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 execution, making mobile computers more competent than ever before.

In this work, we explore process-based parallel computing 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 multiprocessing heuristics 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. We also solved a three-dimensional packing problem in the item-pallet allocations yielded by *Shims*, which can guarantee that the palletization plan will include sets of items that actually fit into the pallets.

Keywords: Air Cargo, Air Palletization, Weight and Balance, Pickup and Delivery, Parallel computing, 3D packing

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 gain 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	MLU	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 (**MLU**) at the intermediate node was modeled 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	l_0	l_1	l_2	l_3	l_4	l_5	l_6
IATA*	GRU	GIG	SSA	CNF	CWB	BSB	REC
GRU	0	343	1,439	504	358	866	2,114
GIG	343	0	1,218	371	677	935	1,876
SSA	1,439	1,218	0	938	1,788	1,062	676
CNF	504	371	938	0	851	606	1,613
CWB	358	677	1,788	851	0	1,084	2,462
BSB	866	935	1,062	606	1,084	0	1,658
REC	2,114	1,876	676	1,613	2,462	1,658	0

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

Figure 1: A route 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.

In this work, we raise four hypotheses for improvements to the work of Mesquita and Sanches (2023): (1) Computer parallelism could improve *Shims*'s performance. (2) The *Shims*' results may have been inferior to that; it could be that a sequential approach to pallet building may lead to a premature stop due to the torque constraints. (3) A procedure for 3D packing running in process-based parallel computing mode would keep the run time adequate for operational use. (4) The distances to move pallets on unloading in the next node could be minimized to benefit ground handling costs.

So, this work seeks to improve their results in terms of quality and performance by using computer parallelism, CG balancing between *Shims*' phases, minimizing the distance between the next node-destined pallets and the cargo door, and performing a 3D packing procedure on each pallet.

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, TSP, and 3D packing has been addressed, and the first time ACLP+RPDP is solved with process-based 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 minimize the total cost of shipping, they developed a 3D packing heuristic, with a loading plan for each pallet. Although the authors do not consider load balancing or positioning of pallets in the cargo hold, this method is relevant in commercial and industrial applications, where cargo items tend to be less dense.

Kaluzny and Shaw (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 maximize the volume used in containers, considering load balancing constraints, the presence of fragile items and the possibility of rotating these boxes. Paquay et al. (2018) developed some heuristics to solve this problem.

Chenguang et al. (2018) modeled 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 minimize 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 model 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 multi-core parallelism to improve performance, nor tried to minimize the distances from the next nodes destined pallets to the cargo ramp door, nor tried to perform a 3D packing to keep in solution only items that fit in the pallets usable volume.

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, include a new feature that is to minimize the distances between the next node's destined pallets and the cargo ramp door, and also include a 3D packing procedure to guarantee that only items that fit into the pallets usable volume are planned to palletization.

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:

- We 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 $14.8m^3$ (depending on the aircraft ceiling height), 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 over-sized cargo in this work, but only cargoes that fit on these pallets.

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.

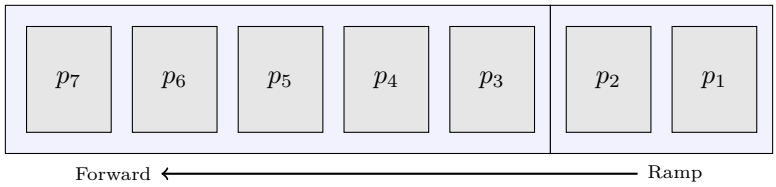

Figure 3: Smaller aircraft layout

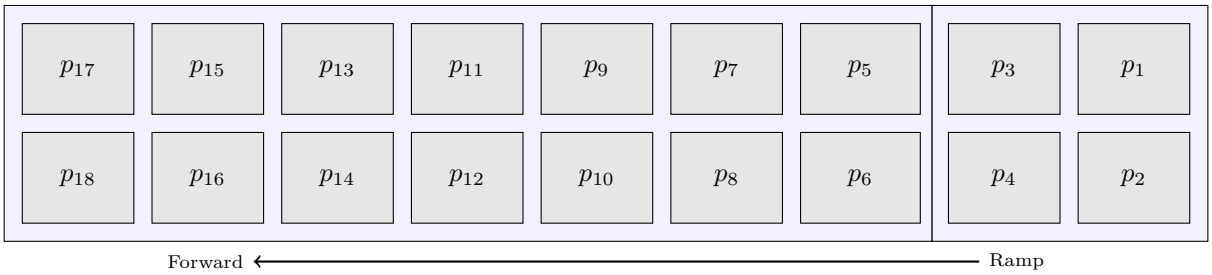

Figure 4: Larger aircraft layout

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.

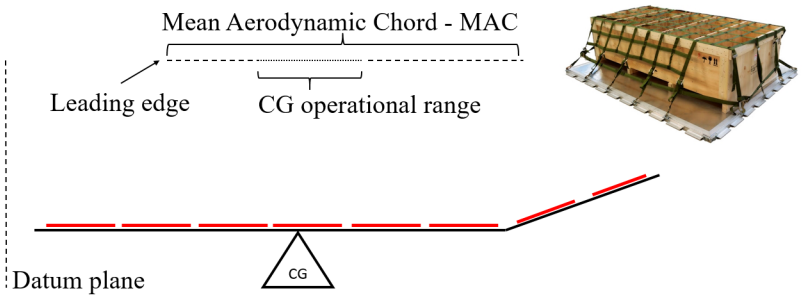

Figure 5: Aircraft longitudinal cut showing red lines as pallets

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 $p_i.D^{lat}$ 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
— from Mesquita and Sanches (2023) —	
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.
— our inclusion —	
s.t.	Pallets destined for the next node should be put as near as possible to the ramp door.
	Items allocated to the pallet must fit on the pallet before consolidation.

4. The mathematical model

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

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 indexed by π, 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 π_k is the node in position k of tour π. τ^{π_k} is the total torque applied by the loaded pallets to the aircraft in this node.

Let $L = \{\pi_0, \pi_1, \ldots, \pi_K\}$ be the set of $K+1$ nodes, where π_0 is the origin and end of a flight path. Let $d(\pi_i,\ \pi_j)$ be the distance from π_i to π_j, where $0 \leq i,\ j \leq K$. By definition, $d(\pi_i,\ \pi_i) = 0$. Let L^{π_k} be the set of remaining nodes when the aircraft is in π_k, $0 \leq k \leq K$. Therefore, $L^{\pi_0} = L$ and $L^{\pi_K} = \{\pi_0\}$.

Let $M^{\pi_k} = \{p_1, p_2, \ldots, p_m\}$ 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 π_k. Attributes for constant values are in uppercase.

Let $N^{\pi_k} = \{t_1^{\pi_k}, t_2^{\pi_k}, \ldots, t_{n^{\pi_k}}^{\pi_k}\}$ be the set of n^{π_k} items to be loaded in node π_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 $Q^{\pi_k} = \{a_1^{\pi_k}, a_2^{\pi_k}, \ldots, a_m^{\pi_k}\}$ be the set of consolidated items loaded in $m_k \leq m$ pallets when the aircraft arrives at node π_k, with $0 \leq k \leq K$. $a_i^{\pi_k}$, $1 \leq i \leq m$, 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^{π_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 $1 \leq q \leq m$. $X_{ij}^{\pi_k} = 1$ if $t_j^{\pi_k}$ is assigned to p_i in node π_k, and 0 otherwise.

$Y_{iq}^{\pi_k} = 1$ if $a_q^{\pi_k}$ is assigned to p_i in node π_k, and 0 otherwise. By definition, $Y_{iq}^0 = 0$.

Allocations of items or consolidated to pallets in node π_k can be seen as a bipartite graph $G^{\pi_k}(V^{\pi_k}, E^{\pi_k})$, where $V^{\pi_k} = M^{\pi_k} \cup N^{\pi_k} \cup Q^{\pi_k}$, $E^{\pi_k} = E_N^{\pi_k} \cup E_Q^{\pi_k}$, $(p_i, t_j^{\pi_k}) \in E_N^{\pi_k}$ if $X_{ij}^{\pi_k} = 1$, and $(p_i, a_q^{\pi_k}) \in E_Q^{\pi_k}$ if $Y_{iq}^{\pi_k} = 1$.

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

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

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

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

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

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

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

$$L^{\pi_0} = L;\ L^{\pi_k} = L^{\pi_{k-1}} - \{\pi_k\};\ k \in \{1, 2, \ldots, K\} \tag{7}$$

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

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

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

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

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

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

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

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

$$s.t.: p_i.D^{lat} \times \Big| LatIt^{\pi_k} + LatCons^{\pi_k} \Big| \leq \sum_{i=1}^{m} p_i.W \times limit_{lat}^{CG};\ k \in \{0,1,\ldots,K\} \tag{16}$$

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

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

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

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

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

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

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

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

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 path will be $\pi_0, \pi_1, \ldots, \pi_K, \pi_0$. $\tilde{s}$ 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 modeling 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 be 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 multiprocessing 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 ← ItemsGeneration(scenario, surplus)
4:     for each method
5:         for each π ∈ S_K
6:             f_π ← SolveTour(π, L, M, C, N, method, timeLim)
7:         answer[scenario, surplus, method] ← 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).

Argument *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 outcome 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 sub-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 π_0 (lines 2-3). After initializing the score and cost values (lines 4-5), there is a loop for the $K + 1$ tours (lines 6-22). Initially we set pallets destination as -1 (line 8). When the aircraft is at node π_0, the initial graph G_1 is empty because it has no consolidated items 11. Otherwise, the set L^{π_k} of remaining nodes when departing from π_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).

Algorithm 2 Solves the sequence of nodes of tour π

1: **procedure** $SolveTour(\pi, L, M, C, N, method, timeLim)$
2: $\quad\pi_0 \leftarrow 0$
3: $\quad\pi_{K+1} \leftarrow 0$
4: $\quad score \leftarrow 0$
5: $\quad cost \leftarrow 0$
6: $\quad$**for** $k \leftarrow 0$ to K
7: $\quad\quad$**for** $i \leftarrow 1$ to m
8: $\quad\quad\quad p_i.to^{\pi_k} \leftarrow -1$ ▷ reset this pallet destination
9: $\quad\quad$**if** $k = 0$
10: $\quad\quad\quad L^{\pi_0} \leftarrow L$
11: $\quad\quad\quad$Let $G_1(M \cup N_0, \varnothing)$
12: $\quad\quad$**else**
13: $\quad\quad\quad L^{\pi_k} \leftarrow L^{\pi_k} - \{\pi_k\}$
14: $\quad\quad\quad Q^{\pi_k}, E_Q^{\pi_k}, M^{\pi_k} \leftarrow UpdateConsolidated(\pi_k)$
15: $\quad\quad\quad$Let $G_1(M^{\pi_k} \cup N^{\pi_k} \cup Q^{\pi_k}, E_Q^{\pi_k})$
16: $\quad\quad M^{\pi_k} \leftarrow SetPalletsDestinations(\pi_k)$
17: $\quad\quad G_2 \leftarrow SolveNode(method, \pi_k, G_1, timeLim)$
18: $\quad\quad G_3 \leftarrow mp3Dpacking(G_2)$
19: $\quad\quad G_4 \leftarrow minRampDist(G_3)$
20: $\quad\quad s, \epsilon \leftarrow ScoreAndDeviation(\pi_k, G_4)$
21: $\quad\quad score \leftarrow score + s$
22: $\quad\quad cost \leftarrow cost + c_{\pi_k, \pi_{k+1}} * (1 + c_g * |\epsilon|)$
23: $\quad$**return** $score/cost$

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. *SetPalletsDestinations* (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 20) and accumulated (lines 21-22), allowing the final result of this tour (line 23).

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 $E_Q^{\pi_k}$ very quickly.

SetPalletsDestinations 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.

In line 18, $mp3Dpacking(G)$ packs the items previously assigned to a pallet, which is feasible in terms of volume, weight, and torque but still not guaranteed to fit into the pallet. In this procedure, some unfit items are excluded from the solution.

In line 19, $minRampDist(G)$ minimizes the distance of the next node destined pallets, keeping the CG in its operational range. This procedure is implemented in integer programming. This method does not affect palletization or the objective function value.

$ScoreAndDeviation(\pi_k, G_4)$ 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*, *SetPalletsDestinations*, 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 π_k and G is the allocation graph of the consolidated items that remain on board at π_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.
Common procedures:	All methods have the same procedures for: solution printing, pallets destinations and initial parameters setting, result integrity checking, and the same procedures for selecting and inserting edges.

The selection of edges for $E_N^{\pi_k}$ uses the *edge attractiveness* $\theta_{ij}^{\pi_k}$, Equation 25, which can be understood as the tendency to allocate the item $t_j^{\pi_k}$ to the pallet p_i. It is directly proportional to the score, and inversely relative to the volume and torque of each item.

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

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 π_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 process-based parallel heuristic that we propose called *Multiprocessing Shims (mpShims)*.

According to (Manfrin et al., 2006, p.226), there are five topologies for multiprocessing 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*, in which there are no communication costs and the best solution is chosen among all processes.

In this work, we had to select the *fully-connected* topology for the *Greedy* heuristics and for *mpShims* because the list of items to embark and the aircraft torque must be accessible for reading and writing by all child processes. This decision require an effective race condition management.

When threads or processes attempt to simultaneously access a shared resource, and the accesses can result in an error, we often say the program has a race condition, because the threads or processes are in a "race" to carry out an operation. (Pacheco and Malensek, 2021, p. 53).

For the *mp3Dpacking* we used *parallel independent runs*, because there is no memory sharing.

5.3. Multiprocessing Shims - mpShims

Finally, we present a new multiprocessing 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

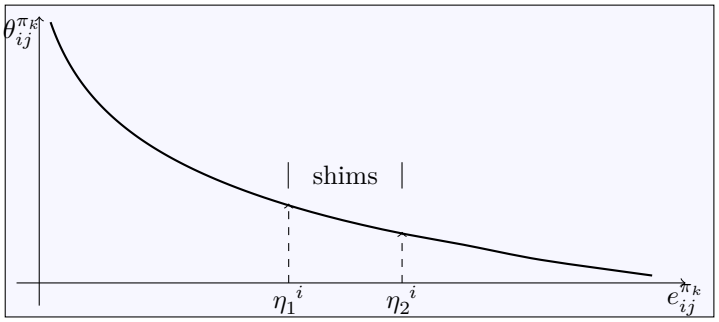

Figure 7: n^{π_k} edges $e_{ij}^{\pi_k}$ of p_i sorted by $\theta_{ij}^{\pi_k}$ in non-ascending order

Figure 7 represents the n^{π_k} possible edges $e_{ij}^{\pi_k}$ of p_i sorted by $\theta_{ij}^{\pi_k}$. *mpShims* starts with a greedy solution for each pallet, stopping at the edge with index in $\eta_1{}^i$ close to the local optimum (first phase). Then, performs an integer programming procedure to minimize the CG deviation (second phase), and finally, 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 (third phase).

Although the third phase solves a *Knapsack Problem*, a faster algorithm for solving a *Bin Packing Problem* was used (*the First Fit Decreasing - FFD*), and the most profitable bin (*Shims*) was selected.

1^{st} - A Greedy phase to partially build pallets.

2^{st} - Minimization of CG deviation.

3^{st} - *Shims* set construction and best *Shims* selection.

Algorithm 3 *mpShims* main process

1: **procedure** $mpShims(N^{\pi_k}, \pi_k, limit, G)$
2: Let $G(M^{\pi_k},\ Q^{\pi_k},\ E^{\pi_k})$ ▷ the initial solution
3: $surplus \leftarrow 1 + 3 \times (1 - limit)$
4: $nodeTorque \leftarrow 0$
5: $ts \leftarrow 1.7$ ▷ torque surplus
6: $lock \leftarrow$ multiprocessing 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: $proc_i.fork$
11: **for** $i \leftarrow 1$ to m
12: $proc_i.join$
13: $minCGdev(M^{\pi_k},\ \pi_k,\ nodeTorque)$ ▷ 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: $proc_i.fork$
17: **for** $i \leftarrow 1$ to m
18: $proc_i.join$
19: **return** $G(M^{\pi_k},\ N^{\pi_k} \cup Q^{\pi_k},\ E^{\pi_k})$

In the first and third phases, we considered each pallet as a parallel process, taking advantage of nowadays multi-core computers, being executed in a multiprocessing run-time, accessing concurrently, reading and writing the list of items and the current value of the CG deviation.

Follow the comments on Algorithm 3:

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 temporary torque unfeasible solutions. We raised the Hypothesis (2) that, in sequential mode, where one pallet is built at a time, torque constraints may be more restrictive than volume or weight constraints, which could stop the palletization process earlier, fostering a solution that may be not as good as it could be. So, we decided to extend torque limit and balance cargo before the next *Shims* phase to allow more room for pallet fulfillment. The value 1.7 was determined by the *iRace* tool developed by López-Ibáñez et al. (2016).

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^{π_k}, $nodeTorque$, and G.

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

Line 13 calls the $minCGdev()$ integer programming method (Subsection 5.3.2) which updates the $nodeTorque$ and G. $minCGdev()$ minimizes the CG deviation, bringing it back to its operational range.

Line 15 calls the $getBestShims()$ method which updates p_i, N^{π_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.

5.3.1. The mpShims first phase

Algorithm 4 generates a greedy allocation of the items available in node π_k, according to the non-ascending order of $\theta_{ij}^{\pi_k}$, and considering the consolidated items already shipped ($E_Q^{\pi_k}$). 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** $Greedy(\pi_k,\ G,\ limit,\ ts)$
2: Let $G(V^{\pi_k}, E_Q^{\pi_k})$
3: $\eta_1 \leftarrow \{0 \mid 1\ to\ m\}$
4: $volume \leftarrow \{0 \mid 1\ to\ m\}$
5: $E_N^{\pi_k} \leftarrow \varnothing$
6: $\tau^{\pi_k} \leftarrow 0$ ▷ aircraft torque in node π_k
7: **for** $q \leftarrow 1$ **to** m
8: **if** $(p_i, a_q^{\pi_k}) \in E_Q^{\pi_k}$
9: $volume_q \leftarrow volume_q + a_q^{\pi_k}.v$
10: $\tau_{max} \leftarrow maxW \times limit_{long}^{CG}$ ▷ maximum aircraft torque
11: **for** each $e_{ij}^{\pi_k}$ in non-ascending order of $\theta_{ij}^{\pi_k}$
12: $\Delta\tau \leftarrow t_j.w \times p_i.D$
13: $\tau_{new} \leftarrow |\tau^{\pi_k} + \Delta\tau|$
14: **if** $(E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\}$ is feasible) **and** $(volume_i \leq p_i.V \times limit)$ **and** $(\tau_{new} \leq \tau_{max} \times ts\ )$
15: $E_N^{\pi_k} \leftarrow E_N^{\pi_k} \cup \{e_{ij}^{\pi_k}\}$
16: $volume_i \leftarrow volume_i + t_j^{\pi_k}.v$
17: ${\eta_1}^i \leftarrow {\eta_1}^i + 1$
18: $\tau^{\pi_k} \leftarrow \tau^{\pi_k} + \Delta\tau$
▷ return the partial solution and the last item indexes array
19: **return** $G(V^{\pi_k}, E_N^{\pi_k} \cup E_Q^{\pi_k}),\ \eta_1$

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.

Line 14 presents the variable *ts* (torque surplus), that is an extra-torque limit to allow each greedy pallet construction to, temporally, exceed the torque limit, that will be turned feasible again by the next *mpShims*' phase, the CG minimization phase.

5.3.2. The mpShims second phase

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

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 π_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 π_k.

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

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

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

Equation 26 minimizes the aircraft absolute torque in node π_k, where 140kg 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.

5.3.3. The mpShims third phase

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

1: **procedure** $getBestShims(i,\ \eta_1,\ \eta_2,\ E,\ \pi_k,\ slack)$
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: $e_{ij}^{\pi_k} \leftarrow {E_x}^i$
9: **for** $shims \in Set$
10: **if** $e_{ij}^{\pi_k} \notin (E_N^{\pi_k} \cup shims)$ **and** $e_{ij}^{\pi_k}$ is feasible **and** $(t_j^{\pi_k}.v + volume) \leq {slack}^i$
11: $shims \leftarrow shims \cup \{e_{ij}^{\pi_k}\}$
12: $volume \leftarrow volume + t_j^{\pi_k}.v$
13: $NewShims \leftarrow$ **False**
14: **break**
15: **if** $NewShims$
16: $volume \leftarrow 0$
17: $b \leftarrow b + 1$
18: $shims_b \leftarrow \{\}$
19: $shims_b \leftarrow shims_b \cup \{e_{ij}^{\pi_k}\}$
20: $Set \leftarrow Set \cup \{shims_b\}$
21: $sh_w \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} t_b^{\pi_k}.w$ is maximum
22: $sh_v \leftarrow shims$, where $shims \in Set$ and $\sum_{e_{ab}^{\pi_k} \in shims} t_b^{\pi_k}.v$ is maximum
23: $sh_{best} \leftarrow shims$, where $shims \in \{sh_w, sh_v\}$ and $\sum_{e_{ab}^{\pi_k} \in x} t_b^{\pi_k}.s$ is maximum
24: **return** sh_{best}

Algorithm 5 describes the procedure to get the top *Shims*, the best subset of edges to fill each pallet gap. This algorithm follows the logic of the *First-Fit Decreasing* method as described by Johnson and Garey (1985). Although it does not aim to minimize the number of bins, its mechanism helped build the set of *Shims* to allow the best to be chosen.

Line 4 creates the first empty shims. Line 5 produce 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.

5.4. Multiprocessing 3D packing

According to Brandt and Nickel (2019), *Technically the task to solve is a three-dimensional Container Loading Problem (CLP) with a multitude of constraints. The CLP alone is known to be NP-hard and extremely hard to solve even for instances of moderate size. The typical number of items per transport segment ranges between 300 and 800 items.* This work also adopts a similar range as items quantities, from 300 to 1000 items per node.

So, to confirm or not Hypothesis (3), we decided to adopt the model for three-dimensional (3D) packing based in the instructions on Dube and Kanavathy (2006), Li et al. (2014) and on Paquay et al. (2018), to pack the items selected for each pallet.

The main logic of the 3D packing procedure (3*Dpacker*) is based on an approximate algorithm like follows:

1 From a list of items assigned to a pallet, items are sorted from the biggest to the smallest and placed in such an order on a pallet. Each item has 1 to 6 orientations to choose from in the moment of placement. The orientation procedure can select the best direction type among possible orientations.

2 A pivot point is used to determine item's position. The pivot is an (x, y, z) coordinate which represents a point in the pallet at which an attempt to pack an item will be made.

3 The back lower left corner of the item will be placed at the pivot. If the item cannot be packed at the pivot position then it is rotated until it can be packed at the pivot point or until we have tried all 6 possible rotation types (first fit).

4 If, after rotating it, the item still cannot be packed at the pivot point, then we move on to packing another item and add the unpacked item to a list of items that will be packed after an attempt to pack the remaining items is made.

5 The first pivot in the empty pallet is always (0, 0, 0). When one item can be placed into multiple optimal pivot point, the placement selection module can help make a choice (best fit).

The algorithm 3*Dpacker* is implemented according to the mathematical model that follows:
Let n^i be the number of items assigned to pallet i and m the amount of pallets.
Let $A_j = 1$ if item t_j is assigned to the pallet, $A_j = 0$ otherwise, for $j \in \{1,\ 2,\ 3,\ ...,\ n^i\}$.
Let $X_{ij} = 1$ if item j was already assigned to pallet i by the previous resolution, $X_{ij} = 0$ otherwise.
Let $t_j.d,\ t_j.wh,\ t_j.h$ be parameters indicating the depth, width, and height of an item.
Let $p_i.Dh,\ p_i.Wh,\ p_i.H$ be parameters indicating the depth, width, and height of the pallet. We included letter h at the end of parameters $p_i.Dh$ and $p_i.Wh$ to make them different from $p_i.D$ (distance from CG) and $p_i.W$ (weight limit).
Let $x_j,\ y_j,\ z_j$ be real type variables for coordinates of item's left-bottom-behind corner.
Let $t_j.o^1,\ t_j.o^2,\ t_j.o^3,\ t_j.o^4,\ t_j.o^5,\ t_j.o^6$ be binary variables, indicating the possible orientations for item t_j.

$$\text{minimize}\Big[p_i.Dh \times p_i.Wh \times p_i.H - \sum_{j\leftarrow 1}^{n^i} X_{ij} \times A_j \times t_j.d \times t_j.wh \times t_j.h\Big] \tag{29}$$

$$\sum_{j\leftarrow 1}^{n^i} A_j \times t_j.d \leq X_{ij} \times p_i.Dh\ i \in \{1,\ 2,\ 3,\ ...,\ m\} \tag{30}$$

$$\sum_{j\leftarrow 1}^{n^i} A_j \times t_j.wh \leq X_{ij} \times p_i.Wh,\ i \in \{1,\ 2,\ 3,\ ...,\ m\} \tag{31}$$

$$\sum_{j\leftarrow 1}^{n^i} A_j \times t_j.h \leq X_{ij} \times p_i.H,\ i \in \{1,\ 2,\ 3,\ ...,\ m\} \tag{32}$$

$$t_j.o^1 + t_j.o^2 + t_j.o^3 + t_j.o^4 + t_j.o^5 + t_j.o^6 \leq 1 \tag{33}$$

Equation 29 minimizes the volume slack; equations 30, 31, and 32 constrains the packing to the three dimensions; and 33 guarantees that at most one orientation is chosen for the item.

Algorithm 6 presents the multiprocessing solution method for 3D packing to be applied to an existent solution, to guarantee that the item to be palletized actually fit into the pallet.

Algorithm 6 Multiprocessing 3D packing procedure

1: **procedure** $mp3Dpacking(\pi_k,\ G)$
2: Let $G(M^{\pi_k},\ N^{\pi_k} \cup Q^{\pi_k},\ E^{\pi_k})$
3: $proc \leftarrow \{p_1, p_2, ..., p_m\}$
4: **for** $i \leftarrow 1$ to m
5: $proc_i \leftarrow 3Dpacker(p_i,\ N_i^{\pi_k},\ \pi_k,\ G)$
6: $proc_i.fork$ ▷ pallet p_i packing processes is forked
7: **for** $i \leftarrow 1$ to m
8: $proc_i.join$
9: $E_i^{\pi_k} \leftarrow E_i^{\pi_k} \setminus proc_i.unfit$
10: **return** $G(M^{\pi_k},\ N^{\pi_k} \cup Q^{\pi_k},\ E^{\pi_k})$

According to line 2, this method receives an initial solution produced by a previous resolution, and according to line 3, each pallet has its own process.

In line 5, $N_i^{\pi_k}$ is the set of items previously assigned to pallet i and the method $3Dpacker()$ conforms to equations 29, 30, 31, 32, and 33.

In line 8, the algorithm waits for the packing process to finish.

Line 9 excludes the unfit items from solution.

5.5. Minimization of the unloading costs

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 aircraft 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.

So, to confirm or not Hypothesis (4), that the distances to move pallets on unloading in the next node could be minimized to benefit unloading costs. We modeled this improvement as an integer programming model.

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 π_k.

Let $Z_{iq}^{\pi_k}$ be the set of binary decision variables that relates pallet i to consolidated q in node π_k.

$$\text{minimize} \sum_{i=1}^{m}\sum_{q=1}^{m} Z_{iq}^{\pi_k} \times (maxD - p_i.D),\ \textbf{if}\ a_q^{\pi_k}.to = \pi(k+1)\} \tag{34}$$

$$s.t.: \sum_{i=1}^{m} Z_{iq}^{\pi_k} = 1;\ q \in \{1, 2, \ldots, m\} \tag{35}$$

$$s.t.: \sum_{q=1}^{m} Z_{iq}^{\pi_k} = 1;\ i \in \{1, 2, \ldots, m\} \tag{36}$$

$$p_i.w = 140 + a_q^{\pi_k}.w;\ i \in \{1, 2, \ldots, m\};\ q \in \{1, 2, \ldots, m\} \tag{37}$$

$$s.t.: \mid \sum\ p_i.D * p_i.w \mid \leq maxW \times limit_{long}^{CG};\ i \in \{1, 2, \ldots, m\} \tag{38}$$

Equation 34 minimizes the sum of distances of the consolidated in node π_k destined to the next node π_{k+1};

Equation 35 states that each consolidated will be assigned to exactly one pallet;

Equation 36 states that each pallet will receive one consolidated;

Equation 37 calculates the total weight on each pallet (note that it is different from $p_i.W$, the capacity of the pallet); and

Equation 38 guarantees that the torque limit will not be exceeded.

The instruction referred in Algorithm 2, line 19 implements this model.

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).

The final experiments are performed on a 64-bit, 16GiB, 3.4GHz, 8-core Intel® i7-3770 CPU, 2 threads per core, with *Linux Ubuntu 22.04 LTS* as the operational system and *Python 3.10.6* as the programming language.

6.1. Parameters tuning

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, 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 the *torque surplus - ts*, we input *iRace* with the range [1.0, 2.0] and this tool chose $ts = 1.7$ as a value that brings torque of the *Greedy* first phase to a feasible region.

6.2. Multiprocessing

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. In our case, the reading and writing process on the shared torque variable and the list of items to be embarked.

According to (Breshears, 2009, p.271), a *Process* is the operating system's spawned and controlled entity that encapsulates an executing application.

It is known that working concurrently opens up synchronization issues. But the *Multiprocessing* 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 *Multiprocessing* module lets the programmer take full advantage of the fact that a machine has more than one processor, normally capable of 2 threads each.

As in the larger aircraft the number of pallets is bigger than the number of threads available in a common handheld computer, for parallelization, we used the *Python Multiprocessing* package, which implements *process-based parallelism* or *parallel shared memory algorithm*. More details on *process-based parallelism* may be found in Python Software Foundation. (2022).

6.3. Results **before** *3D packing*

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

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

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

- **Normalized**: value between 0 and 1, which corresponds to the ratio between the sum of f values obtained by the method in all scenarios and the sum of the best values obtained among all methods in all scenarios. The higher the value of **Normalized**, the closer the method approached the best solutions found.

- **Speedup**: ratio of the sums of the worst run times of all scenarios and the sum of the method run times in all scenarios. The method with the highest **Speedup** 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 **Speedup** values.

Note that, in Table 6 all *Relaxed MIP* f value are italicized to indicate that these values may not be optimal or feasible as the decision variables are not integers.

Table 6: Volume surpluses, methods and scenarios results **before** 3D packing

Volume	**Resolution**		**Scenarios**						**Normalized**
surplus	*method*	**Results**	**1**	**2**	**3**	**4**	**5**	**6**	**Speedup**
1.2	*Relaxed MIP*	f	*12.23*	*8.67*	*12.67*	*15.36*	*16.29*	*16.20*	-
		run time (s)	3	13	20	78	198	1103	-
	Shims	f	10.34	**8.50**	**12.42**	**15.06**	**15.97**	**15.88**	**0.97**
		run time (s)	< 1	3	5	20	58	352	1.00
	mpShims	f	**12.96**	7.53	12.28	13.20	13.71	13.44	0.91
		run time (s)	< 1	2	2	8	22	120	**2.80**
1.5	*Relaxed MIP*	f	*17.01*	*12.45*	*18.02*	*22.03*	*23.09*	*22.35*	-
		run time (s)	4	20	32	115	296	1685	-
	Shims	f	14.68	**12.21**	**17.67**	**21.01**	**22.64**	**21.91**	**0.99**
		run time (s)	< 1	2	3	12	33	209	1.00
	mpShims	f	**16.12**	11.81	17.48	17.66	19.81	17.61	0.90
		run time (s)	< 1	1	2	7	20	120	**1.70**
2.0	*Relaxed MIP*	f	*24.56*	*17.47*	*25.19*	*27.94*	*29.90*	*27.60*	-
		run time (s)	4	23	34	119	317	1762	-
	Shims	f	21.31	**17.35**	24.24	**27.39**	**29.31**	**27.06**	**0.98**
		run time (s)	< 1	2	4	14	35	220	1.00
	mpShims	f	**24.33**	17.26	**24.70**	24.81	27.71	24.93	0.96
		run time (s)	< 1	2	3	9	26	151	**1.70**

The upper bound is the objective function value of a linear programming relaxation of the problem, demonstrating that no integer viable solution can have an objective value greater (in maximization) than this.

We solved all scenarios and instances with a MIP solver in a relaxed linear model. This is good for the reader to have an idea of the real performance differences between serial and process-based parallel *Shims*.

Shims had a quality performance very close to the *Relaxed MIP solver* upper bound solution, and, as to the **Speedup**, on average, *mpShims* was 2 times faster than *Shims*. Theoretically, it could be more, but due to the burden of sharing, for reading and writing, the CG deviation and the list of items among the processes (pallets), the *race condition* management limited *mpShims* performance.

We confirmed the hypothesis raised in 1 and in line 13 of Algorithm 3 that, in sequential mode, where one pallet is built at a time, torque constraints could be more restrictive than volume or weight constraints, which could stop the palletization process earlier, yielding a solution that may be not as good as it could be. This integer programming procedure inclusion noticeably improved all *Shims* results.

After this inclusion *Shims* results surpassed that of *mpShims*. In multiprocessing mode (*mpShims*) this inclusion (line 13) made a negligible difference.

6.4. Results **after** *3D packing*

Table 7 presents the results of the *Relaxed MIP solver* compared to *Shims* and *mpShims*, after the post-optimization with the multiprocessing 3D packing.

Although the *Relaxed MIP solver* results are not always feasible, the multiprocessing 3D packing turns them feasible regarding dimensions and weight. So, in Table 7, we considered it's results in the performance comparisons.

Table 7: Volume surpluses, methods and scenarios results **after** 3D packing (3Dp)

Volume	**Resolution**		**Scenarios**						**Normalized**
surplus	*method*	**Results**	**1**	**2**	**3**	**4**	**5**	**6**	**Speedup**
1.2	*Relaxed* $MIP + 3Dp$	f	11.47	7.24	**10.78**	12.06	12.97	12.44	0.95
		run time (s)	12	51	81	258	601	3161	**1.15**
	$Shims + 3Dp$	f	9.05	5.11	9.75	10.96	12.86	13.25	0.86
		run time (s)	16	68	94	323	719	3549	1.00
	$mpShims + 3Dp$	f	**12.14**	**7.29**	10.36	**12.63**	**14.04**	**13.92**	**0.99**
		run time (s)	9	36	54	192	427	2118	**1.70**
1.5	*Relaxed* $MIP + 3Dp$	f	15.83	**10.08**	**15.14**	**16.87**	18.58	17.52	0.97
		run time (s)	20	83	142	362	996	4741	**1.22**
	$Shims + 3Dp$	f	11.55	7.61	12.28	15.00	17.55	17.57	0.84
		run time (s)	25	107	156	498	1217	5743	1.00
	$mpShims + 3Dp$	f	**17.05**	10.01	15.01	16.75	**20.09**	**17.85**	**1.00**
		run time (s)	18	64	101	351	828	3879	1.50
2.0	*Relaxed* $MIP + 3Dp$	f	23.73	**15.23**	21.96	24.22	26.69	24.29	0.99
		run time (s)	28	99	185	648	1680	8774	1.00
	$Shims + 3Dp$	f	16.28	10.93	18.16	22.35	25.12	24.53	0.85
		run time (s)	37	141	196	653	1453	6851	1.22
	$mpShims + 3Dp$	f	**24.52**	15.18	**21.99**	**24.61**	**26.94**	**24.83**	**1.00**
		run time (s)	27	87	146	496	1080	5160	1.63

The average value of f obtained by *mpShims* was 0.997, followed by the *Relaxed MIP* with 0.971. *Shims* obtained 0.851, on average.

As to the performance, the average **Speedup** obtained by *mpShims* was 1.61, followed by the *Relaxed MIP* with 1.12. *Shims* reached 1.07, on average.

So, in terms of quality and speed, *mpShims* with the *Multiprocessing 3D packing* was the best method to solve the ACLP-RPDP.

6.5. Results on the minimization of the unloading costs

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

Table 8: Average distances minimization

Scenario	**Before (m)**	**After (m)**	**Improvement (%)**
1	32.7	28.7	−7.3
2	193.3	188.9	−6.3
3	165.1	153.3	−7.1
4	137.1	132.0	−3.7
5	126.7	122.0	−3.8
6	115.3	110.4	−4.2

Table 8 presents the distances minimization of the next node destined pallets to the ramp door. Columns **Before** and **After** refer to the application of the current method. On average, the distances from the pallets and the last ramp door position was diminished by 5% along a tour.

6.6. Discussion summary

In this work, we raised four hypotheses for improvements to the work of Mesquita and Sanches (2023):

(1) Computer parallelism could improve *Shims*'s performance: *mpShims* was twice as faster as *Shims*. This hypothesis was confirmed.

(2) The *Shims*' results may have been inferior in that of Mesquita and Sanches (2023); it could be that a sequential approach to pallet building may have lead to a premature stop due to the torque constraints. This hypothesis was confirmed as the results were improved by 5% on average with the inclusion of a CG balancing integer programming procedure.

(3) The procedure for 3D packing generated an unfit item rate of, on average, 13%. We are conscious that this unfit rate could be smaller if we had chosen Paquay et al. (2018) approach, but as we are concerned with

an operational application where the running time is crucial, we stick to faster approximate approaches like the *First Fit Decreasing* (first orientation that fits) and the *Best Fit* (best pivot point if more than one is available).

mpShims consider that, theoretically, all items selected for the pallets would fit. But, due to the variability in item dimensions, an actual complete fit is not possible, causing some unfit items to be excluded from the previous solution.

This solution phase was a multiprocessing 3D packing with each pallet as a process, which decreased this phase's run time by approximate a $n-1$ factor where n is the number of processes. This was possible because each packing process is completely independent; there is no multiprocessing shared memory. It is also important to highlight that, after the items were excluded, the CG deviation was kept within its operational and feasible range.

It is important to emphasize that, as the *Multiprocessing 3D packing* is a post-optimization procedure, it benefited all three resolution methods.

(4) The distances to move pallets on unloading in the next node could be minimized to benefit ground handling costs. This hypothesis was confirmed by the results on Table 8.

Another important fact is the inversion of quality results between *Shims* and *mpShims*. In Table 6, *Shims* was better, but, in Table 7 (with the inclusion of the *Multiprocessing 3D packing*), *mpShims* surpassed *Shims*' results.

The rationale behind this is that, in sequential mode (*Shims*), items with smaller volumes (higher edge attractiveness $\eta_{ij}^{\pi_k}$, Equation 25) tend to be assigned first to central pallets, and items with bigger volumes tend to be assigned to the ramp door or to the forward-positioned pallets; these bigger items are more difficult for the 3D packer actions. On *mpShims*, as all pallets are built in parallel, the size distribution is more uniform, with a better size diversity to be allocated to each pallet, which is less difficult for the 3D packer actions due to the variety of item dimensions for packing. This explains why *mpShims* exceeded *Shims*' results after the *Multiprocessing 3D packing*.

7. Conclusions

In this work we dealt with a hard problem, named *Air Cargo Load Planning with Routing, Pickup, and Delivery Problem* considering standardized pallets in fixed positions, obeying the center of gravity constraints, delivering each item to its destination, and minimizing fuel consumption costs.

We converted *Shims* into a process-based parallel computing heuristic, which we called *Multiprocessing Shims* or *mpShims*, that quickly finds good solutions for a wide range of problem sizes, our first contribution.

We also carried out 126 experiments with each method on synthetic data based on real data from the *Brazilian Air Force* transportation history.

Other important contributions are that we verified four hypotheses: Computer parallelism improved *Shims* performance; the intermediate integer programming procedure added to *Shims* algorithm enhanced it's results; a 3D packing method was turned possible for operational use due to a process-based parallel computing heuristic, and the integer programming procedure to minimize pallets' travel in the cargo bay was successful.

And another important contribution is that we employed the *iRace* package to fine tune the *mpShims* volume threshold and torque surplus 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.

Dube, E., Kanavathy, L.R., 2006. Optimizing three-dimensional bin packing through simulation, in: Modelling, simulation, and optimization, Durban, South Africa. pp. 507–034.

Fok, K., Chun, A., 2004. Optimizing air cargo load planning and analysis, in: Proceedings of the International Conference on Computing, Communications and Control Technologies, p. 200.

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.

Li, X., Zhao, Z., Zhang, K., 2014. A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins., in: Institute of Industrial and Systems Engineers (IISE), p. 200.

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 , XX–XX.

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.

Pacheco, P., Malensek, M., 2021. An Introduction to Parallel Programming 2nd Edition. Morgan Kaufmann. doi:https://doi.org/10.1016/C2015-0-01650-1.

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 Software Foundation., 2022. Python 3.10.10 documentation: 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.