Air cargo load and route planning in pickup and delivery operations

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

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

Abstract

In the aerial pickup and delivery of goods in a distribution network, transport aviation faces risks of load imbalance due to the urgency required for loading, immediate take-off, and mission accomplishment. Transport planners deal with trip itineraries, prioritisation of items, building up pallets, and balanced loading, but there are no commercially available systems that can integrally assist in all these requirements. This enables other risks, such as improper delivery, excessive fuel burn, and possible safety issues due to cargo imbalance, as well as a longer than necessary turn-around time. This *NP-hard* problem, named *Air Cargo Load Planning with Routing, Pickup, and Delivery Problem (ACLP+RPDP)*, is mathematically modelled using standardised pallets in fixed positions. We developed a strategy to solve this problem, considering historical transport data from some Brazilian hub networks, and performed several experiments with a commercial solver, five known meta-heuristics, and a new heuristic designed specifically for this problem. By using a portable computer, our strategy quickly found practical solutions to a wide range of real problems in much less than operationally acceptable time.

Keywords: OR in Airlines, Load Planning, Air Palletization, Weight and Balance, Pickup and Delivery, Vehicle Routing

1. Introduction

The aviation industry adapts during global crises to keep supply chains moving. Air cargo provided complex expertise and the ability to access diverse destinations, delivering essential goods such as medicines, vaccine supplies, testing kits and other necessities with exceptional speed. This mode of transportation has become a preferred choice for governments, corporations and global companies in urgent need of transportation solutions.

Air cargo services are specially designed for organisations that require customised transportation, handle sensitive goods, or serve remote locations with limited routes. Air carriers typically use high-capacity cargo planes for economies of scale. Many cargo airlines have worldwide networks spread across destinations around the world.

A few years ago, (Brandt and Nickel, 2019) 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: item characteristics to be transported (dimensions, scores, dangerousness, etc.); types and quantities of *unit load devices* (ULDs); when these pallets are assembled; how items are allocated to pallets; in which positions these pallets are to be placed; how total cargo weight is balanced; etc.

Recently, (Zhao et al., 2023) developed integer programming solution that address ACLPP and WBP separately, impacting payload optimization and aircraft centre of gravity (CG). The authors argue that, with improvements in computer processing power, joint combinatorial optimization of ACPP and WBP is now feasible. They propose three integer linear programming models: a bi-objective optimization model (BOM), a combinatorial optimization model (COM), and an improved combinatorial optimization model (IOM). The models aim to determine the maximum loading capacity and lowest CG deviation from a specified target CG.

The study uses Gurobi to solve four scenarios with various conditional metrics for three models. Results show that the BOM has the fastest solution speed, but the CG deviation is the largest, making the results unacceptable in some cases. The COM has the longest solution time, making it difficult to tolerate in

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

practice. The IOM offers the best optimization solution, despite taking a little longer to solve computationally than the BOM.

Using a Boeing 777F, four scenarios and 25 cases were tested, finding that the BOM's variable feasible domain is limited, the optimization result is insufficient, and the CG deviation is sometimes too large. The COM takes too long to solve, making it difficult to meet the needs of the actual scene. The IOM accelerates the solving process compared to the COM, but some scenarios still require significant computation time.

Inefficient air transport plans can lead to unnecessary costs, extra routes, longer distances, and incorrect destinations. Unbalanced cargo increases fuel consumption due to altered aircraft pitch angles.

A study by (van Es, 2007, p. 22) highlights the risk of weight- and balance-related accidents on cargo flights compared to passenger flights. So, balancing cargo is crucial for safe aerial transportation, as an improperly positioned centre of gravity can result in dangerous take-off and landing conditions and stall recovery issues.

Another important issue is managing fuel consumption. To get an idea of the influence of CG displacement on fuel consumption, (Mongeau and Bes, 2003, p. 140) report that *"a displacement of the CG of less than 75 cm in a long-range aircraft yields, over a 10,000 km flight, a saving of 4,000 kg of fuel"*. This reinforces the importance of balancing cargo.

Despite technological advancements, many airlines still rely on manual aircraft loading and balancing, using simple computer-aided tools for CG computation and constraint testing. Loadmasters often create load plans manually, which is time-consuming and can lead to flight delays.

As to the right delivery, an organisation's purchasing department may acquire supplies in the main capitals of the country and abroad to meet the company's needs, and it wants these materials to be delivered to the requesting organisations within the stipulated deadlines.

A rational decision-making process is crucial to avoid creating inefficient or unsafe transport plans, considering the high costs of fuel, maintenance, operation, outsourcing expenses, potential operational impairments, and safety risks due to unbalanced cargo.

Solving the ACLPP+RPDP is vital for optimising strategic scores of what must be transported, saving time and effort in loading, ensuring safety and balance, ensuring correct pickups and deliveries, and finding the best routes considering fuel use due to potential cargo imbalance.

However, it is crucial to highlight that there are still other important challenges in air cargo transport that go beyond the definition of ACLPP, especially with regard to routes, and pickup and delivery at each destination. In this context, at least two more important sub-problems can be considered: simultaneous pickup and delivery at each node, called the *Pickup and Delivery Problem* (PDP), and searching for the best benefit-cost route, which is a special case of the *Travelling Salesman Problem* (TSP).

The problem addressed in this work is the definition of the route of an aircraft that loads and unloads hundreds of items in up to seven hubs, dealing with weight, volume, and balance constraints, and maximising the benefit-cost ratio. This challenge has become even more acute during the COVID-19 pandemic due to the need for urgent medical supplies. Delays in shipping essential items such as respirators to critical areas highlighted the need for optimised solutions.

This problem is extremely complex, as it deals with different objectives: defining a route that visits all hubs; maximising the items transported along this route, prioritising the essential ones; making sure items reach the correct destinations; ensuring aircraft safety constraints; and saving fuel so that the flight is sustainable.

We have developed a heuristic process that can be run on a handheld computer, quickly providing a good solution to real instances of this problem. Solutions consist of flight itineraries, pickup and delivery plans, and the allocation of items onto pallets, ensuring the load-balancing constraints. Our method also reduces the stresses that transport planners are subject to, as they have to deal with extensive information in a short time frame.

To the best of our knowledge, this is the first time that an air cargo problem involving simultaneously 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). As we will describe in our mathematical modelling, these four sub-problems appear in an interconnected way in ACLP+RPDP and therefore cannot be solved independently.

As a real case study, we consider a crucial network for the *Brazilian Air Force*, as can be seen in Table 1 and Figure 1. Although there are other airports of interest, these nodes were chosen due to their high demand. Other Brazilian airports tend to have smaller transport requests, which are generally met less expensively by cabotage, rail, or road transport.

This article is organised into six more sections. In Section 2, we make the literature review. In Section 3, we present the context and requirements of ACLP+RPDP. In Section 4, we describe its mathematical modelling. In Section 5, we describe the developed algorithms, whose results are presented in Section 6.

Table 1: Brazilian airports distances (km)

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 airports

Finally, our conclusions are in Section 7.

2. Literature review

The vast majority of operational research applied to air cargo is focused on challenges related to WBP, that is, the distribution of items on pallets to ensure load balancing. We can mention: (Larsen and Mikkelsen, 1980); (Brosh, 1981); (Ng, 1992); (Heidelberg et al., 1998); (Fok and Chun, 2004); (Kaluzny and Shaw, 2009); (Verstichel et al., 2011); (Limbourg et al., 2012); (Roesener and Barnes, 2016); (Chenguang et al., 2018); (Zhao et al., 2021); (Miguel et al., 2023).

Other authors have addressed pallet assembly (APP) on aircraft, possibly also considering load balancing (WBP): (Mongeau and Bes, 2003); (Chan et al., 2006); (Roesener and Hall, 2014); (Vancroonenburg et al., 2014); (Paquay et al., 2016); (Paquay et al., 2018); (Wong and Ling, 2020); (Wong et al., 2021); (Zhao et al., 2023).

In all these works, there is a great diversity of scenarios and solutions: some consider items in two dimensions, and others in three dimensions; some used integer programming, and others developed specific heuristics.

(Lurkin and Schyns, 2015) is the only work that simultaneously addresses an air cargo (WBP) and a flight itinerary (PDP) sub-problem. The authors demonstrated that this problem is *NP-hard*. Although it is innovative, strong simplifications were imposed by these authors: in relation to loading, APP was ignored; regarding routing, it is assumed that a predefined tour plan is restricted to only two legs. Referring directly to this work, (Brandt and Nickel, 2019) comment: *However, not even these sub-problems are acceptably solved for real-world problem sizes, or models omit some practically relevant constraints.*

Table 2 lists this literature and the involved sub-problems. We also indicate whether the dimensions of the items were considered (**2D** or **3D**) and which solution method was used: heuristic search methods (**Heu**), integer programming (**Int**), or linear programming (**Lin**).

As can be seen, none of these papers address air cargo palletization and load balancing with route optimisation in a multi-leg transport plan for a single aircraft. Our work is the first to address a real air transport problem in which APP, WBP, PDP and TSP arise in an interconnected way.

3. Context and assumptions

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

As for the size of the problem dealt with in this work, addressing the scale of the Travelling Salesperson Problem (TSP) in practical, real-world air transport contexts indeed reveals a fascinating and complex challenge. In the realm of academic research, TSP problems often explore theoretical models that can scale to hundreds or even larger numbers of cities.

In practical terms, especially within the air transport sector, the size and complexity of a TSP can vary significantly depending on several factors:

Fleet and Route Planning: For airline operations, the problem isn't just about finding the shortest path through a set of points; it also involves scheduling, aircraft assignment, crew scheduling, maintenance planning, and delivering the right cargo to the right client. This last operation task complexity transforms a standard TSP into a more intricate problem in our study, the Pickup and Delivery Problem (PDP).

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

Work	APP	WBP	PDP	TSP	2D	3D	Heu	Int	Lin
(Larsen and Mikkelsen, 1980)	.	★	.	.	.	.	★	.	.
(Brosh, 1981)	.	★	.	.	.	.	.	.	★
(Ng, 1992)	.	★	.	.	.	.	.	★	.
(Heidelberg et al., 1998)	.	★	.	.	★	.	★	.	.
(Mongeau and Bes, 2003)	★	★	.	.	.	.	.	★	.
(Fok and Chun, 2004)	.	★	.	.	.	.	.	★	.
(Chan et al., 2006)	★	.	.	.	.	★	★	.	.
(Kaluzny and Shaw, 2009)	.	★	.	.	★	.	.	★	.
(Verstichel et al., 2011)	.	★	.	.	.	.	.	★	.
(Limbourg et al., 2012)	.	★	.	.	.	.	.	★	.
(Roesener and Hall, 2014)	★	★	.	.	.	★	.	★	.
(Vancroonenburg et al., 2014)	★	★	.	.	.	.	.	★	.
(Lurkin and Schyns, 2015)	.	★	★	.	.	.	.	★	.
(Roesener and Barnes, 2016)	.	★	.	.	.	.	★	.	.
(Paquay et al., 2016, 2018)	★	★	.	.	.	★	★	★	.
(Chenguang et al., 2018)	.	★	.	.	★	.	★	.	.
(Wong and Ling, 2020)	★	★	.	.	.	.	.	★	.
(Wong et al., 2021)	★	★	.	.	.	.	.	★	.
(Zhao et al., 2021)	.	★	.	.	.	.	.	★	.
(Zhao et al., 2023)	★	★	.	.	.	.	.	★	.
(Miguel et al., 2023)	.	★	.	.	.	.	.	★	.
This article	★	★	★	★	.	.	★	★	.

Operational Constraints: Factors like weather, air traffic control restrictions, and the availability of take-off and landing slots make real-world TSP scenarios in air transport even more challenging. Although we do not deal with these constraints, they add layers of complexity that significantly depart from the theoretical TSP model.

Dynamic Nature: Unlike the static nature of classical TSP, real-world problems in air transport are highly dynamic. Routes need to be optimized continuously as conditions change, requiring algorithms that can adapt in real-time or near-real-time.

In practical air transport scenarios, the size of a TSP can be less about the sheer number of nodes and more about the complexity and dynamic nature of the problem, which in our case (a single aircraft in a single day) is rarely expected to be more than seven nodes. Despite this, at the end of this work, we to submit the best method found to a more challenging 15-node problem as a validation test.

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 content* a set of items of the same destination stacked on a pallet and covered with a restraining net. 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, where Table 3 shows the aircraft parameters. p_i are pallets, $1 \leq i \leq 18$, whose weight and volume limits are W_i and V_i, respectively. D_i^{long} and D_i^{lat} are, respectively, the longitudinal and lateral distances of each pallet centroids to the aircraft centre of gravity (CG) 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 (p_1, p_2, p_3, and p_4).

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 kilometre flown and can carry up to W_{max} of cargo distributed on the pallets. The fuel penalty c_g is the percentage of cost increase due to the CG deviation on the longitudinal axis, estimated at 5.0%. It is important to consider that c_g tends to zero as the aircraft attitude tends to be level. As the CG deviation varies from 0 to $limit_{long}^{CG}$, this fuel penalty varies from 0 to c_g.

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 2).

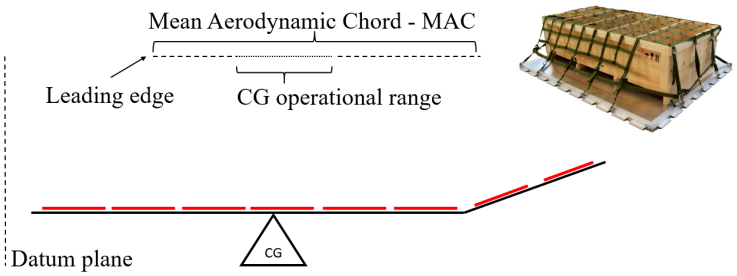


Figure 2: Aircraft longitudinal cut, where red lines are pallets positions

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, because builders are well-trained to do so;
- 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 centreline of the aircraft;
- when there are items or packed contents in p_i, the common destination of this load will be assigned to variable T_i.

[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).

4. The mathematical modelling

In this section, we present the mathematical modelling of ACLP+RPDP in Tables 4, 5, 6, and 7, with their corresponding descriptions.

In Table 4, we describe the problem structure: nodes and their permutations, distances and associated costs, pallets characteristics, items available for shipment at each node, and packed contents shipped. The item j in node k has score s_j, weight w_j, volume v_j, and destination $to_j \in L_k$. Similarly, the packed content q, that remains on board at node k, has total weight w_q, total volume v_q, and destination $to_q \in L_k$. Packed contents that were destined to node k are unloaded when the aircraft arrives there; that is, they are not considered in Q_k.

Table 4: Problem structure

Notation	Description
$L = \{0, 1, \ldots, K\}$	Set of $K+1$ nodes of the tour, where node 0 is the base
π	A permutation between nodes 1, ..., K
S_K	Set of $K!$ permutations
π_k	The k^{th} node of tour π, $1 \leq k \leq K$
Tour π	$\{0, \pi_1, \ldots, \pi_K, 0\}$ For ease of notation, $\pi_0 = \pi_{K+1} = 0$
L_k	Set of remaining nodes of tour at node k, $0 \leq k \leq K$ By definition, $L_0 = L$
$d(a, b)$	Distance from node a to node b, where $0 \leq a, b \leq K$ By definition, $d(a, a) = 0, \forall a$
$C = [c_{a,b}]$	Cost matrix of flights, where $c_{a,b} = c_d \times d(a, b)$
$M = \{1, \ldots, m\}$	Set of m pallets in specific positions within the aircraft See Table 3, where $m = 18$
$N_k = \{1, \ldots, n_k\}$	Set of n_k items available for loading at node k, $1 \leq j \leq n_k$, $0 \leq k \leq K$
$N = \bigcup_{0 \leq k \leq K} N_k$	Set of items in all nodes along a tour
$Q_k = \{1, \ldots, m_k\}$	Set of $m_k \leq m$ packed contents at node k, $1 \leq q \leq m_k$, $0 \leq k \leq K$ By definition, $m_0 = 0$ and $Q_0 = \varnothing$

Table 5 contains decision variables and the ACLP+RPDP allocation graph.

Table 5: Decision variables and allocation graph

Notation	Description
$X_{ij}^{\pi_k}$ and $Y_{iq}^{\pi_k}$	Binary variables, where $1 \leq i \leq m$, $1 \leq j \leq n_{\pi_k}$, $1 \leq q \leq m_{\pi_k}$ and $0 \leq k \leq K$
$X_{ij}^{\pi_k} = 1$	If item j at node π_k is assigned to pallet i, and 0 otherwise
$Y_{iq}^{\pi_k} = 1$	If packed content q at node π_k is assigned to pallet i, and 0 otherwise
$T_i^{\pi_k} \in L_{\pi_k}$	Destination of items and packed contents assigned to pallet i at node π_k
$G_{\pi_k}(V_{\pi_k}, E_{\pi_k})$	Allocation graph at node π_k
$V_{\pi_k} = M \cup N_{\pi_k} \cup Q_{\pi_k}$	Allocation graph vertices at node π_k: pallets, items and packet contents
$E_{N_{\pi_k}}$	Allocation graph edges at node π_k, corresponding to shipped items
$E_{Q_{\pi_k}}$	Allocation graph edges at node π_k, corresponding to packed contents
$E_{\pi_k} = E_{N_{\pi_k}} \cup E_{Q_{\pi_k}}$	Allocation graph edges at node π_k
$(i, j) \in E_{N_{\pi_k}}$	If $X_{ij}^{\pi_k} = 1$, where i is a pallet and j is a item at node π_k
$(i, q) \in E_{Q_{\pi_k}}$	If $Y_{iq}^{\pi_k} = 1$, where i is a pallet and q is a packed content at node π_k

The calculus functions of ACLP+RPDP are described in Table 6.

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

$$\tau_{\pi_k} = \sum_{i=1}^{m} \left[D_i^{long} \times \left(\sum_{j=1}^{n_{\pi_k}} X_{ij}^{\pi_k} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq}^{\pi_k} \times w_q \right) \right] \Big/ W_{max} \times limit_{long}^{CG};\ k \in \{0, \ldots, K\} \tag{2}$$

Table 6: Calculus functions

Function	Description
(1)	Total score of transported items throughout tour π
(2)	Longitudinal torque applied by loaded pallets at node π_k
(3)	Total cost of fuel on tour π (distances and CG longitudinal deviations)
(4)	Set of not visited nodes at node π_k
(5)	Lateral torque at node π_k (shipped items)
(6)	Lateral torque at node π_k (packed contents)
(7)	Objective function of ACLP+RPDP

$$\tilde{c}_\pi = \sum_{k=0}^{K} \Big[c_{\pi_k,\pi_{k+1}} \times (1 + c_g \times |\tau_{\pi_k}|) \Big] \tag{3}$$

$$L_{\pi_k} = L_{\pi_{k-1}} - \{\pi_k\};\ k \in \{1,\dots,K\} \tag{4}$$

$$\epsilon^t_{\pi_k} = \sum_{i=1}^{m} \Big[D^{lat}_i \times \sum_{j=1}^{n_{\pi_k}} \Big(X^{\pi_k}_{ij} \times w_j \times (i\%2) - X^{\pi_k}_{ij} \times w_j \times (i+1)\%2 \Big) \Big] \Big/ W_{max} \times limit^{CG}_{lat} \tag{5}$$

$$\epsilon^a_{\pi_k} = \sum_{i=1}^{m} \Big[D^{lat}_i \times \sum_{q=1}^{m_{\pi_k}} \Big(Y^{\pi_k}_{iq} \times w_q \times (i\%2) - Y^{\pi_k}_{iq} \times w_q \times (i+1)\%2 \Big) \Big] \Big/ W_{max} \times limit^{CG}_{lat} \tag{6}$$

$$\max_{\pi \in S_K} f_\pi = \tilde{s}_\pi / \tilde{c}_\pi \tag{7}$$

Longitudinal (2) and lateral torques (5, 6) are calculated in proportion to the highest torque supported by the aircraft. As there are two rows of pallets, one on each side of the centerline, we use the operator *modulo* (%) to calculate lateral torques. In our experiments, we found that the magnitude of these lateral torques was always minimal, so we decided to ignore them in the fuel consumption (3). The objective of ACLP+RPDP (7) is to find a permutation $\pi \in S_K$ with the corresponding allocation of items on pallets at each node that maximises the function $f_\pi = \tilde{s}_\pi / \tilde{c}_\pi$.

Finally, ACLP+RPDP constraints related to each node π_k are described in Table 7.

Table 7: Constraints

Constraint	Description
(8, 9)	Longitudinal and lateral torques must be within aircraft limits
(10, 11)	Items allocated to each pallet cannot exceed its weight and volume limits
(12)	At most, each item is associated with a single pallet
(13)	Packed contents that have not yet reached their destination must remain on board
(14, 15)	Items allocated on the same pallet must have the same destinations
(16, 17)	If there is a packed content on the pallet, it must also have the same destination as other items

$$|\tau_{\pi_k}| \leq 1;\ k \in \{0,\dots,K\} \tag{8}$$

$$|\epsilon^t_{\pi_k} + \epsilon^a_{\pi_k}| \leq 1 \tag{9}$$

$$\sum_{j=1}^{n_{\pi_k}} X^{\pi_k}_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y^{\pi_k}_{iq} \times w_q \leq W_i;\ i \in \{1,\dots,m\} \tag{10}$$

$$\sum_{j=1}^{n_{\pi_k}} X^{\pi_k}_{ij} \times v_j + \sum_{q=1}^{m_{\pi_k}} Y^{\pi_k}_{iq} \times v_q \leq\ V_i;\ i \in \{1,\dots,m\} \tag{11}$$

$$\sum_{i=1}^{m} X_{ij}^{\pi_k} \leq 1;\ j \in \{1,\ldots,n_{\pi_k}\} \tag{12}$$

$$\sum_{i=1}^{m} Y_{iq}^{\pi_k} = 1;\ to_q \in L_{\pi_k};\ q \in \{1,\ldots,m_{\pi_k}\} \tag{13}$$

$$X_{ij}^{\pi_k} <= X_{ij}^{\pi_k} \times (T_i^{\pi_k} - to_j + 1);\ i \in \{1,\ldots,m\};\ j \in \{1,\ldots,n_{\pi_k}\} \tag{14}$$

$$X_{ij}^{\pi_k} <= X_{ij}^{\pi_k} \times (to_j - T_i^{\pi_k} + 1);\ i \in \{1,\ldots,m\};\ j \in \{1,\ldots,n_{\pi_k}\} \tag{15}$$

$$Y_{iq}^{\pi_k} <= Y_{iq}^{\pi_k} \times (T_i^{\pi_k} - to_q + 1);\ i \in \{1,\ldots,m\};\ q \in \{1,\ldots,m_{\pi_k}\} \tag{16}$$

$$Y_{iq}^{\pi_k} <= Y_{iq}^{\pi_k} \times (to_q - T_i^{\pi_k} + 1);\ i \in \{1,\ldots,m\};\ q \in \{1,\ldots,m_{\pi_k}\} \tag{17}$$

Once the assumptions and the mathematical modelling are presented, it is possible to see that ACLP+RPDP is *NP-hard*. In a similar way to (Lurkin and Schyns, 2015), 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*. For more details, see (Lurkin and Schyns, 2015, p. 6).

5. Solution strategies

Throughout our research, we have thoughtfully described ACLP+RPDP in standard MIP format and found that no solver can handle its practical cases in a feasible time. Thus, as ACLP+RPDP is highly complex, involving four intractable and interconnected sub-problems, we decided to focus only on *real cases*, developing quick node-by-node solutions, not necessarily optimal, but which would allow us to obtain a complete tour.

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

- We consider that the number of destinations is smaller than the number of pallets, that is, $K < m$. We also avoid the trivial case where $K = 1$. With this premise, we can preset the destinations of the pallets at each shipping node, reserving for each destination a number of pallets proportional to the volume available. We could have used another criterion, but it was observed in the experiments that the volume is more constrictive in airlift.
- As the number of nodes is small, we have the possibility to check all possible tours, selecting the one that provides the best value for the objective function. For this reason, a maximum time per solution ($tmax$) is established. In the tests carried out in this work, these times were in the range [240s, 1200s, 2400s, 3600s].
- At each node, we will use a MIP solver or heuristics to check which method offers the best result within the time limit for each node.
- The time limit to solve a node will be adjusted to be proportional to the sum of volumes of the candidate items of the node in relation to the sum of volumes of the entire tour.

Our complete strategy is summarized in Algorithm 1.

In this algorithm, we use six values for *scenario*, according to Tables 1 and 8, which define the number K of destinations, the set L of nodes, the costs C, and the shortest tours π_{TSP1} and π_{TSP2}.

Scenario 6 will be used only for the best method validation.

Figure 3 depicts the Algorithm 1 for easy understanding of the main flow. (1.) Select a scenario from Table 8. (2.) Select a volume surplus (1.2, 1.5, or 2.0). (3.) Establish a time limit (240s, 1200s, 2400s, or 3600s). (4.) Configure the parameters presented in the mathematical formulation. (5.) Load the distance matrix and calculate the matrix cost according to the aircraft characteristics. (6.) Load the pallet configuration parameters. (7.) Load the item instance (all items to be transported) into memory. (8.) Choose a tour schema. Until 7 nodes, it is practical to generate K! tours. For a larger number of nodes, we

Algorithm 1 Solving the ACLP+RPDP

ACLP+RPDP *in: scenario, surplus, tmax, tsp, ntours* *out: answer*
Let M be the set of pallets (*cfr.* Table 3)
Let K, L and C be according to the *scenario* (*cfr.* Tables 1, 3 and 8)
$N \leftarrow ItemsGeneration(scenario, surplus)$
if tsp = True **then**
    $tmax \leftarrow tmax/ntours$ ▷ max. tour time to solve the number of tours from the TSP heuristic
else
    $tmax \leftarrow tmax/K!$
end if
    for each *method* **do**
        for each $\pi \in S_K$ **do**
            $f_\pi \leftarrow SolveTour(\pi, L, M, C, N, method, tmax)$
        end for
        $answer[scenario, surplus, method] \leftarrow \max f_\pi$
    end for

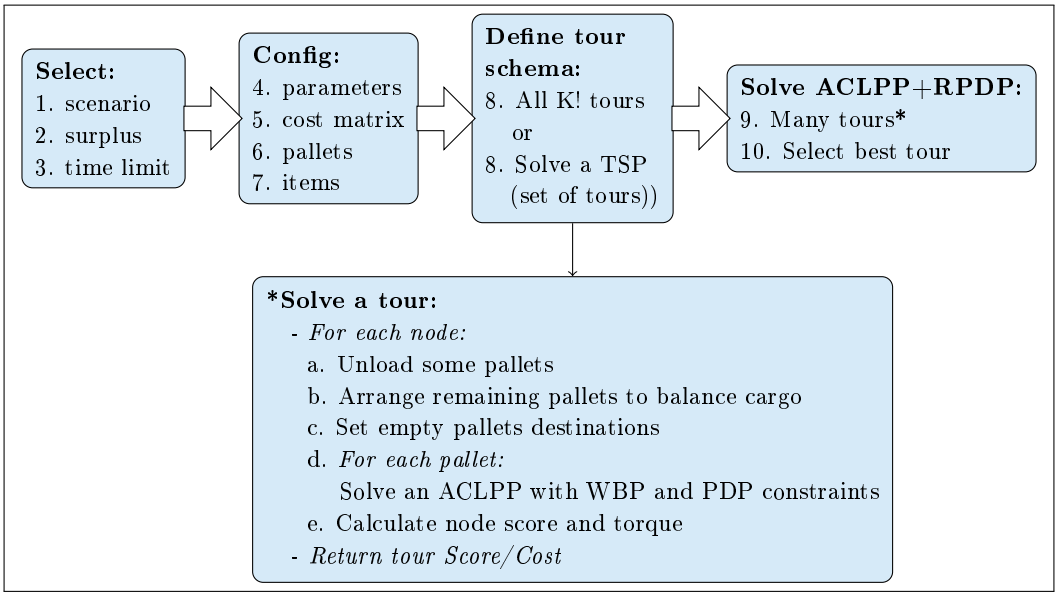


Figure 3: Solution main flow

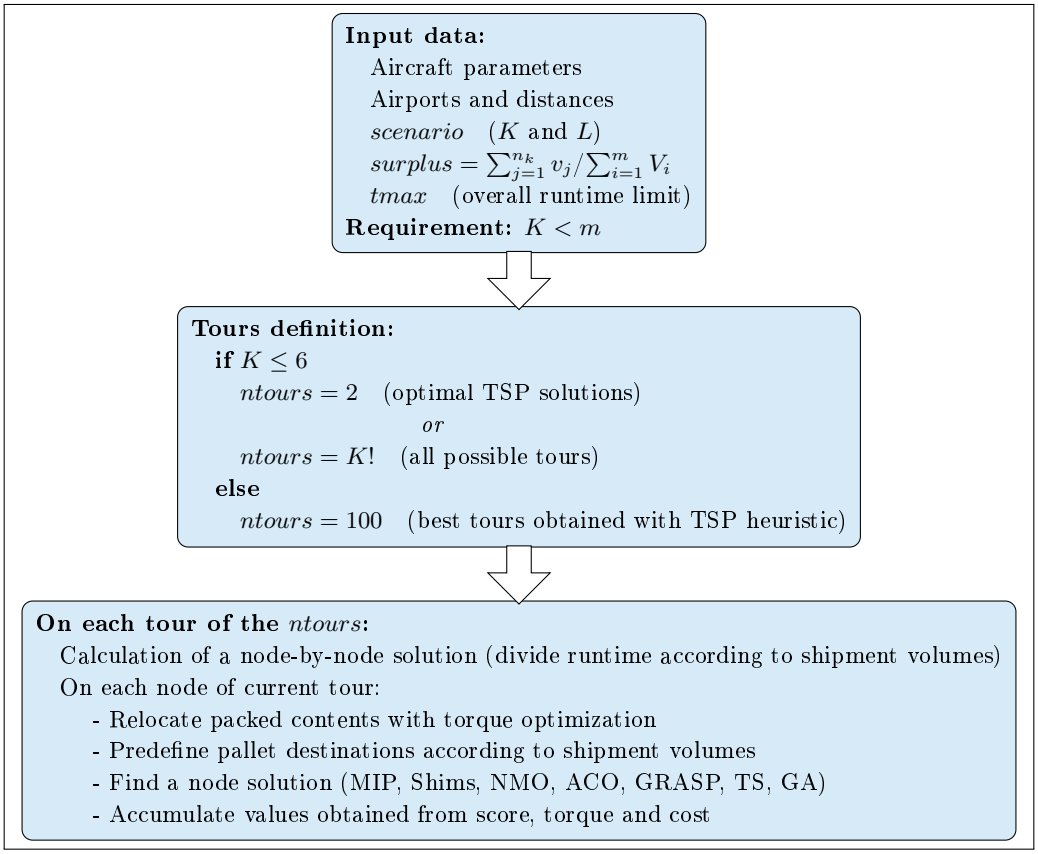


Figure 4: Solution process

Table 8: Testing scenarios

Scenario	K	L
1	2	$\{0, 1, 2\}$
2	3	$\{0, 1, 2, 3\}$
3	4	$\{0, 1, 2, 3, 4\}$
4	5	$\{0, 1, 2, 3, 4, 5\}$
5	6	$\{0, 1, 2, 3, 4, 5, 6\}$
6	15	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$

recommend using a TSP heuristic that generates a set of similar suboptimal tours. (9.) and (10.) solve all tours and select the best.

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 behaviour of each *method*, according to *scenario* and the quantity of items available for shipment. It is passed to *ItemsGeneration* (line 4), responsible for creating the items to be shipped, which will be presented in the next section (Algorithm 7).

tmax is the runtime limit, which will be distributed among the tours (lines **??**, **??** and 12). *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 obtained by testing all tours are stored in *answer* (line 14).

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 of the tour.

Algorithm 2 Solving the tour π with *method*

1: **SolveTour** *in:* $\pi, L, M, C, N, method, tmax$ *out:* $score/cost$
2: $\pi_0 \leftarrow 0$ ▷ all tours start and end at the base
3: $\pi_{K+1} \leftarrow 0$
4: $score \leftarrow 0$
5: $cost \leftarrow 0$
6: $V_\pi \leftarrow 0$ ▷ tour volume
7: **for** $k \leftarrow 0$ to K **do**
8: $v_{\pi_k} \leftarrow \sum_{j=1}^{n_k} v_j$
9: $V_k \leftarrow V_k + v_{\pi_k}$
10: **end for**
11: **for** $k \leftarrow 0$ to K **do**
12: $tnode_{\pi_k} = (v_{\pi_k}/V_\pi) * tmax$ ▷ node time limit calculation
13: $L_{\pi_k} \leftarrow L - \{\pi_0, \pi_1, \ldots, \pi_k\}$ ▷ the set of remaining nodes is updated
14: $T_i^{\pi_k} \leftarrow -1,\ 1 \le i \le m$ ▷ the pallet destination is unset
15: **if** $k = 0$ **then**
16: Let $G_1(M \cup N_0, \varnothing)$ ▷ no packed contents at the base
17: **else**
18: $E_{Q_{\pi_k}}, M \leftarrow UpdatePacked(M, Q_{\pi_k}, \pi_k)$
19: Let $G_1(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}})$
20: **end if**
21: $M \leftarrow SetPalletsDestinations(M, \pi_k)$
22: $G_2 \leftarrow SolveNode(method, \pi_k, G_1, tnode_{\pi_k})$
23: $s, \tau \leftarrow ScoreAndTorque(\pi_k, G_2)$
24: $score \leftarrow score + s$
25: $cost \leftarrow cost + c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau|)$
26: **end for**

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 11-26). Initially, the set L_{π_k} of remaining nodes is updated (line 13), and the pallet destinations are unset (line 14).

When the aircraft is at the base, the initial graph G_1 is empty, and there are no packed contents 16. Otherwise, *UpdatePacked* (line 18) 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 19).

SetPalletsDestinations (line 21) 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 21). The score and the CG deviation of G_2 are calculated (line 23) and accumulated (lines 24-25), allowing the final result of this tour as output.

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 node π_k

1: **UpdatePacked** *in:* M, Q_{π_k}, π_k *out:* $E_{Q_{\pi_k}}, M$
2: $E_{Q_{\pi_k}} \leftarrow MinCGDeviation(E_{Q_{\pi_k}})$
3: **for** $i \leftarrow 1$ to m **do**
4: **for** $q \leftarrow 1$ to m_{π_k} **do**
5: $T_i^{\pi_k} \leftarrow -1$
6: **if** $(i, q) \in E_{Q_{\pi_k}}$ **then**
7: $T_i^{\pi_k} \leftarrow to_q$ ▷ reassign pallet destinations
8: **end if**
9: **end for**
10: **end for**

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 (18) and the constraints (19) and (20). 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 3-10).

$$\min f = \left| \sum_{i=1}^{m} \sum_{q=1}^{m_{\pi_k}} Y_{iq}^k \times w_q \times D_i^{long} \right| \tag{18}$$

$$\sum_{i=1}^{m} Y_{iq}^k = 1;\ q \in \{1, \ldots, m_{\pi_k}\} \tag{19}$$

$$\sum_{q=1}^{m_{\pi_k}} Y_{iq}^k \leq 1;\ i \in \{1, \ldots, m\} \tag{20}$$

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 node π_k

```
1: SetPalletsDestinations   in: M, π_k   out: M
2: vol_x ← 0, x ∈ L_{π_k}
3: max ← 0                                   ▷ destination with maximum volume demand
4: total ← 0
5: for j ← 1 to n_{π_k} do
6:     if to_j ∈ L_{π_k} then
7:         vol_{to_j} ← vol_{to_j} + v_j
8:         total ← total + v_j
9:         if vol_{to_j} > vol_max then
10:            max ← to_j
11:        end if
12:    end if
13: end for
14: for x ∈ L_{π_k} do
15:    if vol_x ≠ 0 then
16:        needed ← max{1, ⌊(m − m_{π_k}) × vol_x/total⌋}
17:        np ← 0
18:        for i ← 1 to m do
19:            if (np < needed) and (T_i^{π_k} = −1) then
20:                T_i^{π_k} ← x
21:                np ← np + 1              ▷ number of necessary pallets to node x
22:            end if
23:        end for
24:    end if
25: end for
26: for i ← 1 to m do
27:    if T_i^{π_k} ← −1 then
28:        T_i^{π_k} ← max   ▷ any remaining pallet is assigned to the maximum demand destination
29:    end if
30: end for
```

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 14-25). *max* is the destination with maximum volume demand (line 10), and *needed* is the number of necessary pallets to node x (line 16). The destination with the maximum volume defines any remaining pallets (lines 26-30).

ScoreAndTorque, described in Algorithm 5, evaluates the allocation graph G generated by *SolveNode* at node π_k and returns the corresponding cargo score and aircraft torque.

Algorithm 5 Cargo score and aircraft torque

1: **ScoreAndTorque** *in:* π_k, G *out:* s, τ
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 **do**
6: **for** $j \leftarrow 1$ to n_{π_k} **do**
7: **if** $X_{ij}^{\pi_k} = 1$ **then**
8: $s \leftarrow s + s_j$ ▷ accumulates cargo score
9: $\tau_i \leftarrow \tau_i + w_j \times D_i^{long}$ ▷ accumulates aircraft torque
10: **end if**
11: **end for**
12: **for** $q \leftarrow 1$ to m_{π_k} **do**
13: **if** $Y_{iq}^{\pi_k} = 1$ **then**
14: $s \leftarrow s + s_q$ ▷ accumulates cargo score
15: $\tau_i \leftarrow \tau_i + w_q \times D_i^{long}$ ▷ accumulates aircraft torque
16: **end if**
17: **end for**
18: **end for**
19: $\tau \leftarrow \sum_{i=1}^{m} \tau_i / (W_{max} \times limit_{long}^{CG})$ ▷ final calculation of the aircraft torque

Algorithm 5 consists of a loop that goes through all the pallets (lines 5-18), accumulating the scores (lines 8 and 14) and the torques (lines 9 and 15) of the shipped items, allowing the final calculation of the aircraft torque (line 19).

5.2. Node-by-node solutions

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

5.2.1. Node-by-node solutions 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 (21) with the calculus equations (22) to (24), subject to the constraints (25) to (31). The binary variables X_{ij} and Y_{iq} define the sets of edges $E_{N_{\pi_k}}$ and $E_{Q_{\pi_k}}$, respectively, included in graph G.

$$\max f = \tilde{s}/\tilde{c} \tag{21}$$

$$\tilde{s} = \sum_{i=1}^{m} \sum_{j=1}^{n_{\pi_k}} X_{ij} \times s_j \tag{22}$$

$$\tau_{\pi_k} = \sum_{i=1}^{m} \left[D_i^{long} \times \left(\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q \right) \right] \Big/ W_{max} \times limit_{long}^{CG} \tag{23}$$

$$\tilde{c} = c_{\pi_k, \pi_{k+1}} \times (1 + c_g \times |\tau_{\pi_k}|) \tag{24}$$

$$|\tau_{\pi_k}| \leq 1 \tag{25}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times w_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times w_q \leq W_i;\ i \in \{1, \ldots, m\} \tag{26}$$

$$\sum_{j=1}^{n_{\pi_k}} X_{ij} \times v_j + \sum_{q=1}^{m_{\pi_k}} Y_{iq} \times v_q \leq V_i;\ i \in \{1,\ldots,m\} \tag{27}$$

$$\sum_{i=1}^{m} X_{ij} \leq 1;\ j \in \{1,\ldots,n_{\pi_k}\} \tag{28}$$

$$X_{ij} = 0;\ to_j \notin L_{\pi_k};\ i \in \{1,\ldots,m\};\ j \in \{1,\ldots,n_{\pi_k}\} \tag{29}$$

$$X_{ij} \leq X_{ij} \times (T_i^{\pi_k} - to_j + 1);\ i \in \{1,\ldots,m\};\ j \in \{1,\ldots,n_{\pi_k}\} \tag{30}$$

$$X_{ij} \leq X_{ij} \times (to_j - T_i^{\pi_k} + 1);\ i \in \{1,\ldots,m\};\ j \in \{1,\ldots,n_{\pi_k}\} \tag{31}$$

The constraints (30) and (31) are equivalent to $X_{ij} = 1$ if $to_j = T_i^{\pi_k}$, and $X_{ij} = 0$ otherwise .

5.2.2. Node-by-node solutions with heuristics

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 four known meta-heuristics: *Ant Colony Optimization* (ACO) (Dorigo, 1992; Dorigo et al., 1996), *Noising Method Optimization* (NMO) (Charon and Hudry, 1993, 2001; Zhan et al., 2020), *Tabu Search* (TS) (Glover, 1986), *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo and Resende, 1989), and the Genetic Algorithm (GA) (Holland, 1992). We considered several ideas from the literature (Niar and Freville, 1997; Fidanova, 2006; Alonso et al., 2019; Zhan et al., 2020; Shah, 2020), and we were careful to use the same data structures and procedures in all implementations to enforce fair results comparison.

As (Peerlinck and Sheppard, 2022) applied the DEAP (Distributed Evolutionary Algorithms in Python) to solve a Multi-Objective Knapsack Problem, we also decided to apply this evolutionary package to implement a GA solution for the ACLPP.

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} (32), 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.

$$\theta_{ij} = \frac{s_j}{v_j} \times \left(1 - \frac{w_j \times |D_i^{long}|}{\max w_j \times \max |D_i^{long}|}\right);\ i \in \{1,\ldots,m\},\ j \in \{1,\ldots,n_{\pi_k}\} \tag{32}$$

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

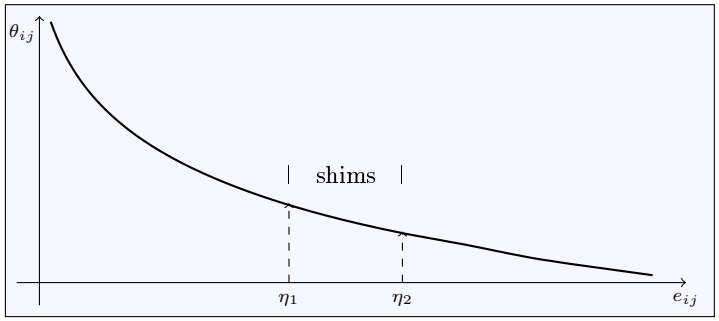


Figure 6: n_{π_k} possible edges e_{ij} sorted by θ_{ij} in non-ascending order

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 (*greedy phase*). Then, with the edges between η_1 and η_2, it elaborates different possible complements (*composition phase*), including later the best ones in the same pallet (*selection phase*). *Shims* is depicted in Algorithm 6.

Algorithm 6 *Shims* heuristic at node π_k

SolveNode *in:* $Shims, \pi_k, G, tmax, threshold_1, threshold_2$ *out:* $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}} \cup E_{N_{\pi_k}})$
$T_{begin} \leftarrow$ current system time
Let $G(M \cup N_{\pi_k} \cup Q_{\pi_k}, E_{Q_{\pi_k}})$
Sort M by $|D_i^{long}|$ in non-descending order
$E_{N_{\pi_k}} \leftarrow \varnothing$
$\tau_{max} \leftarrow W_{max} \times limit_{long}^{CG}$
for $i \leftarrow 1$ to m **do**
    $\tau_{\pi_k} \leftarrow \sum_{(i,q) \in E_{Q_{\pi_k}}} w_q \times D_i^{long}$
    $vol_i \leftarrow \sum_{(i,q) \in E_{Q_{\pi_k}}} v_q$
    Let E be an array of n_{π_k} possibles edges of the pallet i sorted by θ_{ij} in non-ascending order
    $\eta_1 \leftarrow 1$
    repeat
        $e_{ij} \leftarrow E_{\eta_1}$
        if ($E_{N_{\pi_k}} \cup \{e_{ij}\}$ is feasible) **and** ($vol_i \leq V_i \times threshold_1$) **and** ($|\tau_{\pi_k} + w_j \times D_i^{long}| \leq W_{max} \times limit_{long}^{CG}$) **then**
            $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup \{e_{ij}\}$
            $vol_i \leftarrow vol_i + v_j$
            $\tau_{\pi_k} \leftarrow \tau_{\pi_k} + w_j \times D_i^{long}$
            $\eta_1 \leftarrow \eta_1 + 1$
        end if
    until ($vol_i > V_i \times threshold_1$) **or** ($\eta_1 > n_{\pi_k}$)
    $slack_i \leftarrow V_i - vol_i$
    $\eta_2 \leftarrow \eta_1$
    while ($\eta_2 \leq n_{\pi_k}$) **and** ($vol_i < V_i \times threshold_2$) **do**
        $e_{ij} \leftarrow E_{\eta_2}$
        $vol_i \leftarrow vol_i + v_j$
        $\eta_2 \leftarrow \eta_2 + 1$
    end while
        $vol \leftarrow 0$; $b \leftarrow 1$; $shims_b \leftarrow \varnothing$; $Set \leftarrow \{shims_b\}$
        for $x \leftarrow \eta_1$ to η_2 **do**
            if $T_{current} - T_{begin} > tmax$ **then**
                break
            end if
            $NewShims \leftarrow$ **True**
            $e_{ij} \leftarrow E_x$
            for $shims \in Set$ **do**
                if ($e_{ij} \notin (E_{N_{\pi_k}} \cup shims)$) **and** ($e_{ij}$ is feasible) **and** ($(v_j + vol) \leq slack_i$) **then**
                    $shims \leftarrow shims \cup \{e_{ij}\}$
                    $vol \leftarrow vol + v_j$
                    $NewShims \leftarrow$ **False**
                    break
                end if
            end for
            if $NewShims$ **then**
                $vol \leftarrow 0$; $b \leftarrow b + 1$; $shims_b \leftarrow \{e_{ij}\}$
                $Set \leftarrow Set \cup \{shims_b\}$
            end if
        end for
        $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ij} \in shims} w_j$ is maximum
        $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ij} \in shims} v_j$ is maximum
        $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ij} \in shims} s_j$ is maximum
        $E_{N_{\pi_k}} \leftarrow E_{N_{\pi_k}} \cup sh_{best}$
    end for

The arguments $threshold_1$ and $threshold_2$ are the volume thresholds for η_1 and η_2 calculations, respectively. The argument $tmax$ is the maximum time for *Shims* to solve a node. It is proportional to the sum of

the candidate items volumes in each node, in relation to the sum of all nodes volumes, and the maximum time to solve the tour.

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 the procedure $UpdatePacked(M, Q_{\pi_k}, \pi_k)$ and the procedure $SetPalletsDestinations(M, \pi_k)$. 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 the *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 $V_i \times threshold_1$. In η_2 index, this same accumulated volume reaches $V_i \times threshold_2$. The values of these thresholds were defined empirically by the *irace* tool (Lopez-Ibanez et al., 2016).
- In the *composition phase* (lines 21-27), 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* (Johnson and Garey, 1985). Basically, shims are created by accumulating the following edges, taking $slack_i$ as a limit.
- In the *selection phase* (lines 28-51), 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 $E_{N_{\pi_k}}$.

irace (Iterated Racing for Automatic Algorithm Configuration) streamlines the tuning of optimization algorithms by iteratively testing and eliminating weaker parameter configurations through statistical comparisons. Its application may be understood, from a Machine Learning standpoint, as the task of identifying optimal parameter values for solving new problem instances by leveraging knowledge gained from a set of training problem examples. This method significantly cuts down on the time and effort needed to find optimal or near-optimal parameters, proving effective across various application domains.

6. Implementation and results

This section is composed of two parts: the generation of the 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 modelled 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 delivery of supplies 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 9, where P represents the group probability. On the other hand, the average density of these items is approximately $246kg/m^3$.

Table 9: Items weight distribution

x	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 two types of random selections:

- $RandomInt(i_1, i_2)$: randomly selects a integer number in $[i_1, i_2]$, where i_1 and i_2 are integer numbers;
- $Roulette()$: a proportional selection function that is biased through P (Table 9).

The procedure *ItemsGeneration*, which generates N (all items to be moved among the nodes), is described in Algorithm 7.

Algorithm 7 Generating items

1: **ItemsGeneration** *in: scenario, surplus* *out: N*
2: Let L be the set of nodes and M the set of pallets
3: $limit \leftarrow surplus \times \sum_{i=1}^{m} V_i$
4: **for** $k \leftarrow 0$ to K **do**
5: $N_k \leftarrow \varnothing$
6: $j \leftarrow 0$
7: $vol \leftarrow 0$
8: **while** $vol < limit$ **do**
9: $j \leftarrow j + 1$
10: Let t_j^k be the item j at the node k
11: **repeat**
12: $to_j \leftarrow RandomInt(0, K)$
13: **until** $to_j \neq k$
14: $x = Roulette()$ ▷ biased through P (*cfr.* Table 9)
15: $w_j \leftarrow RandomInt(low(x), high(x))$
16: $s_j \leftarrow \lfloor 100 \times (1 - \log_{10}(RandomInt(1, 9)))\rceil$
17: $v_j \leftarrow w_j / RandomInt(148, 344)$
18: $vol \leftarrow vol + v_j$
19: $N_k \leftarrow N_k \cup \{t_j^k\}$
20: **end while**
21: **end for**
22: $N \leftarrow \bigcup_{0 \leq k \leq K} N_k$

scenario defines L and M (line 2), and the argument *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 9 (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% around the average density of $246 kg/m^3$ (line 17).

After the items were generated, some experiments with the *irace* tool were executed to determine the values for parameters $threshold_1$ and $threshold_2$ used in *Shims*, whose results are presented in Table 10.

Table 10: *irace* testing results

surplus	***threshold*$_1$** for η_1	***threshold*$_2$** for η_2	**run time** (min)
1.2	0.8621	1.0539	47
1.5	0.9199	1.1399	59
2.0	0.9617	1.5706	63

For these tests, we considered the odd instances (1, 3, 5, 7) as the **training** set and the even instances (2, 4, 6) as the **testing** set for the *irace* runs. We supplied *irace* for $threshold_1$ determination, the range [0.8, 1.0], and for $threshold_2$, the range [1.0, 2.0]. The number of iterations in each experiment was limited to 3000 executions for *irace* to have plenty of data for its statistical tests. More detail about these configurations may be found on the *irace* user guide in `cran.r-project.org/web/packages/irace/irace.pdf`.

6.2. Obtained solutions: quality and runtimes

In the tests performed, we used a 64-bit, 16 GB, 3.6 GHz, 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 the five scenarios from Table 8, three values for *surplus* from $\{1.2, 1.5, 2.0\}$, four values for *tmax* from $\{240s, 1200s, 2400s, 3600s\}$, and seven different methods for the node-by-node solution: *Gurobi* (see 5.2.1), ACO, NMO, TS, GRASP, GA 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 `www.support.gurobi.com`.

Table 11: Overall results

Method	Best scenarios	Worst scenarios	Worst run times (min)
NMO	4	5	60
ACO	2, 3	4, 5	25, 61
GRASP	1	4, 5	28, 55
TS	-	5	44
GA	-	1,2,3,4,5	did not solve
Gurobi	1, 2, 3, 4	5	did not solve
Shims	1, 2, 3, 4, 5	-	**3.26**

As to the genetic algorithm exclusion from the comparisons tests, we must consider that:

- The palletization planning of 200 to 1000 items per node in a set of 18 aircraft pallets may be an inadequate problem to be solved with GA, because large chromosome sizes, like 18,000-dimension arrays, can pose significant challenges for each generation to evaluate individual fitness, which demonstrated to be resource-intensive.
- Large chromosomes can slow GA convergence due to the vast search space, requiring more generations to find optimal solutions.
- Efficient encoding and decoding of the problem into the chromosome and decoding solutions is crucial for a 18,000-dimension array.
- Also, the design of the fitness function becomes more critical as chromosome size increases, as it needs to guide the evolutionary process without being overly computationally demanding.

6.2.1. Shims results

Among the heuristics, we will present only the results obtained by *Shims*, which had the best performance. For each *scenario*, *surplus* and *tmax*, seven 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 $K!$ tours.

The results obtained with $tmax = 3600s$, which is the highest tested runtime limit, are in Tables 12, 13 and 14, with *surplus* values of 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 random-access memory (RAM) usage.

Table 12: Solutions with $surplus = 1.2$ and $tmax = 3600s$

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

From these data, we can draw some conclusions:

- The strategy of testing all $K!$ tours often provide a better-quality solution, even with less time on each node. This shows that the four sub-problems are interconnected in such a way that it is not enough to solve them separately.

Table 13: Solutions with $surplus = 1.5$ and $tmax = 3600s$

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

Table 14: Solutions with $surplus = 2.0$ and $tmax = 3600s$

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

- *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 99.9% of that value.
- *Shims* always finds a solution, being 8 to 41 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. On the other hand, when $scenario = 5$ and $surplus = 1.5$, *Gurobi* spent almost 40 minutes.

Table 15 shows the results obtained with the strategy of testing the $K!$ tours in all scenarios with different $tmax$. We can observe more cases where *Gurobi* fails, even in smaller scenarios. When *Gurobi* finishes, *Shims* finds a solution of similar quality (99% or better). In all cases, *Shims* finds a good solution in less than 4 minutes.

Table 15: 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	8.60	12.20	13.67	x	x	11.77	16.38	18.10	x	x	17.89	25.42	x	x	x
		time (s)	31	35	124	x	x	52	59	200	x	x	188	145	x	x	x
	1200s	f	8.61	12.20	13.31	10.00	x	11.77	17.02	18.25	20.64	x	17.75	25.24	26.49	27.97	x
		time (s)	28	37	129	320	x	46	61	190	304	x	161	139	384	579	x
	2400s	f	8.60	12.21	13.67	15.00	13.41	11.77	16.37	18.03	20.95	x	17.89	25.44	26.15	29.13	x
		time (s)	26	38	134	310	1520	46	60	199	461	x	164	140	383	786	x
	3600s	f	8.60	12.20	13.66	15.00	x	11.76	16.37	18.01	20.95	17.60	17.90	25.44	26.15	29.13	x
		time (s)	30	35	123	314	x	64	58	195	472	2258	178	143	378	862	x
Shims	240s	f	8.61	12.20	13.46	14.99	13.35	11.78	16.29	17.99	20.93	17.50	17.94	25.45	26.14	28.84	26.22
		time (s)	1	2	4	10	51	1	2	5	15	100	1	3	10	31	196

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

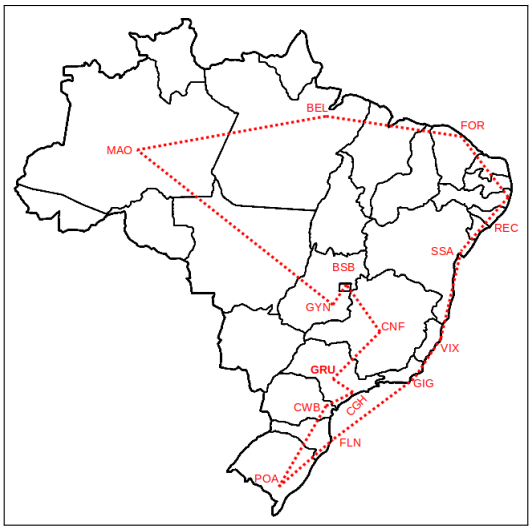
6.3. Shims validation with more nodes

To validate the effectiveness of the *Shims* heuristic, we solve a 15-node TSP containing the 15 main Brazilian airports. For this task, we implemented a TSP solution method based on the Genetic Algorithm

(GA) that returns a population of 100 tours as input to the ACLP-RPDP solved by *Shims*. The GA run time was around three seconds, and the total run time was around 33 seconds.

We implemented the GA with DEAP (Distributed Evolutionary Algorithms in Python) , a evolutionary computation framework available in `github.com/deap/deap`. More details can be found in (Fortin et al., 2012).

Figure 7: *Shims/GA* best ACLP+RPDP tour
Solution for scenario 7, instance 1



The airports cited in Figure 7 are: São Paulo (GRU), Congonhas (CGH), Curitiba (CWB), Porto Alegre (POA), Florianópolis (FLN), Vitória (VIX), Salvador (SSA), Recife (REC), Fortaleza (FOR), Belém (BEL), Manaus (MAO), Brasília (BSB), Goiânia (GYN), Confins (CNF), and Rio de Janeiro (GIG), returning to GRU.

It is important to remember that this is not the shortest tour, but the one with the best relationship with score, cost, and fuel consumption.

It is also important to highlight that three strategies were essential for us to obtain a solution in the instances created:

(1) consider $K < m$ so that each pallet received a predefined destination;
(2) on each tour, calculate a node-by-node solution; and
(3) at each intermediate node of the considered tour, relocate the packed contents, minimizing the torque on the aircraft.

Without adopting these three strategies, we were unable to carry out any experiments, even in smaller instances.

Table 16: Shims solutions with $surplus = 1.2$ and $tmax = 240s$

scenario	**7**	**8**	**9**
Num. nodes	15	14	13
f	50.15	47.00	47.18
time (s)	32	31	29
Occupancy rate	0.89	0.84	0.88

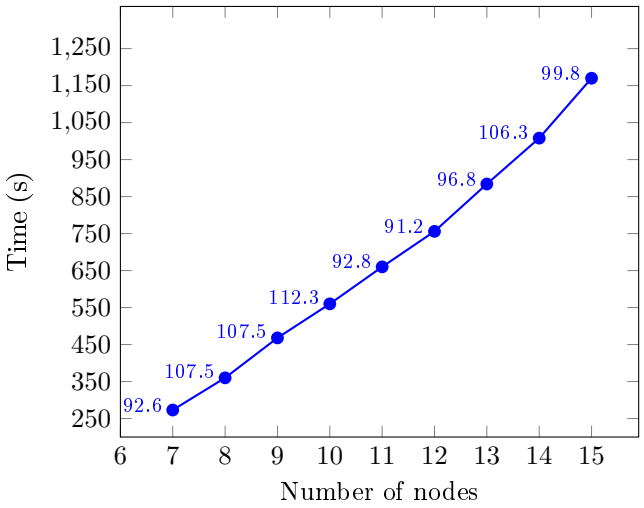


Figure 8: Shims performance

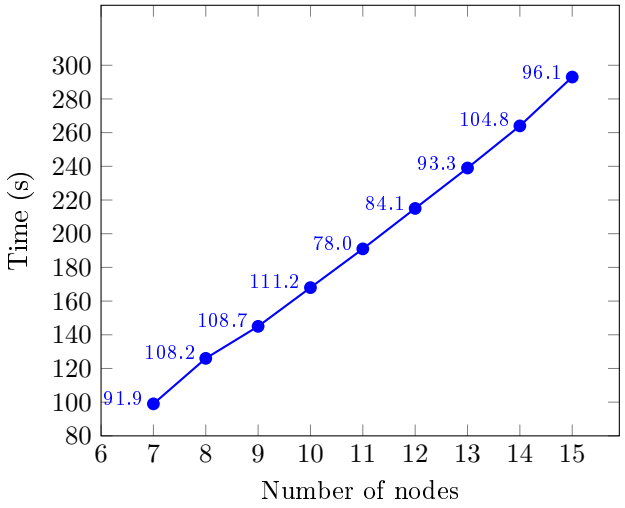


Figure 9: Shims performance (new time computation)

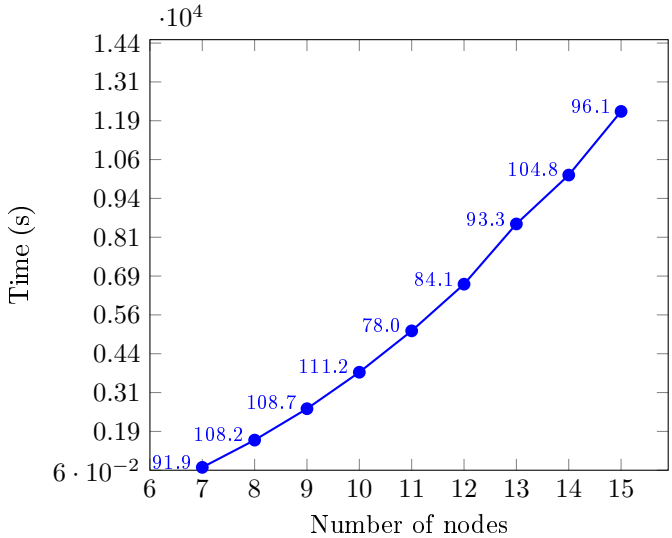


Figure 10: Shims performance (total time)

7. Conclusions

In this work, we modelled 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. Currently, there is no commercial software available for this problem.

We adopted some simplifications that are not critical, but that allowed for an unprecedented solution to this problem considering more than two nodes. In practical cases, the number K of nodes, excluding the base, is small ($K \leq 6$), each of which has hundreds of items to be shipped. Considering a real aircraft, we have developed some node-by-node solutions. The complete process can be executed quickly on a handheld computer, offering good results and reducing stress for the transport planners.

As validation, we carried out tests in several real air transport scenarios. In these cases, the solution process can establish, in less than four minutes, a flight tour for a single aircraft with a good distribution of load on pallets to be put in the cargo bay, enforcing the loaded aircraft balance, maximising the total score, and minimising fuel consumption along the planned route, which is beneficial to reduce carbon emissions. This output is an essential part of airlift: it guarantees flight safety, makes ground operations more efficient, and makes sure that each item gets to its right destination.

Our main contributions were the mathematical modelling of ACLP+RPDP, involving four well-known and interconnected *NP-hard* sub-problems, a complete process to solve it, and a new heuristic named *Shims* that offers fast node solutions with good quality. The method of this work is not exclusive to aircraft and airports: it can be adapted 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.

Because we applied this heuristic process to all $K!$ tours, perhaps it seems that our methodology is not generalizable. This is not true; we did this because *Shims* is very fast and K is small in practical cases of air transport. In other contexts where K may have larger values, this heuristic process remains valid. It is enough to apply it only to some tours that offer a better expectation of a good result, for example, on the TSP's shortest path or other similar ones.

As this is ongoing research, we thought about some possible future improvements: consider more than one aircraft, implement parallel algorithms in some steps of the solution to improve computational efficiency, and model 3-D items.

CRediT authorship contribution statement

Antonio Celio Pereira de Mesquita: Conceptualization, Methodology, Software, Writing - original draft preparation, Investigation, Validation. **Carlos Alberto Alonso Sanches:** Conceptualization, Methodology, Resources, Supervision, Writing - reviewing & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset and code used in this work are available in `github.com/celiomesquita/ACLP_RPDP_P`.

Acknowledgments

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

References

References

Alonso, M. T., Alvarez-Valdes, R., and Parreno, F. (2019). A grasp algorithm for multi-container loading problems with practical constraints. *A Quarterly Journal of Operations Research*, 18:49–72.

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

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

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

Charon, I. and Hudry, O. (1993). The noising method: a new method for combinatorial optimization. *Operations Research Letters*, 14(3):133–137.

Charon, I. and Hudry, O. (2001). The noising methods: A generalization of some metaheuristics. *European Journal of Operational Research*, 135(1):86–101.

Chenguang, Y., Hu, L., and Yuan, G. (2018). Load planning of transport aircraft based on hybrid genetic algorithm. *MATEC Web of Conferences*, 179:1–6.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano.

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

Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71.

Fidanova, S. (2006). *Ant Colony Optimization and Multiple Knapsack Problem*, volume Chapter 33. J–Ph. Renard editor.

Fok, K. and Chun, A. (2004). Optimizing air cargo load planning and analysis. In *Proceedings of the International Conference on Computing, Communications and Control Technologies*, pages 520–531.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549.

Heidelberg, K. R., Parnell, G. S., and Ames, J. E. (1998). Automated air load planning. *Naval Research Logistics*, 45(8):751–768.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Johnson, D. S. and Garey, M. R. (1985). A 7160 theorem for bin packing. *Journal of Complexity*, 1(1):65–106.

Kaluzny, B. L. and Shaw, R. H. A. D. (2009). Optimal aircraft load balancing. *International Transactions in Operational Research*, 16(6):767–787.

Larsen, O. and Mikkelsen, G. (1980). An interactive system for the loading of cargo aircraft. *European Journal of Operational Research*, 4(6):367–373.

Limbourg, S., Schyns, M., and Laporte, G. (2012). Automatic aircraft cargo load planning. *Journal of the Operational Research Society*, 63(9):1271–1283.

Lopez-Ibanez, M., Dubois-Lacoste, J., Cáceres, L., Birattari, M., and Stützle, T. (2016). The irace package: iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Lurkin, V. and Schyns, M. (2015). The airline container loading problem with pickup and delivery. *European Journal of Operational Research*, 244(3):955–965.

Miguel, J., Macalintal, V., and Ubando, A. T. (2023). Optimal aircraft payload weight and balance using fuzzy linear programming model. volume 103, pages 613–618.

Mongeau, M. and Bes, C. (2003). Optimization of aircraft container loading. *IEEE Transaction on Aerospace and Electronic Systems*, 39(1):140–150.

Ng, K. Y. K. (1992). A multicriteria optimization approach to aircraft loading. *Operations Research*, 40(6):1200–1205.

Niar, S. and Freville, A. (1997). A parallel tabu search algorithm for the 0-1 multidimensional knapsack problem. In *Proceedings 11th International Parallel Processing Symposium*, pages 512–516.

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

Paquay, C., Schyns, M., Limbourg, S., and Oliveira, J. F. (2018). MIP-based constructive heuristics for the three-dimensional Bin Packing Problem with transportation constraints. *International Journal of Production Research*, 56(4):1581–1592.

Peerlinck, A. and Sheppard, J. (2022). Multi-objective factored evolutionary optimization and the multi-objective knapsack problem. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Roesener, A. and Barnes, J. (2016). An advanced tabu search approach to the dynamic airlift loading problem. *Logistics Research*, 9(1):1–18.

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

Shah, S. (2020). Genetic algorithm for the 0/1 multidimensional knapsack problem.

van Es, G. (2007). Analysis of aircraft weight and balance related safety occurrences. Technical Report NLR-TP-2007-153, National Aerospace Laboratory (NLR). (RR 95-02).

Vancroonenburg, W., Verstichel, J., Tavernier, K., and Vanden Berghe, G. (2014). Automatic air cargo selection and weight balancing: A mixed integer programming approach. *Transportation Research Part E: Logistics and Transportation Review*, 65:70–83.

Verstichel, J., Vancroonenburg, W., Souffriau, W., and Berghe, G. V. (2011). A mixed integer programming approach to the aircraft weight and balance problem. *Procedia Social and Behavioral Sciences*, 20:1051–1059.

Wong, E. Y. and Ling, K. K. T. (2020). A mixed integer programming approach to air cargo load planning with multiple aircraft configurations and dangerous goods. In *7th International Conference on Frontiers of Industrial Engineering (ICFIE)*, pages 123–130.

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

Zhan, S., Wang, L., Zhang, Z., and Zhong, Y. (2020). Noising methods with hybrid greedy repair operator for 0-1 knapsack problem. *Memetic Computing*, 12:37–50.

Zhao, X., Dong, Y., and Zuo, L. (2023). A combinatorial optimization approach for air cargo palletization and aircraft loading. *Mathematics*, 11(13):1–16.

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