Air cargo load and route planning in pickup and delivery operations

Abstract

In the aerial pickup and delivery of goods in a distribution network, transport aviation faces risks of cargo unbalancing due to the urgency required for loading, immediate take-off, and mission accomplishment. It is especially critical for disaster relief, short deadlines, or any pressure for immediate takeoff. In addition, there are no commercially available systems that can assist load and trip planners with pallet building and aircraft-balanced loading plans, with demands for transport at each hub. This enables other risks, such as improper delivery, excessive fuel burn, and a 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. 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 practical solutions to a wide range of problem sizes, a key contribution that resolved all real scenarios in much less time than is operationally acceptable.

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 [32, p. 401] defined the *Air Cargo Load Planning Problem* (ACLPP) as four sub-problems: the *Aircraft Configuration Problem* (ACP), the *Build-up Scheduling Problem* (BSP), the *Air Cargo Palletization Problem* (APP), and the *Weight and Balance Problem* (WBP).

Several aspects were considered in these authors' model: 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).

Lurkin and Schyns [24] show that this problem is NP-hard and therefore compare their results to current manually generated solutions vice optimal (or even near optimal) solutions. They have shown that the WBP is also NP-hard.

Karp [1] previously proved the 0-1 Knapsack problems to be NP-Hard, as the pallets may be seen as multiple knapsacks, the APP is also NP-hard.

Feng *et al.* (2015) [25] presented a comprehensive literature assessment of commercial air cargo operations. They stated that the airplane loading problem is one of the fundamental issues in combinatorial optimization and described it as an NP-hard problem.

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 1 and in Figure 1. Although there are many other airports of interest to *Brazilian Air Force*, these seven nodes were chosen due to their high demand and short transport times. The other Brazilian airports tend to have lower demand, which is usually met in a less expensive way by cabotage, rail, or road transport.

Table 1: 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 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. 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.

However, as reported by Fok and Chun, *load planning (...) is usually done roughly two hours before departure, when all the details of the cargo are present.* [15]. We can infer that this is because adequate commercial software does not exist. In addition, there is still the necessary time to plan the route 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 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 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 2, which 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**).

Table 2: Air cargo transport: literature, problems and features

Work	APP	WBP	PDP	TSP	2D	3D	Heu	Int	Lin
Larsen and Mikkelsen [2]	.	★	.	.	.	.	★	.	.
Brosh [3]	.	★	.	.	.	.	.	.	★
Ng [7]	.	★	.	.	.	.	.	★	.
Heidelberg *et al.* [12]	.	★	.	.	★	.	★	.	.
Mongeau and Bes [14]	★	★	.	.	.	.	.	★	.
Fok and Chun [15]	.	★	.	.	.	.	.	★	.
Chan *et al.* [17]	★	.	.	.	.	★	★	.	.
Kaluzny and Shaw [18]	.	★	.	.	★	.	.	★	.
Verstichel *et al.* [20]	.	★	.	.	.	.	.	★	.
Mesquita and Cunha [19]	.	.	★	.	.	.	★	.	.
Limbourg *et al.* [21]	.	★	.	.	.	.	.	★	.
Roesener and Hall [22]	★	★	.	.	.	★	.	★	.
Vancroonenburg *et al.* [23]	★	★	.	.	.	.	.	★	.
Lurkin and Schyns [24]	.	★	★	.	.	.	.	★	.
Roesener and Barnes [26]	.	★	.	.	.	.	★	.	.
Paquay *et al* [27, 28]	★	★	.	.	.	★	★	★	.
Chenguang *el al.* [30]	.	★	.	.	★	.	★	.	.
Wong and Ling [33]	★	★	.	.	.	.	.	★	.
Wong *et al.* [35]	★	★	.	.	.	.	.	★	.
Zhao *et al.* [36]	.	★	.	.	.	.	.	★	.
Zhao *et al.* [37]	★	★	.	.	.	.	.	★	.
This article	★	★	★	★	.	.	★	★	.

As can be seen, so far Lurkin and Schyns [24] 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 these authors: in relation to loading, APP was ignored; with regard to routing, it is assumed that a predefined tour plan is restricted to only 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 [32, 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.*

Larsen and Mikkelsen [2] 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 from 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 [3] 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 [7] 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 by solving WBP with pallets in fixed positions and considering 20 different items.

Heidelberg *et al.* [12] 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 [14] 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 [15] 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 plan with mathematical optimization is obtained. This container load planning is usually done roughly 2 hours before departure, when all cargo details are in place.

Chan *et al.* [17] 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 the 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 [18] 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.* [20] solved WBP by selecting the most profitable subset of containers to be loaded into an aircraft by using mixed-integer programming. Experimental results on real-life data showed significant improvements compared to those obtained manually by an experienced planner.

Mesquita and Cunha [19] 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.* [21] developed a mixed-integer program for optimally rearranging a set of pallets into a compartmentalized cargo aircraft, specifically the *Boeing 747*.

Roesener and Hall [22] 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* [23] presented a mixed integer linear programming model that selects the most profitable pallets while 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 [24] were the first to simultaneously model 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 [26] 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 given 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.* [27] presented a mathematical model to optimize the loading of heterogeneous 3D boxes on pallets with a truncated parallelepiped 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.* [28] developed some heuristics to solve this problem.

Chenguang *et al.* [30] 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 [33]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.* [35] integrated this tool into a digital simulation model with a visualization and validation system based on sensors that alert about load deviations.

Zhao *et al.* [36] proposed a new model 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.

Zhao *et al.* [37] proposed three integer linear programming models: a Bi-objective Optimization Model (BOM), a Combinatorial Optimization Model (COM), and an Improved Combinatorial Optimization Model (IOM). The objectives of their solutions were the maximum loading possible and the lowest CG deviation from a specified target CG. Four scenarios with various metrics for three models are solved for the B777F aircraft using Gurobi. The results of the computations demonstrated that the BOM has the fastest solution speed, but the CG deviation is the worst, and in several cases, the CG deviation results are unacceptable. The COM has the longest solution time, which is difficult to tolerate in practice.

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 four 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 for 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 weigh $140kg$. For more information, see `www.463LPallet.com`.
- All items allocated to a pallet must have the same destination. A pallet that 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 other costs not directly associated with aircraft flight, such as handling.

Figure 2: A packed content on a 463L pallet inside a *Boeing C-17*
Source: From Wikimedia Commons, the free media repository

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

3.2. Aircraft parameters and load balancing

We consider real-world scenarios with a single aircraft with a payload of $75,000kg$, and enough fuel for a range of $4,400km$. The aircraft layout is presented in Figure 3, where the pallet positions are identified by p_i, $1 \leq i \leq 18$.

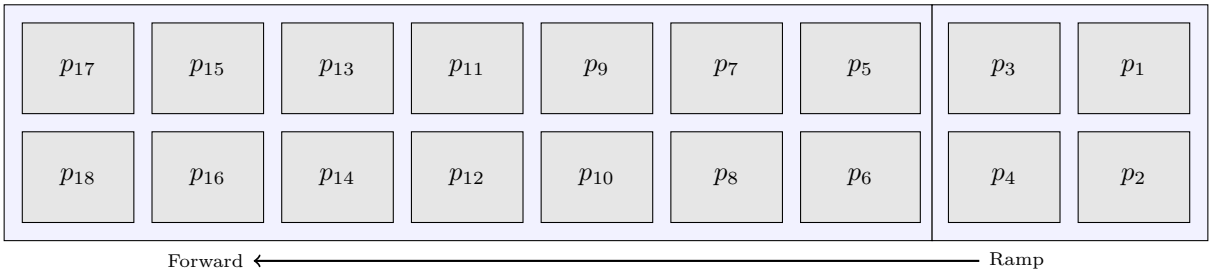

Figure 3: Aircraft layout

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 4).

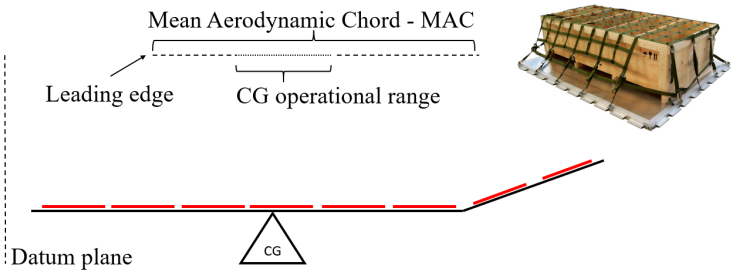

Figure 4: Aircraft longitudinal cut, where red lines are pallets positions

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 lateral 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 has an inclination of 25°, we made the necessary corrections in D_i^{long}, W_i and V_i of the corresponding pallets.

[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).

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 that should be better evaluated in future research. It is important to consider that the c_g tends to zero as the aircraft's attitude tends to be level. As the CG deviation varies from 0 to $limit_{long}^{CG}$, the *fuel cost* increase varies from 0 to c_g.

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 of $limit_{long}^{CG}$ from the CG of the aircraft.
- The CG of the total load must be at a maximum lateral distance of $limit_{lat}^{CG}$ 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 a distance D_i^{lat} from the centerline of the aircraft.

4. The mathematical model

Given the assumptions and parameters described in the previous section, we are ready to present the mathematical model 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

In the definitions below and in the tests of Section 6, we use the values assigned in Tables 1 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 $Q_0 = \varnothing$. Packed 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 $S_K = \{s : \{1, \ldots, K\} \to \{1, \ldots, K\}\}$ 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 $\{0, \pi_1, \ldots, \pi_K, 0\}$. For ease of notation, we can define $\pi_0 = \pi_{K+1} = 0$.

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

We have the following calculation equations:

- $\tilde{s}_\pi$ 2 is the sum of the scores of the items loaded on the aircraft throughout the tour π.
- τ_{π_k} 3 is the longitudinal torque applied by the loaded pallets at the node π_k, in proportion relative to the highest torque supported by the aircraft.
- $\tilde{c}_\pi$ 4 is the cost of to the total fuel consumed in the tour π due to the distances traveled and the CG longitudinal deviations [2]
- The sets of nodes not yet visited along the tour π are defined in 5 and 6.
- At the beginning of the tour π, there are no packed contents 7.
- $\epsilon^t_{\pi_k}$ 8 and $\epsilon^a_{\pi_k}$ 9 are the lateral torque applied by the loaded pallets at the node π_k, in proportion relative to the highest torque supported by the aircraft. For ease of notation, $\epsilon^t_{\pi_k}$ corresponds to the items, and $\epsilon^a_{\pi_k}$ to the packed contents.

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

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

[2] In our experiments, we found that the magnitude of the lateral torque was always very small, so we decided to ignore it in the fuel consumption calculation in each pallet.

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

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

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

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

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

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

4.6. The constraints

Finally, we can consider the constraints at 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;
- Packed 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.

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

$$|\epsilon^t_{\pi_k} + \epsilon^a_{\pi_k}| \leq 1;\ k \in \{0, \ldots, K\} \tag{11}$$

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

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

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

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

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

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

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

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

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 model of the problem are presented, it is possible to see that ACLP+RPDP is *NP-hard*.

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 us 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 runtime of the 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 fewer than six nodes, and each node has hundreds of items to be shipped. We also know that missions with fewer nodes are more frequent than longer missions. Under these circumstances, we can adopt two more strategies:

- We will consider that the number of destinations is greater than 1, to avoid the case of a single and trivial tour, but less than the number of pallets, that is, $1 < K < m$. So we can preset the destinations of the pallets at 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 we have quick node-by-node solutions, as the number of nodes is small, 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 Solving ACLP+RPDP

procedure $ACLP + RPDP(scenario, surplus, tmax)$
  Let M be the set of pallets ▷ parameters in Table 3
  Let K, L and C be according to $scenario$ ▷ Nodes and costs from Tables 1 and 4
  Let π_{TSP1} and π_{TSP2} ▷ The shortest tours from Tables 1 and 4
  $N \leftarrow ItemsGeneration(scenario, surplus)$ ▷ Items available for shipment
  for each $method$ ▷ $method$ is a MIP solver or a heuristic
    $f_1 \leftarrow SolveTour(\pi_{TSP1}, L, M, C, N, method, tmax)$
    $f_2 \leftarrow SolveTour(\pi_{TSP2}, L, M, C, N, method, tmax)$
    $answer1[scenario, surplus, method] \leftarrow max(f_1, f_2)$ ▷ Best result between the shortest tours
    for each $\pi \in S_K$ ▷ π is a permutation of nodes (a tour)
      $f_\pi \leftarrow SolveTour(\pi, L, M, C, N, method, tmax/K!)$ ▷ f_π is a solution to the tour π
    $answer2[scenario, surplus, method] \leftarrow \max f_\pi$ ▷ Best result among all permutations
  return $answer1, answer2$

The aircraft parameters and the set M of pallets are in Table 3. In this algorithm, we use five values for *scenario*, according to Tables 1 and 4, which define the number K of destinations, the set L of nodes, the costs

C, and the shortest tours π_{TSP1} and π_{TSP2}.

Table 4: Testing scenarios

Scenario	K	L	π_{TSP1}	π_{TSP2}
1	2	$\{0,1,2\}$	0 1 2 0	0 2 1 0
2	3	$\{0,1,2,3\}$	0 1 2 3 0	0 3 2 1 0
3	4	$\{0,1,2,3,4\}$	0 4 1 2 3 0	0 3 2 1 4 0
4	5	$\{0,1,2,3,4,5\}$	0 4 1 2 5 3 0	0 3 5 2 1 4 0
5	6	$\{0,1,2,3,4,5,6\}$	0 4 1 2 6 5 3 0	0 3 5 6 2 1 4 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).

tmax is a runtime limit, which will be distributed among the tours (lines 7, 8 and 11). *method* corresponds to a MIP solver or a heuristic to the node-by-node solution *SolveTour*, which will be presented in subsection 5.2.

The best results corresponding to the shortest tours are stored in *answer*1 (line 9), and those obtained by testing all $K!$ 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 heuristics 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 $1 < K < m$. 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 at 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 *tmax* is the runtime limit that will be distributed among the $K + 1$ legs of the tour.

Algorithm 2 Solving the tour π with *method*

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

When the aircraft is at the base, the initial graph G_1 is empty, and there are no packed contents 10. Otherwise, *UpdatePacked* (line 12) 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 13).

SetPalletsDestinations (line 14) presets the destination of each pallet based on the volume demands of the current node without changing the pallet's destination with packed contents.

Finally, *SolveNode* includes the edges corresponding to the items shipped at the current node, returning the graph G_2 (line 15). The score and the CG deviation of G_2 are calculated (line 16) and accumulated (lines 17-18), allowing the final result of this tour (line 19).

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 Updating the packed contents that remain boarded at the node π_k

1: **procedure** $UpdatePacked(M, Q_{\pi_k}, \pi_k)$
2: $\quad E_{Q_{\pi_k}} \leftarrow MinCGDeviation(E_{Q_{\pi_k}})$ ▷ Using a MIP solver
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) relocates the packed contents on the pallets, minimizing torque and ensuring that they all remain on board, one packed content on each pallet. It is run through a MIP solver with the objective function 20 and the constraints 21 and 22. As there are few variables, $E_{Q_{\pi_k}}$ is obtained in less than 30 milliseconds. Finally, the destination of each pallet with packed content is updated (lines 3-7).

$$\min f = \Big| \sum_{i=1}^{m} \sum_{q=1}^{m_{\pi_k}} Y_{iq}^k \times w_q \times D_i^{long} \Big| \tag{20}$$

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

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

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

Algorithm 4 Setting pallets destination based on the items to be embarked at the node π_k

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

vol stores the demand volume of items destined for the non-visited nodes (line 2). The destination of empty pallets is defined proportionally to the volume of items to be embarked (lines 11-18). The destination with the maximum volume defines any remaining pallets (lines 19-21).

ScoreAndDeviation, described in Algorithm 5, evaluates the allocation graph G generated by *SolveNode* at node π_k and returns the corresponding score and CG deviation.

Algorithm 5 Calculating the score and the relative torque deviation of the graph G at the 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: $\tau_i \leftarrow 0$, $1 \leq i \leq m$
5: **for** $i \leftarrow 1$ to m
6: **for** $j \leftarrow 1$ to n_{π_k}
7: **if** $X_{ij}^{\pi_k} = 1$
8: $s \leftarrow s + s_j$
9: $\tau_i \leftarrow \tau_i + w_j \times D_i^{long}$
10: **for** $q \leftarrow 1$ to m_{π_k}
11: **if** $Y_{iq}^{\pi_k} = 1$
12: $s \leftarrow s + s_q$
13: $\tau_i \leftarrow \tau_i + w_q \times D_i^{long}$
14: $\tau \leftarrow \sum_{i=1}^{m} \tau_i / (W_{max} \times limit_{long}^{CG})$
15: **return** s, τ

This algorithm consists of a loop that goes through all the pallets (lines 5-13), accumulating the scores

(lines 8 and 12) and the torques (lines 9 and 13) of the shipped items, allowing the final calculation of the CG deviation (line 14).

5.2. Node-by-node solutions

In this subsection, we present two implementations of *SolveNode* algorithm: a MIP solver and a heuristic.

5.2.1. With a MIP solver

Our strategy adopted in *SolveTour* defines the values of some variables: the set of nodes to be visited is updated, the packed contents that remain on board are reallocated to minimize the CG deviation, and the pallet's destinations are determined according to the volume of items available for shipment.

In this way, the mathematical model for $SolveNode(MIP, \pi_k, G, tmax)$ becomes simpler, which finds an allocation of available items at the node π_k using previously defined values of L_{π_k}, $T_i^{\pi_k}$, and $a_q^{\pi_k}$. Thus, we use a MIP solver with a runtime limit $tmax$ at the node π_k to maximize the objective function

$$\max f = \tilde{s}/\tilde{c} \tag{23}$$

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

$$\tau_{\pi_k} = \sum_{i=1}^{m} \Big[D_i^{long} \times (\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q) \Big] \Big/ W_{max} \times limit_{long}^{CG} \tag{25}$$

$$\tilde{c} = c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau_{\pi_k}|) \tag{26}$$

$$|\tau_{\pi_k}| \leq 1 \tag{27}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q \leq W_i;\ i \in \{1, \ldots, m\} \tag{28}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times v_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times v_q \leq\ V_i;\ i \in \{1, \ldots, m\} \tag{29}$$

$$\sum_{i=1}^{m} X_{ij} \leq 1;\ j \in \{1, \ldots, n_{\pi_k}\} \tag{30}$$

$$X_{ij} = 0;\ to_j \notin L_{\pi_k};\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{31}$$

$$X_{ij} \leq X_{ij} \times (T_i^{\pi_k} - to_j + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{32}$$

$$X_{ij} \leq X_{ij} \times (to_j - T_i^{\pi_k} + 1);\ i \in \{1, \ldots, m\};\ j \in \{1, \ldots, n_{\pi_k}\} \tag{33}$$

The constraints 32 and 33 are equivalent to $X_{ij} = 1$ if $to_j = T_i^{\pi_k}$, and $X_{ij} = 0$ otherwise .

5.2.2. With the Shims heuristic

One of the main objectives of this work was to find a quick heuristic that offers a good-quality solution for the node-by-node problem. Taking this into account, we design algorithms based on known meta-heuristics: *Ant Colony Optimization* (ACO) [8, 10], *Noising Method Optimization* (NMO) [9, 13, 34], *Tabu Search* (TS) [5] and *Greedy Randomized Adaptive Search Procedure* (GRASP) [6]. We considered several ideas from the literature [11, 16, 29, 31, 34], and we were careful to use the same data structures and procedures in all implementations.

However, 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 5). 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* θ_{ij}, Equation 34, which can be understood as the tendency to allocate the item j to the pallet i at the node π_k. It is directly proportional to the score, and inversely to the volume and the torque of each item.

$$\theta_{ij} = \frac{s_j}{v_j} \times \left(1 - \frac{w_j \times |D_i^{long}|}{\max w_j \times \max |D_i^{long}|}\right);\ i \in \{1, \ldots, m\},\ j \in \{1, \ldots, n_{\pi_k}\} \tag{34}$$

Figure 5: *Shims* of various thicknesses
Source: www.mscdirect.com/product/details/70475967

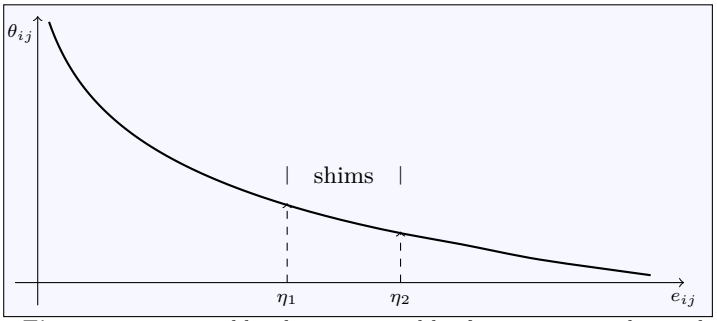

Figure 6: n_{π_k} possible edges e_{ij} sorted by θ_{ij} in non-ascending order

Considering only the items that can be shipped at the node π_k, Figure 6 represents n_{π_k} possible edges e_{ij} of the pallet i sorted by θ_{ij} in non-ascending order. Initially, *Shims* builds a greedy solution for the pallet i selecting edges up to index η_1 (first phase). Then, with the edges between η_1 and η_2, it elaborates different possible complements (second phase), including later the best ones in the same pallet (third phase). *Shims* is described in Algorithm 6.

Algorithm 6 *Shims* heuristic at the node π_k

1: **procedure** $SolveNode(Shims, \pi_k, G, tmax)$
2: $T_{begin} \leftarrow$ current system time
3: Let $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}})$
4: Sort M by $|D_i^{long}|$ in non-descending order
5: $E_{N_{\pi_k}} \leftarrow \varnothing$ ▷ An empty set of pallet-item edges
6: $\tau_{max} \leftarrow W_{max} \times limit_{long}^{CG}$ ▷ Maximum aircraft torque
7: $limit \leftarrow 0.92$
8: **for** $i \leftarrow 1$ to m
9: $\tau_{\pi_k} \leftarrow \sum_{(i,q) \in E_{Q_{\pi_k}}} w_q \times D_i^{long}$ ▷ Torque due to the packed contents
10: $vol_i \leftarrow \sum_{(i,q) \in E_{Q_{\pi_k}}} v_q$ ▷ Volume at the pallet i due to the packed contents
11: Let E be an array of n_{π_k} possibles edges of the pallet i sorted by θ_{ij} in non-ascending order
12: $\eta_1 \leftarrow 1$
13: **repeat**
14: $e_{ij} \leftarrow E_{\eta_1}$ ▷ e_{ij} is the possible allocation of the item j to the pallet i
15: **if** $(E_{N_{\pi_k}} \cup \{e_{ij}\}$ is feasible) **and** $(vol_i \leq V_i \times limit)$ **and** $(|\tau_{\pi_k} + w_j \times D_i^{long}| \leq W_{max} \times limit_{long}^{CG})$
16: $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup \{e_{ij}\}$
17: $vol_i \leftarrow vol_i + v_j$
18: $\tau_{\pi_k} \leftarrow \tau_{\pi_k} + w_j \times D_i^{long}$
19: $\eta_1 \leftarrow \eta_1 + 1$
20: **until** $(vol_i > V_i \times limit)$ **or** $(\eta_1 > n_{\pi_k})$ ▷ End of the first phase
21: $slack_i \leftarrow V_i - vol_i$
22: $\eta_2 \leftarrow \eta_1$
23: **while** $(\eta_2 \leq n_{\pi_k})$ **and** $(vol_i < (1 + 2 \times (1 - limit)) \times V_i)$
24: $e_{ij} \leftarrow E_{\eta_2}$
25: $vol_i \leftarrow vol_i + v_j$
26: $\eta_2 \leftarrow \eta_2 + 1$ ▷ End of the second phase
27: $vol \leftarrow 0$; $b \leftarrow 1$; $shims_b \leftarrow \varnothing$; $Set \leftarrow \{shims_b\}$
28: **for** $x \leftarrow \eta_1$ to η_2
29: **if** $T_{current} - T_{begin} > tmax$
30: **break** ▷ Runtime limit exceeded
31: $NewShims \leftarrow$ **True**
32: $e_{ij} \leftarrow E_x$
33: **for** $shims \in Set$
34: **if** $(e_{ij} \notin (E_{N_{\pi_k}} \cup shims))$ **and** (e_{ij} is feasible) **and** $((v_j + vol) \leq slack_i)$
35: $shims \leftarrow shims \cup \{e_{ij}\}$
36: $vol \leftarrow vol + v_j$
37: $NewShims \leftarrow$ **False**
38: **break**
39: **if** $NewShims$
40: $vol \leftarrow 0$; $b \leftarrow b + 1$; $shims_b \leftarrow \{e_{ij}\}$
41: $Set \leftarrow Set \cup \{shims_b\}$
42: $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ij} \in shims} w_j$ is maximum
43: $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ij} \in shims} v_j$ is maximum
44: $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ij} \in shims} s_j$ is maximum
45: $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup sh_{best}$ ▷ End of the third phase
46: **return** $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}} \cup E_{N_{\pi_k}})$

Initially, Q_{π_k} (line 3) corresponds to the packed contents that remain on board. It is important to remember that $E_{Q_{\pi_k}}$ and M were modified by $UpdatePacked(M, Q_{\pi_k}, \pi_k)$ and $SetPalletsDestinations(M, \pi_k)$. Then,

the pallets i are considered in non-descending order of $|D_i^{long}|$.

For each pallet i, its n_{π_k} possible edges e_{ij} are considered in non-increasing order of θ_{ij}:

- In *greedy phase* (lines 4-20), a partial solution for each pallet i is constructed by adding edges following θ_{ij} order. The η_1 index corresponds to the accumulated volume equal to $V_i \times limit$. In η_2 index, this same accumulated volume reaches $(1 + 2 \times (1 - limit)) \times V_i$. The size of this range $[\eta_1, \eta_2]$ was defined empirically, and the value $limit = 0.92$ (line 7) was determined by the *iRace* tool [38].

- In *composition phase* (lines 21-26), a set of shims named *Set* is created for each pallet i, where each shim is formed by a set of edges in the range $[\eta_1, \eta_2]$, 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* [4]. Basically, shims are created by accumulating the following edges, taking $slack_i$ as a limit.

- In *selection phase* (lines 27-45), the best shim in *Set* is chosen. Initially, two shims are found: sh_w with larger weight and sh_v with larger volume. Between the two, the one with the highest score will be chosen, and its edges will be inserted into $E_{N_{\pi_k}}$.

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 that until now had not been modeled in the literature, we have to create our own benchmarks. For this, we based it 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

Group	*P*	*low* (kg)	*high* (kg)
1	0.23	10	20
2	0.22	21	40
3	0.24	41	80
4	0.23	81	200
5	0.08	201	340

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.

$ItemsGeneration$, which generates N, is described in Algorithm 7.

Algorithm 7 Generating items

```
1:  procedure ItemsGeneration(scenario, surplus)
2:      Let L and M                                          ▷ From Tables 3 and 4
3:      limit ← surplus × Σ_{i=1}^{m} V_i
4:      for k ← 0 to K
5:          N_k ← ∅
6:          j ← 0
7:          vol ← 0
8:          while vol < limit
9:              j ← j + 1
10:             Let t_j^k be the item j at the node k
11:             repeat
12:                 to_j ← RandomInt(0, K)
13:             until to_j ≠ k
14:             x = Roulette(item) biased through P          ▷ From Table 5
15:             w_j ← RandomReal(low(x), high(x))
16:             s_j ← ⌊100 × (1 − log_10(RandomInt(1, 9)))⌉
17:             v_j ← w_j / RandomReal(148, 344)
18:             vol ← vol + v_j
19:             N_k ← N_k ∪ {t_j^k}
20:     N ← ⋃_{0≤k≤K} N_k
21:     return N
```

scenario defines L and M (line 2), and *surplus* sets a limit on the total volume of items at each node (line 3). To avoid simply loading all items, we use $surplus \in \{1.2, 1.5, 2.0\}$. This also represents more instances for tests in 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 $246 kg/m^3$ (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 considering 5 values for *scenario* from Table 4, 3 values for *surplus* from $\{1.2, 1.5, 2.0\}$, 4 values for *tmax* from $\{240s, 1200s, 2400s, 3600s\}$, and 6 different *method* for the node-by-node solution: *Gurobi* (see 5.2.1), ACO, NMO, TS, GRASP, and *Shims* (Algorithm 6).

For Gurobi to be able to solve the largest possible number of tests without memory overflow, we set its parameter *MIPgap* to 1%. This shortens its runtime, in addition to ensuring that its objective function f is at most 1% of the optimal solution. For more details, see `https://support.gurobi.com`.

Among the heuristics, we will present only the results obtained by *Shims*, which had the best performance. For each *scenario*, *surplus* and *tmax*, 7 different instances were generated. We will present the average of the objective function f and the runtime of *Gurobi* and *Shims* in these instances. To facilitate the comparison between both, we added a last column in the 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 by both 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 runtimes of all scenarios and the sum of the method runtimes in all scenarios. The method with the highest **Speed-up** is the fastest.

We also indicate the two adopted strategies: dedicating all the processing time to the two shortest tours or distributing it among all the $K!$ tours.

The results obtained with $tmax = 3600s$, which is the highest tested runtime limit, are in Tables 6, 7 and 8, with *surplus* values 1.2, 1.5 and 2.0, respectively. We indicate with an **x** the cases where *Gurobi* did not find a feasible solution within this runtime limit or had to be aborted due to high RAM consumption.

Table 6: Solutions with $surplus = 1.2$ and $tmax = 3600s$

Tested tours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Shortest	*Gurobi*	f	8.52	11.70	13.07	13.54	12.44	1.00
		time (s)	17	17	16	17	26	1.0
	Shims	f	8.54	11.63	12.97	13.43	12.36	0.99
		time (s)	1	1	2	2	2	11.6
All $K!$	*Gurobi*	f	8.52	12.25	13.32	14.61	x	1.00
		time (s)	25	36	121	297	x	1.0
	Shims	f	8.54	12.26	13.23	14.53	13.42	0.99
		time (s)	1	2	6	17	106	18.4

Table 7: Solutions with $surplus = 1.5$ and $tmax = 3600s$

Tested tours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Shortest	*Gurobi*	f	11.80	16.74	18.04	18.86	16.92	1.00
		time (s)	38	30	27	29	81	1.00
	Shims	f	11.83	16.74	18.01	18.89	16.91	1.00
		time (s)	1	2	2	3	3	20.5
All $K!$	*Gurobi*	f	11.81	17.01	18.25	20.59	18.36	1.00
		time (s)	53	61	193	470	2241	1.0
	Shims	f	11.83	17.00	18.21	20.45	18.00	0.99
		time (s)	1	2	9	26	157	15.5

Table 8: Solutions with $surplus = 2.0$ and $tmax = 3600s$

Tested tours	*method*	*scenario*	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Shortest	*Gurobi*	f	17.75	24.39	26.46	27.09	24.00	1.00
		time (s)	162	95	74	68	67	1.0
	Shims	f	17.78	24.40	26.36	27.01	23.94	0.99
		time (s)	2	3	3	4	5	27.4
All $K!$	*Gurobi*	f	17.75	25.24	26.46	29.11	x	1.00
		time (s)	151	123	319	689	x	1.0
	Shims	f	17.78	25.25	26.38	28.98	26.08	1.00
		time (s)	2	3	13	37	229	23.3

From these data, we can draw some conclusions:

- The strategy of testing all $K!$ tours often gives a better-quality solution.
- *Gurobi* fails in some cases when $scenario = 5$ and the strategy is to check all $K!$ tours. This occurs because the runtime limit per node is smaller and there tend to be more packed contents on the aircraft, reducing the space for allocating items and making the solution difficult.
- When *Gurobi* finishes, it finds the best solution, but the one obtained by *Shims* reaches 99% of that value.
- *Shims* always finds a solution, being 11 to 27 times faster.
- All runtimes are much lower than the limit because the solution on many nodes can be fast. Anyway, in all the tests performed, the maximum time spent by *Shims* did not reach 4 minutes, i.e., it was practically instantaneous. On the other hand, when $scenario = 5$ and $surplus = 1.5$, Gurobi spent almost 40 minutes.

Table 9 shows the results obtained with the strategy of testing the $K!$ tours in all scenarios with different runtime limits. We can observe more cases where *Gurobi* fails, even in smaller scenarios. When *Gurobi* finishes, *Shims* finds a solution of similar quality (98% or better). In all cases, *Shims* finds a good solution in less than 4 minutes.

Table 9: Solutions testing all $K!$ tours with different runtime limits

		surplus	**1.2**					**1.5**					**2.0**				
method	*tmax*	*scenario*	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
Gurobi	240s	f	8.53	12.24	13.34	x	x	11.81	17.02	18.31	x	x	17.75	25.24	x	x	x
		time (s)	41	43	123	x	x	50	62	181	x	x	166	140	x	x	x
	1200s	f	8.52	12.25	13.31	14.67	x	11.81	17.02	18.25	20.64	x	17.75	25.24	26.49	27.97	x
		time (s)	39	41	129	217	x	52	61	190	304	x	161	139	384	579	x
	2400s	f	8.53	12.25	13.30	14.59	13.59	11.80	17.02	18.25	20.60	x	17.75	25.24	26.46	29.10	x
		time (s)	33	39	127	313	1444	51	62	188	327	x	158	127	381	543	x
	3600s	f	8.52	12.25	13.32	14.61	x	11.81	17.01	18.25	20.59	18.36	17.75	25.24	26.46	29.11	x
		time (s)	25	36	121	297	x	53	61	193	470	2241	151	123	319	689	x
Shims	240s	f	8.54	12.26	13.23	14.53	13.42	11.83	17.00	18.21	20.45	18.00	17.78	25.25	26.38	28.98	26.08
		time (s)	1	2	6	17	109	1	2	8	26	160	2	3	13	37	229

Most of *Gurobi* executions consumed over 12 GB of RAM memory. Considering that the computer used in the experiments has a standard RAM consumption when idle of 3.5 GB (with *Python Interface Development Environment*, a PDF viewer, a system monitor, and a LaTeX IDE simultaneously open), the actual RAM consumption of *Gurobi* was over 8.5 GB. On the other hand, all *Shims* executions consumed at most 1.5 GB of RAM memory.

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 simultaneously is addressed, where the cost-effectiveness of transport is maximized. We adopted some simplifications that are not critical but that allowed for an unprecedented solution to 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 a real aircraft, we have developed some node-by-node solutions that are not necessarily optimal. This allows us to test two strategies: find a solution using the shortest tours, or 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*. At the moment, there is no commercial software available for this problem.

We have developed a solution process that, in less than 4 minutes, can establish a flight plan for a single aircraft with 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. 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. In this way, at each node of the tour, the time required is restricted exclusively to handling the airport equipment.

Our main contributions were the mathematical model of ACLP+RPDP that involves four sub-problems *NP-hard*, a complete process to solve it on a simple handheld computer, and a new heuristic that offers fast node solutions with good quality. Finally, by focusing on the node-to-node solution, the method of this work is not exclusive to aircraft and airports; it can be adapted, for example, to ships and ports, vehicles and warehouses, or wagons and railways, provided that their practical cases are similar to those considered here. In these situations, it would be necessary to make some changes in the model: for example, modify the load balancing constraints and consider the available space in vehicles or wagons instead of pallets.

As this is ongoing research, we thought about some possible future improvements:

- consider an aircraft fleet rather than a single one;
- model three-dimensional items;
- discard the highest-cost tours to reduce the runtime;
- implement parallel algorithms in some steps of the solution.

The dataset used in this study is available in [39].

References

References

[1] R. Karp, *Reducibility among combinatorial problems. Complexity of Computer Computations.* (R.E. Miller and J.M. Thatcher, eds.), Plenum Press, 85-103 (1972).

[2] 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.

[3] 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.

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

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Feng, B.; Li, Y.; Shen, Z.J.M. *Air cargo operations: Literature review and comparison with practices.* Transp. Res. Part C Emerg. Technol. 2015, 56, 263?280.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Zhao, Xiangling, Yun Dong, and Lei Zuo. *A Combinatorial Optimization Approach for Air Cargo Palletization and Aircraft Loading.* Mathematics (Basel) 11.13 (2023): 2798. Web.

[38] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari and T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, *Operations Research Perspectives*, Vol. 3, pp. 43-58, 2016.

[39] Mesquita, Antonio Celio, and Carlos Alberto Alonso Sanches. Air Cargo Load and Route Planning in Pickup and Delivery Operations (2022). Web.