4OR (2026) 24:81–121
https://doi.org/10.1007/s10288-026-00614-5

INDUSTRY

A recursive algorithm for synchronized rich vehicle routing in air cargo ground handling

Jenny Tonka[1] · **Célia Paquay**[1] · **Michaël Schyns**[1]

Received: 14 October 2025 / Revised: 12 January 2026 / Accepted: 24 January 2026 /
Published online: 27 March 2026
© The Author(s) 2026

Abstract
We study a rich vehicle coordination problem met by Liège Airport, a major cargo airport in Europe. We focus on air cargo ground handling: a set of services should be provided by different vehicle types to different clients at different locations within defined time intervals, and various vehicle interdependencies exist. We first formalize the problem as a set of rich Vehicle Routing Problems that are bound together by multiple synchronization constraints. In particular, we consider compulsory vehicle pairings, precedence constraints between services, and goods transfers between vehicles of different types. The latter can involve more than two vehicles, implying cascading dependencies that greatly complicate the problem. We then develop a client-centered greedy heuristic using a recursive procedure that is able to solve the problem as a whole without having to handle constraints separately, as is often the case in vehicle routing problems with such complex structures. We seek to minimize total service time to produce safety time buffers that would help absorb the impact of disruptions and subsequently reduce the number and duration of aircraft delays. Tests on real instances show that the algorithm performs well.

Keywords Industrial application · Air cargo ground handling · Routing · Synchronization · Recursive heuristic

1 Introduction

This paper investigates a vehicle coordination problem experienced by Liège Airport (LGG), a major European cargo airport. Specifically, the paper formalizes the prob-

✉ Jenny Tonka
j.tonka@uliege.be

Célia Paquay
cpaquay@uliege.be

Michaël Schyns
M.Schyns@uliege.be

[1] Centre for Quantitative methods and Operations Management, HEC Liège - Management School of the University of Liège, Liège, Belgium

lem as a set of highly interdependent and rich vehicle routing problems (VRPs), and proposes a solution method that could be adapted to other applications presenting the same problem structure.

Most authors agree that the first paper to approach the topic of VRPs was that of Dantzig and Ramser (1959). Given the importance of VRPs for the logistics activities of many industries, considerable scientific research has been carried out since then. Over time, VRPs have been enriched with various features and many subfields have emerged (see, e.g., Laporte 1992, 2009; Eksioglu et al. 2009; Toth and Vigo 2014; Braekers et al. 2016; Elshaer and Awad 2020; Mor and Speranza 2020; Konstantakopoulos et al. 2022, Archetti et al. 2025). In particular, VRPs that incorporate many complex features of real-life problems are grouped under the name of rich VRPs (Caceres-Cruz et al. 2014; Lahyani et al. 2015). Since VRPs are known to be NP-hard, solving them at optimality is not an easy task. Some exact methods have been developed, but authors traditionally prefer heuristic approaches, which are often more reasonable in terms of computing time when dealing with richer or larger problems. Another common approach is to focus on simplified problems that address only a limited subset of constraints. It is therefore challenging to find good solutions for real-life problems that integrate a considerable number of different characteristics.

The objective of this paper is to optimize air cargo ground handling by enhancing the coordination of ground service vehicles. Air cargo ground handling refers to the logistical activities that are performed between the time a cargo aircraft lands and takes off again. As can be seen in Figures 1 and 2, it requires many different vehicle types to provide all those services to aircraft. On Figure 2, services presented in parallel can be performed simultaneously, while those in series should be conducted sequentially due to technical field constraints (e.g., cleaning cannot be carried out until the toilets have been emptied). This corresponds to the reality of LGG, but other services or ways of dealing with the services may be possible in other airports. Airport ground handling (AGH) is already the focus of many articles as it is a key issue in air transportation. However, most of them focus only on the passenger side of AGH. We focus on the cargo side, which entails new constraints that make the problem much more complex.

The first contribution of this paper is to formalize the problem of cargo AGH that is experienced by LGG. This is a problem where a set of services should be provided by different vehicle types to different clients at different locations within defined time

[1] Lift between ground and aircraft door, [2] Transfer vehicle between HL and dollies, [3] Non-motorized trolleys, [4] Towing vehicle for dollies

Fig. 1 Air cargo ground handling vehicles

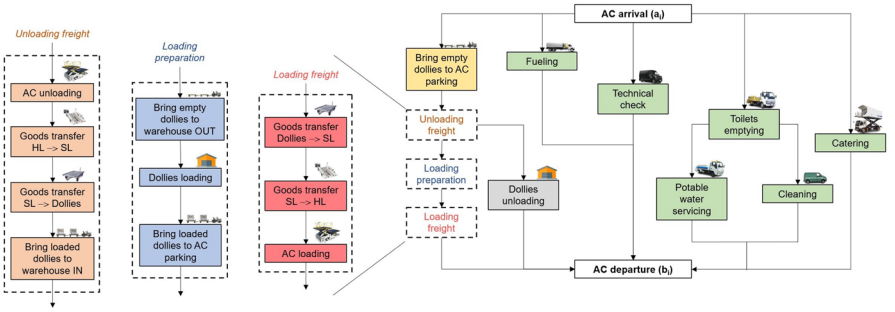


Fig. 2 Air cargo ground handling precedence diagram

intervals, and where various synchronization constraints exist. In particular, compulsory vehicle pairings, precedence constraints between services, and goods transfers between vehicles of different types may appear. Moreover, many additional problem features should be considered to reflect the reality of the field, such as time windows, heterogeneous capacitated vehicle fleets, forbidden pairings, split deliveries, multi-depot, multi-trip, and specific constraints like multi-vehicle services or multi-operation vehicles.

The second contribution of this paper is to propose an effective algorithm to solve a problem with such a complex structure. In this regard, we have designed a client-centered greedy heuristic that uses a recursive procedure to handle the cascading dependencies caused by goods transfer chains of variable length. Goods transfers can indeed involve more than two vehicles, as in the case of aircraft unloading, where containers are transferred from a high loader to speed loaders and then to dollies (see Figure 2). We chose to minimize total service time so as to produce safety time buffers that would help absorb the impact of disruptions and subsequently reduce the number and duration of aircraft delays. This requires services to be completed as soon as possible, thereby maximizing responsiveness. In addition, we believe that our solution method could be adapted to other applications as long as they can also be modeled as a set of highly interdependent rich VRPs that interact with each other through multiple synchronization constraints, thus extending the contributions of this paper to the growing research stream on VRPs with synchronization (Soares et al. 2024).

The remainder of this article is organized as follows: Sect. 2 provides a brief literature review. Sect. 3 details the problem and its main characteristics. Sect. 4 is dedicated to the solution method description, while results on real instances are presented in Sect. 5. Conclusions are finally drawn in Sect. 6.

2 Literature review

The following literature review does not aim to address all generic aspects of VRPs. Instead, Sect. 2.1 provides an overview of AGH, while Sect. 2.2 focuses on VRPs with synchronization, which is the most relevant VRP aspect for our study.

2.1 Airport ground handling

Air transportation is a major actor in e-commerce, with many logistical challenges yet to be overcome. In particular, ground handling, and more precisely ground service vehicle coordination, have been identified by the industry as the most critical part of the supply chain (Padrón and Guimarans 2019). With the continued growth of e-commerce, it becomes urgent to limit aircraft delays as much as possible. Indeed, beyond their negative impact on client satisfaction, aircraft delays result in penalties, fines and additional operation costs for airlines concerned, especially when they exceed 15 minutes (Britto et al. 2012). From a sustainable viewpoint, they may also cause additional environmental damage by increasing fuel consumption and gas emissions (Ryerson et al. 2014; Krstić Simić and Babić 2015).

All of these incentives have led to extensive research on the topic, but no solutions have yet been developed that encompass the entire problem structure. In the literature, many authors focus only on a specific ground operation type and consequently on a single vehicle fleet and its respective VRP or scheduling problem, meaning that they do not address the prioritization problem induced by precedence constraints nor the vehicle interdependency problem generated by goods transfer chains and movement synchronization. Among them, Norin et al. (2012) address the routing of de-icing vehicles, while Du et al. (2014) do the same for aircraft towing vehicles, Schyns (2015) for refueling trucks, and Han et al. (2023) for ferry vehicles. In addition, for those considering several service types at once (e.g., Ip et al. 2013; Padrón et al. 2016; Padrón and Guimarans 2019; Gök et al. 2020; Gök et al. 2023; Wu et al. 2023; Zhou et al. 2023), none considers goods transfer between vehicles, which is fundamental to reflect the real-life problem. Most authors also consider very simplified hypotheses or solve the problem by iteratively solving each VRP or scheduling problem individually. Interested readers in airport ground handling may additionally refer to Andreatta et al. (2014), Fink et al. (2019), Tomasella et al. (2019), Wang et al. (2020), Evler et al. (2021), Guimarans and Padrón (2022), Zhu et al. (2022), or Chen et al. (2023).

The solution method proposed in this article considers all the AGH vehicle synchronization constraints, including goods transfers, and manages to solve the different VRPs simultaneously. To the best of our knowledge, this paper is the first to provide such a complete solution method for the AGH problem.

2.2 VRPs with synchronization

Although the literature on VRPs is extensive, vehicle synchronization remains much less covered despite its high relevance to real-life problems. According to Drexl (2012), the high complexity of this constraint explains why it has long been not considered

by the scientific community. When VRPs have synchronization constraints related to either spatial, temporal, or load aspects, the different vehicles are no longer independent from each other. This has a significant impact on the possible solution approaches since most VRP exact and heuristic algorithms rely on the assumption that routes are mutually independent (Drexl 2012). Additional work is therefore required to successfully deal with the underlying complexity and combinatorial nature of VRPs with synchronization (Soares et al. 2024).

Drexl (2012) was the first to propose a classification scheme for vehicle dependencies. However, Soares et al. (2024) pinpointed a lack of consensus on the different types of synchronization that can appear in VRPs. They therefore decided to rearrange the categorization of Drexl (2012) and present a classification scheme that groups all dependency types found in the literature into four categories:

- *Local aspects* refer to interdependencies that occur within the route of a particular vehicle;
- *Synchronization aspects* refer to interdependencies that appear between different routes. Here appear *movement* and *operation synchronizations* of Drexl (2012);
- *Global aspects* refer to interdependencies among the whole routing problem. They include *task*, *resource*, and *load synchronizations* of Drexl (2012);
- *Integration aspects* refer to problems where the interdependency is established between the routing problem and another optimization problem.

AGH combines different local, synchronization, and global aspects. Beyond this distinction between dependency types, Soares et al. (2024) also consider different categories of VRPs with synchronization. AGH belongs to three of them:

- *VRP with synchronization of schedules*: much of the literature on VRPs with synchronization falls into this category, with precedence constraints being its predominant aspect. It is also frequent to consider qualified vehicles (or skilled staff members), meaning that not all vehicles (resp. staff members) may be able to perform certain services. Schedule synchronization is considered in various application areas, such as construction, electric vehicle charging, or railway transportation for instance (see, e.g., Fedtke and Boysen 2017; Abdzadeh et al. 2022; Froger et al. 2022). However, home health care is the most prevalent problem application of this category (see, e.g., Bredström and Rönnqvist 2008; Cappanera et al. 2020). A robust optimization approach for such problems is provided by Soares et al. (2025). Regarding cargo AGH, both precedence constraints (see Figure 2) and qualified vehicles (see Figure 1) are considered.
- *VRP with trailers or passive vehicles*: the main characteristic of this category is that it considers vehicles that require another vehicle to move in space. Here also, various articles can be found in the literature (e.g., Chao 2002; Drexl 2013; Meisel and Kopfer 2014; Tilk et al. 2018; Hu and Wei 2018). Regarding AGH, two types of vehicles are considered: active vehicles, which can move by themselves and visit clients (e.g., refueling trucks), and passive vehicles, which cannot move on their own but can be used as mobile depots to which active vehicles may transfer their load (e.g., dollies).
- *VRP with transfer or cross-docking requirements*: such VRPs require temporal precedence between the unloading and loading tasks of different vehicles to ensure

that the unloading of merchandise at transfer/cross-docking locations is over before their collection by another vehicle. Representative papers in this category are those considering pick-up and delivery problems with transfers (e.g., Masson et al. 2013; Aguayo et al. 2025), VRPs with cross-docking (e.g., Grangier et al. 2021), and multiple-echelon VRPs (e.g., Dellaert et al. 2021). Another example is the feeder vehicle routing problem, which involves transfers between vehicles of different sizes at certain customer locations, enabling rapid customer service in urban areas (e.g., Huang et al. 2019). Cargo AGH is different in that it deals with the direct transfer of goods between vehicles (high loaders, speed loaders, and dollies) without having to consider specific transfer locations, which makes the problem much more complex and has important algorithmic implications. This has been partly introduced by Wang et al. (2024) in the context of container transfers with automated stacking cranes and guided vehicles.

So far, vehicle dependencies have been partially discussed in the scientific literature, and only in an individual and simplified manner. In contrast, the present article proposes a solution method that addresses a problem with multiple synchronizations of a complex form that perfectly matches the ground realities.

3 Problem description

To formalize the cargo AGH problem experienced by LGG, we consider a set of aircraft (i.e., clients) in different locations that require a set of services (Figure 2) to be provided by different types of vehicles (Figure 1) within defined time intervals. Vehicle types differ from each other in terms of the services they are able to perform. Formally, we consider a set of vehicle fleets $\mathcal{V}$, where each fleet $\mathcal{V}_\alpha \subset \mathcal{V}$ consists of vehicles of a certain type α. For each service $p \in \mathcal{P}$, there is only one vehicle fleet $\mathcal{V}_\alpha \subset \mathcal{V}$ able to perform the service. In addition, for each fleet $\mathcal{V}_\alpha \subset \mathcal{V}$, there exists a set $\mathcal{P}_\alpha$ containing all services $p \in \mathcal{P}$ that $\mathcal{V}_\alpha$ is able to perform. This brings a set of rich vehicle routing problems, each of which refers to a specific vehicle type fleet $\mathcal{V}_\alpha$.

Each VRP can be represented by a complete digraph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \mathcal{D}_\alpha \cup \mathcal{C}$ is the set of nodes and $\mathcal{A} = \{(i, j) \mid i, j \in \mathcal{N}, i \neq j\}$ is the set of arcs. $\mathcal{D}_\alpha$ denotes the set of available depots for vehicles of type α, while $\mathcal{C}$ denotes the set of clients to be served. Notations are summarized in Table 1. This generic problem is exemplified in Figure 3, where we see on the left the routes of a fleet of vehicles of one type and on the right the routes of a fleet of vehicles of another type, both serving the same set of clients. These VRPs are actually interdependent due to various synchronization constraints described later in this section.

Note that this section does not provide a complete mathematical model. Indeed, due to the complexity induced by the numerous problem features considered and the dependencies that bind the different VRPs together, it is very likely that exact methods would not succeed in finding an optimal solution, or at least a good feasible one, within a reasonable time frame. The cascading dependencies caused by goods transfer chains of variable length are, moreover, extremely difficult to handle with an exact approach. This therefore led us to adopt a recursive heuristic approach, for

Table 1 Notations

Sets (indices)	
$\mathcal{V}$	Set of vehicle fleets
$\mathcal{V}_\alpha$	Set of vehicles of type α (k, l)
$\mathcal{K}_\alpha$	Set of vehicle fleets with which vehicles of type α can be paired
$\mathcal{P}$	Set of services $(m,\ p)$
$\mathcal{P}_\alpha$	Set of services vehicles of type α are able to perform
$\mathcal{D}_\alpha$	Set of depots for vehicles of type α (d)
$\mathcal{C}$	Set of clients $(i,\ j)$
$\mathcal{C}_k$	Set of clients vehicle k is allowed to serve
Parameters	
q_{ip}	Quantity that should be delivered to client i for service p
a_i	Earliest start time of any services at client i
b_i	Latest allowed end time of any services at client i
cap_k	Capacity of vehicle k
c_p^k	Constant time required by vehicle k for setup and post-operations of service p
v_p^k	Variable time required by vehicle k to deliver one unit of product in service p
c_d^k	Constant time required by vehicle k for setup and post-operations at depot d
v_d^k	Variable time required by vehicle k to replenish one unit of product at depot d
t^{kl}	Transfer time of one unit of product from vehicle k to vehicle l
Variables	
A_{ip}	Earliest start time of service p at client i
S_{ip}	Start time of service p at client i
E_{ip}	End time of service p at client i
S_{ip}^k	Start time of vehicle k performing service p at client i
E_{ip}^k	End time of vehicle k performing service p at client i
E_{ip}^*	Updated end time of service p at client i in case of goods transfer
Q_{ip}^k	Part of q_{ip} covered by vehicle k
R_k	Remaining capacity of vehicle k updated after each visit
O_d^k	Refilling time of vehicle k at depot d
O_{ip}	Operating time of service p at client i
O_{ip}^k	Operating time of vehicle k for its part of service p at client i
ST	Total service time

which a complete mathematical representation is not necessary. Still, to help the reader better understand the AGH problem experienced by LGG, we chose to mathematically formalize most of the constraints and provide concrete examples whenever possible. However, we deliberately omit the mathematical formulation of the variable-length cascading constraint presented in Sect. 3.3.2, as it would result in a cumbersome model

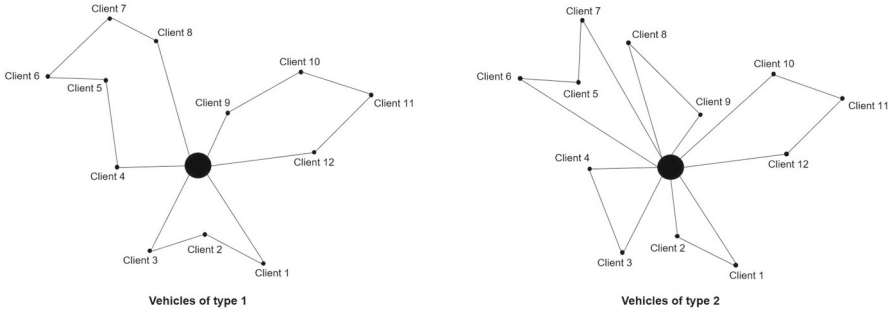


Fig. 3 Set of routes for two fleets of different types

that would be difficult to understand and would fail to accurately reflect the specific recursive approach we have implemented (see Algorithm 3).

3.1 VRP features

The rich VRPs we consider are characterized by many different features. First, services should be provided to clients within **time windows**. Each client $i \in \mathcal{C}$ is available during the time window $[a_i, b_i]$, where a_i (resp. b_i) is the aircraft scheduled arrival (resp. departure) time. We talk about allowed ground time since $[a_i, b_i]$ refers to the time available to carry out all ground operations on aircraft i. Soft time windows are considered because delays can occur in real life, but are penalized by an increase in the value of the objective function. In addition, a vehicle is allowed to arrive in advance (before the earliest service start time of a given client), but has to wait until the beginning of the time window to start the service. Note also that it is essential to introduce the variable A_{ip}, the earliest start time of service p at client i, to deal with the additional constraints on start times imposed by the synchronization between vehicles described in Sect. 3.3. The service time window of service $p \in \mathcal{P}$ at client $i \in \mathcal{C}$ is therefore $[A_{ip}; b_i]$. Obviously, one has $A_{ip} \geq a_i$ for all services $p \in \mathcal{P}$. The start time of service p at client i, denoted by S_{ip}, should be such that $S_{ip} \geq A_{ip}$. Its end time is referred to as E_{ip}.

Then, for each client $i \in \mathcal{C}$ and service $p \in \mathcal{P}$, there is a positive quantity q_{ip} that should be delivered. Some vehicles may have a limited capacity (**capacitated vehicles**), as refuelling trucks for instance, meaning that they cannot deliver more volume than their full capacity on a given route. Mathematically, each time a vehicle $k \in \mathcal{V}_\alpha$ serves a client $i \in \mathcal{C}_k$ for a given service $p \in \mathcal{P}_\alpha$, its remaining capacity R_k is decreased by the part of q_{ip} it covers, denoted by Q^k_{ip}, with $Q^k_{ip} \leq q_{ip}$. Therefore, if $\mathcal{M}$ is the set of clients to be served by a vehicle $k \in \mathcal{V}_\alpha$ for a service $p \in \mathcal{P}_\alpha$ on a given route, the inequality $\sum_{i \in \mathcal{M}} Q^k_{ip} \leq cap_k$, where cap_k is the full capacity of vehicle k, should be satisfied. Yet, capacity recovery is possible. We consider a set of depots $\mathcal{D}_\alpha$ (**multi-depot**) which are always open and where compatible capacitated vehicles are allowed to return for replenishment. Once completed ($R_k = cap_k$), the considered vehicle is available again and can start a new route if needed. **Multi-trips** are therefore

allowed. Note finally that **split deliveries** are also accepted, although naturally limited by our objective function (see Sect. 3.2), meaning that a given client i may be served several times for the same service p. A service p at client i is considered completed only when the whole demand for that service has been satisfied, i.e., $\sum_k Q_{ip}^k = q_{ip}$. Note that this is easily adapted to pick-up services such as freight unloading or toilets emptying.

Next, in addition to various vehicle capacities, each fleet of vehicles $\mathcal{V}_\alpha \subset \mathcal{V}$ is also **heterogeneous** in terms of vehicle-client compatibilities. Indeed, in each fleet, some vehicles are not allowed or appropriate to serve some clients for different reasons (e.g., huge refueling trucks will not be suitable for small aircraft). As a consequence, **forbidden pairings** appear, meaning that a vehicle cannot visit clients for which an incompatibility has been pinpointed. This can be extended to the case of multiple service providers serving only aircraft of specific airlines. Formally, for each vehicle $k \in \mathcal{V}$, there exists a set of clients $\mathcal{C}_k$ that vehicle k can serve. In addition, vehicles in the same fleet may have different service and replenishment operating speeds. The operating time of vehicle k for service p at client i is defined by $O_{ip}^k = c_p^k + (v_p^k \times Q_{ip}^k)$ and the replenishment time of vehicle k at depot d is given by $O_d^k = c_d^k + (v_d^k \times (cap_k - R_k))$, where c_p^k (resp. c_d^k) is the constant time required by vehicle k for setup and post-operations of service p (resp. at depot d) and v_p^k (resp. v_d^k) is the variable time required by vehicle k to deliver (resp. replenish) one unit of product in service p (resp. at depot d). This must be distinguished from $O_{ip} = E_{ip} - S_{ip}$, that denotes the total time required to perform service $p \in \mathcal{P}$ at client $i \in \mathcal{C}$ considering all the vehicles involved. $S_{ip} = \min_{k \in \mathcal{L}} S_{ip}^k$ and $E_{ip} = \max_{k \in \mathcal{L}} E_{ip}^k$, where $\mathcal{L}$ is the set of vehicles performing service p at client i, and S_{ip}^k (resp. E_{ip}^k) is the start (resp. end) time of vehicle k service p at client i.

In addition to all of these classic VRP features, the problem at hand also considers other VRP characteristics that are more complex. First, it includes some **multi-vehicle services**, which allow multiple vehicles of the same type to perform part of the service simultaneously at a given client location. In the case of split deliveries, the time at which the second vehicle will be able to start the service depends on whether it is a single-vehicle or multi-vehicle service. Formally, let $k_1, k_2 \in \mathcal{V}_\alpha$ be the vehicles assigned to client $i \in \mathcal{C}_{k_1} \cap \mathcal{C}_{k_2}$ for service $p \in \mathcal{P}_\alpha$, with vehicle k_1 being the first vehicle performing the service. In the case of a single-vehicle service (e.g., only one tanker truck is allowed to refuel an aircraft at a time), vehicle k_2 cannot start serving client i until vehicle k_1 has left the service area and therefore $S_{ip}^{k_2} \geq E_{ip}^{k_1}$. On the other hand, in the case of a multi-vehicle service (e.g., two speed loaders are allowed to operate simultaneously when transferring outgoing containers from dollies to the high loader), vehicle k_2 can start serving client i as soon as it is permitted to start the service, regardless of whether vehicle k_1 is still in the service area or not. Consequently, the only constraint to consider here is $S_{ip}^{k_2} \geq A_{ip}$. Note, however, that for some multi-vehicle services, the maximum number of vehicles of the same type that may be present at the client location at any one time is restricted by the capacity of the service area.

Another distinctive characteristic of the studied problem is that a fleet of vehicles $\mathcal{V}_\alpha$ can be **multi-operations** (i.e., $|\mathcal{P}_\alpha| > 1$), meaning that all vehicles $k \in \mathcal{V}_\alpha$ are

able to perform different services. Yet, the services a vehicle is able to perform are closely related to each other, we therefore assume that a multi-operation vehicle k has a single capacity R_k that is common to all the services it can perform. For example, high loaders can be used to both load and unload aircraft with containers, but has only one capacity that increases or decreases depending on the service considered. Of course, this could be extended to multiple service-related capacities if needed.

To summarize, each VRP considered in AGH can therefore be defined as a Multi-Depot Heterogeneous Multi-Operation Capacitated VRP with Time Windows, Forbidden Pairings, Multi-Trip, Split Deliveries, and Multi-Vehicle Services. Considering the classification scheme of Soares et al. (2024), those VRP features include both local and global aspects.

3.2 Objective function

Even though each subproblem takes the form of a rich VRP, we do not try to minimize the total distance traveled or the number of vehicles used, as it is often the case. Instead, we seek to minimize the total time all clients must wait until their entire set of required services is completed. We call this time the total service time, labeled ST, and define it as follows:

$$ST = \sum_i \max_{p \in \mathcal{P}} E_{ip} - a_i \quad \forall i \in \mathcal{C}$$

where $\mathcal{P}$ is the set of services requested by client i, E_{ip} is the end time of service p at client i, and a_i is the earliest start time of any service at client i.

The main idea here is to create safety time buffers that help produce more robust solutions. Indeed, looking at a client $i \in \mathcal{C}$, minimizing the service time of client i is equivalent to minimizing the end time of the last service requested by client i (i.e., $\max_{p \in \mathcal{P}} E_{ip}$). This in turn maximizes the size of the safety time buffer before aircraft scheduled departure time $b_i - \max_{p \in \mathcal{P}} E_{ip}$, or minimizes the duration of the delay produced in case $\max_{p \in \mathcal{P}} E_{ip}$ exceeds b_i. Looking at the management side of the problem, we seek to maximize responsiveness, as defined in Schyns (2015). Such an objective is much more in line with AGH's field reality than those based solely on distance traveled or number of vehicles used. Nevertheless, these objectives remain partially considered, since our algorithm tends to both reduce travel distance and maximize vehicle utilization (see Sect. 4.3).

3.3 Synchronization constraints

The rich VRPs we consider interact with each other through compulsory vehicle pairings, precedence between services, and goods transfers. These synchronization constraints appear at the client level and bind the different vehicle types and their respective VRPs together in different ways. They fall within the scope of the local and synchronization aspects described by Soares et al. (2024).

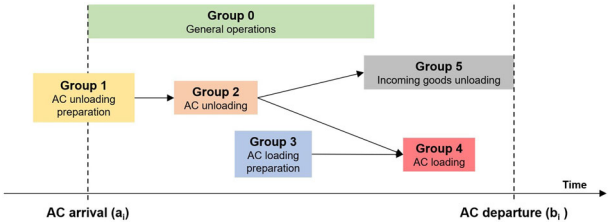


Fig. 4 Precedence between groups of services in cargo AGH

3.3.1 Precedence between services

If, on a given client, a service cannot be started before another one or several other ones are completed, that service is said to be subject to precedence constraints. On the one hand, this dependency may appear between two services provided by two different fleets of vehicles, and therefore between two distinct VRPs. For instance, we can see in Figure 2 that on a given aircraft, potable water servicing as well as cleaning can only take place after toilets have been emptied. On the other hand, the precedence may also appear between services performed by vehicles belonging to the same fleet and therefore to the same VRP. This comes from the fact that some vehicles are able to perform different services (multi-operation vehicles). A simple example here is that high loaders must unload the aircraft first before loading it again.

Precedence constraints impact service time windows, and particularly service earliest start times: the earliest start time of a service at a given client should be later than or equal to the latest service end time of its direct predecessors at the same client. Formally, let $\mathcal{P}^1 \subset \mathcal{P}$ be the set of services that must be completed before service m can start at client $i \in \mathcal{C}$, the earliest start time of service m at client i, A_{im} (as defined in Sect. 3.1), should satisfy:

$$\max_{p \in \mathcal{P}^1} E_{ip} \leq A_{im} \qquad \forall i \in \mathcal{C} \tag{1}$$

This can be extended to precedence between groups of services. A group of services refers to a set of services that form a particular global process with its own time window. While some global processes can run in parallel, there may be precedence relationships between others. Groups of services of cargo AGH are indicated by colors in Figure 2, and is summarized in Figure 4, where arrows represent precedence constraints. For instance, we can see that services belonging to group 0, which refers to general operations such as refueling, cleaning, or toilet emptying, can run in parallel with any other groups of services. In contrast, group 4, which refers to aircraft loading services, should wait until the end of group 2, aircraft unloading services, as well as group 3, which includes loading preparation services, before being allowed to start.

Formally, suppose a group of services, denoted by $\tilde{x}$, should precede a group of services, denoted by $\tilde{y}$. This means that all the services in $\tilde{x}$ should be finished before any of the services in $\tilde{y}$ can start. In particular, this implies that the end time of the

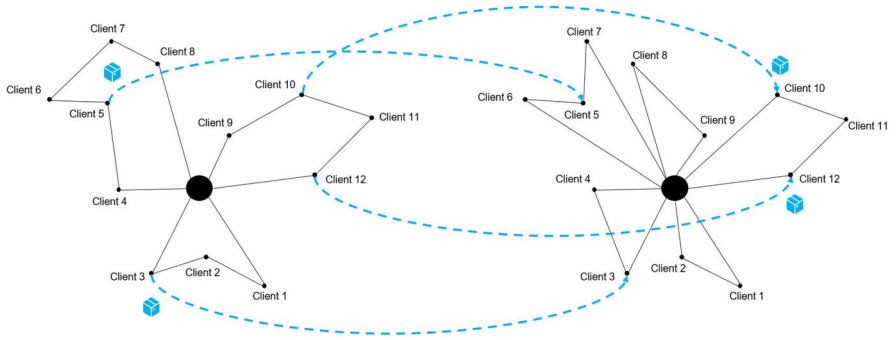


Fig. 5 Goods transfers between different vehicle types

last service in group $\tilde{x}$ should be earlier than or equal to the earliest start time of the first service in group $\tilde{y}$. Thus, constraints (1) can be extended as follows:

$$\max_{p \in \tilde{x}} E_{ip} \leq \min_{m \in \tilde{y}} A_{im} \quad \forall i \in \mathcal{C}$$

3.3.2 Goods transfers

To the best of our knowledge, this paper is the first to consider goods transfers, which is fundamental for cargo AGH. It is a major contribution since goods transfers between vehicles performing different services, represented by the blue arrows in Figure 5, are the most impactful constraints of AGH. Moreover, such a constraint may entail vehicle chains of variable length, implying cascading dependencies. This complicates further the problem and leads to the use of a recursive procedure in the solution method to ensure both feasibility and efficiency.

A vehicle involved in a goods transfer belongs to different VRPs and takes on different roles. More precisely, a vehicle involved in a goods transfer takes on both the role of the main vehicle performing services in its own VRP and the role of a depot in the VRP of the next vehicle in the chain. This goes far beyond the synchronization types described in Drexl (2012) and has many implications for the algorithmic approach developed in Sect. 4. In addition, goods transfer can involve more than two vehicles, creating chains that can be of variable length. For instance, looking at the unloading group of services, detailed below in Figure 6, we can see a goods transfer chain of three vehicles going from a high loader (HL) to speed loaders (SL) and then to dollies (D). Here, the HL plays the vehicle role in its own VRP when unloading the aircraft, but it takes on the role of the depot providing the goods when considering the VRP where the SL is the main vehicle performing the services. Therefore, in a goods transfer chain, although we start with a single depot and end up with a single client, as in a classic VRP, the path between the two is not carried out by a single vehicle, but rather by a succession of vehicles that alternately take on the roles of vehicle and depot. From this, and knowing that some services may involve multiple vehicles, sometimes there are multiple vehicles with the role of depot to be emptied with the same product, we

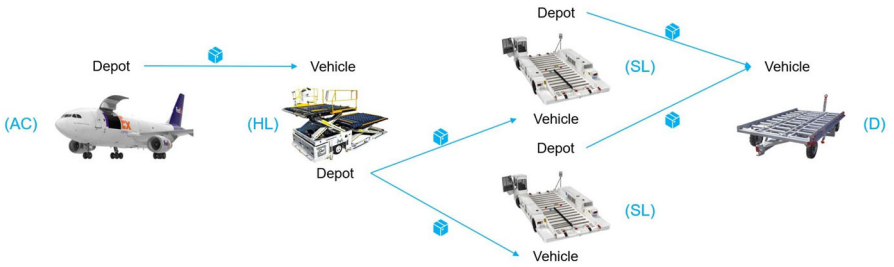


Fig. 6 Vehicle role evolution in a goods transfer chain

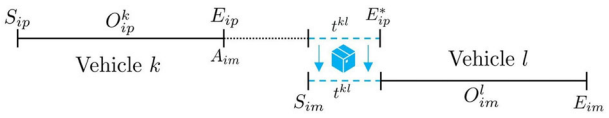


Fig. 7 Goods transfers and time management

talk about multiple resource providers. For instance, during aircraft cargo unloading (see Figure 6), two SL can be used to unload the HL. At some point, dollies have thus two depots offering the same product to be emptied. Beyond selecting the vehicle performing the service, it is therefore also necessary to select the intermediate depot from which the product is to be retrieved, which further complicates the problem.

In addition, when there is a transfer between two vehicles, the operating time of the loaded vehicle depends on the arrival time of the unloading vehicle, as depicted in Figure 7. Indeed, even though it has performed its service, the first vehicle is not released until its load is completely transferred to the second vehicle. Therefore, here, the end time of a service does not indicate the moment when the vehicle becomes available, since it is necessary to take into account an additional time, that of goods transfer. Similarly, the second vehicle in the chain cannot start its service until the first vehicle requests unloading. Its earliest start time is therefore conditioned by the service end time of the vehicle to be unloaded. For instance, in Figure 7, vehicle k completes service p at client i by E_{ip}, waits until S_{im} to be unloaded by vehicle l, and is available again from E^*_{ip} when the goods transfer is over, while vehicle l completes service m at client i by E_{im}.

Formally, consider service $m \in \mathcal{P}_\beta$ at client $i \in \mathcal{C}_k \cap \mathcal{C}_l$, which consists of collecting the goods gathered upstream by service $p \in \mathcal{P}_\alpha$ at client i. For the sake of clarity, we assume that the quantity requested by client i for service p (and thus the quantity to be transferred between vehicles) is 1, and that service p (resp. m) at client i is performed by a single vehicle $k \in \mathcal{V}_\alpha$ (resp. $l \in \mathcal{V}_\beta$). We then get the following constraints:

$$A_{im} \geq E_{ip} \tag{2}$$

$$E^*_{ip} = E_{ip} + (S_{im} - E_{ip}) + t^{kl} = S_{im} + t^{kl} \tag{3}$$

$$E_{im} = S_{im} + t^{kl} + O_{im} \tag{4}$$

where $E_{ip} = S_{ip} + O_{ip}$ is the end time of service p at client i, $(S_{im} - E_{ip})$ is the waiting time for vehicle k for vehicle l to arrive, and t^{kl} is the transfer time of one unit of product from vehicle k to vehicle l. Equation (2) implies that vehicle l cannot start its service m until vehicle k has completed its service p. Equation (3) defines the actual time, E^*_{ip}, at which vehicle k is available again after completing service p at client i and transferring goods to vehicle l. Equation (4) adds the goods transfer time to the operating time of vehicle l. These equations are obviously much more complex when vehicles are involved in a goods transfer chain with more than two vehicles, as this creates cascading dependencies. For instance, looking again at the unloading group of services (Figure 6), defining the earliest start time of the service performed by the dolly requires defining the end time of the service performed by the speed loader, which itself depends on the end time of the service performed by the high loader, which itself depends on both the end time of the previous group of services and the aircraft arrival time. This is further complicated when multi-vehicle services are involved, when the vehicles have different capacities and more than one unit must be transferred, or when many transfers must be made before the initial client demand is satisfied. In addition, other processes may require longer chains, making it even more challenging to mathematically model the constraint without loss of generality. This is addressed by a recursive procedure in our solution method (see Sect. 4.4).

Finally, goods transfers between vehicles also impact the demand quantity definition. De facto, when vehicles are independent from each other, the demand to be served is defined at the main client level. However, when two vehicles are involved in a transfer, they are interdependent and the demand for the second vehicle in the chain, for each transfer, depends on the previous vehicle's current capacity. In addition, considering the definition of the total quantity to be served, if service $m \in \mathcal{P}$ at client i consists of collecting the goods gathered upstream by service $p \in \mathcal{P}$ at client i, then $q_{im} = q_{ip}$.

3.3.3 Compulsory vehicle pairings

Passive vehicles should be paired with other vehicles to move in space. It is called compulsory vehicle pairing and refers to the movement synchronization described by Drexl (2012). In cargo AGH, for instance, dollies can only move at the same time and along the same route as cargo tractors. Formally, for all vehicle fleets $\mathcal{V}_\alpha \subset \mathcal{V}$, there exists a set $\mathcal{K}_\alpha$ of vehicle fleets with which vehicles $k \in \mathcal{V}_\alpha$ should be paired to move, an empty set indicates active vehicles that can move on their own. Paired vehicles share their start time, operating time, and possible movements. Thereby, if vehicles $k \in \mathcal{V}_\alpha$ and $l \in \mathcal{K}_\alpha$ should be paired during a given service $p \in \mathcal{P}_\alpha$ at client $i \in \mathcal{C}_k \cap \mathcal{C}_l$, then their common start time is set equal to the maximum between vehicles k, l start times if they were not paired, that is, $\max(S^k_{ip}; S^l_{ip})$, and their common operating time $O^k_{ip} = O^l_{ip}$ includes both vehicles pairing time and transport time.

As we can see, cargo AGH involves numerous synchronization constraints that greatly complicate the problem characteristics presented in Sect. 3.1. In particular, the cascading structure of the goods transfer chains leads us to consider an original solution method.

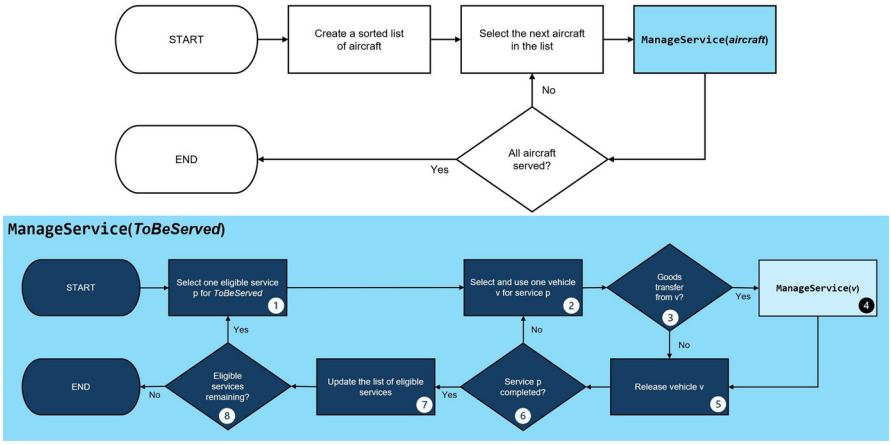


Fig. 8 Solution method's general scheme

4 Solution method

To coordinate efficiently the different airport ground service vehicles, we developed a solution method that optimizes their routes, taking into account all the synchronization constraints and problem features described in Sect. 3. As mentioned before, we chose a heuristic approach due to the recursive and complex nature of the problem.

Figure 8 illustrates the key components of the proposed solution method. As can be seen, the main process focuses on successively serving aircraft from a sorted list. The sorting operator will be explained in more detail in Sect. 4.2. The core of the algorithm is a procedure designed to manage a service, which is detailed in the blue area of Figure 8. This procedure appears to be recursive for services involved in a goods transfer chain. Indeed, while most vehicles serve aircraft, vehicles involved in a goods transfer chain serve the vehicles from which they depend. For instance, as shown in Figure 6, when the service consists of unloading a high loader (HL) using a speed loader (SL), the SL is the vehicle for which a route is computed, while the HL becomes the vehicle to be served. Consequently, this entity is passed as an argument *ToBeServed* to the recursive procedure `ManageService`.

To illustrate this logic, Figure 9 depicts the sequence of operations that the algorithm performs when considering the first steps of the aircraft unloading process, shown in the left box of Figure 2. In the main process (Figure 8), a first aircraft (A_1) is selected from the list, and the recursive procedure is called with A_1 as the argument. Then, from the procedure contained in the blue area of Figure 8, in box (1), a service p is selected among the eligible services for A_1. A service is considered eligible if it is associated with A_1, since A_1 is the argument *ToBeServed* at this point, and if there is no other service to be completed before. In our example, the first service is "Aircraft Unloading", where *ToBeServed* is A_1, and the vehicle to be selected is a HL (the operation is labeled "$A_1 \rightarrow HL$" in Figure 9). In box (2), a specific vehicle, HL_1, is selected to perform at least part of the service. Once this operation is completed, the algorithm checks in box (3) whether any other subsequent service involves HL_1. This

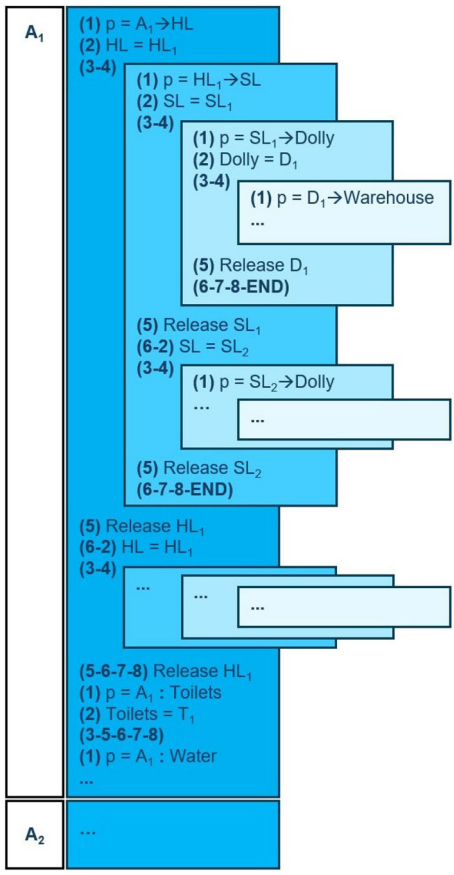


Fig. 9 Algorithm's sequence of operations example

is indeed the case here, as two containers typically must be unloaded from HL_1 by SLs. Therefore, HL_1 cannot be released in box (5) until two SLs have fully completed this task. This triggers a recursive call to the procedure in box (4), this time with HL_1 as the *ToBeServed* vehicle. This scheme leads to a chain of recursive calls: $A_1 \rightarrow HL_1$, $HL_1 \rightarrow SL_1$, $SL_1 \rightarrow Dolly_1$, $Dolly_1 \rightarrow Warehouse_1$ (see Figure 9). When HL_1 is finally released in box (5), the service for A_1 is typically not yet fully completed. Other containers may still need to be handled by the released HL_1 or by another HL vehicle, which is verified in box (6). Once all containers have been processed, the algorithm reaches box (8) to determine if another service should be initiated, such as those depicted in Figure 2. The precedence between services must be enforced. Some services, like potable water servicing, which was not eligible initially, will become eligible as soon as the preceding service is completed (toilet servicing, in our example). Before the test in box (8), we must therefore update the list of eligible services in box (7).

Since the logic within `ManageService` (depicted in the blue area of Figure 8) remains consistent for any successive service, one might assume the process could be managed via nested loops. However, such an approach is not computationally tractable

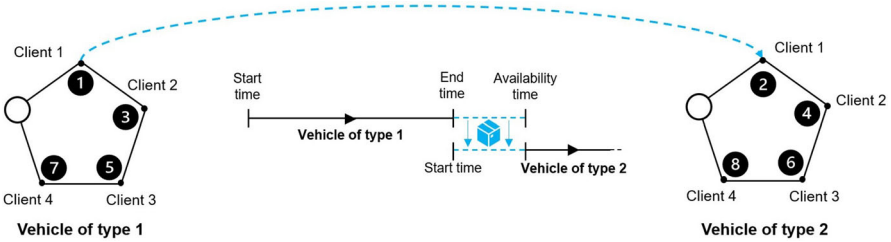


Fig. 10 Interdependency of vehicles involved in a goods transfer

because the depth of the service chain varies depending on the initial service. As shown in Figure 2, the unloading chain has a length of 4, while loading preparation and freight loading have a length of 3. This variability justifies the use of this innovative recursive approach, which constitutes a key element of our proposed solution. It should also be noted that the actual problem is significantly more complex, and the implementation detailed in the following sections will depart slightly from the conceptual overview in Figure 8.

4.1 Main algorithm

At first glance, a classical vehicle routing problem approach that builds the routes of each vehicle one at a time may seem like a natural option. However, to elegantly and efficiently handle vehicle dependencies induced by goods transfers, we adopt a different approach. Indeed, since a vehicle involved in a transfer alternately takes on the role of a vehicle serving a client in its own VRP and the role of a depot in the VRP of the next vehicle in the chain (see Sect. 3.3.2), it seems consistent not to plan the remainder of the vehicle's route in its own VRP before knowing the end of its service as a depot in the VRP of the next vehicle. Yet, to know when the first vehicle will be released in the second VRP, we must plan the route of the second vehicle. We therefore decided to produce a complete solution for one **client** before moving on to another, and the routes of the different vehicles are built up gradually. We illustrate this concept on a specific example presented in Figure 10. The numbers in the black circles indicate the order in which the algorithm proceeds and the blue arrow represents a goods transfer. As can be seen, the algorithm first considers the service of vehicle of type 1 at client 1, then the service of vehicle of type 2 at the same client to define the availability time of vehicle of type 1, and only then the service of vehicle of type 1 at client 2 is considered.

The main structure of our algorithm is depicted in Algorithm 1, where the solution path is first by client (sorted according to a given strategy, see Sect. 4.2), then by group of services with special consideration of precedence between groups of services, and finally by service with particular attention to service precedence and goods transfers. In particular, focusing on a given client i and a given group of services $\tilde{x}$, we define $t_{max}(\tilde{x})$ as the end time of group $\tilde{x}$ (i.e., the end time of the last service belonging to group $\tilde{x}$) and set its initial value to $A_{i\tilde{x}}$, that is, the earliest start time of group $\tilde{x}$ at client i. Note that the earliest start time of all groups of services is set equal to

a_i when initializing the data ($A_{i\tilde{x}} = a_i, \forall \tilde{x}$). Then, for each service p belonging to $\tilde{x}$ and having no direct predecessor, a vehicle k is selected to visit and serve client i according to the procedure contained in the `SelectVehicle`$(p, i, \mathcal{V})$ function depicted in Algorithm 2. The quantity to be served (q_{ip}) and the service end time (E_{ip}), as well as $t_{max}(\tilde{x})$, are then updated accordingly. Services that ensue from service p due to precedence or goods transfers are handled next, as described in the `HandleDependencies`(p, i, k) function depicted in Algorithm 3. From this, t_{new} is defined as the end time of the last service in the goods transfer/precedence chain ensuing from service p, and is simply equal to E^k_{ip} in case of independent service. The value of $t_{max}(\tilde{x})$ is finally updated, and the process is repeated until the total quantity to be served (q_{ip}) is satisfied. Once group $\tilde{x}$ has been fully processed, the precedence constraints it may have with other groups of services are handled. For each client $i \in \mathcal{C}$, Algorithm 1 deals with the different groups of services and successively updates the time windows of all groups of services $\tilde{y}$ for which group $\tilde{x}$ is a direct predecessor ($\tilde{y} \succ \tilde{x}$), revising the earliest start time of each successor to ensure that it is greater than or equal to $t_{max}(\tilde{x})$.

Algorithm 1 `SolveAGH`$(\mathcal{C}, \mathcal{P}, \mathcal{V})$

Require: All clients ($\mathcal{C}$), services ($\mathcal{P}$), vehicles ($\mathcal{V}$)
$\mathcal{C}_{sort} \Leftarrow \mathcal{C}$ sorted according to the selected strategy
for all client $i \in \mathcal{C}_{sort}$ **do**
  for all group $\tilde{x}$ of services **do**
    $t_{max}(\tilde{x}) \Leftarrow A_{i\tilde{x}}$
    for all service p of group $\tilde{x}$ that does not have any direct predecessor **do**
      while $q_{ip} > 0$ **do**
        $k, Q^k_{ip}, E^k_{ip} \Leftarrow$ `SelectVehicle`$(p, i, \mathcal{V})$ ▷ see Algorithm 2
        $q_{ip} \Leftarrow q_{ip} - Q^k_{ip}$
        $E_{ip} \Leftarrow E^k_{ip}$
        $t_{max}(\tilde{x}) \Leftarrow \max(E^k_{ip}; t_{max}(\tilde{x}))$
        // Manage service p dependencies (precedence and goods transfers)
        if there are goods transfers ensuing from p **or** $q_{ip} = 0$ **then**
          $t_{new} \Leftarrow$ `HandleDependencies`(p, i, k) ▷ see Algorithm 3
          $t_{max}(\tilde{x}) \Leftarrow \max(t_{new}; t_{max}(\tilde{x}))$
        end if
      end while
    end for
    // Manage precedence between groups of services
    for all group of services $\tilde{y} \succ \tilde{x}$ **do**
      $A_{i\tilde{y}} \Leftarrow \max(A_{i\tilde{y}}; t_{max}(\tilde{x}))$
    end for
  end for
end for

4.2 Client sorting strategies

Since we have opted for a greedy algorithm approach, it is important to note that the order in which clients are processed in Algorithm 1 ($\mathcal{C}_{sort}$) should be carefully

chosen to be in line with the objective. The general rule, denoted *AC 1*, is to work on a first-come first-served basis, that is, ordering aircraft based on their scheduled time of arrival (a_i). This is justified by our goal to minimize the total time all aircraft must wait to be fully served. However, other sorting methods may be expected to lead to different results, we therefore tested different client sorting strategies in Sect. 5. Most of them are initially built based on *AC 1*, then modified by switching aircraft in different ways: *AC 2* (resp. *AC 2b*) considers a switch between two aircraft (resp. with a probability of 50%) if the subsequent has a scheduled departure time that is smaller than the first one; *AC 3* (resp. *AC 3b*) considers a switch between two aircraft (resp. with a probability of 50%) if the subsequent has a smaller allowed ground time than the first one; *AC 4* considers a switch between two aircraft with a probability of 50% if their allowed ground times overlap; *AC 5* sorts the aircraft in their order of scheduled departure time (without considering *AC 1* initially).

4.3 Vehicles selection

An important decision is how to select the vehicles so that the total service time, ST, is minimized and the responsiveness maximized. The associated procedure is contained in the `SelectVehicle`$(p, i, \mathcal{V})$ function detailed in Algorithm 2. Note that while the client's ordering strategy should preferably be aligned with the objective, algorithmically, the consideration of responsiveness only appears in the present function (i.e., if the objective function were to change, only `SelectVehicle`$(p, i, \mathcal{V})$ would need to be modified). The algorithm first checks the number of vehicles already allocated to service p at client i. If the maximum number of vehicles allowed for that service (given the capacity of the service area) has already been reached, then a set $\mathcal{Z}$ is built with all the vehicles already used to perform service p at client i. The vehicle v selected from $\mathcal{Z}$ is the first one to end its current service and become available again. Although other vehicles could be available before those in $\mathcal{Z}$, they would not be allowed to start the service until at least one vehicle in $\mathcal{Z}$ ends its service and leaves the service area. Yet, in an effort to minimize travel distance and client waiting time, as soon as a vehicle in $\mathcal{Z}$ ends its current service, it becomes the preferred choice since it is already in the service area of client i. Otherwise, the set $\mathcal{Z}$ is built with all vehicles that are compatible with both service p and client i and that are able to provide the service given their current status and remaining capacity. The vehicle v selected from $\mathcal{Z}$ is then the one that can start the service as close as possible to its earliest start time, considering both its previous service end time and the travel time to reach client i location. In this regard, note that, for simplicity and due to strict airport regulations, we consider that all vehicles travel at the same speed, which also tends to minimize the distance traveled. In both cases, Algorithm 2 first tries to select a vehicle that is ready to start the service before its earliest start time, but as close as possible to this last one, in an effort to maximize vehicle utilization. If no vehicle can start the service at its earliest start time, the algorithm simply selects the first vehicle ready to operate at the client location. Vehicle v serves a quantity equal to the minimum value between its remaining capacity and the total demand of client i for service p. Its operating and end time are then computed, while its current location and remaining capacity are

updated. In the case of allowed replenishment, when the selected vehicle v runs out of capacity, it goes to the nearest compatible depot to be replenished to full capacity and its parameters are updated again. Given this architecture, `SelectVehicle`$(p, i, \mathcal{V})$ maximizes responsiveness, but also tends to minimize the distance traveled and the number of vehicles used.

Algorithm 2 `SelectVehicle`$(p, i, \mathcal{V})$

Require: Service p, client i, all vehicles ($\mathcal{V}$)
Ensure: Selected vehicle v and its associated parameters Q_{ip}^v and E_{ip}^v
$\mathcal{Z} \Leftarrow \emptyset$
if max number of vehicles allowed for service p is already reached **then**
  $\mathcal{Z} \Leftarrow$ Vehicles already used to perform service p at client i
  for all vehicle $k \in \mathcal{Z}$ **do**
    $ready_k \Leftarrow$ Time at which vehicle k can start again service p at client i at the earliest
  end for
else
  for all vehicle $k \in \mathcal{V}$ **do**
    if service p can be performed by k **and** k is available to perform p **and** $R_k > 0$ **and** $i \in \mathcal{C}_k$ **then**
      $ready_k \Leftarrow$ Time at which vehicle k can start service p at client i at the earliest
      $\mathcal{Z} \Leftarrow \mathcal{Z} \cup \{k\}$
    end if
  end for
end if
if $\exists\, v \in \mathcal{Z}$ such that $ready_v \leq A_{ip}$ **then**
  $v \Leftarrow \arg\min_v (A_{ip} - ready_v | ready_v \leq A_{ip})$
else
  $v \Leftarrow \arg\min_v (ready_v - A_{ip})$
end if
$S_{ip}^v \Leftarrow ready_v$
$Q_{ip}^v \Leftarrow \min(R_v; q_{ip})$
$R_v \Leftarrow R_v - Q_{ip}^v$
Set O_{ip}^v and update v current location
$E_{ip}^v \Leftarrow S_{ip}^v + O_{ip}^v$
if $R_v = 0$ **and** replenishment is allowed for v **then**
  Send v to the nearest compatible depot and replenish it to full capacity
  Update v current location, remaining capacity, and availability time
end if
return v, Q_{ip}^v, E_{ip}^v

4.4 Dependencies handling

To deal with services that ensue from other ones, as opposed to `SolveAGH`$(\mathcal{C}, \mathcal{P}, \mathcal{V})$ that only handles services without direct predecessors, we use the `HandleDependencies`(p, i, k) function detailed in Algorithm 3. As can be seen, for each service m that directly depends on service p (because of precedence constraint or goods transfer), t_{max} is set to E_{ip}. Here, t_{max} is a local variable that refers to the end time of the last service in the chain ensuing from service p. A vehicle l is then selected with Algorithm 2 to perform the service m, the quantity that remains to be served (q_{im}) and the service

Algorithm 3 `HandleDependencies`(p, i, k)

Require: Service p, client i, vehicle k
Ensure: End time of the last service in the chain ensuing from service p (t_{max})
1: **for all** service m ensuing from service p **do**
2: $t_{max} \Leftarrow E_{ip}$
3: **while** $q_{im} > 0$ **do**
4: $l, Q^l_{im}, E^l_{im} \Leftarrow$ `SelectVehicle`$(m, i, \mathcal{V})$ ▷ see Algorithm 2
5: $q_{im} \Leftarrow q_{im} - Q^l_{im}$
6: $E_{im} \Leftarrow E^l_{im}$
7: $t_{max} \Leftarrow \max(E^l_{im}; t_{max})$
8: *// In case of goods transfer from vehicle k to vehicle l*
9: **if** there are goods transfers from p to m **then**
10: $E^k_{ip} \Leftarrow S^l_{im} + (t^{kl} \times Q^l_{im})$
11: **end if**
12: *// Manage service m precedence constraints*
13: **if** there is no goods transfer starting from m **and** $q_{im} = 0$ **then**
14: $t_{new} \Leftarrow$ `HandleDependencies`(m, i, l)
15: $t_{max} \Leftarrow \max(t_{new}; t_{max})$
16: **end if**
17: *// Manage service m goods transfers*
18: **if** $R_l = 0$ **or** (there are goods transfers starting from m **and** it is the last units to be transferred) **then**
19: $t_{new} \Leftarrow$ `HandleDependencies`(m, i, l)
20: $t_{max} \Leftarrow \max(t_{new}; t_{max})$
21: **end if**
22: **end while**
23: **end for**
24: **return** t_{max}

end time (E_{im}) are updated accordingly. Beyond inputs and outputs, the difference with `SolveAGH`$(\mathcal{C}, \mathcal{P}, \mathcal{V})$ is twofold. First, in case of goods transfer between vehicles k and l, E^k_{ip} is updated according to what is prescribed by Equation (3). Second, and most importantly, the cascading dependencies created by service precedence or goods transfer chains should be handled. This is done by a recursive procedure that involves calls to `HandleDependencies`(p, i, k) from within the function itself and updates t_{max} at the end of each call. There is no restriction on the length of the dependency chains and thus on the number of successive calls. It is indeed important to fully investigate any cascading dependencies before going back to `SolveAGH`$(\mathcal{C}, \mathcal{P}, \mathcal{V})$ to ensure both feasibility and efficiency. For instance, when unloading an aircraft (see Figure 6), once the high loader has been unloaded by speed loaders, it is essential that the algorithm handles the unloading of the speed loaders before the high loader continues to unload the aircraft. Otherwise, the speed loaders already in use for that aircraft will be considered as unavailable (already loaded) to unload the new containers being unloaded by the high loader. This would result in either inefficient solutions, with more vehicles used than needed and the selection of more distant vehicles, or infeasible solutions if the maximum number of speed loaders for that aircraft has already been reached. Note that dealing with such cascading dependencies would not have been possible with repeated calls through a more conventional programming

loop, confirming the need for a recursive approach. The procedure is repeated until the total quantity (q_{im}) has been served.

4.5 Additional aspects

In addition to the previous considerations, note that the following aspects are also included in our solution method. First, each service must be completed exactly once, and a client is not considered fully served until all of its requested services have been performed. Moreover, time consistency should be maintained, meaning that if vehicle k serves client i first and client j next, then service at client j cannot start before vehicle k has completed its service at client i and has moved from its location to that of client j. Finally, regarding multi-operation vehicles, once a vehicle $k \in \mathcal{V}_\alpha$ is selected to provide a given service $p \in \mathcal{P}_\alpha$ to a client i, this vehicle can temporarily serve only that client, as it has to fully complete its service (or service sequence) for client i before being available for the others again.

5 Results

To validate the solution method developed in the previous section, tests were performed on real instances with varying degrees of tightness. The algorithm was coded in Python (version 3.12.2) and the results presented below were obtained on a laptop running Windows 11, equipped with Intel® Core™ i7-9750H and 16 GB RAM. Full code is publicly available online (https://hdl.handle.net/2268/323814).

5.1 Instances description

Real data was collected from Liège Airport (LGG), one of the main cargo airports in Europe, that carries about 3000 tons of freight daily and hosts several airlines and cargo handlers. A 24-hour time horizon is considered, instances are therefore built to consider all cargo aircraft that land and take off from LGG between noon on day 1 and noon on day 2.

To assess the ability of the algorithm to adapt to any application condition, 18 instances were selected to cover various situations. The chosen instances encompass different periods of economic activity: low (less than 15 aircraft to be served during the time horizon), normal (25 to 35 aircraft to be served), and intense (more than 35 aircraft to be served). The 18 test instances also show a high degree of variability in their mean allowed ground time (i.e., the mean time between aircraft scheduled arrival time (a_i) and scheduled departure time (b_i) across all aircraft in the instance), which refers to the mean time available to carry out all ground operations. Test instances finally differ in their respective distribution of aircraft arrivals, although there is often a peak between 10 pm and midnight that is specific to LGG, one of the few airports allowed to operate at night. The details of all these instances are summarized in Table 2.

These characteristics also allow us to evaluate the tightness of an instance. An instance is assumed more difficult to solve if there are more aircraft to be served, as

Table 2 Instances description and LGG initial result

ID	Aircraft flow	Number of aircraft	Aircaft arrivals distribution[a]	Mean allowed ground time	LGG initial results	
					Delayed aircraft	Mean (max) delay
1	Intense	36		05:24:05	19	00:42:05 (05:07:40)
2	Intense	38		04:09:22	19	00:26:56 (01:09:00)
3	Intense	40		04:50:51	21	00:35:20 (04:12:26)
4	Low	13		06:02:42	7	00:28:57 (01:40:34)
5	Low	9		02:45:40	3	01:11:22 (01:55:00)
6	Normal	32		04:19:09	18	00:15:45 (01:59:38)
7	Intense	37		04:15:05	21	01:05:09 (06:46:00)

Table 2 continued

ID	Aircraft flow	Number of aircraft	Aircaft arrivals distribution[a]	Mean allowed ground time	LGG initial results	
					Delayed aircraft	Mean (max) delay
8	Normal	35		04:44:34	16	00:21:31 (01:48:31)
9	Normal	34		04:03:32	21	00:29:40 (01:57:29)
10	Normal	27		03:39:03	11	00:39:57 (02:17:00)
11	Intense	39		04:21:32	25	00:52:16 (09:36:00)
12	Intense	41		04:57:41	15	00:17:02 (01:16:00)
13	Intense	40		04:12:44	21	00:22:21 (01:38:32)
14	Intense	38		04:18:36	15	00:20:13 (01:27:00)

Table 2 continued

ID	Aircraft flow	Number of aircraft	Aircaft arrivals distribution[a]	Mean allowed ground time	LGG initial results	
					Delayed aircraft	Mean (max) delay
15	Normal	30		05:03:06	11	00:41:23 (03:20:29)
16	Normal	29		04:47:48	14	00:19:15 (01:45:00)
17	Intense	40		04:23:02	21	00:36:10 (02:36:23)
18	Intense	40		03:56:45	21	00:56:42 (06:17:00)
Mean over all instances				04:27:31	16.61	00:35:40 (03:09:26)

[a] Bins correspond to time slots of 2 hours from 12:00 on day 1 to 12:00 on day 2

this directly affects the global workload. However, the tightness of an instance also increases if the mean allowed ground time is short or if the aircraft arrival distribution is peaked. For example, although instances 4 and 5 appear to be equivalent, instance 5 is actually much tighter to solve due to its very short mean allowed ground time. Similarly, instances 15 and 16 may seem comparable, but the former benefits from a scattered distribution of aircraft arrivals and is thus much easier to solve. It is then important to be careful about the combination of these attributes.

Although datasets for many VRP variants have been made publicly available since Solomon (1987) (see, e.g., Mendoza et al. 2014; Sim et al. 2019; Gunawan et al. 2021), there is still no set of instances for VRPs with multiple fleet synchronization. Since that characteristic is key to the problem at hand, we do not have the opportunity to evaluate our work based on recognized benchmarks. We therefore decided to compare our results to the ones initially produced by LGG for each instance in terms of delays. They are reported in the last two columns of Table 2. Note that the causes of the identified delays are not all known, some may be due to unexpected events, such as engine failures. Nevertheless, this is the exception rather than the rule. In addition, all delays caused by late arrivals have been eliminated, so it is reasonable to assume that the result comparisons and main conclusions presented in this section are fair and accurate.

The results produced in the next subsections are assessed based on the mean service time, the number of delayed aircraft, and the mean and maximum delay durations. The mean service time ($ST/|\mathcal{C}|$) refers to the mean time to complete all ground operations on an aircraft. Deducted from the mean allowed ground time, defined as $\sum_i (b_i - a_i) \ / \ |\mathcal{C}|$, such a measure allows to compute the size of the mean safety time buffer generated (on all aircraft in the instance), which in turn helps to assess the responsiveness of the produced solution. Indeed, the safety time buffer of an aircraft i is defined as $(b_i - \max_p E_{ip})$, which, in case of a positive value, indicates the time left before the theoretical departure of aircraft i, while a negative value indicates a delay. The number of delayed aircraft (aircraft for which $\max_p E_{ip} > b_i$) as well as mean and maximum delays (computed only on delayed aircraft) are also considered.

5.2 Experiment results

When running the algorithm on the 18 test instances using the different client sorting strategies described in Sect. 4.2, equivalent results were obtained for most sorting methods. Note that we carried out the experiment ten times for the sorting methods that include randomness to ensure that similar conclusions could be drawn each time. We therefore decided to present only the results produced with the general *AC 1* rule in Table 3. Substantial improvements can be observed compared to the results initially produced by LGG (see Table 2). First, it appears that delayed aircraft are eliminated for 5 instances, while others benefit from a reduction in the number of delayed aircraft ranging from 33.33% to 95.24%, with an average of 89.69%. Mean and maximum delays are also greatly reduced for most instances. Only instances 8 and 14 show an increased mean delay, but this is directly related to the fact that they now have only

Table 3 Experiment results

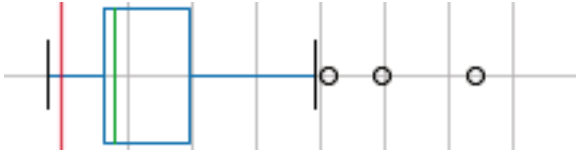
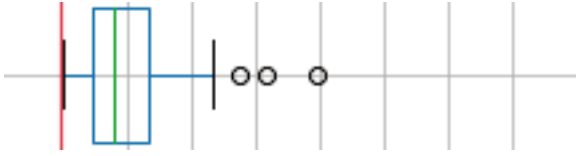
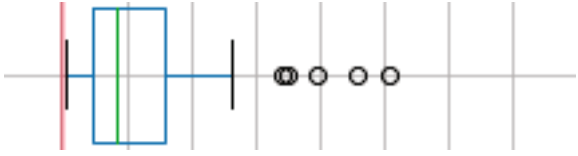
ID	Mean service time	Mean buffer	Buffer/Delay distribution	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)
1	00:59:38	04:24:27		1 (-94.74%)	00:38:30 (-8.51%)	00:38:30 (-87.49%)	7.25
2	01:02:45	03:06:37		0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	8.65
3	01:04:26	03:46:25		0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	9.28
4	01:27:54	04:34:48		0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	3.85

Table 3 continued

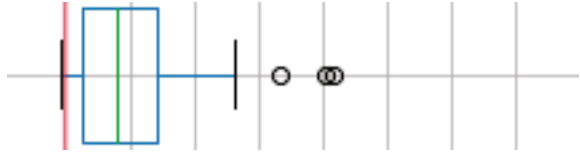
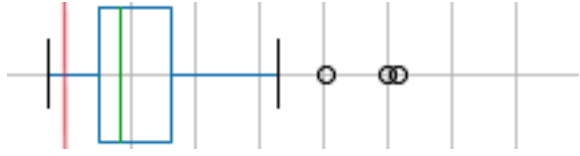
ID	Mean service time	Mean buffer	Buffer/Delay distribution	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)
5	01:27:53	01:17:47		2 (-33.33%)	00:42:30 (-40.45%)	00:46:30 (-59.57%)	2.95
6	00:59:43	03:19:26		1 (-94.44%)	00:03:20 (-78.84%)	00:03:20 (-97.21%)	7.71
7	01:08:05	03:07:00		3 (-85.71%)	00:06:55 (-89.38%)	00:08:45 (-97.84%)	10.27
8	01:01:23	03:43:11		1 (-93.75%)	00:43:44 (+103.25%)	00:43:44 (-59.7%)	7.06

Table 3 continued

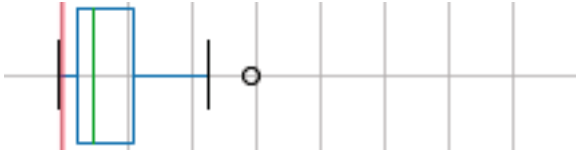
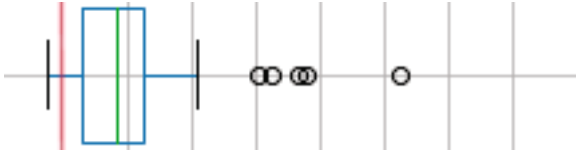
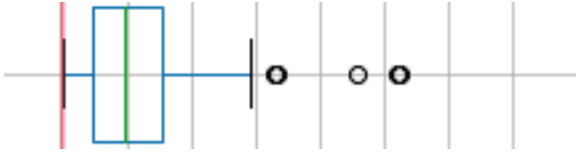
ID	Mean service time	Mean buffer	Buffer/Delay distribution	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)
9	01:04:13	02:59:19		1 (-95.24%)	00:13:23 (-54.89%)	00:13:23 (-88.61%)	7.06
10	01:20:27	02:18:36		2 (-81.82%)	00:05:00 (-87.48%)	00:06:30 (-95.26%)	7.04
11	01:05:02	03:16:30		2 (-92%)	00:23:39 (-54.75%)	00:35:48 (-93.78%)	8.4
12	01:01:02	03:56:39		0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	8.19

Table 3 continued

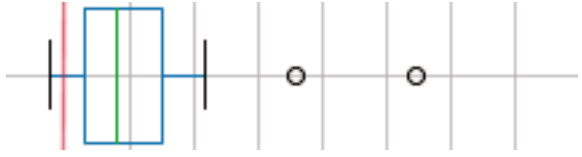
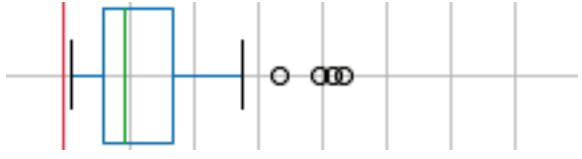
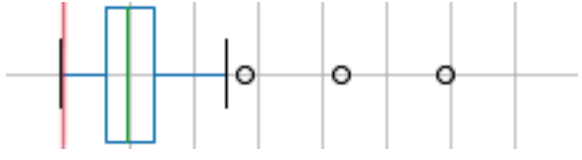
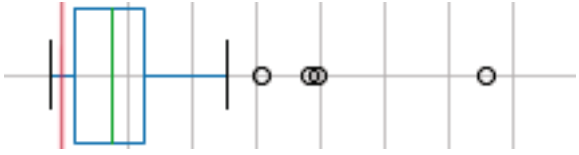
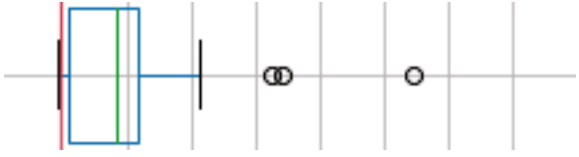
ID	Mean service time	Mean buffer	Buffer/Delay distribution	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)
13	01:06:05	03:06:39		2 (-90.48%)	00:07:45 (-65.32%)	00:09:53 (-89.97%)	8.44
14	01:08:01	03:10:35		1 (-93.33%)	00:36:28 (+80.38%)	00:36:28 (-58.08%)	8.52
15	01:02:49	04:00:17		0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	6.15
16	01:02:06	03:45:42		1 (-92.86%)	00:09:07 (-52.64%)	00:09:07 (-91.32%)	5.85

Table 3 continued

ID	Mean service time	Mean buffer	Buffer/Delay distribution	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)
17	01:07:35	03:15:27		2 (-90.48%)	00:25:21 (-29.91%)	00:32:11 (-79.42%)	10.81
18	01:09:44	02:47:01		5 (-76.19%)	00:05:56 (-89.54%)	00:08:30 (-97.75%)	10.02
	01:07:43	**03:19:48**		**1.33 (-89.69%)**	**00:14:32 (-53.78%)**	**00:16:16 (-88.67%)**	**7.64**

one delayed aircraft. On average, the mean (resp. maximum) delay is cut by 53.78% (resp. 88.67%). Then, mean service times vary from 00:59:38 to 01:27:54, while, at the aggregate level, the global mean is 01:07:43, that is to say, 03:19:48 less than the mean allowed ground time for all instances presented in Table 2 (04:27:31). This comfortable safety time buffer helps reduce the total aircraft delay, as described above. The boxplots of Table 3 illustrate this trend, each observation refers to the safety time buffer produced for a single aircraft $i \in \mathcal{C}$. The observations on the left-hand side of the red line (representing value 0) refer to delays, while the others refer to positive safety time buffers. As can be seen, few delays are produced, and mostly small ones, regardless of the instance considered. Finally, in addition to the quality of the generated solutions, the algorithm also proves to be fast, with a mean computing time of about 7.7 seconds.

5.3 Experiment results for challenging instances

The quality of the results of the algorithm makes us wonder if too many resources are being considered. It is indeed common for airports to have oversized fleets of service vehicles, since delays have important negative consequences. The same holds for LGG. This implies that the real instances considered are not that challenging to resolve, which in turn explains why such good results are obtained. In the following, we designed new instances to test the performance of the algorithm in more challenging contexts.

Testing the different client sorting strategies described in Sect. 4.2 on these tighter instances, the results produced are no longer equivalent. In Sections 5.3.1 and 5.3.2, we identified the best sorting strategy for each instance, first based on the number of delays over 15 minutes, and then using the total number of delays, the maximum delay duration, and the mean buffer/service time as tiebreakers. Although *AC 2b* seems to be a good option for many instances, a Friedman test followed by a Nemenyi post hoc test leads to the conclusion that no strategy is significantly better than another based on these criteria. We therefore decided to systematically test the different sorting methods and choose the best solution the algorithm can achieve for each instance (note that the algorithm is fast enough to make this conceivable).

5.3.1 Increasing the number of aircraft to be served

We created two artificial instances based on a combination of two real instances to ensure realistic aircraft arrival distributions and mean allowed ground times, while inflating the number of aircraft to be served during the 24 hours considered. Instance 19 considers all the aircraft of instance 12 and one over two from instance 13, whereas instance 20 considers all aircraft of both instances. Note that they remain realistic, as LGG expects to double the number of cargo flights in the next few years based on the new export permit it has received. Instances' characteristics are described in Table 4. Testing the algorithm on these two instances, the results produced remain very good (see Table 5). Indeed, it appears that mean service times and buffers are very close to those produced for real instances. It can also be seen that the number

Table 4 Artificial instances description

ID	Aircraft flow	Number of aircraft	Aircaft arrivals distribution[b]	Mean allowed ground time
19	Artificial	61		04:31:17
20	Artificial	81		04:36:04

[b] Bins are defined as in Table 2

Table 5 Experiment results for artificial instances

ID	Mean service time	Mean buffer	Delayed aircraft	Mean delay	Maximum delay	CPU time (sec)	Best client order
19	01:03:55	03:27:22	2	00:08:25	00:13:30	13.42	*AC 1*
20	01:09:01	03:27:03	2	00:08:49	00:13:30	19.71	*AC 3b*

Table 6 Ground service vehicle fleet sizes

Vehicle type	LGG fleet size	Reduced fleet size
Toilet car	8	3
Refueling truck	9	7
Catering truck	8	3
Technical car	8	4
Cleaning car	8	3
Water truck	8	3
Batch of 3 dollies	484	$2.75 \times \lvert\mathcal{C}\rvert$ (Min. 70)
High loader	38	$0.125 \times \lvert\mathcal{C}\rvert$ (Min. 4)
Speed loader	27	$0.25 \times \lvert\mathcal{C}\rvert$ (Min. 8)
Cargo tractor	48	$0.5 \times \lvert\mathcal{C}\rvert$ (Min. 14)

of delayed aircraft produced is very low, despite the increased number of aircraft to be served. In addition, mean and maximum delays are quite short (less than 15 minutes). Finally, the computing time remains reasonable, despite the increase in instance tightness.

5.3.2 Reducing the number of ground service vehicles

To further increase the tightness of the instances, we considered arbitrarily reduced ground service vehicle fleet sizes. Loading and unloading being bottleneck operations, associated vehicle fleets are resized based on instances' number of aircraft, while a fixed fleet size is determined for the others. To remain realistic regarding LGG aircraft flow, a minimum number of each vehicle type has also been set. Still, we designed the instances with significantly smaller fleets than those used in the original solutions. That way, we are able to test the performance of our algorithm in a more challenging context. However, optimizing fleet size is a more complex management issue that is not addressed in this article. Table 6 summarizes initial and new fleet sizes.

Looking at the produced solution values in Table 7, we can see that the mean service time increases for all instances, as expected, since there are fewer vehicles available. On average, the mean service time is equal to 01:37:25, which is 00:29:42 longer than the experiment results for the initial use case. Still, it remains reasonable and provides a comfortable mean safety time buffer of 02:50:06. In addition, when comparing the previous experiment results with these new ones for each instance, identical or better results in terms of number and duration of delays are obtained for

13 cases, while very close results (less than 15 minutes and 1 delayed aircraft difference) are produced for 3 cases, meaning that the poorest results represent only 2 cases (*instances 10* and *16*). At the aggregate level, the cut in vehicle fleet sizes allows to decrease the mean computing time to about 2.1 seconds, and the number of delayed aircraft is reduced for all instances. On average, the number of delayed aircraft is decreased by 86.61% compared to the results initially produced by LGG (see Table 2), while the mean (resp. maximum) delay is cut by 51.01% (resp. 85.78%). These aggregated performances are therefore very close to what the algorithm produced when considering all vehicles available at LGG. The last two rows of Table 7 show that performances on artificial instances are also good, despite the increase in clients number, with few delayed aircraft and mean/maximum delay below 15 minutes.

6 Conclusion

We have formalized the rich vehicle coordination problem of air cargo ground handling that is experienced at Liège Airport, a major cargo airport in Europe. The model appears to be a set of Multi-Depot Heterogeneous Multi-Operation Capacitated VRPs with Time Windows, Forbidden Pairings, Multi-Trip, Split Deliveries, and Multi-Vehicle Services. These rich VRPs are furthermore bound together through compulsory vehicle pairings, precedence between services, and goods transfers between vehicles of different types. Goods transfer chains with a variable number of vehicles involved can also occur, which introduces cascading dependencies and greatly complicates the problem.

We have built a greedy heuristic method centered on clients to solve the problem in its entirety. This one optimizes the routes of the different service vehicles, taking into account all the VRP features and all the vehicle dependencies that bind them together. In particular, we make use of a recursive procedure to handle the cascading dependencies caused by goods transfer chains of variable length. We chose to minimize the total service time, which is equivalent to maximizing responsiveness. This criterion is relevant to the air transportation industry since it helps minimize the number and duration of aircraft delays. The resulting algorithm proves to produce promising results when tested on real instances coming from Liège Airport, with the creation of comfortable safety time buffers for each client (responsiveness), a significant reduction in the number of delayed clients as well as delay duration (efficiency), and a computing time short enough to allow reactivity in case of unexpected events that would cause an increase in service time (speed). The algorithm also keeps performing well when considering more challenging cases, where the number of clients is artificially increased or the size of the fleet is reduced, which in turn proves its robustness. Given these results, we decided not to implement local improvement algorithms. In addition, we believe that our solution method could be adapted to other situations presenting the studied problem structure, thus extending the contributions of this paper.

Still, further research remains to be done, especially with respect to handling the dynamic nature of the data. The cargo AGH environment is indeed highly dynamic,

Table 7 Experiment results with client processing order modification and resources reduction

ID	Mean service time	Mean buffer	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)	Best client order
1**	01:03:30	04:20:35	1 (-94.74%)	00:38:30 (-8.51%)	00:38:30 (-87.49%)	2.09	*AC 1*
2**	01:15:15	02:54:07	0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	2.28	*AC 2b*
3**	01:09:46	03:41:05	0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	2.43	*AC 1*
4**	01:47:28	04:15:14	0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	0.78	*AC 2b*
5*	01:48:42	00:56:58	2 (-33.33%)	00:51:44 (-27.51%)	00:56:58 (-50.46%)	0.47	*AC 1*
6**	01:02:23	03:16:46	1 (-94.44%)	00:03:20 (-78.84%)	00:03:20 (-97.21%)	1.67	*AC 1*
7**	01:19:37	02:55:28	3 (-85.71%)	00:06:55 (-89.38%)	00:08:45 (-97.84%)	2.34	*AC 2b*
8**	01:08:53	03:35:41	1 (-93.75%)	00:43:44 (+103.25%)	00:43:44 (-59.7%)	1.9	*AC 2b*
9*	02:41:46	01:21:46	2 (-90.48%)	00:11:35 (-60.96%)	00:13:23 (-88.61%)	2.45	*AC 5*
10	01:45:31	01:53:32	6 (-45.45%)	00:10:18 (-74.22%)	00:27:34 (-79.88%)	1.53	*AC 2b*
11*	01:24:05	02:57:27	2 (-92%)	00:24:13 (-53.67%)	00:36:56 (-93.59%)	2.59	*AC 2*
12**	01:09:22	03:48:19	0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	2.57	*AC 2b*
13**	01:14:07	02:58:37	2 (-90.48%)	00:07:45 (-65.32%)	00:09:53 (-89.97%)	2.56	*AC 2b*
14**	01:17:09	03:01:27	1 (-93.33%)	00:32:51 (+62.49%)	00:32:51 (-62.24%)	2.46	*AC 2b*
15**	01:29:45	03:33:21	0 (-100%)	00:00:00 (-100%)	00:00:00 (-100%)	1.47	*AC 2b*

Table 7 continued

ID	Mean service time	Mean buffer	Delayed aircraft (change)	Mean delay (change)	Maximum delay (change)	CPU time (sec)	Best client order
16	03:28:00	01:19:48	3 (-78.57%)	00:18:04 (-6.15%)	00:42:11 (-59.83%)	1.73	*AC 5*
17**	02:46:20	01:36:42	2 (-90.48%)	00:25:21 (-29.91%)	00:32:11 (-79.42%)	3.4	*AC 5*
18**	01:21:43	02:35:02	5 (-76.19%)	00:05:56 (-89.54%)	00:08:30 (-97.75%)	2.73	*AC 2b*
	01:37:25	**02:50:06**	**1.72 (-86.61%)**	**00:15:34 (-51.01%)**	**00:19:43 (-85.78%)**	**2.08**	
19	02:59:07	01:32:10	2	00:08:25	00:13:30	6.93	*AC 5*
20	01:12:49	03:23:15	2	00:08:49	00:13:30	10.02	*AC 4*

* Less than 15 minutes delay and 1 delayed aircraft difference compared to Table 3. ** Identical or improved number and duration of delays compared to Table 3

with many stochastic elements, including aircraft arrival times. Although our algorithm is already very comprehensive, we rely on its speed to repeatedly start new optimizations as soon as a disruption occurs. However, the more synchronization aspects are introduced into a problem, the more changes there will be in the reconstructed solution compared to the initial one. An interesting future work would therefore be to adapt the algorithm to take into account other criteria that help to minimize changes to the initial solution during updates.

Acknowledgements We are grateful to ASL Airlines Belgium and especially to Jean-Marc Urbani, CIO of ASLB, for their openness to sharing data. We extend our thanks to Liège Airport for their continued support.

Author Contributions Conceptualization: Jenny Tonka, Célia Paquay, Michaël Schyns; Methodology: Jenny Tonka, Michaël Schyns; Formal analysis and investigation: Jenny Tonka; Writing - original draft preparation: Jenny Tonka; Writing - review and editing: Jenny Tonka, Célia Paquay, Michaël Schyns; Supervision: Célia Paquay, Michaël Schyns.

Funding Jenny Tonka is a Research Fellow of the Fonds de la Recherche Scientifique – FNRS.

Declarations

Conflicts of interest The authors have no competing interests to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

References

Abdzadeh B, Noori S, Ghannadpour SF (2022) Simultaneous scheduling of multiple construction projects considering supplier selection and material transportation routing. Autom Constr 140:104336. https://doi.org/10.1016/j.autcon.2022.104336

Aguayo MM, Avilés FN, Sarin SC et al (2025) The vehicle routing problem with transfers. Comput Oper Res 177:106980. https://doi.org/10.1016/j.cor.2025.106980

Andreatta G, Capanna L, De Giovanni L et al (2014) Efficiency and Robustness in a Support Platform for Intelligent Airport Ground Handling. J Intell Transp Syst 18(1):121–130. https://doi.org/10.1080/15472450.2013.802160

Archetti C, Coelho LC, Speranza MG et al (2025) Beyond fifty years of vehicle routing: Insights into the history and the future. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2025.06.014

Braekers K, Ramaekers K, Van Nieuwenhuyse I (2016) The vehicle routing problem: State of the art classification and review. Comput Ind Eng 99:300–313. https://doi.org/10.1016/j.cie.2015.12.007

Bredström D, Rönnqvist M (2008) Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. Eur J Oper Res 191(1):19–31. https://doi.org/10.1016/j.ejor.2007.07.033

Britto R, Dresner M, Voltes A (2012) The impact of flight delays on passenger demand and societal welfare. Transp Res Part E Logist Transp Rev 48(2):460–469. https://doi.org/10.1016/j.tre.2011.10.009

Caceres-Cruz J, Arias P, Guimarans D et al (2014) Rich Vehicle Routing Problem: Survey. ACM Comput Surv 47(2):1–28. https://doi.org/10.1145/2666003
Cappanera P, Requejo C, Scutellà MG (2020) Temporal constraints and device management for the Skill VRP: Mathematical model and lower bounding techniques. Comput Oper Res 124:105054. https://doi.org/10.1016/j.cor.2020.105054
Chao IM (2002) A tabu search method for the truck and trailer routing problem. Comput Oper Res 29(1):33–51. https://doi.org/10.1016/S0305-0548(00)00056-3
Chen ST, Ermiş G, Sharpanskykh A (2023) Multi-agent planning and coordination for automated aircraft ground handling. Robot Auton Syst 167:104480. https://doi.org/10.1016/j.robot.2023.104480
Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6(1):80–91. https://doi.org/10.1287/mnsc.6.1.80
Dellaert N, Van Woensel T, Crainic TG et al (2021) A multi-commodity two-Echelon capacitated vehicle routing problem with time windows: Model formulations and solution approach. Comput Oper Res 127:105154. https://doi.org/10.1016/j.cor.2020.105154
Drexl M (2012) Synchronization in Vehicle Routing–A Survey of VRPs with Multiple Synchronization Constraints. Trans Sci 46(3):297–316. https://doi.org/10.1287/trsc.1110.0400
Drexl M (2013) Applications of the vehicle routing problem with trailers and transshipments. Eur J Oper Res 227(2):275–283. https://doi.org/10.1016/j.ejor.2012.12.015
Du J, Brunner J, Kolisch R (2014) Planning towing processes at airports more efficiently. Transp Res Part E Logist Transp Rev 70(1):293–304. https://doi.org/10.1016/j.tre.2014.07.008
Eksioglu B, Vural AV, Reisman A (2009) The vehicle routing problem: A taxonomic review. Comput Ind Eng 57(4):1472–1483. https://doi.org/10.1016/j.cie.2009.05.009
Elshaer R, Awad H (2020) A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Comput Ind Eng 140:106242. https://doi.org/10.1016/j.cie.2019.106242
Evler J, Asadi E, Preis H et al (2021) Airline ground operations: Schedule recovery optimization approach with constrained resources. Transp Res Part C Emerg Technol 128:103129. https://doi.org/10.1016/j.trc.2021.103129
Fedtke S, Boysen N (2017) Gantry crane and shuttle car scheduling in modern rail-rail transshipment yards. OR Spectr 39(2):473–503. https://doi.org/10.1007/s00291-016-0461-z
Fink M, Desaulniers G, Frey M et al (2019) Column generation for vehicle routing problems with multiple synchronization constraints. Eur J Oper Res 272(2):699–711. https://doi.org/10.1016/j.ejor.2018.06.046
Froger A, Jabali O, Mendoza JE et al (2022) The Electric Vehicle Routing Problem with Capacitated Charging Stations. Trans Sci 56(2):460–482. https://doi.org/10.1287/trsc.2021.1111
Gök YS, Guimarans D, Stuckey PJ, et al (2020) Robust resource planning for aircraft ground operations. In: 17th International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR), Vienna, Austria, pp 222–238, https://doi.org/10.1007/978-3-030-58942-4_15
Gök YS, Padrón S, Tomasella M et al (2023) Constraint-based robust planning and scheduling of airport apron operations through simheuristics. Ann Oper Res 320(2):795–830. https://doi.org/10.1007/s10479-022-04547-0
Grangier P, Gendreau M, Lehuédé F et al (2021) The vehicle routing problem with cross-docking and resource constraints. J Heuristics 27(1):31–61. https://doi.org/10.1007/s10732-019-09423-y
Guimarans D, Padrón S (2022) A stochastic approach for planning airport ground support resources. Int Trans Oper Res 29(6):3316–3345. https://doi.org/10.1111/itor.13104
Gunawan A, Kendall G, McCollum B et al (2021) Vehicle routing: Review of benchmark datasets. J Oper Res Soc 72(8):1794–1807. https://doi.org/10.1080/01605682.2021.1884505
Han X, Zhao P, Kong D (2023) Two-stage optimization of airport ferry service delay considering flight uncertainty. Eur J Oper Res 307(3):1103–1116. https://doi.org/10.1016/j.ejor.2022.09.023
Hu ZH, Wei C (2018) Synchronizing vehicles for multi-vehicle and one-cargo transportation. Comput Ind Eng 119:36–49. https://doi.org/10.1016/j.cie.2018.03.023
Huang YH, Blazquez CA, Huang SH et al (2019) Solving the feeder vehicle routing problem using ant colony optimization. Comput Ind Eng 127:520–535. https://doi.org/10.1016/j.cie.2018.10.037
Ip WH, Wang D, Cho V (2013) Aircraft Ground Service Scheduling Problems and Their Genetic Algorithm With Hybrid Assignment and Sequence Encoding Scheme. IEEE Syst J 7(4):649–657. https://doi.org/10.1109/JSYST.2012.2196229

Konstantakopoulos GD, Gayialis SP, Kechagias EP (2022) Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. Oper Res 22(3):2033–2062. https://doi.org/10.1007/s12351-020-00600-7

Krstić Simić T, Babić O (2015) Airport traffic complexity and environment efficiency metrics for evaluation of ATM measures. J Air Transp Manag 42:260–271. https://doi.org/10.1016/j.jairtraman.2014.11.008

Lahyani R, Khemakhem M, Semet F (2015) Rich vehicle routing problems: From a taxonomy to a definition. Eur J Oper Res 241(1):1–14. https://doi.org/10.1016/j.ejor.2014.07.048

Laporte G (1992) The vehicle routing problem: An overview of exact and approximate algorithms. Eur J Oper Res 59(3):345–358. https://doi.org/10.1016/0377-2217(92)90192-C

Laporte G (2009) Fifty Years of Vehicle Routing. Trans Sci 43(4):408–416. https://doi.org/10.1287/trsc.1090.0301

Masson R, Lehuédé F, Péton O (2013) An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers. Trans Sci 47(3):344–355. https://doi.org/10.1287/trsc.1120.0432

Meisel F, Kopfer H (2014) Synchronized routing of active and passive means of transport. OR Spectr 36(2):297–322. https://doi.org/10.1007/s00291-012-0310-7

Mendoza J, Hoskins M, Guéret C, et al (2014) VRP-REP: A vehicle routing community repository. In: VeRoLog'14, Oslo, Norway

Mor A, Speranza MG (2020) Vehicle routing problems over time: a survey. 4OR-Q J Oper Res 18(2):129–149. https://doi.org/10.1007/s10288-020-00433-2

Norin A, Yuan D, Granberg TA et al (2012) Scheduling de-icing vehicles within airport logistics: A heuristic algorithm and performance evaluation. J Oper Res Soc 63(8):1116–1125. https://doi.org/10.1057/jors.2011.100

Padrón S, Guimarans D (2019) An Improved Method for Scheduling Aircraft Ground Handling Operations From a Global Perspective. Asia-Pac J Oper Res 36(4):1950020. https://doi.org/10.1142/S0217595919500209

Padrón S, Guimarans D, Ramos J et al (2016) A bi-objective approach for scheduling ground-handling vehicles in airports. Comput Oper Res 71:34–53. https://doi.org/10.1016/j.cor.2015.12.010

Ryerson MS, Hansen M, Bonn J (2014) Time to burn: Flight delay, terminal efficiency, and fuel consumption in the National Airspace System. Transp Res Part A Policy Pract 69:286–298. https://doi.org/10.1016/j.tra.2014.08.024

Schyns M (2015) An Ant colony system for responsive dynamic vehicle routing. Eur J Oper Res 245(3):704–718. https://doi.org/10.1016/j.ejor.2015.04.009

Sim K, Hart E, Urquhart N, et al (2019) A new rich vehicle routing problem model and benchmark resource. In: Minisci E, Vasile M, Periaux J, et al (eds) Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences. Springer International Publishing, p 503–518, https://doi.org/10.1007/978-3-319-89988-6_30

Soares R, Marques A, Amorim P et al (2024) Synchronisation in vehicle routing: Classification schema, modelling framework and literature review. Eur J Oper Res 313(3):817–840. https://doi.org/10.1016/j.ejor.2023.04.007

Soares R, Parragh SN, Marques A et al (2025) The Robust Vehicle Routing Problem With Synchronization: Models and Branch-And-Cut Algorithms. Netw 86(3):296–324. https://doi.org/10.1002/net.22287

Solomon MM (1987) Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Oper Res 35(2):254–265. https://doi.org/10.1287/opre.35.2.254

Tilk C, Bianchessi N, Drexl M et al (2018) Branch-and-Price-and-Cut for the Active-Passive Vehicle-Routing Problem. Trans Sci 52(2):300–319. https://doi.org/10.1287/trsc.2016.0730

Tomasella M, Clare A, Gök Y, et al (2019) STTAR: A Simheuristics-Enabled Scheme for Multi-Stakeholder Coordination Of Aircraft Turnaround Operations. In: Winter Simulation Conference (WSC), National Harbor, MD, USA, pp 488–499, https://doi.org/10.1109/WSC40007.2019.9004787

Toth P, Vigo D (2014) Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, https://doi.org/10.1137/1.9781611973594

Wang S, Che Y, Zhao H et al (2020) Accurate Tracking, Collision Detection, and Optimal Scheduling of Airport Ground Support Equipment. IEEE Internet Things J 8:572–584. https://doi.org/10.1109/JIOT.2020.3004874

Wang YZ, Hu ZH, Tian XD (2024) Scheduling ASC and AGV considering direct, buffer, and hybrid modes for transferring containers. Comput Oper Res 161:106419. https://doi.org/10.1016/j.cor.2023.106419

Wu Y, Zhou J, Xia Y et al (2023) Neural Airport Ground Handling. IEEE Trans Intell Transp Syst 24(12):15652–15666. https://doi.org/10.1109/TITS.2023.3253552

Zhou J, Wu Y, Cao Z et al (2023) Learning Large Neighborhood Search for Vehicle Routing in Airport Ground Handling. IEEE Trans Knowl Data Eng 35(9):9769–9782. https://doi.org/10.1109/TKDE.2023.3249799

Zhu S, Sun H, Guo X (2022) Cooperative scheduling optimization for ground-handling vehicles by considering flights' uncertainty. Comput Ind Eng 169:108092. https://doi.org/10.1016/j.cie.2022.108092

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.