compartmentalized cargo aircraft, specifically the *Boeing 747*.

Roesener and Hall [21] solved APP and WBP as an integer programming problem, where the items are selected for shipment based on a utility score, then assigned to pallets, which will be loaded into an aircraft in a specific pallet position. The pallets are then packed in a manner to optimize both the pallet and aircraft characteristics, such as item utility, pallet occupancy, and aircraft center of balance. This method groups items by destinations to be served by multiple aircraft.

Vancroonenburg *et al* [22] 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, these authors solved real problems in less than an hour.

As already mentioned, Lurkin and Schyns [23] was the first work that simultaneously modeled 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.

Roesener and Barnes [24] 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 in those aircraft.

Paquay *et al.* [25] presented a mathematical modeling to optimize the loading of heterogeneous 3D boxes on pallets with a truncated parallelepipeds format. Its objective is to maximize the volume used in containers, considering load balancing constraints, the presence of fragile items and the possibility of rotating these boxes. Paquay *et al.* [26] developed some heuristics to solve this problem.

Chenguang *et al.* [28] modeled the air transport problem as a 2D packing problem, and presented a heuristic for its optimization in several aircraft, considering load balancing to minimize fuel consumption.

Wong and Ling [31]A developed a mathematical model and a tool based on mixed integer programming for optimizing cargo in aircraft with different pallet configurations. Balance constraints and the presence of dangerous items were considered. Wong *et al.* [33] integrated this tool to a digital simulation model, with a visualization and validation system, based on sensors that alert about load deviations.

Zhao *et al.* [34] proposed a new modeling for WBP based on mixed integer programming. Instead of focusing on the 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 transportation plan for a single aircraft. 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 tour, the items to be allocated are characterized by weight, volume, scores, and previously known destinations. We leave the consideration of 2D or 3D items to a future work.
- 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 $4,500kg$ and $13.7m^3$, which may be limited by its position along the cargo bay. It 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 cargo items that fit on these pallets.
- Finally, as we are interested in minimizing fuel costs, we disregarded others costs not directly associated with aircraft flight, such as handling.

Throughout this text, we call *Packed Contents* (PC) a set of items of same destination stacked on a pallet and covered with a restraining net (see Figure 3). It is considered unique, having the same attributes of its components, whose values are the sum of individual scores, weights and volumes. To ensure accuracy in pickup and delivery operations, a PC must remain on board until its destination.

3.2. Aircraft parameters and load balancing

We consider real scenarios with an aircraft with payload of $75,000kg$, with enough fuel for a range of $4,400km$. The aircraft layout is presented in Figure 4, where the pallets positions are identified by p_i, $1 \leq i \leq 18$.

Figure 3: A PC on a 463L pallet inside a *Boeing C-17*
Source: From Wikimedia Commons, the free media repository

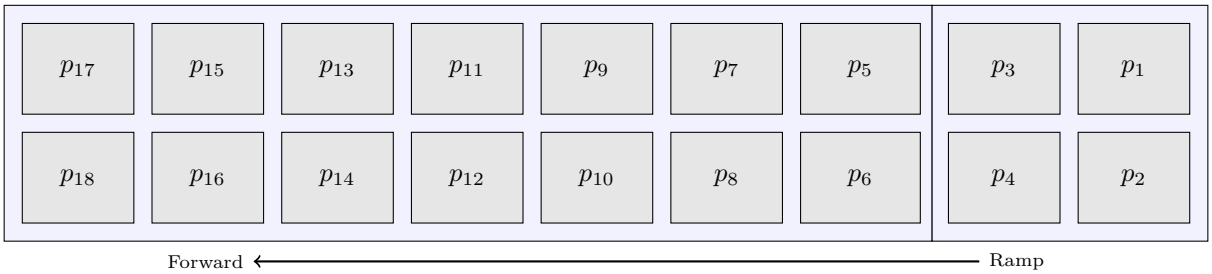

Figure 4: Aircraft layout

The torque applied to the aircraft must keep its CG in the operational range, which corresponds to a fixed percentage of the *Mean Aerodynamic Chord* [1] which is considered $1.17m$ for the aircraft of this work (see Figure 5).

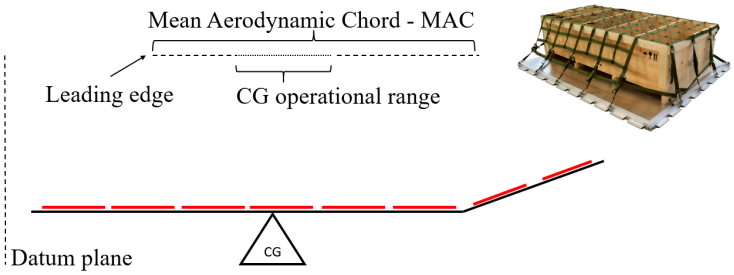

Figure 5: Aircraft longitudinal cut, where red lines are pallets positions

Table 3 shows the parameters of this aircraft. p_i are pallets, $1 \leq i \leq 18$, with weight W_i and V_i volume limits, respectively. D_i^{long} and D_i^{lat} are, respectively, the longitudinal and latitudinal distances of each pallet centroids to the CG of aircraft along both axes. These distances will be used in the calculation of the torque referring to the items allocated on each pallet. In this aircraft, as the ramp have an inclination of 25ř, we made the necessary corrections in D_i^{long}, W_i and V_i of the corresponding pallets.

Table 3: Aircraft parameters

	Payload: 75,000kg			$limit_{long}^{CG}$: $1.170m$			$limit_{lat}^{CG}$: $0.19m$		
$\boldsymbol{p_i}$	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
$\boldsymbol{D_i^{long}}$ (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
$\boldsymbol{D_i^{lat}}$ (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)	4,500	4,500	4,500	4,500	4,500	4,500	4,500	3,000	3,000
$\boldsymbol{V_i}$ (m^3)	14.8	14.8	14.8	14.8	14.8	14.8	14.8	10.0	7.0
Fuel cost	c_d = US$ 4.90/km								
Fuel consumption rate	$c_g = 5\%$								
Maximum weight	$W_{max} = \sum_i W_i = 75,000kg$								

This aircraft spends c_d dollars per kilometer flown, and can carry up to W_{max} of cargo distributed on the pallets. The *fuel consumption rate* c_g is the percentage limit of cost increase due to the CG deviation on the longitudinal axis of the aircraft. c_g depends on the characteristics of the aircraft, and is an arbitrated value for this work that should be better evaluated in future research. It is important to consider that the c_g tends to zero as the aircraft attitude tends to level. As the CG deviation varies from 0 to $limit_{long}^{CG}$, the *fuel cost* increase varies from 0 to c_g.

[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).

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 total load must be at a maximum longitudinal distance $limit_{long}^{CG}$ from the CG of the aircraft;
- the CG of the total load must be at a maximum lateral distance $limit_{lat}^{CG}$ from the CG of the aircraft;
- the pallets are distributed in two identical rows (with odd and even indices, respectively), and the centroid of p_i is at distance D_i^{lat} from the center-line of the aircraft.

4. The mathematical modeling

Given the assumptions and parameters described in the previous section, we are ready to present the mathematical modeling of ACLP+RPDP.

ACLP+RPDP has the objective function 1, and the calculus equations 2 to 9 subject to constraints 11 to 19, which will be described below.

4.1. Problem structure and input data

In the definitions below and in the tests of Section 6, we use the values assigned in Tables 2 and 3. To simplify the notation throughout this text, p_i will be called pallet i, and l_k will be called node k.

Let $L = \{0, 1, \ldots, K\}$ be the set with the $K + 1$ nodes, and let L_k be the set of remaining nodes when the aircraft is in the node k, $0 \leq k \leq K$. Therefore, $L_0 = L$.

Let $d(a, b)$ be the distance from node a to node b, where $0 \leq a, b \leq K$. By definition, $d(a, a) = 0, \forall a$.

Let $C = [c_{a,b}]$ be the cost matrix of flights, where $c_{a,b} = c_d * d(a, b), 0 \leq a, b \leq K$.

Let $M = \{1, 2, \ldots, m\}$ the set of m empty pallets assigned to specific positions within the aircraft. In Table 3, $m = 18$. 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}, and lateral distance to the center line of aircraft D_i^{lat}.

Let $N_k = \{1, \ldots, n_k\}$ be the set of n_k items t_j^k available for loading on the node k, $1 \leq j \leq n_k$, $0 \leq k \leq K$. For ease of notation, t_j^k will be called item j of the node k, with score s_j, weight w_j, volume v_j, and destination $to_j \in L_k$. Let $N = \bigcup_{0 \leq k \leq K} N_k$ be the set of items of all nodes along a tour.

Let $Q_k = \{1, \ldots, m_k\}$ be the set of $m_k \leq m$ packed contents a_q^k at the node k, $1 \leq q \leq m_k$, $0 \leq k \leq K$. For ease of notation, a_q^k will be called packed content q of the node k, with total weight w_q, total volume v_q, and destination $to_q \in L_k$. By definition, $m_0 = 0$, and therefore $Q_0 = \varnothing$. Packets contents that were destined to node k are unloaded when the aircraft arrives at this node, that is, they are not considered in Q_k.

4.2. The decision variables

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

$X_{ij}^k = 1$ if the item j of the node k is assigned to the pallet i, and 0 otherwise.

$Y_{iq}^k = 1$ if the packed content q of the node k is assigned to the pallet i, and 0 otherwise.

4.3. The allocation graph

Allocations of items or packed contents to the pallets in the node k can be seen as a bipartite graph $G_k(V_k, E_k)$, where:

- $V_k = M \cup N_k \cup Q_k$;
- $E_k = E_{N_k} \cup E_{Q_k}$;
- $(i, j) \in E_{N_k}$ if $X_{ij}^k = 1$, where i is a pallet and j is a item of the node k;
- $(i, q) \in E_{Q_k}$ if $Y_{iq}^k = 1$, where i is a pallet and q is a packed content of the node k.

4.4. The objective function

Let $S_K = \{s : \{1, \ldots, K\} \rightarrow \{1, \ldots, K\}\}$ be the set of $K!$ permutations π, which correspond to all possible tours (or itineraries) 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 $\{0, \pi_1, \ldots, \pi_K, 0\}$, $K > 1$. For ease of notation, we can define $\pi_0 = \pi_{K+1} = 0$.

We consider K > 1 because the case K = 1 has a single and trivial solution for the tour.

The objective of ACLP+RPDP is to find the permutation $\pi \in S_K$, with the corresponding allocation of items on the pallets at each node, that maximizes the function $f_\pi = \tilde{s}_\pi / \tilde{c}_\pi$ 1, where $\tilde{s}_\pi$ is the total score of transported items, and $\tilde{c}_\pi$ is the total cost of fuel consumed.

$$\max_{\pi \in S_K} f_\pi = \tilde{s}_\pi / \tilde{c}_\pi \tag{1}$$

4.5. The calculus equations

$\tilde{s}_\pi$ 2 is the sum of the scores of the items loaded on the aircraft throughout the tour π.

$$\tilde{s}_\pi = \sum_{k=0}^{K} \sum_{i=1}^{m} \sum_{j=1}^{n_{\pi_k}} X_{ij}^{\pi_k} \times s_j \tag{2}$$

τ_{π_k} 3 is the longitudinal torque applied by the loaded pallets, in the node π_k, in proportion relative to the highest torque supported by the aircraft.

$$\tau_{\pi_k} = \sum_{i=1}^{m} \left[D_i^{long} \times \left(\sum_{j=1}^{n_{\pi_k}} X_{ij}^{\pi_k} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq}^{\pi_k} \times w_q \right) \right] \Big/ W_{max} \times limit_{long}^{CG};\ k \in \{0, \ldots, K\} \tag{3}$$

In this way, we can calculate the cost $\tilde{c}_\pi$ 4, referring to the total fuel consumed in the tour π due to the distances traveled and the CG longitudinal deviations in each pallet.

$$\tilde{c}_\pi = \sum_{k=0}^{K} \left[c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau_{\pi_k}|) \right] \tag{4}$$

The sets of nodes not yet visited are defined in 5 and 6.

$$L_0 = L \tag{5}$$

$$L_{\pi_k} = L_{\pi_{k-1}} - \{\pi_k\};\ k \in \{1, \ldots, K\} \tag{6}$$

At the beginning of the tour π, there are no packed contents 7.

$$m_0 = 0 \tag{7}$$

$\epsilon_{\pi_k}^t$ 8 and $\epsilon_{\pi_k}^a$ 9 are the lateral torque applied by the loaded pallets, in the node π_k, in proportion relative to the highest torque supported by the aircraft. For ease of notation, $\epsilon_{\pi_k}^t$ corresponds to the items, and $\epsilon_{\pi_k}^a$ to the packed contents.

$$\epsilon_{\pi_k}^t = \sum_{i=1}^{m} \left[D_i^{lat} \times \sum_{j=1}^{n_{\pi_k}} \left(X_{ij}^{\pi_k} \times w_j \times (i\%2) - X_{ij}^{\pi_k} \times w_j \times (i+1)\%2 \right) \right] \Big/ W_{max} \times limit_{lat}^{CG};\ k \in \{0, \ldots, K\} \tag{8}$$

$$\epsilon_{\pi_k}^a = \sum_{i=1}^{m} \left[D_i^{lat} \times \sum_{q=1}^{m_{\pi_k}} \left(Y_{iq}^{\pi_k} \times w_q \times (i\%2) - Y_{iq}^{\pi_k} \times w_q \times (i+1)\%2 \right) \right] \Big/ W_{max} \times limit_{lat}^{CG};\ k \in \{0, \ldots, K\} \tag{9}$$

4.6. The constraints

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

- The longitudinal 10 and the lateral 11 torques must be within the limits of the aircraft.
- The items allocated to each pallet cannot exceed its weight 12 and volume 13 limits.
- At most, each item is associated with a single pallet 14;
- Packet contents that have not yet reached their destination must remain on board 15.
- Items allocated on the same pallet must have the same destinations. In this case, we need two constraints: 16 and 17. If $X_{ia}^{\pi_k} = X_{ib}^{\pi_k} = 1$, both constraints require that $to_a = to_b$; otherwise these constraints have no effect.
- If there is a packed content on the pallet, it must also have the same destination as the other items. Similarly, we use two constraints: 18 and 19.

$$|\tau_{\pi_k}| \leq 1;\ k \in \{0, \ldots, K\} \tag{10}$$

$$|\epsilon_{\pi_k}^t + \epsilon_{\pi_k}^a| \leq 1;\ k \in \{0, \ldots, K\} \tag{11}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij}^{\pi_k} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq}^{\pi_k} \times w_q \leq W_i;\ i \in \{1, \ldots, m\};\ k \in \{0, \ldots, K\} \tag{12}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij}^{\pi_k} \times v_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq}^{\pi_k} \times v_q \leq\ V_i;\ i \in \{1, \ldots, m\};\ k \in \{0, \ldots, K\} \tag{13}$$

$$\sum_{i=1}^{m} X_{ij}^{\pi_k} \leq 1;\ j \in \{1, \ldots, n_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{14}$$

$$\sum_{i=1}^{m} Y_{iq}^{\pi_k} = 1;\ to_q \in L_{\pi_k};\ q \in \{1, \ldots, m_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{15}$$

$$to_a - to_b \geq -K \times (1 - X_{ia}^{\pi_k} \times X_{ib}^{\pi_k});\ i \in \{1, \ldots, m\};\ a, b \in \{1, \ldots, n_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{16}$$

$$to_a - to_b \leq K \times (1 - X_{ia}^{\pi_k} \times X_{ib}^{\pi_k});\ i \in \{1, \ldots, m\};\ a, b \in \{1, \ldots, n_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{17}$$

$$to_j - to_q \geq -K \times (1 - X_{ij}^{\pi_k} \times Y_{iq}^{\pi_k});\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\};\ q \in \{1, \ldots, m_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{18}$$

$$to_j - to_q \leq K \times (1 - X_{ij}^{\pi_k} \times Y_{iq}^{\pi_k});\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\};\ q \in \{1, \ldots, m_{\pi_k}\};\ k \in \{0, \ldots, K\} \tag{19}$$

Constraints 16 to 19 are not within standard *Mixed-Integer Programming* (MIP), and therefore need to be handled in a different way. In the next section, we will explain the strategy adopted to solve ACLP+RPDP.

5. Solution strategy

Once the assumptions of this work and the mathematical modeling of the problem are presented, it is possible to see that ACLP+RPDP is *NP-hard*. In a similar way to Lurkin and Schyns [23, 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 node 0, and no items in node 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*.

Throughout our research, we have thoughtfully described the ACLP+RPDP model in standard MIP format and found that no MIP solver can handle its practical cases in a feasible time. Thus, as ACLP+RPDP is highly complex, involving four intractable subproblems (APP, WBP, PDP and TSP), our strategy will be to focus only on real cases and to develop quick node-by-node solutions, not necessarily optimal, that allow to build a complete tour.

The ACLP+RPDP solution defines a tour and a corresponding loading and unloading plan at each node. In this way, the loading and unloading time will be limited only by the use of the equipment at each node. Therefore, there will only be waiting time at the base node, corresponding to the run time of ACLP+RPDP algorithm. For this reason, our method will also be parameterized by the maximum time that can be waited until the start of the load in the base node.

On the other hand, we know that transport aircraft generally have a few dozen pallets, the flight plan has less than 6 nodes, and each node has hundreds of items to be shipped. Under these circumstances, we can adopt two more strategies:

- Considering the number of destinations less than the number of pallets ($K < m$), we can preset the destinations of the pallets in each 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.
- If the number of nodes is small and we have quick node-by-node solutions, we have the possibility to check the solution corresponding to the tour with the shortest total distance, and even test all possible tours, selecting the one that provides the best value for the objective function.

Our complete strategy is summarized in Algorithm 1.

Algorithm 1 ACLP+RPDP solving

1: **procedure** $ACLP + RPDP(scenario, surplus, time)$
2: Let aircraft and M be according to Table 3 ▷ Aircraft parameters
3: Let K, L and C be according to *scenario* and Table 2 ▷ Nodes, distances and costs
4: Let π_{TSP} be the tour with the shortest distance traveled
5: $N \leftarrow ItemsGeneration(scenario, surplus)$ ▷ Items available for shipment
6: **for** each *method* ▷ *method* is a MIP solver or a heuristic
7: $f_{\pi_{dir}} \leftarrow SolveTour(\pi_{dir}, L, M, C, N, method, time)$
8: $f_{\pi_{rev}} \leftarrow SolveTour(\pi_{rev}, L, M, C, N, method, time)$
9: $answer1[scenario, surplus, method] \leftarrow max(f_{\pi_{dir}}, f_{\pi_{rev}})$
10: **for** each $\pi \in S_K$ ▷ π is a permutation of nodes (a tour)
11: $f_\pi \leftarrow SolveTour(\pi, L, M, C, N, method, time/(K+1)!)$ ▷ f_π is a solution to the tour π
12: $answer2[scenario, surplus, method] \leftarrow \max f_\pi$ ▷ Best result among all permutations
13: **return** *answer*

In line 7, *SolveTour* solves the lowest-cost tour, and in line 8, its reverse.

The aircraft parameters are in Table 3 (line 2). In this algorithm, we use five values for *scenario*, according to Table 4, which defines K and the set L of nodes (line 3).

Table 4: Testing scenarios

Scenario	K	L	Shortest tours
1	2	$\{1, 2\}$	0 1 2 0
2	3	$\{1, 2, 3\}$	0 1 2 3 0
3	4	$\{1, 2, 3, 4\}$	0 4 1 2 3 0
4	5	$\{1, 2, 3, 4, 5\}$	0 4 1 2 5 3 0
5	6	$\{1, 2, 3, 4, 5, 6\}$	0 4 1 2 6 5 3 0

surplus is a value in $\{1.2, 1.5, 2.0\}$, which 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 behavior of each *method*, according to *scenario* and the quantity of items available for shipment. It is passed to *ItemsGeneration* (line 5), responsible for creating the items to be shipped, which will be presented in the next section (Algorithm 7).

time is a run time limit, which will be distributed among the node-by-node solutions (lines **??** and 11). Each tour consists of $K + 1$ trips, and the total number of tours is $K!$.

method corresponds to a MIP solver or a heuristic that we will present in subsection 5.2, where *SolveTour* is the node-by-node solution.

The results corresponding to the shortest tour are stored in *answer*1 (line 9), and those obtained by testing all tours are stored in *answer*2 (line 12).

Next, we will present two subsections: in the first we explain how *SolveTour* is executed, presetting the destinations of the pallets. In the second we will present the heuristic developed for node-by-node solutions.

5.1. SolveTour algorithm

As we commented in the previous subsection, we will adopt the strategy of presetting the destinations of each pallet throughout the tour. This is feasible in practical cases, where $K < m$. For this, each pallet i also has a field T_i^k, $0 \leq k \leq K$, which stores its next destination after being loaded in node k. For this reason, $T_i^k \in L_k$, $1 \leq i \leq m$, $0 \leq k \leq K$.

SolveTour is described in Algorithm 2, where π is a permutation of the nodes (excluding the base) that defines the order of visits in this tour, *method* corresponds to a MIP solver or a heuristic for solving the node-by-node problems, and *time* is the run time limit that will be distributed among the $K + 1$ legs of the tour.

Algorithm 2 Procedure to solve tour π with *method* (MIP solver or heuristic)

1: **procedure** $SolveTour(\pi, L, M, C, N, method, time)$
2: $\quad \pi_0 \leftarrow 0$ ▷ Base 0 is the first node
3: $\quad \pi_{K+1} \leftarrow 0$ ▷ Base 0 is the last node
4: $\quad score \leftarrow 0$
5: $\quad cost \leftarrow 0$
6: $\quad$ **for** $k \leftarrow 0$ to K
7: $\quad\quad L_{\pi_k} \leftarrow L - \{\pi_0, \pi_1, \ldots, \pi_k\}$ ▷ π_k is the current node
8: $\quad\quad$ **for** $i \leftarrow 1$ to m
9: $\quad\quad\quad T_i^{\pi_k} \leftarrow -1$ ▷ Unset the pallet destinations
10: $\quad\quad$ **if** $k = 0$
11: $\quad\quad\quad$ Let $G_1(M \cup N_0, \varnothing)$
12: $\quad\quad$ **else**
13: $\quad\quad\quad E_{Q_{\pi_k}}, M \leftarrow UpdatePacked(M, Q_{\pi_k}, \pi_k)$
14: $\quad\quad\quad$ Let $G_1(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}})$
15: $\quad\quad M \leftarrow SetPalletsDestinations(M, \pi_k)$
16: $\quad\quad G_2 \leftarrow SolveNode(method, \pi_k, G_1, time/(K + 1))$
17: $\quad\quad s, \tau \leftarrow ScoreAndDeviation(\pi_k, G_2)$
18: $\quad\quad score \leftarrow score + s$
19: $\quad\quad cost \leftarrow cost + c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau|)$
20: $\quad$ **return** $score/cost$

As we mentioned in the previous section, all tours start and end at the base 0 (lines 2-3). After initializing the score and cost values (lines 4-5), there is a loop for the $K + 1$ flights (lines 6-19). Initially, the set L_{π_k} of remaining nodes is updated (line 7), and the pallet destinations are unset (line 9).

When the aircraft is at the base, the initial graph G_1 is empty because it has no packed content 11. Otherwise, *UpdatePacked* (line 13) returns the set of packed contents 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).

SetPalletsDestinations (line 15) presets the destination of each pallet based on the volume demands of the current node, without changing the pallets destination with packed contents.

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 G_2 are calculated (line 17) and accumulated (lines 18-19), allowing the final result of this tour (line 20).

UpdatePacked, described in Algorithm 3, finds the best packed-pallet allocation, in terms of CG deviation, for the packed contents that remain on board.

Algorithm 3 Procedure to update the PC that remain boarded on node π_k

1: **procedure** $UpdatePacked(M, Q_{\pi_k}, \pi_k)$
2: $\quad E_{Q_{\pi_k}} \leftarrow MinCGDeviation(E_{Q_{\pi_k}})$
3: $\quad$ **for** $i \leftarrow 1$ to m
4: $\quad\quad$ **for** $q \leftarrow 1$ to m_{π_k}
5: $\quad\quad\quad T_i^{\pi_k} \leftarrow -1$
6: $\quad\quad\quad$ **if** $(i, q) \in E_{Q_{\pi_k}}$
7: $\quad\quad\quad\quad T_i^{\pi_k} \leftarrow to_q$ ▷ Set the pallet destinations
8: $\quad$ **return** $E_{Q_{\pi_k}}, M$

MinCGDeviation (line 2) is run through a MIP solver with the equations 20, 21 and 22, to relocate the packed contents on the pallets, minimizing torque and ensuring that they all remain on board, one packed content on each pallet. As there are few variables, the MIP solver returns an allocation $E_{Q_{\pi_k}}$ in less than 30 milliseconds. Finally, the destination of each pallet with a packed content is updated (lines 3-7).

MinCGDeviation objective function:

$$\text{minimize} \left| \sum_{i=1}^{m} \sum_{q=1}^{m_{\pi_k}} Y_{iq}^k \times w_q \times D_i^{long} \right| \tag{20}$$

s.t.:

$$\sum_{i=1}^{m} Y_{iq}^k = 1;\ q \in \{1, \ldots, m_{\pi_k}\} \tag{21}$$

$$\sum_{q=1}^{m_{\pi_k}} Y_{iq}^k \leq 1;\ i \in \{1, \ldots, m\} \tag{22}$$

SetPalletsDestinations, which sets the pallets destination not yet defined, is described in Algorithm 4.

Algorithm 4 Procedure to set pallets destination based on the items to be embarked on node π_k

1: **procedure** $SetPalletsDestinations(M, \pi_k)$
2: **for** $x \in L_{\pi_k}$
3: $vol_x \leftarrow 0$
4: $max \leftarrow 0$
5: $total \leftarrow 0$
6: **for** $j \leftarrow 1$ to n_{π_k}
7: **if** $to_j \in L_{\pi_k}$
8: $vol_{to_j} \leftarrow vol_{to_j} + v_j$
9: $total \leftarrow total + v_j$
10: **if** $vol_{to_j} > vol_{max}$
11: $max \leftarrow to_j$ ▷ max is the destination with maximum volume demand
12: **for** $x \in L_{\pi_k}$
13: **if** $vol_x \neq 0$
14: $needed \leftarrow \max\{1, \lfloor (m - m_{\pi_k}) \times vol_x/total \rfloor\}$ ▷ Number of pallets to node x
15: $np \leftarrow 0$
16: **for** $i \leftarrow 1$ to m
17: **if** $np < needed$ **and** $T_i^{\pi_k} = -1$
18: $T_i^{\pi_k} \leftarrow x$
19: $np \leftarrow np + 1$
20: **for** $i \leftarrow 1$ to m
21: **if** $T_i^{\pi_k} \leftarrow -1$ ▷ Remaining pallets
22: $T_i^{\pi_k} \leftarrow max$
23: **return** M

vol stores the demand volume of items destined to the non-visited nodes (line 3). The destination of empty pallets is defined proportionally to the volume of items to be embarked (lines 12-19). The destination with the maximum volume defines any remaining pallets (lines 20-22).

ScoreAndDeviation is described in Algorithm 5, which evaluates the allocation graph generated by *SolveNode*, returning the corresponding score and CG deviation.

Algorithm 5 Procedure to calculate the score and the relative torque deviation of graph G in node π_k

1: **procedure** $ScoreAndDeviation(\pi_k, G)$
2: Let $G(V_{\pi_k}, E_{Q_{\pi_k}} \cup E_{N_{\pi_k}})$
3: $s \leftarrow 0$
4: **for** $i \leftarrow 1$ to m
5: $\tau_i \leftarrow 0$
6: **for** $i \leftarrow 1$ to m
7: **for** $j \leftarrow 1$ to n_{π_k}
8: **if** $X_{ij}^{\pi_k} = 1$
9: $s \leftarrow s + s_j$
10: $\tau_i \leftarrow \tau_i + w_j \times D_i^{long}$
11: **for** $q \leftarrow 1$ to m_{π_k}
12: **if** $Y_{iq}^{\pi_k} = 1$
13: $s \leftarrow s + s_q$
14: $\tau_i \leftarrow \tau_i + w_q \times D_i^{long}$
15: $\tau \leftarrow \sum_1^m \tau_i / (W_{max} \times limit_{long}^{CG})$
16: **return** $s,\ \tau$

This algorithm consists of a loop that goes through all the pallets (lines 6-14), accumulating the scores (lines 9 and 13) and the torques (lines 10 and 14) of the shipped items, allowing the final calculation of the CG deviation (line 15).

5.2. Node-by-node solutions

In this subsection we present two implementations of $SolveNode(method,\ \pi_k,\ G)$, where π_k is the current node and G is the allocation graph of the PC that remain on board at node π_k. These implementations correspond to the two possible values of the *method* parameter: MIP (*Gurobi*) or *Shims* (a heuristic).

5.2.1. MIP solver

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 PC 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 modeling for $SolveNode(MIP,\ \pi_k,\ G)$ becomes simpler, which finds an allocation of available items in node π_k using previously defined values of L_{π_k}, $to_i^{\pi_k}$, and $a_q^{\pi_k}$.

This simplified modeling is described in the equations 23 to 33, which maximizes the score of the n_{π_k} items to be shipped, maintaining load balancing.

The variables X_{ij} and Y_{iq} define the sets of edges $E_{N_{\pi_k}}$ and $E_{Q_{\pi_k}}$, respectively, which will be added to the graph G.

Objective function:

$$maximize\ \ \tilde{s}/\tilde{c} \tag{23}$$

Where:

$$\tilde{s} = \sum_{i=1}^{m}\sum_{j=1}^{n_{\pi_k}} X_{ij} \times s_j \tag{24}$$

$$W_{max} = \sum_{i=1}^{m} W_i \tag{25}$$

$$\tau^{\pi_k} = \sum_{i=1}^{m}\left[D_i^{long} \times (\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q)\right] \Big/ W_{max} \times limit_{long}^{CG} \tag{26}$$

In Equation 27, the first parcel denotes the fuel cost from the base until the last node, and the second the fuel cost in the last leg, from the last node back to the base. Note that c_g is the fuel consumption potential increase due to the loaded aircraft CG deviation in the node (τ_{π_k}).

$$\tilde{c} = \sum_{k=0}^{K}\left[c_{\pi_k,\pi_{k+1}} \times (1 + c_g \times |\tau_{\pi_k}|)\right] \tag{27}$$

s.t.:

It was observed in the experiments that the lateral torque was always in the order of magnitude -3. So we did not consider these constraints.

$$|\tau_{\pi_k}| \leq 1 \tag{28}$$

$$\sum_{i=1}^{m}(\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q) \leq W_{max} \tag{29}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q \leq W_i;\ i \in \{1,\ldots,m\} \tag{30}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times v_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times v_q \leq \ V_i;\ i \in \{1, \ldots, m\} \tag{31}$$

$$\sum_{i=1}^{m} X_{ij} \leq 1;\ j \in \{1, \ldots, n_{\pi_k}\} \tag{32}$$

$$X_{ij} = 0;\ to_j \notin L_{\pi_k};\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{33}$$

$$X_{ij} \leq X_{ij} \times (T_i^{\pi_k} - to_j + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{34}$$

$$X_{ij} \leq X_{ij} \times (to_j - T_i^{\pi_k} + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{35}$$

5.2.2. Shims heuristic

Our goal is to find a heuristic that offers a good-quality solution for the node-by-node problem. As we are testing all $K!$ tours, its run time is a crucial requirement. Taking this into account, we perform a series of implementations based on known meta-heuristics: *Ant Colony Optimization* (ACO) [7, 9], *Noising Method Optimization* (NMO) [8, 12, 32], *Tabu Search* (TS) [4] and *Greedy Randomized Adaptive Search Procedure* (GRASP) [5].

These meta-heuristics were implemented in their standard forms. We did not use any specialized features to allow for the exploitation of the problem structure.

We also considered several ideas from the literature [10, 15, 27, 29, 32], and we were careful to use the same data structures and procedures in all implementations. The decision for a simple and fast heuristic is due to the need to obtain, in less than an hour (because the *Brazilian Air Force*'s transportation teams usually performs cargo planning in more than 2 hours), a complete solution for ACLP+RPDP, allowing its use in an operational environment.

However, when we established the run time limit of 0.7 seconds per node, the heuristic that presented better solutions was none of the previous ones. In this subsection, we will present a new heuristic for the node-by-node problem, called *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 available space.

The selection of edges for $E_{N_{\pi_k}}$ uses the *edge attractiveness* $\theta_{ij}^{\pi_k}$ 36, which can be understood as the tendency to allocate the item j to pallet i. It is directly proportional to the score, and inversely proportional to the volume and the torque of each item.

$$\theta_{ij}^{\pi_k} = \frac{s_j}{v_j} \ \times \left(1 - \frac{w_j \times |D_i^{long}|}{w_{max} \times \ |D_{max}^{long}|}\right);\ i \in \{1, \ldots, m\},\ j \in \{1, \ldots, n_{\pi_k}\} \tag{36}$$

The denominator $w_{max} \times \ |D_{max}^{long}|$ represents the heaviest item put on the most distant pallet.

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

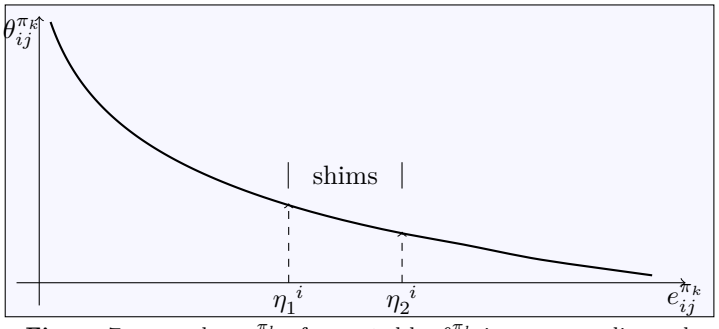

Figure 7: n_{π_k} edges $e_{ij}^{\pi_k}$ of p_i sorted by $\theta_{ij}^{\pi_k}$ in non-ascending order

Figure 7 represents the n_{π_k} possible edges $e_{ij}^{\pi_k}$ of i sorted by $\theta_{ij}^{\pi_k}$. *Shims* starts with a greedy solution, stopping at the edge with index η_1^i (first phase). Then, considering even the edge with the index η_2^i, it elaborates different

possible complements for this pallet (second phase) and selects the best of these complements (third phase).

In this heuristic, described in Algorithm 6, pallets i are considered in non-descending order of $|D_i^{long}|$ (line 4). For each pallet, the n_{π_k} possible edges $e_{ij}^{\pi_k}$ are considered in non-increasing order of $\theta_{ij}^{\pi_k}$ (line 6) in three phases: the greedy phase (until line 23), the composition phase of the shims (lines 24-30) and the selection phase of the best shim (lines 31-55).

In the beginning of the first phase (line 3), the set Q_{π_k} represents the PC whose destination is not π_k, the current node, thus remaining on board.

It is important to remember that Q_{π_k} and M were modified by the procedures $UpdatePacked(M, Q_{\pi_k}, \pi_k)$ and $SetPalletsDestinations(M, \pi_k)$, having their positions and destinations reassigned.

Until line 23, a greedy and partial solution for pallet i is constructed by edges inclusion, until a certain volume limit ($V_i \times limit$) is reached. The value of the *limit* (0.92) was empirically determined by the *iRace* tool developed by [35].

In the second phase (lines 24 to 30), a set of shims named *Set* is created, where each *Shim* is formed by a set 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 *First-Fit Decreasing*, which is an approximation algorithm for the *Bin Packing Problem* [3]. Basically, shims are created by accumulating the following edges, taking $slack_i$ as a limit.

In the third phase (lines 31-55), the best *Shim* in the *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 $E_{N_{\pi_k}}$.

Algorithm 6 *Shims* heuristic on node π_k

procedure $SolveNode(Shims,\ \pi_k,\ G,\ time)$
  $T_{begin} \leftarrow$ current system time
  $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}})$ ▷ a graph with pallets positions (M), PC (Q_{π_k}), and edges ($E_{Q_{\pi_k}}$)
  Sort M by $|D_i|$ in non-descending order
  $E_{N_{\pi_k}} \leftarrow \{\ \}$ ▷ an empty set of pallet-item edges
  Let E_{ij} be an array with $m \times n_{\pi_k}$ edges $(i,\ j)$ sorted by $\theta_{ij}^{\pi_k}$ in non-ascending order
  $\tau^{\pi_k} \leftarrow 0$ ▷ aircraft torque in node π_k
  $\tau_{max} \leftarrow W_{max} \times limit_{long}^{CG}$ ▷ maximum aircraft torque
  $volume \leftarrow \{0 \mid [1, \ldots, m]\}$
  $slack \leftarrow \{0 \mid [1, \ldots, m]\}$
  $limit \leftarrow 0.92$ ▷ 0.92 was obtained by the *iRace* package [35]
  for $i \in \{1, \ldots, m\}$
    for $q \in \{1, \ldots, m_{\pi_k}\}$
      if $(i, q) \in E_{Q_{\pi_k}}$
        $volume_i \leftarrow volume_i + v_q$
    for $e_{ij} \in E_{ij}$
      $\Delta\tau \leftarrow w_j \times D_i$
      $\tau_{new} \leftarrow |\tau^{\pi_k} + \Delta\tau|$
      if $(E_{N_{\pi_k}} \cup \{e_{ij}\}$ is feasible) **and** $(volume_i \leq V_i \times limit)$ **and** $(\tau_{new} \leq \tau_{max}\ )$
        $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup \{e_{ij}\}$
        $volume_i \leftarrow volume_i + v_j$
        ${\eta_1}^i \leftarrow {\eta_1}^i + 1$
        $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + \Delta\tau$
    ▷ end of the first phase
    $slack_i \leftarrow V_i - volume_i$
    ${\eta_2}^i \leftarrow {\eta_1}^i$
    repeat
      ${\eta_2}^i \leftarrow {\eta_2}^i + 1$
      $e_{ij} \leftarrow E_{i,{\eta_2}^i}$
      $volume_i \leftarrow volume_i + v_j$
    until $({\eta_2}^i = n_{\pi_k})$ **or** $(volume_i \geq (1 + 3 \times (1 - limit)) \times V_i))$ ▷ end of the second phase
    $volume \leftarrow 0$
    $volume \leftarrow 0,\ b \leftarrow 1\ shims_b \leftarrow \{\},\ Set \leftarrow Set \cup \{shims_b\}$
    $shims_b \leftarrow \{\}$
    $Set \leftarrow Set \cup \{shims_b\}$
    for $x \in \{{\eta_1}^i, \ldots, {\eta_2}^i\}$
      if current system time - $T_{begin} > time$
        break
      $NewShims \leftarrow$ **True**
      $e_{ij} \leftarrow {E_x}^i$
      for $shims \in Set$
        if $e_{ij} \notin (E_{N_{\pi_k}} \cup shims)$ **and** e_{ij} is feasible **and** $(v_j + volume) \leq slack_i$
          $shims \leftarrow shims \cup \{\ e_{ij}\ \}$
          $volume \leftarrow volume + v_j$
          $NewShims \leftarrow$ **False**
          break
      if $NewShims$
        $volume \leftarrow 0$
        $b \leftarrow b + 1$
        $shims_b \leftarrow \{\ \}$
        $shims_b \leftarrow shims_b \cup \{e_{ij}\}$
        $Set \leftarrow Set \cup \{\ shims_b\ \}$
    $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab} \in shims} w_b$ is maximum
    $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab} \in shims} v_b$ is maximum
    $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ab} \in x} s_b$ is maximum
    $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup sh_{best}$ ▷ end of the third phase
  return $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}} \cup E_{N_{\pi_k}})$

6. Implementation and results

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

6.1. Instances generation

As we are dealing with a new problem, which until now had not been modeled 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 5, where P represents the group probability. On the other hand, the average density of these items is approximately $246kg/m^3$.

Table 5: Items weight distribution (kg) in Brazil

Group	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

In the generation of test instances, we use three 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;
- $Roulette(set)$ 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.

Algorithm 7 Procedure to generate items

procedure $ItemsGeneration(scenario, surplus)$
  Let L and M be according to $scenario$
  $limit \leftarrow surplus \times \sum_{i=1}^{m} V_i$
  Let t_j^k be an item j in node k.
  for $k \in \{0, \ldots, K\}$
    $N_k \leftarrow \varnothing$
    $j \leftarrow 0$
    $vol \leftarrow 0$
    while $vol < limit$
      $j \leftarrow j + 1$
      repeat
        $to_j \leftarrow RandomInt(0, K)$
      until $to_j \neq k$
      $x = Roulette(item)$ biased through P ▷ From Table 5
      $w_j \leftarrow RandomReal(low(x), high(x))$
      $s_j \leftarrow \lfloor 100 \times (1 - \log_{10}(RandomInt(1, 9))) \rceil$
      $v_j \leftarrow w_j / RandomReal(148, 344)$
      $vol \leftarrow vol + v_j$
      $N_k \leftarrow N_k \cup \{t_j^k\}$
  $N \leftarrow \bigcup_{0 \leq k \leq K} N_k$
  return N

ItemsGeneration, which generates N, is described in Algorithm 7. The parameter *scenario* defines L and M (line 2), and the parameter *surplus* sets a limit on the total volume of items at each node (line 3). To avoid simply loading all items, we use $surplus > 1$ (1.2, 1.5, and 2.0). This represents more test instances for each scenario.

For each generated t_j^k item, its destination is randomly selected (line 12), its weight has a distribution according to Table 5 (lines 14-15), its score varies 100 (highest) and 5 (lowest) according to a logarithmic scale (line 16, 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).

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.

We ran Algorithm 1 in the 5 scenarios described in Table 4, considering 6 methods for node-by-node solution: Gurobi (see 5.2.1), ACO, NMO, TS, GRASP, and Shims (Algorithm 6). 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 Tables 6 ($surplus = 1.2$), 7 ($surplus = 1.5$) and 8 ($surplus = 2.0$). For each *method*, *scenario* and *surplus*, 7 different instances were generated. Therefore, 105 tests (5 scenarios $\times$ 3 volumes $\times$ 7 instances) were performed for each method.

The average values were presented for f_π and, for the run time, the worst result obtained. To facilitate the comparison between the methods, we added a last column in these tables, where two values are indicated:

- **Normalized**: value between 0 and 1, which corresponds to the ratio between the sum of 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 **Normalized**, the closer the method approached the best solutions found.
- **Speed-up**: 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 **Speed-up** is the fastest.
- **run time (s)** is the average time to solve a problem instance.

In each *scenario*, we indicate in bold the best value of f found. In each table, we also indicate in bold the best **Normalized** and **Speed-up** values.

As the MIP solver was not capable of solving all scenarios in polynomial time or reached the computer's RAM (16 GiB) too early, we decided to set the Gurobi *MIPgap* parameter to 1% to control the minimal quality of the returned solutions and abbreviate the solution time.

More details on the Gurobi *MIPgap* refer to [36].

Table 6: Solutions with $surplus = 1.2$ with the direct and reverse least cost tours.

method	**Scenarios**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
*Gurobi**	f	8.52	11.70	13.07	13.54	12.44	1.00
	run time (s)	17	17	16	17	26	1.0
Shims	f	7.83	11.34	12.77	13.20	12.22	0.97
	run time (s)	4	5	6	7	10	2.9
Average f and run time values of K! tours.							
*Gurobi**	f	8.51	12.25	13.32	14.61	x	1.00
	run time (s)	25	36	121	297	x	1.0
Shims	f	8.11	12.18	13.25	14.57	14.33	0.99
	run time (s)	4	8	33	95	572	3.4

**MIPgap* parameter set to 1%. The f values are $\pm 1\%$ distant from the optimal solution

97% of 16GiB was the Read-Only Memory (RAM) state before we aborted the Gurobi execution on solving scenario 5 in Table 6.

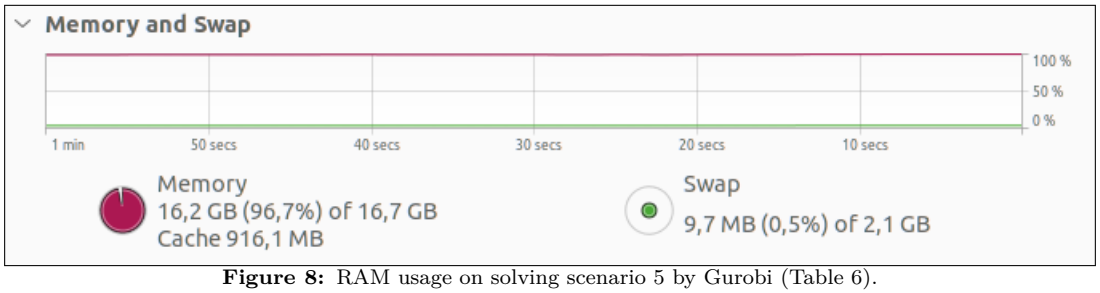

Figure 8: RAM usage on solving scenario 5 by Gurobi (Table 6).

Table 7: Solutions with *surplus* = 1.5 with the direct and reverse least cost tours.

method	**Scenarios**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
*Gurobi**	f	11.80	16.74	18.04	18.86	16.92	1.00
	run time (s)	38	30	27	29	81	1.00
Shims	f	10.89	16.65	17.80	18.69	16.45	0.98
	run time (s)	4	5	6	7	8	7.1
Average f and run time values of K! tours.							
*Gurobi**	f	11.81	17.01	18.25	20.59	18.36	1.00
	run time (s)	53	61	193	470	2241	1.0
Shims	f	11.33	16.97	17.56	20.43	17.94	0.98
	run time (s)	4	8	34	96	591	4.1

**MIPgap* parameter set to 1%. The f values are ±1% distant from the optimal solution

Table 8: Solutions with *surplus* = 2.0 with the direct and reverse least cost tours.

method	**Scenarios**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
*Gurobi**	f						
	run time (s)	38	30	27	29	81	
Shims	f						
	run time (s)	4	5	6	7	8	
Average f and run time values of K! tours.							
*Gurobi**	f						
	run time (s)	53	61	193	470	2241	
Shims	f						
	run time (s)	4	8	34	96	591	

**MIPgap* parameter set to 1%. The f values are ±1% distant from the optimal solution

It is important to note that the best tour found in each test was seldom the tour with the lowest cost (the minimal cost tour without considering the CG deviation), but any of the first 25% of the best tours. This indicates that our strategy of enumerating and solving all tours was assertive, although it is possible to devise a more proactive strategy to discard most of the higher-cost tours and improve performance. This must be one of the next steps in this research.

If a posterior application of this method requires 8, 9, or more nodes, it is possible to use this 25% as a threshold to discard the higher-cost tours and keep the run time within an operational acceptable limit for the client.

It is worth noting that the other heuristics always obtained **Normalized** values greater than or equal to 0.9, and were always faster than *Gurobi*.

It is not possible to state how far these results are from the optimal solutions, but we can report that all occupancy rates in terms of volume were above 97%, and in terms of weight above 79%, which denotes that the results achieved are good, considering that the CG deviation constraints keeps these values from going higher.

6.2.1. Proposed strategy assessment

Another important test is to compare the best tour score/cost ratio with the least expensive tour score/cost ratio. This is important to see how far our solution strategy is from the simplest form, just solving a TSP.

For this achievement, we ran *Shims* in 5 scenarios with 7 instances each, and 3 volume surpluses, collecting the averages of the best tour result and the least cost tour result.

ESTA TABELA DEVERÁ REFEITA DEPOIS QUE AS ANTERIORES ESTIVEREM PRONTAS.

Table 9: Shims best and least cost tour results

surplus	**Tour**	**1**	**2**	**3**	**4**	**5**	**Sum**	**Improvement**
1.2	Least cost	6.23	11.56	12.88	13.54	12.47	56.68	
	Best	7.96	13.04	14.30	15.56	15.28	66.14	16.7%
1.5	Least cost	8.44	16.61	18.18	19.68	17.64	80.55	
	Best	11.47	18.25	20.61	22.22	20.94	93.49	16.1%
2.0	Least cost	13.38	22.46	25.97	26.23	22.51	110.55	
	Best	17.93	25.59	28.61	31.28	29.09	132.50	19.9%
						Average improvement		17.6%

As the selection of the best tour, considering what was collected and delivered in each node, and considering the cost increase due to CG deviation, was 17.6% better than the option for a simple TSP solution (*Least cost* results), this confirms that our solution strategy was adequate.

7. Conclusions

In this work, we modeled and solved a real air transport problem, named *Air Cargo Load Planning with Routing, Pickup and Delivery Problem* (ACLP+RPDP). For the first time in the literature, a NP-hard problem that involves *simultaneously* pallet assembly, load balancing and route planning is addressed, where the cost-effectiveness of transport is maximized. We adopted some simplifications that are not critical, but that allowed the unprecedented solution of this problem considering more than two nodes.

We consider that, in practical cases, the number K of nodes, excluding the base, is small ($K \leq 6$), each of them with hundreds of items to be shipped. Thus, considering real aircraft models, we developed some node-by-node heuristics, which allow us to enumerate all $K!$ tours and choose the best. The complete process can be executed quickly on a simple handheld computer, offering good results and reducing stress for the transport planners. As validation, we performed tests in several scenarios with real data from the *Brazilian Air Force*.

In less than one hour, this development can establish a good distribution of load on pallets to be put in the cargo bay, enforcing the loaded aircraft balance, maximizing the total score and minimizing fuel consumption for the airlift. The output, which includes the tour plan and the pallet building and arrangement plan, is an essential part of airlift: it improves flight safety, makes ground operations more efficient, and makes sure that each item gets to its right destination.

Our main contributions were the mathematical modeling of this complex problem, which involves four NP-hard sub-problems, and a complete process that offers a fast and good-quality solution.

As this is ongoing research, some improvements are possible, which are the next steps of the development:

(1) This work addresses all possible tours, but we plan bounding schemes to allow for discarding most of the higher-cost tours. This is expected to reduce the running time, favoring the discovery of better search mechanisms.

(2) We are also working on computer parallelism to accelerate the algorithms.

(3) In the event that these gains in performance are achieved, we strongly believe that they will permit the application of a 3-dimensional packing procedure to maximize the occupancy rate among the selected items.

The simple heuristic *Shims* presented a solution with the best node-by-node results in a restricted time interval. Without it, it would not be possible to test all routes: after all, the greater the number of nodes, the shorter the run time limit for node-by-node solution. We also show that, as the shipment volume increases, the exact node-by-node solution becomes unfeasible, indicating the importance of a fast and good-quality heuristic.

Finally, by focusing on node-by-node solution, the method of this work is not exclusive to aircraft and airports: it can be adapted, for example, to ships and ports, or vehicles and warehouses, or freight cars and railways. In these cases, it would be necessary to make some changes in the modeling: for example, modify the load balancing constraints, and consider the available space in vehicles or freight cars instead of pallets.

Acknowledgments

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

References

[1] O. Larsen and G. Mikkelsen, An interactive system for the loading of cargo aircraft, *European Journal of Operational Research*, Vol. 4 (6), pp. 367-373, 1980.

[2] Israel Brosh, Optimal cargo allocation on board a plane: a sequential linear programming approach, *European Journal of Operational Research*, Vol. 8 (1), pp. 40-46, 1981.

[3] D.S. Johnson and M.R. Garey, A 71/60 theorem for bin packing, *Journal of Complexity*, Vol. 1 (1), pp. 65-106, 1985.

[4] Fred Glover, Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, Vol. 13, pp. 533-549, 1986.

[5] Thomas A. Feo and Mauricio G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters*, Vol. 8 (2), pp. 67-71, 1989.

[6] Kevin Y.K. Ng, A multicriteria optimization approach to aircraft loading, *Operations Research*, Vol. 40 (6), pp. 1200-1205, 1992.

[7] Marco Dorigo, *Optimization, Learning and Natural Algorithms*, PhD Thesis, Politecnico di Milano, 1992.

[8] I. Charon and O. Hudry, The noising method: a new method for combinatorial optimization, *Operations Research Letters*, Vol. 14 (3), pp. 133-137, 1993.

[9] M. Dorigo, V. Maniezzo and A. Colorni, The ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26 (1), pp. 29-41, 1996.

[10] S. Niar and A. Freville, A parallel tabu search algorithm for the 0-1 multidimensional knapsack problem, Proceedings of *11th International Parallel Processing Symposium*, 1997.

[11] K.R. Heidelberg, G.S. Parnell and J.E. Ames, Automated air load planning, *Naval Research Logistics*, Vol. 45 (8), pp. 751-768, 1998.

[12] I. Charon and O. Hudry, The noising methods: a generalization of some metaheuristics, *European Journal of Operational Research*, Vol. 135 (1), pp. 86-101, 2001.

[13] M. Mongeau and C. Bes, Optimization of aircraft container loading, *IEEE Transaction on Aerospace and Electronic Systems*, Vol. 39 (1), pp. 140-150, 2003.

[14] M.K.K. Fok and A.H.W. Chun, Optimizing air cargo load planning and analysis, Proceedings of *The International Conference on Computing, Communications and Control Technologies*, 2004.

[15] S. Fidanova, Ant colony optimization and multiple knapsack problem and model bias, *Numerical Analysis and Its Applications*, Springer Berlin Heidelberg, pp. 280-287, 2005.

[16] F.T.S. Chan, R. Bhagwat, N. Kumar. M.K. Tiwari and P. Lam, Development of a decision support system for air-cargo pallets loading problem: a case study, *Expert Systems with Applications*, Vol. 31 (3), pp. 472-485, 2006.

[17] B.L. Kaluzny and R.H.A.D. Shaw, Optimal aircraft load balancing, *International Transactions in Operational Research*, Vol. 16 (6), pp. 767-787, 2009.

[18] A.C.P. Mesquita and C.B. Cunha, 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*, Vol. 19, pp. 33-42, 2011.

[19] J. Verstichel, W. Vancroonenburg, W. Souffriau and G.V. Berghe, A mixed integer programming approach to the aircraft weight and balance problem, *Procedia Social and Behavioral Sciences*, Vol. 20, pp. 1051-1059, 2011.

[20] S. Limbourg, M. Schyns and G. Laporte, Automatic aircraft cargo load planning, *Journal of the Operational Research Society*, Vol. 63 (9), pp. 1271-1283, 2012.

[21] A.G. Roesener and S. Hall, A nonlinear integer programming formulation for the airlift loading problem with insufficient aircraft, *Journal of Nonlinear Analysis and Optimization: Theory and Applications*, Vol. 5 (1), pp. 125-141, 2014.

[22] W. Vancroonenburg, J. Verstichel, K. Tavernier and G.V. Berghe, Automatic air cargo selection and weight balancing: a mixed integer programming approach, *Transportation Research Part E: Logistics and Transportation Review*, Vol. 65, pp. 70-83, 2014.

[23] V. Lurkin and M. Schyns, The airline container loading problem with pickup and delivery, *European Journal of Operational Research*, Vol. 244 (3), pp. 955-965, 2015.

[24] A.G. Roesener and J.W. Barnes, An advanced tabu search approach to the dynamic airlift loading problem, *Logistics Research*, Vol. 9 (1), pp. 1-18, 2016.

[25] C. Paquay, M. Schyns and S. Limbourg, A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application, *International Transactions in Operational Research*, Vol. 23 (1-2), pp. 187-213, 2016.

[26] C. Paquay, S. Limbourg, M. Schyns and J.F. Oliveira, MIP-based constructive heuristics for the three-dimensional bin packing problem with transportation constraints, *International Journal of Production Research*, Vol. 56 (4), pp. 1581-1592, 2018.

[27] S. Laabadi, M. Naimi, H. El Amri and B. Achchab, The 0/1 multidimensional knapsack problem and its variants: a survey of practical models and heuristic approaches, *American Journal of Operations Research*, Vol. 8, pp. 395-439, 2018.

[28] Y. Chenguang, L. Hu and G. Yuan, Load planning of transport aircraft based on hybrid genetic algorithm, *MATEC Web of Conferences*, Vol. 179, pp. 1-6, 2018.

[29] M.T. Alonso, R. Alvarez-Valdes and F. Parreno, A GRASP algorithm for multi-container loading problems with practical constraints, *A Quarterly Journal of Operations Research*, Vol. 18, pp. 49-72, 2019.

[30] F. Brandt and S. Nickel, The air cargo load planning problem - a consolidated problem definition and literature review on related problems, *European Journal of Operational Research*, Vol. 275 (2), pp. 399-410, 2019.

[31] E.Y.C. Wong and K.K.T. Ling, A mixed integer programming approach to air cargo load planning with multiple aircraft configurations and dangerous goods, Proceedings of *7th International Conference on Frontiers of Industrial Engineering*, 2020.

[32] S. Zhan, L. Wang, Z. Zhang and Y. Zhong, Noising methods with hybrid greedy repair operator for 0-1 knapsack problem, *Memetic Computing*, Vol. 12, pp. 37-50, 2020.

[33] E.Y.C. Wong, D.Y. Mo and S. So, Closed-loop digital twin system for air cargo load planning operations, *International Journal of Computer Integrated Manufacturing*, Vol. 34 (7-8), pp. 801-813, 2021.

[34] X. Zhao, Y. Yuan, Y. Dong and R. Zhao, Optimization approach to the aircraft weight and balance problem with the centre of gravity envelope constraints, *IET Intelligent Transport Systems*, Vol. 15 (10), pp. 1269-1286, 2021.

[35] López-Ibáñez, Manuel and Dubois-Lacoste, Jérémie and Pérez Cáceres, Leslie and Birattari, Mauro and Stützle, Thomas. 2016. The irace package: Iterated racing for automatic algorithm configuration. Elsevier. pp. 43-58. Volume 3.

[36] What is the MIPGap? `https://support.gurobi.com/hc/en-us/articles/8265539575953-What-is-the-MIPGap-`, 2023.