Parallel air palletization and tour planning for simultaneous pickup and delivery

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

Parallelism is crucial for mobile computers to be capable of doing complicated operations while conserving energy. By leveraging parallelism, handheld computers can divide tasks into numerous smaller sections and execute these parts simultaneously, so utilizing multiple cores of the processor and offering a substantial performance improvement. This enables for more complicated applications and speedier performance, making mobile computers more competent than ever before.

In this work, we explore computer parallelism in solving the problem involving scheduling the loading and routing of an airplane in a tour of simultaneous pickup and delivery at intermediate hubs while taking into account a utility score, weight and balance rules, and fuel usage.

This hard problem, named *Air Cargo Load Planning with Routing, Pickup, and Delivery Problem* considers using standardized pallets in fixed positions, obeying the center of gravity constraints, delivering each item to its destination, and minimizing fuel consumption costs.

We also contributed by carrying out multiple experiments with the Shims heuristic on synthetic data based on real data from the *Brazilian Air Force* transportation history and a new procedure to minimize the distance from the next node destined pallet to the ramp door.

We converted Shims into a process-based parallel computing heuristic that quickly finds good solutions for a wide range of problem sizes, an essential contribution as it surpassed all Shims results and some of the integer programming results, all methods under a time limit to guarantee that a distribution plan is yielded in less than one hour.

Keywords: Air Cargo, Air Palletization, Weight and Balance, Pickup and Delivery, Vehicle Routing, Parallel computing

1. Introduction

Air cargo transport involves several sub-problems that are difficult to solve. Recently, Mesquita and Sanches (2023) modeled and solved the *Air Cargo Load Planning with Routing, Pickup and Delivery* (ACLP+RPDP) composed by four sub-problems: *Build-up Scheduling Problem*, *Air Palletization Problem* (APP), *Weight and Balance Problem* (WBP), and a special case of the *Traveling Salesman Problem*, similar to the proposed by Kaspi et al. (2019), where the profit per unit time is defined as the profit gained after completing the tour divided by the total time (cost in our case) required to complete the tour.

However, there are still other important challenges in air cargo transport that go beyond the definition of the ACLP+RPDP, especially with regard to algorithms performance and the easiness of loading operations at each destination.

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

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

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

	APP	WBP	SPDP	TSP	2D	3D	Heu	Int	Lin	Parallel heuristics
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)	.	★	.	.	.	.	.	★	.	.
Mesquita and Cunha (2011)	.	.	★	.	.	.	★	.	.	.
Limbourg et al. (2012)	.	★	.	.	.	.	.	★	.	.
Kaspi et al. (2019)	.	.	.	★	.	.	★	.	.	.
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)	.	★	.	.	.	.	.	★	.	.
Mesquita and Sanches (2023)	★	★	★	★	.	.	★	★	.	.
This work	★	★	★	★	.	.	★	★	.	★

As can be seen, so far Lurkin and Schyns (2015) is the only work that simultaneously addresses an air cargo (WBP) and a flight itinerary (PDP) sub-problem. Although it is innovative, strong simplifications were imposed by the authors: in relation to loading, APP was ignored; with regard to routing, it is assumed a pre-defined flight plan restricted to two legs. It is important to note that these authors consider an aircraft with two doors, and the minimization of loading and unloading costs at the intermediate node was modelled through a container sequencing problem. Referring directly to this work, (Brandt and Nickel, 2019, p. 409) comment: *However, not even these sub-problems are acceptably solved for real-world problem sizes or the models omit some practically relevant constraints.*

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

Table 2: Brazilian airports distances (km)

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

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

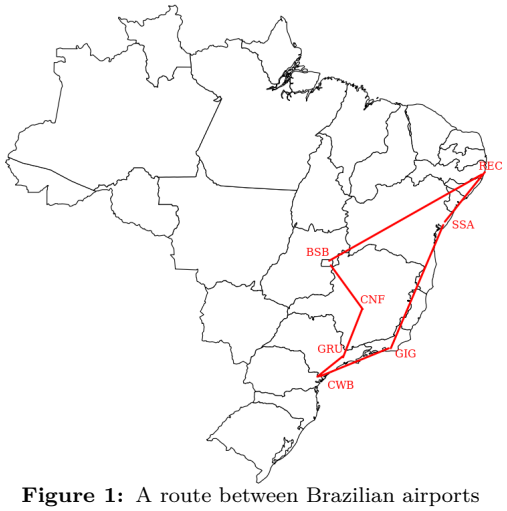

Figure 1: A route between Brazilian airports

In the *Brazilian Air Force* missions, hundreds of items can be carried at each node, where the objectives are to prioritize the transport of the most important items and minimize the cost of fuel along the route. As standardized pallets are used with predefined positions on the aircraft, it is possible to carry out loading and

unloading at each node in around two hours. However, there is no technological assistance that guarantees the achievement of these objectives.

Mesquita and Sanches (2023) proposed a method that attends these objectives: a pallet building and arrangement plan with a routing option that maximizes the benefit-cost ratio for the smooth execution of pickup and delivery transport missions. They develop a heuristic that can be executed on a simple handheld computer (like a laptop or a tablet) and that provides a solution quickly enough to keep this cargo handling time under an hour. This work seeks to improve their results in terms of quality and performance by the use of computer parallelism and by minimizing the distance between the next node destined pallets and the cargo door.

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

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, the mathematical model and how we dealt with its issues. In Section 5, we describe the elaborate algorithms, whose results are presented in Section 6. Finally, our conclusions are in Section 7.

2. Related literature

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

Larsen and Mikkelsen (1980) developed an interactive procedure for loading 14 types of Boeing 747 into a two-leg flight plan. Seven types of items were considered to be allocated in 17 to 42 positions. With non-linear programming and heuristics, they present a solution that minimizes positioning changes in the intermediate node, optimizing the load balancing in the aircraft.

Brosh (1981) 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 (1992) developed a multi-criteria optimization approach to load the *C-130* aircraft of the *Canadian Air Force*. Based on integer programming, this model provides timely planning and improves airlift support for combat operations, solving WBP with pallets in fixed positions, and considering 20 different items.

Heidelberg et al. (1998) developed a heuristic for 2D packing in air loading, comparing it with methods for solving the *Bin Packing Problem*. Authors conclude that the classical algorithms are inadequate in this context, because they ignore the aircraft balancing constraints.

Mongeau and Bes (2003) presented a method based on linear integer programming to solve the problem of choosing and positioning containers on the *Airbus 340-300*. Safety and stability constraints were considered, with the objective of minimizing fuel consumption.

Fok and Chun (2004) developed a web-based application to make efficient use of space and load balancing for an air cargo company. Based on an analysis of historical data, an operational load planning with mathematical optimization is obtained. This container load planning is usually done roughly 2 hours before departure, when all cargo details are in place.

Chan et al. (2006) carried out a case study with heterogeneous pallets. In order to minimise the total cost of shipping, they developed a 3D packing heuristic, with a loading plan for each pallet. Although the authors do not consider load balancing or positioning of pallets in the cargo hold, this method is relevant in commercial and industrial applications, where cargo items tend to be less dense.

Kaluzny and Shaw (2009) developed a mixed integer linear programming model to arrange a set of items in a military context that optimizes the load balance.

Verstichel et al. (2011) solved WBP by selecting the most profitable subset of containers to be loaded onto an aircraft using mixed-integer programming. Experimental results on real-life data showed significant improvements compared to those obtained manually by an experienced planner.

Mesquita and Cunha (2011) 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. They used the *Scatter Search* metaheuristic as solution method.

Limbourg et al. (2012) developed a mixed-integer program for optimally rearranging a set of pallets into a compartmentalized cargo aircraft, specifically the *Boeing 747*.

Kaspi et al. (2019) define and solve a new extension of the TSP to maximize the financial contribution per invested time and present an optimal iterative solution procedure for the problem which converges after a limited number of iterations.

Roesener and Hall (2014) solved APP and WBP as an integer programming problem, which also allows items to be loaded into pallets according to a specific orientation (e.g., this side up).

Vancroonenburg et al. (2014) presented a mixed integer linear programming model that selects the most profitable pallets, satisfying safety and load balancing constraints on the *Boeing 747-400*. Using a solver, authors solved real problems in less than an hour.

As already mentioned, Lurkin and Schyns (2015) was the first work that simultaneously modeled WBP and SPDP 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 (2016) proposed a heuristic to solve the *Dynamic Airlift Loading Problem* (DALP). Given a set of palletized cargo items that require transport between two nodes in a time frame, the objective of this problem is to select an efficient subset of aircraft, partition the pallets into aircraft loads and assign them to allowable positions on those aircraft.

Paquay et al. (2016) presented a mathematical modeling to optimize the loading of heterogeneous 3D boxes on pallets with a truncated parallelepipeds format. Its objective is to maximise the volume used in containers, considering load balancing constraints, the presence of fragile items and the possibility of rotating these boxes. Paquay et al. (2018) developed some heuristics to solve this problem.

Chenguang et al. (2018) modelled the air transport problem as a 2D packing problem, and presented a heuristic for its optimization in several aircraft, considering load balancing in order to minimise fuel consumption.

Wong and Ling (2020)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. (2021) integrated this tool to a digital simulation model, with a visualization and validation system, based on sensors that alert about load deviations.

Zhao et al. (2021) proposed a new modelling for WBP based on mixed integer programming. Instead of focusing on the center of gravity (CG) deviation, the authors consider the original CG envelope of the aircraft, with a linearization method for its non-linear constraints.

Mesquita and Sanches (2023) contributed with a complex model and elaborate heuristics for the ACLP-RPDP that simultaneously solve 4 intractable sub-problems: APP, WBP, SPDP and TSP. They also compare the performances of four well known heuristics: *Ant Colony Optimization*, the *Noising Method Optimization*, the *Greedy Randomized Adaptive Search Procedure*, and *Tabu Search*. They also create a new heuristic called *Shims* which is fast and may be run in a handheld computer. They did not use computer parallelism to improve performance, nor tryed to minimize the distances from the next nodes destined pallets to the cargo ramp door.

As can be seen, except for Mesquita and Sanches (2023), the remaining works do not address air cargo palletization and load balancing with route optimization in a multi-leg flight plan, and none of them employ parallelization in their algorithms. This is the objective of our work: to reshape Mesquita and Sanches (2023) solution process and algorithms to accommodate parallel features that can improve solution quality and performance, and also include a new feature that is to minimize the distances between the next node's destined pallets and the cargo ramp door.

3. Problem context and assumptions

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

3.1. Operational premises

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

- At each node of the flight plan, the items to be allocated are characterized by weight, volume, scores, and previously known destinations, but do not have dimensions. We leave the consideration of 2D or 3D items for future work.

- We also disregarded *hazardous* items, which eventually could be treated as high score items and other specific constraints.
- We considered a unique pallet type: the *463L Master Pallet*, a common size platform for bundling and moving air cargo. It is the primary air cargo pallet for more than 70 Air Forces and many air transport companies. This pallet has a capacity of $4500kg$ and $14.8m^3$, is equipped for locking into cargo aircraft rail systems, and includes tie-down rings to secure nets and cargo loads, which in total weighs $140kg$. For more information, see `www.463LPallet.com`.
- All items allocated on a pallet must have the same destination. A pallet which has not yet reached its destination may receive more items, although it is known that these operations of removing restraining nets increase handling time and the risk of improper delivery. We do not consider oversized cargo in this work, but only cargoes that fit on these pallets.
- Pallets with destinations set to the next node should be put as near as possible to the cargo ramp door.
- Finally, as we are interested in minimizing fuel costs while keeping the CG in its operational range, we disregarded some lower ones as handling costs.

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

Figure 2: Consolidated items on 463L pallets inside a *Boeing C-17*
Source: From Wikimedia Commons, the free media repository

3.2. Aircraft and load balancing

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

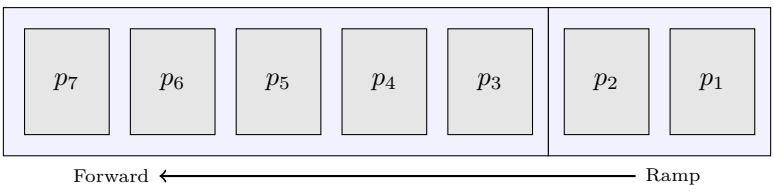

Figure 3: Smaller aircraft layout

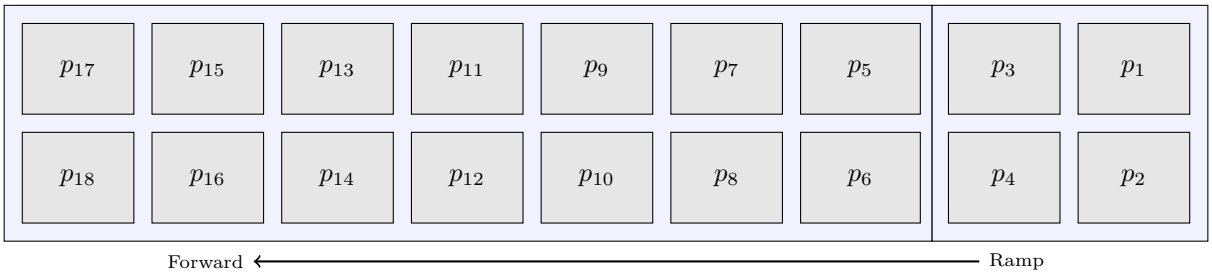

Figure 4: Larger aircraft layout

In both cases, the torque applied to the aircraft must keep its CG in the operational range, which corresponds to a percentage of the *Mean Aerodynamic Chord* [1]: $0.556m$ in the smaller aircraft and $1.17m$ in the larger one. See Figure 5.

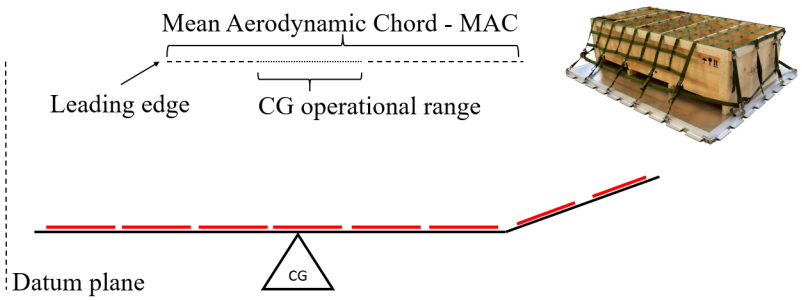

Figure 5: Aircraft longitudinal cut showing red lines as pallets

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

Table 3: Smaller aircraft parameters

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

[1] Chord is the distance between the leading and trailing edges of the wing, measured parallel to the normal airflow over the wing (Houghton and Carpenter, 2003, p.18). The average length of the chord is known as the *Mean Aerodynamic Chord* (MAC).

Table 4: Larger aircraft parameters

Limits	*Payload*: $75,000kg$			$limit^{CG}_{long}$: $1.170m$			$limit^{CG}_{lat}$: $0.19m$		
Pallets	p_{17} p_{18}	p_{15} p_{16}	p_{13} p_{14}	p_{11} p_{12}	p_9 p_{10}	p_7 p_8	p_5 p_6	p_3 p_4	p_1 p_2
CGx (m)	-17.57 -17.57	-13.17 -13.17	-8.77 -8.77	-4.40 -4.40	0 0	4.40 4.40	8.77 8.77	11.47 11.47	14.89 14.89
CGy (m)	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32	1.32 -1.32
Weight limits (kg)	4,500	4,500	4,500	4,500	4,500	4,500	4,500	4,000	3,000
Volume limits (m^3)	14.8	14.8	14.8	14.8	14.8	14.8	14.8	10.0	7.0
Costs	c_d: US$ 4.900/km						$c_g = 0.05$		

We accept the same assumptions as stated by Mesquita and Sanches (2023):

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

3.3. Problem Summary

Informally, ACLP+RPDP can be summarized as follows:

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

4. The mathematical modelling

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

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

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

Let $S_K = \{s : \{1, \ldots, K\} \rightarrow \{1, \ldots, K\}\}$ be the set of $K!$ permutations, which correspond to all possible tours (or itineraries) that have l_0 as origin and end, passing through the other K nodes. k is the node index and $\pi(k)$ is the node in position k of tour π.

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

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

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

Let $X_{ij}^{\pi(k)}$ and $Y_{iq}^{\pi(k)}$ be binary variables, where $0 \leq k \leq K$, $1 \leq j \leq n_{\pi(k)}$, $1 \leq i \leq m$ and $1 \leq q \leq m_{\pi(k)}$. $X_{ij}^{\pi(k)} = 1$ if $t_j^{\pi(k)}$ is assigned to p_i in node $l_{\pi(k)}$, and 0 otherwise. $Y_{iq}^{\pi(k)} = 1$ if $a_q^{\pi(k)}$ is assigned to p_i in node $l_{\pi(k)}$, and 0 otherwise. By definition, $Y_{iq}^0 = 0$. Allocations of items or consolidated items to pallets in node $l_{\pi(k)}$ can be seen as a bipartite graph $G_{\pi(k)}(V_{\pi(k)}, E_{\pi(k)})$, where $V_{\pi(k)} = M \cup N_{\pi(k)} \cup Q_{\pi(k)}$, $E_{\pi(k)} = E_{\pi(k)}^N \cup E_{\pi(k)}^Q$, $(p_i, t_j^{\pi(k)}) \in E_{\pi(k)}^N$ if $X_{ij}^{\pi(k)} = 1$, and $(p_i, a_q^{\pi(k)}) \in E_{\pi(k)}^Q$ if $Y_{iq}^{\pi(k)} = 1$.

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

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

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

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

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

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

$$\epsilon_{\pi(k)} = \frac{\tau_{\pi(k)}}{maxW \times limit_{long}^{CG}};\ k \in \{0, 1, \ldots, K\} \tag{6}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5. Resolution strategy

Once the assumptions of this work and the mathematical modelling of the problem are presented, it is easy to see that ACLP+RPDP is NP-hard. In a similar way to (Lurkin and Schyns, 2015, p. 6), consider the simple case where $K = 1$ (one leg), $m = 2$ (two pallets around the aircraft CG), $2n$ sufficiently light items with same scores in l_0, and no items in l_1. Under these conditions, through polynomial reductions for the *Set-Partition Problem*, it is possible to demonstrate that the decision problem associated with ACLP+RPDP is NP-hard.

Real cases are more complex as they have hundreds of different items in each node and involve three intractable sub-problems: APP, WBP and SPDP. Through the mathematical modeling presented in the previous section, we verify that *Mixed-Integer Programming* (MIP) is not able to solve these cases in feasible time. Thus, it is necessary to adopt some strategy to find a viable solution, not necessarily optimal, that seeks to maximize the objective function f.

Mesquita and Sanches (2023) strategy is based on the fact that, in real cases, K is usually small. Specifically, they will consider $K \leq 6$ throughout their and our work, which is a higher value than usual in *Brazilian Air Force* missions. As a result, if we have fast node-by-node solutions that allow us to construct a complete tour, we will be able to test all possible $K!$ tours and thus select the one that provides the best value for the f function.

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

The tactic will be, at each shipping node, to predefine the destinations of the pallets at that node. In this way, we will reserve a number of pallets proportional to the volume demanded by each destination at the shipping node. We could have used another criterion, but it was observed in the experiments that volume is more constrictive in airlift. Once the destinations of the pallets are defined, we will use serial and Multi-processing heuristics to find the best possible node-by-node solutions. This strategy is summarized in Algorithm 1, retrieved from Mesquita and Sanches (2023).

Algorithm 1 Solves ACLP+RPDP for a scenario with certain volume surplus (1.2, 1.5, or 2.0)

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

In this algorithm, there are six values for the *scenario* parameter, according to Table 5, which defines K, the sets of nodes, the aircraft, the pallets and the costs from Tables 3 or 4 that will be used (line 2).

The other parameter *volume* is a value greater than 1, which corresponds, at each node $l_{\pi(k)}$, to the ratio between the sum of the volumes of the items ($\sum_{j=1}^{n_{\pi(k)}} t_j^{\pi(k)}.v$) and the load capacity of the pallets ($\sum_{i=1}^{m} p_i.v$). This parameter is passed to *ItemsGeneration* (line 3), responsible for creating the items to be shipped, which will be presented in the next section.

method corresponds to one of the heuristics that we will present in subsection 5.2. The loop of lines 5-6 goes through all permutations π, where the node-by-node resolutions are performed by *SolveTour*, whose result is stored in f_π. The best result among all $K!$ tours will be the answer for *scenario*, *volume* and *method* (line 7).

Table 5: Testing scenarios retrieved from Mesquita and Sanches (2023).

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

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

5.1. SolveTour algorithm

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

As we mentioned in the previous section, all tours start and end at l_0 (lines 2-3). After initializing the score and cost values (lines 4-5), there is a loop for the $K + 1$ flights (lines 6-20). Initially we set pallets destination as -1 (line 8). When the aircraft is at node l_0, the initial graph G_1 is empty because it has no consolidated items 11. Otherwise, the set L_k of remaining nodes when departing from $\pi(k)$ is updated (line 13), and *UpdateConsolidated* (line 14) returns the set of consolidated items that have not yet reached their destinations and remain on board, rearranging them on the pallets to minimize CG deviation. This allocation is stored in graph G_1 (line 15).

In the context of this work, we know that $m > K$, once the aircraft has 7 or 18 pallets and $K \leq 6$, allowing there to be at least one pallet for each node to be visited. *SetPalletsDestination* (line 16) presets the destination of each pallet based on the volume demands caused by the items to be embarked in the current node, without changing the pallets destination with consolidated items.

Finally, *SolveNode* includes the edges corresponding to the items shipped at the current node, returning the graph G_2 (line 17). The score and the CG deviation of this graph are calculated (line 18) and accumulated (lines 19-20), allowing the final result of this tour (line 21).

Algorithm 2 Solves the sequence of nodes of tour π

procedure $SolveTour(\pi, L, M, C, N, method, timeLim)$
  $\pi(0) \leftarrow 0$
  $\pi(K+1) \leftarrow 0$
  $score \leftarrow 0$
  $cost \leftarrow 0$
  for $k \leftarrow 0$ to K
    for $i \leftarrow 1$ to m
      $p_i.to[k] \leftarrow -1$ ▷ reset this pallet destination
    if $k = 0$
      $L_0 \leftarrow L$
      Let $G_1(M \cup N_0, \varnothing)$
    else
      $L_k \leftarrow L_k - \pi(k)$
      $Q_{\pi(k)}, E^Q_{\pi(k)}, M \leftarrow UpdateConsolidated(\pi(k))$
      Let $G_1(M \cup N_{\pi(k)} \cup Q_{\pi(k)}, E^Q_{\pi(k)})$
    $M \leftarrow SetPalletsDestination(\pi(k))$
    $G_2 \leftarrow SolveNode(method, \pi(k), G_1, timeLim)$
    $s, \epsilon \leftarrow ScoreAndDeviation(\pi(k), G_2)$
    $score \leftarrow score + s$
    $cost \leftarrow cost + c_{\pi(k),\pi(k+1)} * (1 + c_g * |\epsilon|)$
  return $score/cost$

UpdateConsolidated finds the best allocation for the consolidated items that remain on board. Initially, the set Q is created, with the consolidated items that did not reach their destination.

Then *MinCGDeviation* is run through a MIP solver to relocate the consolidated items on the pallets minimizing torque and ensuring that they all remain on board, one on each pallet. As there are few variables, the MIP solver returns an allocation E^Q_k very quickly.

SetPalletsDestination sets pallets destination not yet defined. The total volume of items and the number of pallets with consolidated items destined for each node. The destinations of each pallet are defined proportionally to the volume of items, regarding the pallets with consolidated items.

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

UpdateConsolidated, *MinCGDeviation*, *SetPalletsDestination*, and *ScoreAndDeviation* are thoroughly described by Mesquita and Sanches (2023).

5.2. Node-by-node resolution

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

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

Data structure:	Items, pallets and edges are arrays of *Python* class objects, which are initialized with the same data loaded from text files.
Stopping criterion:	All methods have the same time limit of 1.8 seconds per node.
Common procedures:	All methods have the same procedures for: solution printing, pallets destinations and initial parameters setting, solution integrity checking, and the same procedures for selecting and inserting edges.

The selection of edges for $E^N_{\pi(k)}$ uses the *edge attractiveness* $\theta^{\pi(k)}_{ij}$, Equation 25, which can be understood as the tendency to allocate the item $t^{\pi(k)}_j$ to the pallet p_i. It is directly proportional to the score, and inversely

proportional to the volume and the torque of each item.

$$\theta_{ij}^{\pi(k)} = \frac{t_j^{\pi(k)}.s}{t_j^{\pi(k)}.v \times 3000} \times (1 - \frac{t_j^{\pi(k)}.w \times |p_i.d|}{\max_s\{t_s^{\pi(k)}.w\} \times \max_q\{|p_q.d|\}}); \; i \in \{1, 2, \ldots, m\}, \; j \in \{1, 2, \ldots, n_k\} \tag{25}$$

To compare allocation graphs generated by the heuristics, we choose the one that offer a higher associated score, since the CG deviation is limited by the constraints.

In the pseudo-codes presented below, given an allocation graph G for the node $l_{\pi(k)}$, $f_s(G)$ is equivalent to the score s returned by $ScoreAndDeviation(\pi(k), G)$, described in Mesquita and Sanches (2023).

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

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

5.3. Multi-process Shims - mpShims

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

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

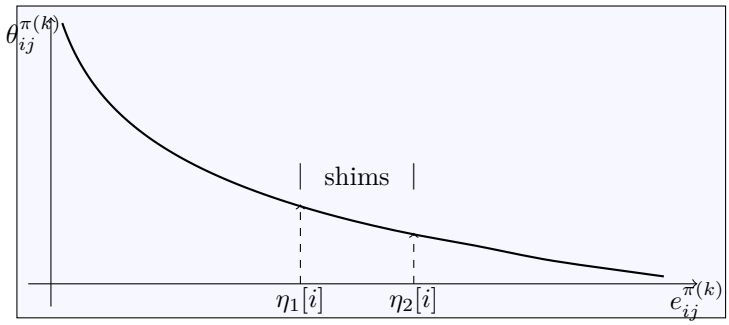

Figure 7: $n_{\pi(k)}$ edges $e_{ij}^{\pi(k)}$ of p_i sorted by $\theta_{ij}^{\pi(k)}$ in non-ascending order

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

Although the second phase solves a knapsack problem, an algorithm for solving a bin packing problem was used (the First Fit Decreasing - FFD), and the most profitable bin (shims) was selected. This method was proved to be faster than a dynamic program solution for the knapsack problem.

In this work, we included a new phase after the first: a minimization of the CG deviation. An integer programming procedure that helps improve the overall *mpShims* results.

1^{st} - Greedy phase with torque surplus

2^{st} - Minimization of CG deviation

3^{st} - Shims construction and best Shims selection

All executed in a multi-process runtime, accessing concurrently, reading and writing, the list of items and the current value of the CG deviation.

As to the multi-process runtime, we considered each pallet as a process.

5.4. mpShims

Algorithm 3 *mpShims* main process

1: **procedure** $mpShims(N_{\pi(k)}, \pi(k), limit, G)$
2: Let $G(M,\ Q_{\pi(k)},\ E_{\pi(k)})$ ▷ the initial solution solution
3: $surplus \leftarrow 1 + 3 \times (1 - limit)$
4: $nodeTorque \leftarrow 0$
5: $ts \leftarrow 2.0$ ▷ torque surplus
6: $lock \leftarrow$ multi-processing lock ▷ avoid race condition
7: $proc \leftarrow \{p_1, p_2, ..., p_m\}$ ▷ each pallet has its own process
8: **for** $i \leftarrow 1$ to m
9: $proc_i \leftarrow Greedy(p_i,\ N_{\pi(k)},\ \pi(k),\ nodeTorque,\ G,\ limit,\ lock,\ ts)$
10: $proc_i.start$
11: **for** $i \leftarrow 1$ to m
12: $proc_i.join$
13: $minCGdev(M,\ \pi(k),\ nodeTorque)$ ▷ minimize the CG deviation.
14: **for** $i \leftarrow 1$ to m
15: $proc_i \leftarrow getBestShims(p_i,\ N_{\pi(k)},\ \pi(k),\ nodeTorque,\ G,\ surplus,\ lock,\ ts)$
16: $proc_i.start$
17: **for** $i \leftarrow 1$ to m
18: $proc_i.join$
19: **return** $G(M,\ N_{\pi(k)} \cup Q_{\pi(k)},\ E_{\pi(k)})$

Line 2 the initial solution comes with some consolidated on board, a bipartite graph between pallets and consolidated.

Line 5 defines the torque surplus (ts). For the greedy phase, the solution may generate torque infeasible solutions with doubled torque (2.0), which will be repaired by the next phase, the minimization of the CG deviation.

Line 6 creates a lock feature for shared data among processes to avoid race condition.

Line 9 calls the greedy method which updates p_i, $N_{\pi(k)}$, $nodeTorque$, and G.

Line 12 wait for all $Greedy$ processes to finish.

Line 13 calls the $minCGdev$ method which updates the $nodeTorque$ and G. $minCGdev$ minimizes the CG deviation and repairs the greedy solution if it is infeasible.

Line 15 calls the $getBestShims$ method which updates p_i, $N_{\pi(k)}$, $nodeTorque$, and G.

Line 18 wait for all $getBestShims$ processes to finish.

Line 19 returns a complete solution, a bipartite graph between pallets and consolidated or items.

5.4.1. Shims first phase

Algorithm 4 generates a greedy allocation of the items available in node $l_{\pi(k)}$, according to the non-ascending order of $\theta_{ij}^{\pi(k)}$, and considering the consolidated items already shipped ($E_{\pi(k)}^{Q}$). Edges are included in this allocation as long as they respect feasibility constraints. Furthermore, the volume of each pallet p_i cannot exceed $p_i.v \times limit$, where $0 < limit \leq 1$ is a given parameter.

Algorithm 4 Mount a greedy solution until the volume limit for each pallet

1: **procedure** $Greedy(\pi(k), G, limit)$
2: Let $G(V_{\pi(k)}, E^{Q}_{\pi(k)})$
3: $\eta_1 \leftarrow \{0\} \times m$
4: $volume \leftarrow \{0\} \times m$
5: $E^{N}_{\pi(k)} \leftarrow \varnothing$
6: **for** $q \leftarrow 1$ to m
7: **if** $(p_i, a_q^{\pi(k)}) \in E^{Q}_{\pi(k)}$
8: $volume[q] \leftarrow volume[q] + a_q^{\pi(k)}.v$
9: **for** each $e_{ij}^{\pi(k)}$ in non-ascending order of $\theta_{ij}^{\pi(k)}$
10: **if** $(E^{N}_{\pi(k)} \cup \{e_{ij}^{\pi(k)}\}$ is feasible) **and** $(volume[i] \leq p_i.v \times limit)$
11: $E^{N}_{\pi(k)} \leftarrow E^{N}_{\pi(k)} \cup \{e_{ij}^{\pi(k)}\}$
12: $volume[i] \leftarrow volume[i] + t_j^{\pi(k)}.v$
13: $\eta_1[i] \leftarrow \eta_1[i] + 1$
14: **return** $G(V_{\pi(k)}, E^{N}_{\pi(k)} \cup E^{Q}_{\pi(k)}),\ \eta_1$ ▷ **return the partial solution and the last item indexes array**

In Algorithm 4, line 3 defines an array with the same size as pallets array to save the set of last edges inserted in the greedy solution. The set η_1 is returned together with the greedy solution.

5.4.2. Shims second phase: improvement on minimizing CG deviation

Between the greedy and the *Shims* selection phases we inserted, in this work, a method to minimize the CG deviation, as stated by the model that follows.

Let π be a tour in the set of permutations of nodes.

Let $Q^{\pi(k)} = \{a_1^{\pi(k)}, a_2^{\pi(k)}, \ldots, a_m^{\pi(k)}\}$ be the set of consolidated items assembled on the set of m pallets in node $\pi(k)$. There is a bi-univocal relation between pallets and consolidated.

Let $H_{iq}^{\pi(k)}$ be the set of decision variables that relates pallet i to consolidated q in node $\pi(k)$.

$$\text{minimize} \left| \sum_{i=1}^{m} \sum_{q=1}^{m} H_{iq}^{\pi(k)} \times (140 + a_q^{\pi(k)}.w) * p_i.d \right| \tag{26}$$

$$s.t.: \sum_{i=1}^{m} H_{iq}^{\pi(k)} = 1;\ q \in \{1, 2, \ldots, m\} \tag{27}$$

$$s.t.: \sum_{q=1}^{m} H_{iq}^{\pi(k)} = 1;\ i \in \{1, 2, \ldots, m\} \tag{28}$$

Equation 26 minimizes the aircraft absolute torque in node $\pi(k)$, where 140 is the weight of an empty pallet, $a_q^{\pi(k)}.w$ the weight of the consolidated, and $p_i.d$ the distance between the pallet centroid to the aircraft CG; Equation 27 states that each consolidated will be assigned to exactly one pallet; and Equation 28 states that each pallet will receive one consolidated.

5.4.3. Shims third and fourth phases

Algorithm 5 Mount shims of edges that fills each pallet gap and return the best shims

1: **procedure** $getBestShims(i,\ \eta_1,\ \eta_2,\ E,\ k,\ slack)$
2: $volume \leftarrow 0$
3: $b \leftarrow 1$
4: $shims[b] \leftarrow \{\}$
5: $Set \leftarrow Set \cup \{shims[b]\}$
6: **for** $x \leftarrow \eta_1[i]$ to $\eta_2[i]$
7: $NewShims \leftarrow$ **True**
8: $e_{ij}^k \leftarrow E[i][x]$
9: **for** $shims \in Set$
10: **if** $e_{ij}^k \notin (E_k^N \cup shims)$ **and** e_{ij}^k is feasible **and** $(t_j^k.v + volume) \leq slack[i]$
11: $shims \leftarrow shims \cup \{e_{ij}^k\}$
12: $volume \leftarrow volume + t_j^k.v$
13: $NewShims \leftarrow$ **False**
14: **break**
15: **if** $NewShims$
16: $volume \leftarrow 0$
17: $b \leftarrow b + 1$
18: $shims[b] \leftarrow \{\}$
19: $shims[b] \leftarrow shims[b] \cup \{e_{ij}^k\}$
20: $Set \leftarrow Set \cup \{shims[b]\}$
21: $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^k \in shims} t_b^k.w$ is maximum
22: $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^k \in shims} t_b^k.v$ is maximum
23: $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ab}^k \in x} t_b^k.s$ is maximum
24: **return** sh_{best}

Algorithm 5 describes the procedure do get the best Shims, the best subset of edges to fill each pallet gap. This algorithm follows the logic of the *First-Fit Decreasing* algorithm as described by Johnson and Garey (1985).

Line 4 creates the first empty shims. Line 5 creates the first set of shims from where the best shims will be chosen.

Line 6 iterate in the edges indexes range to find subsets of shims that fit into each pallet gap.

Initially, new shims creation is permitted (7), but it will only be created if the last edge could not be included in any shims from the set (15).

For each edge, iterate in all sets in a try to include it in any of the previous shims (9). As the last edge was included, forbid a new shims creation (13).

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

6. Implementation and results

As we are dealing with a new problem, which has been modeled by Mesquita and Sanches (2023), we use the same benchmarks of their work, which are based on the characteristics of real airlifts carried out by the *Brazilian Air Force*.

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

6.1. Results obtained

The final experiments are performed on a 64-bit, 16GiB, 3.6GHz, 8-core AMD® Fx-8320e processor, 2 threads per core, with *Linux Ubuntu 20* as the operational system and *Python 3.8.2* as the programming language.

A relevant problem with meta-heuristics is the parameter's adjustment to extract the best possible efficiency in the optimization. The bulk of the tools to fine-tune these parameters utilize a statistical backdrop that enables them to make accurate predictions regarding the differences in candidate configurations' performance. Inadequacies in the design of the statistical experiment may therefore lead to erroneous results from the statistical tests and, subsequently, cause the approach to produce inaccurate comparisons of candidate configurations. That is why all parameters were set using the *iRace* package (López-Ibáñez et al., 2016). Supplying *iRace* with a range for $volThreshold$ as [0.80 to 0.99], *iRace* yielded 0.92, by training on the uneven instances numbers. The values tested for 20%, 50%, and 100% volume surpluses were very close, considering the confidence level offered by *iRace*.

For parallelization, we used the *Python Multi-processing* package, which implements *process-based parallelism* or *parallel shared memory algorithm*. According to Python (2022), *Multi-processing* is a package that supports spawning processes using an API similar to the *Python Threading* package.

According to (Breshears, 2009, p.271), a *Process* is the operating system's spawned and controlled entity that encapsulates an executing application. A process has two main jobs: the first is to hold the application's resources, and the second is to carry out the application's instructions.

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

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

We ran Algorithm 1 in the 6 scenarios described in Table 5, considering 3 methods for node-by-node resolution: the *Gurobi* MIP solver, *Shims*, and *mpShims* (Algorithm **??**).

In generating the items in each node, we consider 3 values for the parameter *surplus*. The results obtained for the function f, with the corresponding runtime in seconds, are shown in Tables 6 ($surplus = 1.2$, $surplus = 1.5$, and $surplus = 2.0$).

For each *method*, *scenario* and *surplus*, 7 different instances were generated. Therefore, 126 tests (6 scenarios $\times$ 3 volumes $\times$ 7 instances) were performed for each method.

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

- **Normalized**: value between 0 and 1, which corresponds to the ratio between the sum of f values obtained by the method in all scenarios and the sum of the best values obtained among all methods in all scenarios. The higher the value of **Normalized**, the closer the method approached the best solutions found.
- **Speed-up**: ratio of the sums of the worst runtimes of all scenarios and the sum of the method runtimes in all scenarios. The method with the highest **Speed-up** is the fastest.

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

Table 6: Volume surpluses, methods and scenarios results

			Scenarios						Normalized
surplus	*method*	**Results**	**1**	**2**	**3**	**4**	**5**	**6**	**Speed-up**
1.2	*Gurobi*	*f*	**12.00**	**8.38**	**12.19**	**13.38**	**14.42**	13.25	**0.99**
		run-time (s)	8	14	25	95	217	1719	1.0
	Shims	*f*	9.69	6.37	9.58	11.07	12.24	11.94	0.82
		run-time (s)	< 1	3	5	11	27	342	5.3
	mpShims	*f*	11.90	7.61	10.86	13.13	13.75	**13.75**	0.96
		run-time (s)	< 1	< 1	3	9	26	156	**10.6**
1.5	*Gurobi*	*f*	16.49	**11.60**	**16.68**	**18.33**	**20.10**	17.94	**0.99**
		run-time (s)	12	25	45	169	437	2377	1.0
	Shims	*f*	13.71	9.06	13.54	15.21	17.09	17.15	0.84
		run-time (s)	< 1	3	6	24	70	419	5.9
	mpShims	*f*	**16.85**	10.65	**16.68**	17.05	19.72	**18.53**	0.98
		run-time (s)	< 1	2	3	10	28	173	**14.1**
2.0	*Gurobi*	*f*	**24.04**	**17.33**	**24.62**	**26.36**	**28.88**	**25.89**	**1.00**
		run-time (s)	14	34	62	236	609	3431	1.0
	Shims	*f*	20.42	13.89	20.36	22.71	25.13	24.69	0.86
		run-time (s)	< 1	4	7	30	85	501	7.7
	mpShims	*f*	23.12	14.89	22.08	23.96	26.98	25.29	0.93
		run-time (s)	< 1	2	3	12	33	203	**19.0**

For the MIP solver in some scenarios, less than 50% of the results were optimal, and the remaining were suboptimal due to the time limit. This explains that *mpShims* surpassed some of the MIP solver results.

In Mesquita and Sanches (2023), *Shims* surpassed the MIP solver, on average, because the time limit they used (0.7s) was sufficient to guarantee a worst-case solution in less than 30min. This decision caused most of the MIP solver results to be poor, making *Shims* outstanding as it was the unique method to perform well with all scenarios and instances.

In this work, we used as a time limit the worst *Shims* run-time, which was 1.8s. This made the MIP solver yield optimal or near-optimal solutions. This is good for the reader to have an idea of the real performance differences among the methods.

On average, *mpShims* is 14 times faster than the MIP solver, and it reached 96% of *Gurobi f* values, on average, which is good for a tool of operational use. The use of parallelism is another important contribution of this work. It has improved *Shims* results and also its speed.

6.2. Yet another improvement

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

Let π be a tour in the set of permutations of nodes.

Let $maxD$ be the distance from the last ramp pallet to the center of gravity.

Let $Q^{\pi(k)} = \{a_1^{\pi(k)}, a_2^{\pi(k)}, \ldots, a_m^{\pi(k)}\}$ be the set of consolidated items assembled on the set of m pallets in node $\pi(k)$.

Let $Z_{iq}^{\pi(k)}$ be the set of decision variables that relates pallet i to consolidated q in node $\pi(k)$.

$$\text{minimize} \sum_{i=1}^{m} \sum_{q=1}^{m} Z_{iq}^{\pi(k)} \times (maxD - p_i.d), \ \textbf{if } a_q^{\pi(k)}.to = \pi(k+1)\} \tag{29}$$

$$s.t.: \sum_{i=1}^{m} Z_{iq}^{\pi(k)} = 1; \ q \in \{1, 2, \ldots, m\} \tag{30}$$

$$s.t.: \sum_{q=1}^{m} Z_{iq}^{\pi(k)} = 1;\ i \in \{1, 2, \ldots, m\} \tag{31}$$

$$p_i.w = 140 + a_q^{\pi(k)}.w;\ i \in \{1, 2, \ldots, m\};\ q \in \{1, 2, \ldots, m\} \tag{32}$$

$$s.t.: \ |\sum\ p_i.d * p_i.w\ | \leq maxW \times limit_{long}^{CG};\ i \in \{1, 2, \ldots, m\} \tag{33}$$

Equation 29 minimizes the sum of distances of the consolidated in node $\pi(k)$ destined to the next node $\pi(k+1)$; Equation 30 states that each consolidated will be assigned to exactly one pallet; Equation 31 states that each pallet will receive one consolidated; Equation 32 calculates the total weight of each pallet; and Equation 33 guarantees that the torque limit will not be exceeded.

As the number of pallets was small (7 or 18) for an integer program solver, the increased time was negligible (less than 1s).

Columns **Before** and **After** on Table 7 refer to the application of the integer programming solver according to equations 29 to 33. Column **Improvement** is the percentage of improvements in distances minimization.

Table 7: Minimization of pallets distances from the ramp door

surplus	**Scenario**	**Before (m)**	**After (m)**	**Improvement (%)**
1.2	1	33	16	52
	2	194	154	21
	3	149	119	20
	4	135	103	23
	5	121	84	31
	6	105	71	33
1.5	1	32	18	44
	2	198	135	32
	3	156	128	18
	4	137	106	22
	5	120	87	28
	6	106	77	27
2.0	1	33	20	38
	2	199	142	28
	3	156	122	22
	4	138	106	23
	5	123	84	31
	6	109	75	31
			Average	**29**

This last procedure favored ground operations as it reduced in 29% the distances pallets will be moved on disembarking operations. This last step is not part of *mpShims* and may be applied as a post improvement for any method. This is another important contribution of this work.

7. Conclusions

xxxxx

Another important contribution is that we employed the *iRace* package to fine tune the meta-heuristics parameters, guaranteeing the best possible performance for the instances under training and testing.

Acknowledgments

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

References

Brandt, F., 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, 399–410.

Breshears, C., 2009. Chapter 8: A Thread Monkey's Guide to Writing Parallel Applications. Kindle 1st Edition.

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

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

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

Fok, K., Chun, A., 2004. Optimizing air cargo load planning and analysis, in: Proceedings of the International Conference on Computing, Communications and Control Technologies.

Heidelberg, K.R., Parnell, G.S., Ames, J.E., 1998. Automated air load planning. Naval Research Logistics 45, 751–768.

Houghton, E.L., Carpenter, P.W., 2003. Aerodynamics for Engineering Students (5th ed.). Butterworth Heinmann.

Johnson, D.S., Garey, M.R., 1985. A 7160 theorem for bin packing. Journal of Complexity 1, 65–106. doi:10.1016/0885-064X(85)90022-6.

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

Kaspi, M., Zofi, M., Teller, R., 2019. Maximizing the profit per unit time for the travelling salesman problem. Computers & Industrial Engineering 135, 702–710. URL: https://www.sciencedirect.com/science/article/pii/S0360835219303808, doi:https://doi.org/10.1016/j.cie.2019.06.050.

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

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

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

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58.

Manfrin, M., Birattari, M., Stützle, T., Dorigo, M., 2006. Parallel Ant Colony for the Traveling Salesman Problem. Springer.

Mesquita, A.C.P., Cunha, C.B., 2011. 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 19, 33–42.

Mesquita, A.C.P., Sanches, C.A.A., 2023. Air cargo load and route planning in pickup and delivery operations. Computers & Operations Research .

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

Ng, K.Y.K., 1992. A multicriteria optimization approach to aircraft loading. Operations Research 40, 1200–1205.

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

Paquay, C., Schyns, M., 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.

Python, 2022. Process-based parallelism. URL: https://docs.python.org/3/library/multiprocessing.html.

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

Roesener, A., 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, 125–141.

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

Verstichel, J., Vancroonenburg, W., Souffriau, W., 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., 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), pp. 123–130. doi:10.1109/ICFIE50845.2020.9266727.

Wong, E.Y.C., Mo, D.Y., So, S., 2021. Closed-loop digital twin system for air cargo load planning operations. International Journal of Computer Integrated Manufacturing 34, 801–813. doi:10.1080/0951192X.2020.1775299.

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