Transportation Research Part E 161 (2022) 102679

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Fleet sizing and routing of healthcare automated guided vehicles

Imadeddine Aziez [a,b], Jean-François Côté [a,*], Leandro C. Coelho [a,b]

[a] *CIRRELT and Université Laval, Québec, Canada*
[b] *Canada Research Chair in Integrated Logistics, Canada*

ARTICLE INFO

Keywords:
Automated guided vehicles
Healthcare logistics
Fleet sizing problem

ABSTRACT

This paper proposes techniques to improve the efficiency of transportation activities within hospitals. Transportation is a major activity in hospitals which requires management efforts to improve its efficiency and increase the quality of patient services. One of the most powerful ways of achieving more efficiency of transportation systems in hospitals is via the labor-saving technology of logistics processes using automated guided vehicles (AGVs). In this paper, we study the fleet sizing and routing problem with synchronization for AGVs with dynamic demands (FSRPS-AGV) in the context of a real-life application. The goal is to simultaneously optimize the number and types of carts and AGVs needed to perform all daily requests in a hospital while optimizing AGVs' routes and respecting time constraints. Different requests require different types of carts, which are transported by the AGVs. Each request is composed of several tasks consisting of moving material from a pickup point to a delivery point. Operation synchronization among the tasks of the same request (task 1 must be performed before task 2, possibly by different AGVs but using the same cart) and movement synchronization with respect to AGVs and carts (an AGV may drop off a cart, and later another AGV picks up the cart to continue serving the same or another request) are major challenges of the addressed problem. In this paper, we describe, model, and solve the FSRPS-AGV. We introduce a mathematical formulation and propose a powerful matheuristic based on a fast and efficient dynamic reoptimization of the routes upon the arrival of new requests. We compare the performance of our matheuristic under different scenarios, including against the solutions obtained by an oracle, showing that it can handle dynamism of demand very well and achieve near-optimal solutions. We assess our methods using small and large instances generated based on real data from an industrial partner. We demonstrate that significant performance improvements and important savings can be achieved with a small degree of flexibility in timing constraints and less conservative speed estimations (while taking into account safety concerns). Finally, we provide managerial insights with respect to the number of AGVs and carts that should be acquired by our industrial partner.

1. Introduction

Healthcare systems around the world are facing many challenges such as increasing costs of treatment, growing elderly population, and shortage of doctors, nurses, and ancillary staff (Bačík et al., 2017). These and other challenges bring up the need for improving hospital operations in order to increase their efficiency and effectiveness. Chikul et al. (2017) state that approximately 80% of healthcare expenses are related to patient care activities, thereby, considerable savings can be obtained by improving

* Corresponding author.
E-mail addresses: imadeddine.aziez.1@ulaval.ca (I. Aziez), jean-francois.cote@fsa.ulaval.ca (J.-F. Côté), leandro.coelho@fsa.ulaval.ca (L.C. Coelho).

https://doi.org/10.1016/j.tre.2022.102679
Received 29 June 2021; Received in revised form 26 January 2022; Accepted 13 March 2022
Available online 4 April 2022

clinical practices by better managing supplies, labor, equipment, logistics, and facilities. Transportation activities in hospitals are complex due to the large number of materials and equipments used, justifying the need for more efficient and productive transportation systems. One of the most powerful ways of achieving these goals is via the automation of logistics processes, for example, employing automated guided vehicles (AGVs) (Bačík et al., 2017). Transportation activities such as daily drug delivery are traditionally performed manually using handcarts, hence, the multiple steps in this process impact the efficiency and increase the risk of contamination (Chen et al., 2021a). The automation of this process is then needed to ensure high safety levels and high system efficiency. In addition, the Covid-19 pandemic has shown the importance of contactless delivery and automation (Chen et al., 2021b). Using AGVs to handle transportation in hospitals can provide flexibility and help better protect the staff by minimizing contact.

The design of AGV systems requires taking into consideration a set of strategic (system design), tactical (fleet sizing, flowpath layout design), and operational issues (dispatching, routing, scheduling) (Hamzheei et al., 2013; Le-Anh and De Koster, 2006; Vis, 2006). In this paper, we study the fleet sizing and routing problem with synchronization for AGVs with dynamic demands (FSRPS-AGV) which combines tactical and operational decisions in the same setting.

The present paper finds a real-life application in the context of a collaboration with the healthcare institution "Centre hospitalier universitaire (CHU) de Québec Université Laval". This setting is characterized by highly uncertain and dynamic demands. In this system, several AGVs must be available to perform automated transportation tasks. A task, such as to deliver food, transport equipment, recover trash, etc. requires that a specific type of cart be transported by an AGV. Thus, the automation of logistics operations using AGVs requires some key optimization decisions: optimizing the number of expensive carts and AGVs used to perform all daily requests while ensuring a high service level.

In this paper, we model and solve the FSRPS-AVG using mathematical programming. In order to cope with large instances and real-time operations, a powerful matheuristic algorithm is also proposed. This algorithm handles dynamic demand as the routes are re-planned immediately upon the arrival of new requests. Each time new requests arrive, we solve a static problem that contains all the known requests that have not been served so far. The solution procedure starts by heuristically optimizing the number of AGVs and their routes, then based on these routes, a mathematical model is solved to optimize the number of carts. We also solve a static version of the problem which assumes the knowledge of all requests at the start of the day. From fleet sizing perspective, to the best of our knowledge, this paper is the first to address a double fleet sizing problem of AGVs and carts while taking into account multiple synchronization constraints and several types of carts.

The remainder of the paper is organized as follows. Section 2 describes the problem and outlines synchronization challenges. Section 3 is devoted to the literature review. Section 4 provides a formal description and a mathematical formulation for the problem. Section 5 presents an overview of our matheuristic. Detailed computational experiments are presented in Section 6 and conclusions follow in Section 7.

2. Problem description

In the FSRPS-AGV, each one of the daily requests is composed of one or multiple transportation tasks, and each task is defined by a pickup–delivery pair. For instance, an AGV performing a request with two tasks starts by driving from the depot to the pickup location of the first task to pickup a cart of a given type. Then, the AGV drives to the delivery location to deliver the cart. Subsequently, this delivery location is the origin of the next task of the AGV. A time window (TW) is associated with each task imposing a lower and an upper bound on the start time of service, and more than one AGV may be used to fulfill the tasks of a given request.

An interdependence problem is raised in this setting which means that any change in one AGV route may affect other routes, so two main aspects of this problem that are present in the FSRPS-AGV are addressed next. The first one concerns the possibility of serving the tasks of the same request using multiple AGVs, hence, a change in one AGV route may affect other routes, and in some scenarios, a change in one AGV route may make all other routes infeasible. The second one concerns the interdependence between carts and AGVs, meaning that a cart must always be transported by an AGV. Drexl (2012) states that addressing the interdependence problem may require different types of synchronization, and introduced five types of synchronization in VRPs with multiple synchronization constraints (VRPMSs). The FSRPS-AGV covers three synchronization types among the five introduced by Drexl (2012), namely: (i) task synchronization: a decision must be made regarding which vehicle(s) to fulfill each task, where a task may be pickup–delivery, visiting a location to provide a service, etc; (ii) operation synchronization: concerns the time coordination for the operations of different vehicles at the same or at different locations with respect to the time at which the vehicles perform their operations, implying that the computation of a schedule for one vehicle needs to consider schedules for other vehicles; (iii) movement synchronization: it refers to space–time coordination of autonomous vehicles and non-autonomous vehicles, which means that for a non-autonomous (cart) to be able to move along a certain arc, a compatible autonomous vehicle (AGV) must move along the same arc at the same time. The last two synchronization types introduced by Drexl (2012) are load and resource synchronization. For the detailed description of the five synchronization types, the reader is referred to Drexl (2012). In what follows, we focus on operation and movement synchronizations raised in the problem in hand, and we neglect discussing task synchronization as it is trivial for the problem.

The first synchronization challenge raised in the FSRPS-AGV is with respect to the temporal coordination of AGVs serving the tasks of the same request due to the existence of overlapping time windows. One may argue that one way to avoid synchronization with respect to temporal aspect in some transportation systems is to have disjoint time windows whenever it is possible; however, it is important to mention that in many cases more efficiency can be achieved by having overlapping time windows. A comparison

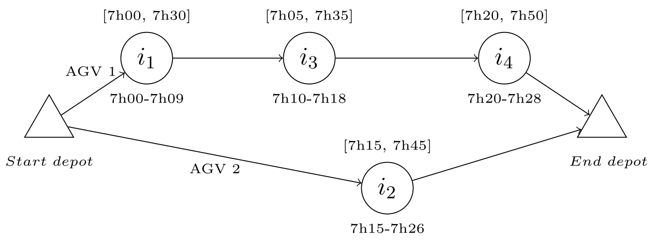

Fig. 1. An example of satisfying requests using AGVs and carts.

between two AGV fleet exploitation strategies is conducted in Section 6.3.3: in the first one, only one AGV is used to satisfy a full request, whereas in the second one many AGVs are synchronized to serve the tasks of a request. Furthermore, the tasks of the same request must be served using the same cart. Consequently, one may observe that this problem is characterized by two fleets with different operating modes. Therefore, another synchronization challenge is related to the space–time coordination of AGVs and carts, meaning that the carts must always move with AGVs (movement synchronization).

It is important to highlight that AGVs and carts are used to serve the requests of 10 different departments at the hospital; while the AGV fleet is homogeneous, the cart fleet is heterogeneous and is composed by 10 different types of carts depending on the transportation task that must be performed. Carts cannot be interchanged, meaning that each cart type is limited to specific transportation tasks.

Fig. 1 depicts a detailed example using AGVs and carts to serve three requests. It also shows how synchronization constraints are intricate and challenging. The requests are r_1, r_2, and r_3 such that $r_1 = \{i_1, i_2\}$, $r_2 = \{i_3\}$, $r_3 = \{i_4\}$ are the sets of tasks of each request. Request r_1 is composed of two tasks, while requests r_2 and r_3 are composed of only one task each. Each task corresponds to a pickup–delivery pair as previously described. For the sake of this example, we assume that requests r_1 and r_3 use the same type of cart while request r_2 uses another type of cart. Tasks' TWs are shown above task nodes, the start and end times of service are specified below each node.

In this example, a first empty AGV must be sent from the depot to perform the first task of request r_1 which consists of picking up cart 1 from a pickup location and transporting it to the delivery location. We then assume that the pickup location of i_3 is closer to the drop location of i_1 than the start depot. Therefore, once AGV 1 has finished serving task i_1, instead of waiting to perform task i_2, it is assigned to serve task i_3 where cart 2 is used, while task i_2 which is served using cart 1 will be done by AGV 2. AGV 1 continues its operations by serving task i_4 of request r_3 which is served using cart 3. While requests r_1 and r_3 use the same type of cart as mentioned earlier, they are not served using the same cart because of the overlapping service times between the second task of request r_1 and the task of request r_3. AGVs return to the depot once their routes are over for storage and charging.

In the example of Fig. 1, it is clear how synchronization plays an important role in satisfying these tasks due to the overlapping TWs of the tasks of many requests and the need to couple/decouple AGVs and carts at each transportation operation. For instance, during the optimization process, if other tasks are performed before task i_1, its service might be delayed which creates a synchronization problem in case where the time of finishing the service at task i_1 is after 7h15 which is the TW lower bound on the start time of service of the task i_2. This means that synchronization constraints must be handled in order to ensure that the start time of service at node i_2 is larger than or equal to the time of finishing the service at node i_1. Furthermore, a delay in the route of AGV 1 may create an opportunity to reuse cart 1 in order to serve task i_4. Space–time synchronization between AGVs and carts is also crucial in this case because carts must be transported using AGVs.

3. Literature review

According to the survey of Vis (2006), AGVs enable a driverless transportation system that is used for the movement of materials. Their use has grown enormously and the number of applications has increased significantly. The main application areas lie in flexible manufacturing systems (Nishi et al., 2011), container terminals (Chen et al., 2020), warehouses (Boysen et al., 2019), and healthcare industry (Chikul et al., 2017). For healthcare facilities, AGVs are used to handle a variety of materials (samples, food, drugs, waste, instruments, laundry, etc.). Different types of carts are also used in material handling operations by AGVs such as: waste carts, food carts, flatbed carts, etc.

The FSRPS-AGV shares many characteristics with other problems from the literature, namely the pickup and delivery problem with TW (PDPTW), the dynamic vehicle routing problem (DVRP), and the VRP with trailers and transshipments (VRPTT). These are briefly reviewed next.

In the PDPTW, when a vehicle serves a customer request, it must pickup a given item at the origin location and deliver it to its destination. The FSRPS-AGV resembles the PDPTW as each task is composed of an origin–destination pair and each request is composed of multiple origin–destination tasks. Different heuristic algorithms have been proposed to solve variants of the PDPTW, including adaptive large neighborhood search (ALNS) (Pisinger and Ropke, 2007; Sun et al., 2020), particle swarm optimization (Ai and Kachitvichyanukul, 2009), parallel neighborhood descent (Subramanian et al., 2010), and tabu embedded simulated annealing (Şahin et al., 2013). Several exact algorithms have also been proposed for the PDPTW including the branch-and-cut (Ropke et al., 2007), branch-cut-and-price of (Ropke and Cordeau, 2009; Veenstra et al., 2017), and a set partitioning-based algorithm (Baldacci et al., 2011; Sun et al., 2019).

The DVRP is a routing problem with many real-life applications in logistics. In the DVRP, the data available to the planner may change during the execution of the routes. For instance, the data can be in the form of customer requests to deliver items or requests for trips from an origin to a destination that appear dynamically. Pillac et al. (2013) surveyed contributions on the DVRP and classified routing problems from the perspective of information quality and evolution. Lund et al. (1996) defined the concept of degree of dynamism δ as the ratio between the number of dynamic requests n_d and the total number of requests n_{tot} as $\delta = \frac{n_d}{n_{tot}}$. The survey of Ritzinger et al. (2016) summarizes the literature in the area of dynamic and stochastic VRPs. Azi et al. (2012) studied a DVRP in which vehicles are used to perform deliveries over multiple routes a day. The problem was solved heuristically using ALNS which takes into account uncertainty by generating scenarios containing possible demand realizations. A generalization of the DVRP is the dynamic PDP (Berbeglia et al., 2010) in which the transportation decisions are performed in real-time (Arslan et al., 2019; Mitrović-Minić and Laporte, 2004).

In the VRPTT, a set of customers with a given supply must be visited by a limited fleet of heterogeneous vehicles (lorries and trailers) to collect it. The lorries can move on their own, whereas trailers must be pulled by a lorry. Customers differ as some of them can be visited by a lorry with a trailer while others can only be visited by a lorry without a trailer. Load transfers from lorries to trailers are performed at specific points called transshipment locations. In addition to TW, synchronization requirements with regard to spatial, temporal, and load aspects must also be taken into consideration. The VRPTT has received less attention in the literature despite its numerous applications (Drexl, 2013). The problem can be solved using the branch-and-cut algorithm of Drexl (2014). A special case of the VRPTT which also deals with trailers is the truck and trailer routing problem (TTRP) (Parragh and Cordeau, 2017; Regnier-Coudert et al., 2016). In the TTRP, which was originally introduced by Chao (2002), a fleet of trucks and trailers serves a set of customers. Some customers with accessibility constraints must be served just by truck, while others can be served either by truck or by a complete vehicle (a truck pulling a trailer) in a similar way as in the VRPTT (Villegas et al., 2013). However, an additional restriction on the truck–trailer assignment is imposed, that is, only one truck is responsible for pulling each trailer, and it is the only one that can transfer a load into the trailer. Therefore, the only synchronization requirement in this case is related to customer covering which is also known as task synchronization. Drexl (2013) stated that the VRPTT constitutes an archetypal representative of the class of VRPMSs (e.g., Fink et al. (2019)). Synchronization requirements may also appear in many real-world VRPs such as delivery and installation routing problems (Ali et al., 2021).

The literature dedicated to the design of AGV systems for healthcare applications has grown in the last 20 years (Bhosekar et al., 2021a). Katevas (2001) provided guidelines for the design of a mobile robotic system in healthcare environments, and Benzidia et al. (2019) proposed specific guidelines related to AGV projects for healthcare facilities. They highlighted the need for new knowledge and skills to improve the design and management of AGV systems in hospitals. Some studies included a comparison between manual and automated material handling systems. Chikul et al. (2017) compared a manual hospital supply chain management model versus a process that uses either Radio Frequency Identification (RFID) technology or AGVs, in a general hospital in Singapore. The results show that combining RFID and AGVs leads to high cost savings in the supply chain model. Another comparison between a manual and an AGV-based material handling system is conducted by Rossetti et al. (2000). They proposed a simulation model whose results show that the AGV-based system is economically viable to handle the delivery of clinical supplies and pharmaceuticals in a hospital. Fragapane et al. (2020) compared five innovative applications of autonomous mobile robots (AMRs) including AGVs with other material handling systems applied in hospitals. The study highlights the benefits of using AMRs in healthcare environments. The authors stated that AMRs can support and collaborate closely with hospital personnel to increase value-added time for patient care.

Simulation is commonly used to study AGV systems in healthcare facilities. Rossetti and Selandari (2001) proposed a simulation model to assess the performance of a robotic healthcare delivery system. The authors provide a methodology for automation introduction evaluation in a hospital environment with multiple technical and economical performance indicators. The results show that the robotic system outperforms a human-based one in handling internal deliveries. Pedan et al. (2017) used a simulation-based case study to demonstrate the advantages of using an AGV system in a healthcare environment. Fragapane et al. (2018) studied healthcare material distribution and transportation based on a Norwegian case study in order to stimulate further research in material handling and distribution in hospitals. They developed an agent-based simulation model to assess the performance of the AGV-based delivery system in place and to investigate potential improvement measures.

On the fleet sizing problem in a healthcare context, Bhosekar et al. (2021b) developed a case study using real data from the delivery of surgical carts to operating rooms via AGVs. They proposed a framework that integrates data analysis with system simulation and optimization to redesign AGV pathways and operational AGV fleet sizing. For the AGV fleet sizing part, which is closely related to our study, the authors use a simulation–optimization model to evaluate the performance of the system for different AGV fleet sizes. Ganesharajah et al. (1998) stated that the complex nature of the issues involved in determining fleet size makes simulation the most promising tool for this job, however, estimating the fleet size using analytical models can be a good starting point for a detailed simulation.

AGV fleet sizing is an important problem which received considerable attention in the literature. In what follows, we review papers discussing AGV fleet sizing problem in different practical contexts. Egbelu (1987) proposed four analytical models to estimate the AGV fleet size in a flexible manufacturing system (FMS) facility, based on several information sources such as the expected number of loaded trips between stations and the number of workstations in the facility. Maxwell and Muckstadt (1982) and Mahadevan and Narendran (1993) also proposed similar approaches to tackle the problem. Rajotia et al. (1998) proposed a mixed integer programming model to determine the optimal AGV fleet size in an FMS. Considering load-handling time, empty travel time, waiting time, and congestion time, the objective function is to minimize empty trips made by AGVs. A simulation approach was used to validate the results of the model. Vis et al. (2001) developed a minimum flow algorithm to determine the number of AGVs required at a semi-automated container terminal. They introduced network flow modeling for the problem. Vis et al. (2005) studied the AGV fleet size problem under time window constraints at a container terminal. They developed an integer linear programming model. The efficiency of this model was tested under various conditions (e.g., unloading cycle times, buffer size). The results were validated using a simulation model. The results show that the analytical model slightly underestimates the fleet size compared to the best fleet size obtained by the simulation approach. They concluded that the analytical model performs well in the context of a container terminal. Vivaldini et al. (2016) studied a problem of estimation of the minimum number of AGVs in a warehouse. The authors applied shortest job first rule and tabu search algorithms to determine a task assignment, in addition to an enhanced Dijkstra algorithm applied for the conflict-free routing. Other analytical approaches used to estimate the required number of AGVs include: queuing models (Choobineh et al., 2012; Talbot, 2003; Tanchoco et al., 1987), statistical approach (Arifin and Egbelu, 2000), and multi-criteria decision modeling (Sinriech and Tanchoco, 1992). Another research stream focuses on using simulation approaches to determine the number of AGVs (Bhosekar et al., 2021b; Chang et al., 2014; Yifei et al., 2010). Ganesharajah et al. (1998) investigated design and operational issues in AGV-served manufacturing systems including AGV fleet sizing with analytical and simulation methods.

In light of the reviewed literature, one may observe that non-autonomous objects (e.g., carts, pallets, etc.) are often neglected when studying or designing AGV systems in general, and in particular when tackling the AGV fleet sizing problem. This may be justified in some environments where the non-autonomous objects transported by AGVs such as pallets in a manufacturing environment do not have a significant impact on the investment cost of the system compared to the cost of acquiring AGVs. However, in a healthcare environment, the situation is somewhat different, where reducing the cost of acquiring carts is as important as aiming to reduce the cost of acquiring AGVs due to their important impact on the investment cost of the overall system. Therefore, investigating the AGV and cart fleet sizing simultaneously is an important problem to tackle in order to lower the total investment cost of the entire AGV system. Furthermore, research on synchronizing autonomous and non-autonomous objects are still rare (Drexl, 2013). Indeed, Drexl (2013) mentioned that the robot coordination is among the relevant fields to the VRPMSs that constitute an interesting and promising research perspective. Hence, the present work is also a contribution to the literature of the VRPMSs as it exhibits in addition to the usual task covering constraints, further synchronization requirements concerning spatial and temporal coordination of one or multiple AGVs to a cart in order to satisfy a given request.

4. Mathematical formulation

Let us start by presenting the static counterpart of the true dynamic version of the FSRPS-AGV. In reality, not all requests are known at the beginning of the planning horizon, but they rather arrive dynamically throughout the day. For the sake of this description, we assume that all requests are known, or equivalently, that we model the problem considering only the requests known at this time.

The problem can be represented on a graph $G = (V, A)$ in which the set of nodes is $V = N \cup \{0, n+1\}$, where $N = \{1, \dots, n\}$ is the set of task nodes, and the starting and ending depots are represented by two copies 0 and $n+1$ of the same physical nodes. Each task node $n \in N$ represents a pickup–delivery pair. Note that modeling each task (a pickup–delivery pair) as a node rather than using separate nodes for the pickup and delivery locations yields a smaller and easier problem to solve due to the reduction of the number of nodes in the problem. Let R be the set of requests to be routed. Each request $r \in R$ is composed of one or multiple task nodes. Let $r(i)$ be the request associated with node $i \in N$. Precedence constraints among the tasks of the same request $r \in R$ must be respected when serving them, such that they are served sequentially. We consider a fleet of homogeneous AGVs sufficiently large to serve all the requests such that K is the set of identical AGVs with capacity of carrying one cart. We also consider a fleet of heterogeneous carts which is composed by h different cart types with $H = \{1, \dots, h\}$. For each cart type $j \in H$, a sufficiently large number of carts is available to serve the requests that require this type of cart. Only one type of cart is used to serve each request, and it is known beforehand. Each time a cart finishes the service of a given request, it can be reused to serve another request which uses the same cart type. We define $W = \{(r_i, r_j) \in R \times R |\ r_i \text{ and } r_j \text{ use the same type of cart}\}$ as the set of pairs of requests which use the same type of cart.

The set of arcs is $A = V \times V$, omiting arcs that lead to infeasible solutions, as described next. Let $A = A_a \cup A_c$ where A_a is the set of arcs traversed by an AGV alone and A_c is the set of arcs traversed by an AGV carrying a cart. Each node $i \in V$ has a service time s_i and a time window $[a_i, b_i]$ allowing an AGV to arrive at i prior to a_i but waiting until a_i to serve the node, and service must start not later than b_i. Note that the service time s_i of each node $i \in V$ is calculated as the sum of: a fixed time to perform the pickup operation, the travel time between the corresponding pickup–delivery pair, and a fixed time to perform the delivery operation. AGVs can depart from the depot at a_0 and must return not later than b_{n+1}. Furthermore, a routing cost $c_{ij} \geq 0$ is associated with each arc $(i, j) \in A$, and a travel time $t_{ij} \geq 0$ is associated with each arc $(i, j) \in A$. The triangle inequality holds for routing costs and travel times as the distances represent shortest paths. Obviously, the travel time between two consecutive tasks of the same request is null due to the fact that the delivery location of a given task is the pickup location of the next task. Assumptions with respect to routing cost c_{ij} are outlined in Section 6.2.

To illustrate the omitted arcs from the set A, we use an example of two requests $r_1, r_2 \in R$ such that $r_1 = \{i_1, i_2\}$, $r_2 = \{i_3, i_4\}$ are the sets of nodes of r_1 and r_2, respectively. Knowing that precedence constraints among the tasks of the same request hold, we have $i_1 \prec i_2$ and $i_3 \prec i_4$. Fig. 2 depicts the feasible and omitted arcs in the graph of the example. For simplicity, the graph is presented in two subfigures to illustrate the feasible and omitted arcs of the set A_c in Fig. 2(a) and of the set A_a in Fig. 2(b). However, in reality,

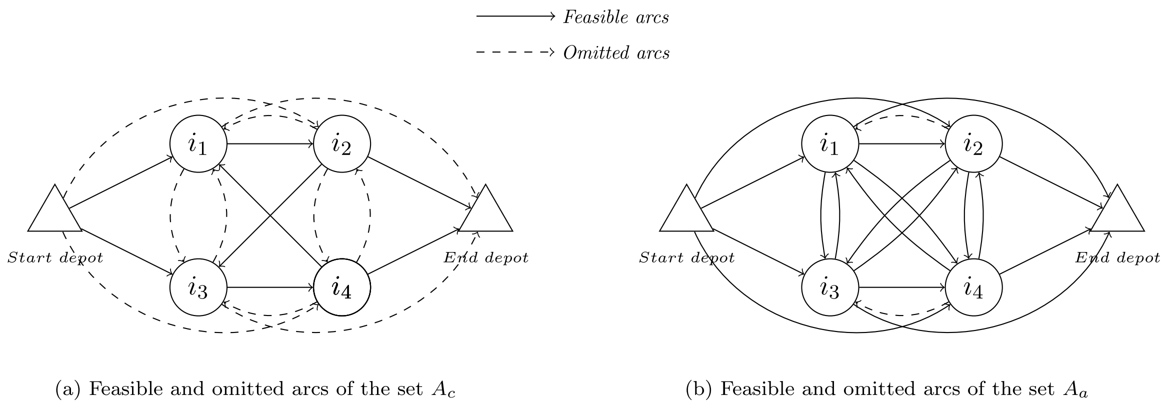
Feasible arcs
Omitted arcs
i_1
i_2
i_3
i_4
Start depot
End depot
(a) Feasible and omitted arcs of the set A_c
(b) Feasible and omitted arcs of the set A_a

Fig. 2. An example of feasible and omitted arcs of the set A.

the arcs in both subfigures are part of the same graph. In what follows, additional arc omission cases which are not illustrated in Fig. 2 are highlighted. In Fig. 2(a), we assume that requests r_1 and r_2 are served using the same type of carts, however, in the case where requests r_1 and r_2 are served by different types of carts, all the arcs between r_1 and r_2 and vice-versa must be omitted. Furthermore, the violation of TW may result in arc omission, for example, if $a_{i_4} + s_{i_4} + t_{i_4 i_1} > b_{i_1}$, then the arcs between i_4 and i_1 are omitted. Finally, additional trivial omitted arcs such as arcs from the end depot to all the nodes are not illustrated in Fig. 2 for simplicity. Let $A_a^+(i)$ be the set of nodes j such that there is an arc from i to j, that is, $A_a^+(i) = \{j \in V | (i,j) \in A_a\}$. Similarly, $A_a^-(i) = \{j \in V | (j,i) \in A_a\}$, $A_c^+(i) = \{j \in V | (i,j) \in A_c\}$, $A_c^-(i) = \{j \in V | (j,i) \in A_c\}$.

A solution to the FSRPS-AGV optimizes the number of AGVs and carts used to serve all the requests while minimizing the routing costs, and assigns the requests to the AGVs and carts while routing them. The static version of the problem can be mathematically formulated with parameters α_j representing the fixed cost of the cart used to serve node j, and β representing the fixed cost of using one AGV such that $\beta > \alpha_j$, $\forall j \in N$. We define the following decision variables: x_{ij} equal to 1 if arc (i,j) is traversed by an AGV, 0 otherwise; y_{ij} equal to 1 if arc (i,j) is traversed by a cart, 0 otherwise; and S_i indicating the start time of service at node $i \in V$. The model can then be formulated as follows:

$$\min \sum_{j \in N} \alpha_j y_{0j} + \sum_{j \in N} \beta x_{0j} + \sum_{(i,j) \in A_a} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j \in A_c^+(i)} y_{ij} = 1 \qquad i \in N \tag{2}$$

$$\sum_{j \in A_c^-(i)} y_{ji} = 1 \qquad i \in N \tag{3}$$

$$\sum_{j \in A_a^+(i)} x_{ij} = 1 \qquad i \in N \tag{4}$$

$$\sum_{j \in A_a^-(i)} x_{ji} = 1 \qquad i \in N \tag{5}$$

$$S_j \geq S_i + s_i + t_{ij} \qquad i,j \in r, r \in R, j \succ i \tag{6}$$

$$S_j \geq S_i + s_i + t_{ij} - M\left(1 - y_{ij}\right) \qquad (i,j) \in A_c, (r(i), r(j)) \in W \tag{7}$$

$$S_j \geq S_i + s_i + t_{ij} - M\left(1 - x_{ij}\right) \qquad (i,j) \in A_a \tag{8}$$

$$a_i \leq S_i \leq b_i \qquad i \in V \tag{9}$$

$$x_{ij}, y_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{10}$$

$$S_i \geq 0 \qquad i \in V. \tag{11}$$

The objective function (1) minimizes the total cost. The first two terms relate to the fixed cost of using carts (of different types) and AGVs, and the third term relates to the routing cost. Constraints (2)–(5) are degree constraints requiring each node to be visited exactly once by an AGV and a cart. Constraints (6) preserve the precedence order of tasks in each request $r \in R$; they also handle time synchronization among the tasks of the same request. The consistency of the time variables is ensured by constraints (7) and (8) where M is a big number and can be set to the maximum return time to the depot b_{n+1}; they also ensure space–time coordination of AGVs and carts. Constraints (9) impose TW constraints. Moreover, constraints (7)–(9) guarantee that there are no subtours in the solutions. Constraints (10) and (11) impose the nature and the domain of the variables.

In the dynamic version of the problem, not all the requests are known in advance but must be assigned to AGVs as the routes for serving previously assigned requests are executed. Once new requests arrive, the routes are re-planned immediately. Each request

$r \in R$ is given a release time $\mathcal{R}_r$ (i.e., the moment it is revealed on the time horizon) which can be just a few minutes before the start of the TW of its first task. When solving our model, we can accommodate dynamic requests as follows. The nodes of an assigned request are fixed in their corresponding routes as soon as an AGV departs (from the depot or a previously visited node) to serve the first node of the request. Accordingly, the nodes of new requests can only be inserted in routes after the nodes whose positions are fixed. Hence, our model (1)–(11) can solve the dynamic problem as a series of static ones.

5. Matheuristic algorithm for the FSRPS-AGV

This section describes the algorithm proposed to solve the FSRPS-AGV, presenting its general dynamic framework. The algorithm is based on a periodic reoptimization of the AGV routes whenever one or multiple requests arise. According to Ozbaygin and Savelsbergh (2019), periodic reoptimization has been used in many existing methods of dynamic and deterministic VRPs, either at pre-determined moments or when a certain number of changes to the input data have occurred. The latter case applies to our algorithm. In this algorithm, upon the arrival of new requests, a static problem is solved with the latest available data following a two-stage "AGV First, Cart Second" approach. In the first stage, an ALNS framework is used to optimize the routes and number of AGVs (see Section 5.1), where mixed integer program (MIP) insertion operators are developed to handle synchronization requirements. In the second stage, we propose a mathematical model to optimize the number of carts needed to serve the requests already assigned to AGVs in the first stage (see Section 5.2). This model aims at better exploiting the AGV fleet so that the number of carts needed in the system is reduced. This is done by conserving the AGV routes defined in the first stage, and changing departure and start times of service of AGVs at their corresponding locations while respecting TW and synchronization constraints.

Algorithm 1 provides an overview of the periodic reoptimization approach proposed in this paper. We assume at the beginning that a sufficiently large number of AGVs is available to serve the requests, then an initial solution is built with all known requests by inserting them sequentially into AGV routes one by one using MIP insertion described in Section 5.1.2. To minimize the number of AGVs in the initial solution, we set a high insertion cost on moves within an AGV route that has not yet been used. More AGVs can be used during the execution of ALNS if the need arises. In the $while$ loop from lines 2–7, we repeatedly reoptimize the previously planned AGV routes by solving a static problem with the latest available data upon the arrival of new requests (new event). In the remainder of the paper, we also refer to the arrival of new requests as an event. At any given moment, previously assigned requests which have not yet been served are also re-planned when reoptimizing the routes. In contrast, the tasks of a request can be fixed in their routes or in their positions under certain conditions. A *fixed in position* task is a task which has already been served or has an AGV currently driving towards it. In this case the decision can no longer be changed, and any insertion operation must be performed after the last fixed in position task of a given route. A *fixed in route* task is a task whose corresponding request is partially served or at least has an AGV currently driving towards its first task. This means that the task is fixed in its AGV route but other tasks may be inserted before it. Therefore, for the tasks of a given request, we distinguish three cases: (i) all the tasks of a request which has already been served are fixed in their positions, (ii) all the tasks of a request that has an AGV currently driving towards its first task are fixed in their routes and the first task is fixed in its position, (iii) when a request is partially served, the served tasks are fixed in their positions, and the non-served tasks are fixed in their routes.

ALNS is executed at line 5 in order to reoptimize the routes and obtain the number of required AGVs. Then, the mathematical model is solved to optimize the number of carts in the solution (line 6). Note that solving the model in line 6 implies updating the start time of service at non-fixed nodes. This dynamic procedure is repeated until there are no new requests to serve. For practical reasons, it is crucial to have the AGV routes into effect as soon as possible, hence, it is critical that the reoptimization is done efficiently and in a reasonable amount of time.

Algorithm 1 Dynamic Optimization Algorithm

```
1: Create a solution $S$ that contains the initial requests
2: Do
3:     Add the new requests to $S$
4:     Fix tasks in routes/positions
5:     $Execute\_ALNS(S)$
6:     Solve $Cart\_Model(S)$ to minimize the number of carts
7: While there are new events
```

This algorithm is designed to be executed in real-time, uninterruptedly. Each new request known at release time $\mathcal{R}$ may need to be served as soon as in a few minutes (depending on how tight the opening and closing of the time window of its first task is). Later in Section 6, we explain the large real-life instances we obtained for a 24 h period, and how we assess the performance of this framework with respect to its sensitivity to different parameters. In what follows, the main features of the ALNS framework that is used to optimize the routes and number of AGVs are described in detail in Section 5.1. The mathematical formulation used to optimize the number of carts is presented in Section 5.2.

5.1. ALNS framework to optimize the routes and number of AGVs

The ALNS is an extension of the Large Neighborhood Search proposed by Shaw (1998). In the ALNS framework presented in Ropke and Pisinger (2006) and Pisinger and Ropke (2007) a number of heuristics compete to modify the current solution. At each iteration, a removal operator is chosen to partially destroy the solution, and an insertion operator is chosen to repair it. Throughout the execution of ALNS, the removal and insertion operators are selected dynamically according to their past performance. Each operator is given a score which increases if it improves the current solution. A simulated annealing criterion is used to determine whether a new solution is accepted. A detailed description of ALNS can be found in Ropke and Pisinger (2006) and Pisinger and Ropke (2007). Algorithm 2 provides an overview of the ALNS framework.

A roulette wheel selection mechanism is used to randomly select the request removal and insertion operators ro and io based on the scores of each operator. A successful operator is the one with a high score, therefore, it should be chosen with a larger probability. The score of an operator is incremented with the following values depending on the new solution S':

- θ_1 if S' is a new best solution;
- θ_2 if S' is better than the current solution;
- θ_3 if S' is not better than the current solution but still accepted;

with $\theta_1 > \theta_2 > \theta_3$.

The measure of how well a given operator i has performed in the past is given by ω_i. An operator j is selected with a probability $\omega_j / \sum_{i=1}^{v} \omega_i$, such that v is the number of operators and ω_i is the weight of operator i.

Weights and scores of the operators are updated at the end of each segment which is defined as φ number of iterations. The weights are updated as follows:

$$\omega_i := \begin{cases} \omega_i & \text{if } o_{ij} = 0 \\ (1-\eta)\omega_i + \eta\pi_i / o_{ij} & \text{if } o_{ij} \neq 0, \end{cases} \tag{12}$$

where π_i is the accumulated score of operator i and o_{ij} is the number of times operator i has been used in the last segment j. The reaction factor $\eta \in [0,1]$ controls how quickly the weight adjustment algorithm reacts to changes in the scores. Note that at the end of each segment all the scores are reset to zero.

A simulated annealing metaheuristic approach is applied as a local search framework, thus, given a current solution S, a new solution S' is accepted with a probability $e^{-(f(S')-f(S))/T}$. The temperature T decreases at each iteration according to the expression $T = T * c$ where c is a standard exponential cooling rate. Increasing the diversification of the algorithm is possible by allowing worse solutions to be easily accepted. A procedure is applied to guarantee more diversification through allowing the temperature to reset to its initial value when it reaches a minimum threshold T_{min}.

Algorithm 2 ALNS framework for the FSRPS-AGV

```
Require: $T_{init}$: initial temperature; $c$: cooling rate
Require: $S$: initial solution
Require: $RO$: set of removal operators with score null
Require: $IO$: set of insertion operators with score null
1: $S_{best} \leftarrow S$
2: $T \leftarrow T_{init}$
3: while stop criterion not met do
4:     $S' \leftarrow S$
5:     $q \leftarrow$ Generate a random number of requests to remove
6:     $ro \leftarrow$ Select an operator from $RO$ (Section 5.1.1) using a roulette wheel based on the weight of the operators
7:     $io \leftarrow$ Select an operator from $IO$ (Section 5.1.2) using a roulette wheel based on the weight of the operators
8:     Remove $q$ requests from $S'$ by applying $ro$
9:     Insert removed requests into $S'$ by applying $io$
10:    if $f(S') < f(S_{best})$ then
11:        $S_{best} \leftarrow S \leftarrow S'$
12:        Increase the scores of the $ro$ and $io$ by $\theta_1$
13:    else if $f(S') < f(S)$ then
14:        $S \leftarrow S'$
15:        Increase the scores of the $ro$ and $io$ by $\theta_2$
16:    else if $acccept(S', S)$ then
17:        $S \leftarrow S'$
18:        Increase the scores of the $ro$ and $io$ by $\theta_3$
19:    end if
20:    $T \leftarrow T * c$
21:    if $T < T_{min}$ then
22:        $T \leftarrow T_{init}$
23:    end if
24:    if end of a segment of $\varphi$ iterations then
25:        Update weights and reset scores of operators
26:    end if
27: end while
```

5.1.1. Removal operators

Removal operators are used to partially destroy a solution by removing some requests from the routes of some AGVs. We use four removal operators from the literature, namely, random removal, related removal, time-oriented removal, and worst removal, removing q requests between $[q_{min}, q_{max}]$ at each iteration, drawn uniformly. The reader is referred to Ropke and Pisinger (2006) for a detailed description of removal operators.

1. **Random removal**: this operator randomly selects q requests and removes them from the solution.
2. **Related removal**: the general idea of this operator is to remove requests that are similar to each other. We start by randomly selecting a seed request and then its $q-1$ most related ones are removed from the solution. The relatedness of two requests r_1 and r_2 is measured by the distance function $g(r_1, r_2) = \frac{1}{B}\sum_{i\in r_1}\sum_{j\in r_2} d(i,j)$, where $d(i,j)$ is the distance between two nodes i and j, and B is the number of nonzero terms in the equation as described in Ropke and Pisinger (2006).
3. **Time-oriented removal**: the time-oriented removal is a variant of the related removal, where $g_t(r_1, r_2) = |S_{i_1} - S_{i_2}|$, where S_{i_1} and S_{i_2} are the start times of service of the first tasks of requests r_1 and r_2, respectively. The intuition behind this formula is to remove requests that are served roughly at the same time in the hope that they are easy to interchange.
4. **Worst removal**: this operator removes requests that are expensive in the current solution. The main idea is to remove q requests with high cost, allowing the algorithm to try and insert them at another place in the solution to obtain a better solution value.

5.1.2. Insertion operators

Insertion operators are used to rebuild a solution after some requests have been removed from it. They require evaluating the positions to insert the nodes of a given request. Synchronization requirements with respect to the temporal aspect must be taken into consideration here. Two MIP insertion operators are proposed to handle synchronization.

1. **MIP insertion**: For each removed request $r \in R$, the MIP below determines the best position (or positions) to insert the nodes of this request. Prior to solving the MIP, we enumerate all possible insertion configurations of request r. A configuration can be a single node of request r, a sequence of part of the nodes of r, or a sequence of all the nodes of r, where the sequences preserve the precedence order of the nodes. The number of configurations of a request r composed of k nodes is $k(k+1)/2$. For example: given a request r composed of three nodes $i_1, i_2, i_3 \in r$, then the number of configurations is 6. The insertion configurations of this request are then: i_1, i_1–i_2, i_1–i_2–i_3, i_2, i_2–i_3, i_3. After generating the configurations, based on the current AGV routes in the solution, we generate the set of feasible insertion moves and calculate their corresponding insertion costs. An insertion move is defined as inserting the node(s) corresponding to a configuration into an insertion position (between two nodes) in the current solution. Note that the insertion cost of a given insertion move between two nodes in the solution is calculated by subtracting the routing cost between the two nodes from the routing cost between the same two nodes but after adding the nodes of the configuration corresponding to the insertion move. The set of feasible insertion moves of all configurations is then used to generate the variables m_j^l described below. Note that intra-route constraints must be respected when generating the set of feasible moves, therefore, infeasible moves with respect to intra-route constraints (such as schedule feasibility constraints) are omitted which reduces the size of the problem.
   We define the set I of already inserted nodes in the solution, the set I_R of already inserted requests in the solution, and the set L_r of configurations of request r. The cost of inserting configuration l after node j is denoted c_j^l. We define the parameter e_i^l equal to 1 if node $i \in r$ is in configuration l, 0 otherwise. The problem can be mathematically formulated with variables: m_j^l equal to 1 if configuration l is inserted after node j, 0 otherwise; and S_i indicating the start time of service at node $i \in V$. The MIP can then be formulated as follows:

$$\min \sum_{l\in L_r}\sum_{j\in I\setminus\{n+1\}} c_j^l m_j^l \tag{13}$$

$$\text{s.t.} \sum_{l\in L_r}\sum_{j\in I} e_i^l m_j^l = 1 \qquad i \in r \tag{14}$$

$$S_i \ge S_j + s_j + t_{ji} - M\left(1 - \sum_{l\in L_r} e_i^l m_j^l\right) \qquad i \in r, j \in I \tag{15}$$

$$S_k \ge S_i + s_i + t_{ik} - M\left(1 - \sum_{l\in L_r} e_i^l m_j^l\right) \qquad j,k \in I, k \succ j, i \in r \tag{16}$$

$$S_k \ge S_j + s_j + t_{jk} \qquad j,k \in I, k \succ j \tag{17}$$

$$S_i \ge S_j + s_j + t_{ji} \qquad i,j \in r, i \succ j \tag{18}$$

$$S_k \ge S_j + s_j + t_{jk} \qquad j,k \in r', r' \in I_R, k \succ j \tag{19}$$

$$a_i \le S_i \le b_i \qquad i \in V \tag{20}$$

$$m_j^l \in \{0,1\} \qquad j \in I, l \in L_r \tag{21}$$

$$S_i \ge 0 \qquad i \in V. \tag{22}$$

The objective function (13) minimizes the overall insertion cost of request r. Constraints (14) ensure that only one configuration is used to insert each node $i \in r$. Constraints (15)–(20) guarantee schedule feasibility with respect to time windows. Constraints (18) and (19) also handle synchronization requirements with regard to temporal aspect among the nodes of the same request, and they preserve the precedence order of these nodes. Constraints (21) and (22) impose the nature and the domain of the variables. M is a big enough number and can be set to the maximum return time to the depot b_{n+1}. This MIP is solved to determine the best insertion position (or positions) and the best insertion configuration (or configurations) to insert the nodes of request r.

2. **MIP insertion with noise**: This operator extends the previous one by adding a noise term to the insertion cost (Ropke and Pisinger, 2006). Each time we calculate a configuration's insertion cost, we add some noise and calculate a modified insertion cost. This adds diversity to the search and helps avoid being trapped in local optima.

5.2. Mathematical formulation to minimize the number of carts

In this section, we present the mathematical programming method used to optimize the number of carts given a set of previously defined AGV routes. This model optimizes the number of carts needed to serve the requests already assigned to the routes. This is achieved by optimizing departure and starting service times, by better exploiting the available fleet, and by reducing the number of (expensive) carts needed to operate the AGV routing solution. We define $U = \{(i,j) \in V \times V \mid i$ and j belong to the same route in the input solution and $j \succ i\}$ as the set of pairs of consecutive nodes belonging to the same route.

A new graph $\bar{G} = (V, \bar{A})$ is defined, where the set of arcs is $\bar{A} \subseteq A_c$ minus omitted arcs. In addition to the omitted arcs from A_c presented in Section 4 which apply to $\bar{A}$, we also omit arc (i,j) if: (i) $(j,i) \in U$, (ii) $i \in r$ and $j \in r'$ with $r \neq r'$, i is the last node of r and j is the first node of r', and $S_i + s_i + t_{ij} - S_j > slack_j$ where $slack_j$ is the slack time at node j. Let $\bar{A}^+(i)$ be the set of nodes j such that there is an arc from i to j, that is, $\bar{A}^+(i) = \{j \in V | (i,j) \in \bar{A}\}$. Similarly, $\bar{A}^-(i) = \{j \in V | (j,i) \in \bar{A}\}$.

The main idea of this model is to conserve the existing routes and change departure and start times of service of AGVs at the nodes in order to further reduce the number of carts used in the system. As stated in Section 5.1, the objective function of ALNS minimizes the total routing cost of AGVs, thus, the start time of service of AGVs at each node is naturally set to its earliest value. It is then possible to manipulate the start time of service in order to optimize the number of carts.

The problem can be mathematically formulated with the following decision variables: y_{ij} equal to 1 if arc (i,j) is traversed by a cart, 0 otherwise; and S_i indicating the start time of service at node $i \in V$. The model can then be formulated as follows:

$$\min \sum_{j \in N} y_{0j} \tag{23}$$

$$\text{s.t.} \sum_{j \in \bar{A}^+(i)} y_{ij} = 1 \qquad i \in N \tag{24}$$

$$\sum_{j \in \bar{A}^-(i)} y_{ji} = 1 \qquad i \in N \tag{25}$$

$$S_j \geq S_i + s_i + t_{ij} - M\left(1 - y_{ij}\right) \qquad (i,j) \in \bar{A}, (r(i), r(j)) \in W \tag{26}$$

$$S_j \geq S_i + s_i + t_{ij} \qquad i,j \in r, r \in R, j \succ i \tag{27}$$

$$S_j \geq S_i + s_i + t_{ij} \qquad (i,j) \in U \tag{28}$$

$$a_i \leq S_i \leq b_i \qquad i \in V \tag{29}$$

$$y_{ij} \in \{0,1\} \qquad (i,j) \in \bar{A} \tag{30}$$

$$S_i \geq 0 \qquad i \in V. \tag{31}$$

The objective function (23) minimizes the number of carts departing from the start depot. Constraints (24) and (25) are degree constraints requiring each node to be visited exactly once. Constraints (26)–(29) guarantee schedule feasibility with respect to time windows. Note that constraints (27) also handle synchronization requirements with regard to temporal aspect among the nodes of the same request for all requests, and they preserve the precedence order of these nodes. Constraints (30) and (31) impose the nature and the domain of the variables. Again, M is a big number and can be set to the maximum return time to the depot b_{n+1}.

6. Computational experiments

This section describes the computational experiments. They have been performed on a computer equipped with a 2.1 GHz Intel processor. All of the proposed algorithms are coded in C++ and CPLEX 12.10 is used with default parameters. Each test was allowed to use up to 8 GB of RAM. Section 6.1 provides a detailed description of the real dataset used and introduces the characteristics of the problem instances. Parameter setting is discussed in Section 6.2. The results of detailed and extensive computational experiments are then presented in Section 6.3.

Table 1
Statistics relative to the number of tasks per request classification.

	# tasks/req			
	2	3	4	5
# Requests	636	265	255	56
Avg T/req (min)	151.7	336.5	277.0	218.6

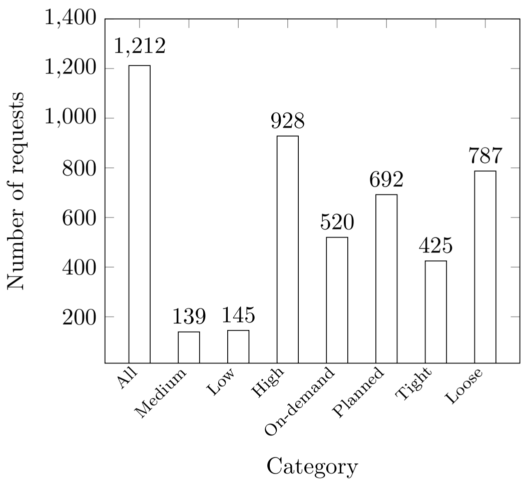

Fig. 3. Number of requests per category.

6.1. *Test instances*

The dataset provided by our partner consists of all potential transportation requests that may be needed in a given day for all the departments in the hospital. The dataset was built by project leaders in collaboration with all the 10 departments at the hospital including: pharmacy, maintenance, food preparation, laboratory, and others. Each department specified their needs in terms of transportation requests which are then added to the dataset. The dataset contains 1212 requests having a total of 3367 tasks. The number of tasks per request in the dataset ranges between two and five. Table 1 presents a classification of the requests based on the number of tasks per request. For each class (# tasks/req), we report the number of requests and the average of the durations of time needed to perform a request where each of these durations is measured as follows: $b_{last\_task} - a_{first\_task}$, where b_{last_task} is the TW upper bound of the last task and a_{first_task} is the TW lower bound of the first task for each request.

Each request in the dataset is defined by a set of characteristics including: demand type, weekly frequency of occurrence, TW intervals, and cart type. The demand type of a request can be *Planned* indicating that the request is known at the start of the day, or *On-demand* indicating that the request is unknown but it may or may not occur throughout the day. The release time of an *On-demand* request in the dataset is set to 5 min before the TW lower bound of its first task. In terms of frequency of occurrence, we have: *High* frequency requests occurring four times a week, *Medium* frequency requests occurring three times a week, and *Low* requests occurring twice a week. In terms of TW intervals, on a typical day, we distinguish two types of requests: requests with *tight TW* whose tasks are characterized by urgency in execution, meaning that the intervals of the TWs are tighter than the ones of the second type of requests known as requests with *loose TW*. In the dataset, a *tight TW* is given 5 min interval, while a *loose TW* is given 30 min interval. Finally, the 10 types of carts that are used to serve requests are: linen carts, flatbed carts with rails, flatbed carts, return tray carts, instruments carts, food carts, cage carts, pharmacy carts, type B carts (insulated, half-sized, lockable), and waste carts. Hence, the cart type used to perform each request is also specified in the dataset. Fig. 3 depicts the number of requests per category based on the different characteristics of the requests in the dataset.

Fig. 4 depicts the scattering of the tasks in the dataset throughout the day. It shows that most of the tasks must be performed in day time. It also shows that there are some peak activity moments during the day where many tasks must be performed at the same time.

For computational experiments, we consider a set of 40 real-size instances, where the number of requests varies between 586 and 658. Each instance is generated by executing several successive selections based on the frequency of occurrence of each request. A random value ρ between 0 and 1 is generated, and if $\rho \leq \frac{Frequency\_of\_occurrence}{7}$, then the current request is added to the instance, otherwise we pass to the next request. Our instances have an average degree of dynamism of 42% which is calculated by dividing the sum of the values of degree of dynamism for all instances by the number of instances.

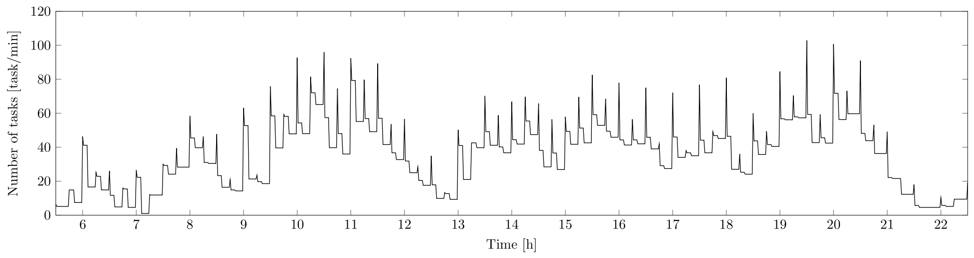

Fig. 4. Number of tasks over time during the day.

Table 2
Quality of the solution when changing the number of ALNS iterations.

# iter.	# AGV		# Carts		Travel t. (min)		T/E (s)
	Best	Avg	Best	Avg	Best	Avg	
100	757	757.0	8009	8009.0	32867.50	32867.50	33.5
200	749	749.0	8003	8003.0	32721.60	32721.60	61.2
300	754	754.0	8003	8003.0	32515.60	32515.60	92.2
400	748	748.0	8001	8007.2	32413.20	32474.76	117.0
500	748	753.6	7997	7998.4	32225.90	32391.86	133.5

6.2. Parameter setting

In our ALNS metaheuristic, the minimum number of requests to remove is set to $q_{min} = min(10, 0.1n)$, and the maximum is $q_{max} = min(20, 0.4n)$ where n is the number of assigned requests. The number of iterations in each segment is set to 100, and the cooling rate is set to 0.8. The remaining parameters are set to the values reported in Ropke and Pisinger (2006). In a real environment, it is critical that the reoptimization is done efficiently. Therefore, the maximum runtime of the algorithm at each reoptimization is set to 5 min. The fixed costs β and α_j are set to the cost (in dollars) of acquiring one AGV (β) and one cart that is used to serve a given node $j \in N$ (α_j). The routing cost c_{ij} associated with each arc $(i,j) \in A$ is assumed to be the travel time between i and j in minutes (t_{ij}), we also assume that one unit routing cost is equivalent to one unit variable cost (one Canadian dollar in general) which includes: depreciation cost, cart handling cost, as well as AGV charging cost, per one unit routing cost (one minute). Note that the same assumption applies to insertion cost in Section 5.1.2. According to our industrial partner, the speed of the AGVs is constant at 1.8 m/s, in general. Thus, it is possible to obtain the theoretical travel times between all the points. To ensure a safety margin, we reduce the speed to 1.5 m/s which allows to somewhat overestimate the travel times.

A preliminary testing of our metaheuristic using a set of 25 real-size instances is conducted in order to choose the number of ALNS iterations allowed to run before stopping at each reoptimization. The initial value was set to 100 iterations, and we have tested running it for 200, 300, 400, and 500 iterations, obtaining the solutions reported in Table 2. The algorithm is executed five times for each parameter configuration per instance. In Table 2, for each configuration (defined by the number of iterations *# iter.*), we report the sum of the values corresponding to the best solutions ($Best$) and the sum of the average values (Avg) of: the number of AGVs, the number of carts, and the travel time. Last column reports the average runtime per event (T/E) in seconds. Note that the value of travel time excludes the travel time between the pickup and delivery locations of each task which is added to the service time of the task. Based on the results in Table 2, we observe that the 500 iterations is the best overall, therefore, it was selected to perform the computational experiments. One can observe that T/E corresponding to 500 iterations is 133.5 s which is somewhat far from the maximum runtime per event which is set to 300 s. The runtime per event may exceed 200 s in some cases such as at the start of the day due to a large number of planned requests. Our tests showed that the runtime needed to perform 500 iterations at the start of the day exceeds 200 s in 10 out of the 25 instances. This may also happen during peak activity moments.

6.3. Computational results

We start our analysis by running our algorithm five times per instance. We report the best as well as the average solutions. The computational results are reported as follows.

6.3.1. Comparison with the exact model

In this section, we present a comparison between our matheuristic and the exact model (1)–(11). Four sets of ten small size instances, each with 100, 200, 300, and 400 requests, are used to perform the tests. We first compare the results of the dynamic problem: the exact model in this case solves a series of static problems by reoptimizing the routes upon the arrival of new requests. As in the case of the matheuristic algorithm, in the exact algorithm, the maximum runtime per event is also set to five minutes.

Table 3 presents a comparison between the exact algorithm and our matheuristic using small size instances. The results are summarized based on the number of requests per instance. In the first two columns, we present the number of requests and the

Table 3
Comparison between the exact algorithm and the matheuristic for the dynamic problem.

# Req.	# Eve.	Exact algorithm					Matheuristic							
		# Com.	# AGV	# Carts	Travel t. (min)	T/E (s)	# AGV		# Carts		Travel t. (min)		Gap (%)	T/E (s)
							Best	Avg	Best	Avg	Best	Avg		
100	28.3	9	64	550	2092.3	14.3	64	65	552	554.2	2223.3	2247.9	1.12	22.8
200	38.9	4	111	1057	3775.1	67.3	111	112.4	1062	1063.6	4132.6	4175.2	1.67	44.7
300	41.9	1	157	1560	5656.5	169.2	158	159.2	1566	1572.6	6297.6	6346.9	2.02	66.5
400	48.7	0	214	2042	7347.9	223.9	203	203.6	2050	2056.6	8318.6	8380.0	−0.94	90.7

average number of events per instance category per day (24 h). In the case of the exact algorithm, for each instance category, we report the following: the number of completed instances (# Com.) indicating the number of instances where all events are solved to optimality, the sum of the number of AGVs (# AGV), the sum of the number of carts (# Carts), the sum of travel times (Travel t.) in minutes, and the average runtime per event (T/E) in seconds. In the case of the matheuristic, for each instance category, we report the sum of the $Best$ and the sum of the average (Avg) values of: the number of AGVs, number of carts, and travel times. We also report the average gap with respect to the objective function of the final solution for all instances per instance category (Gap), and the average of the average runtime per event (T/E). The gap is calculated as $100 * \left(\frac{BestMath - BestEx}{BestMath}\right)$, where *BestMath* is the value of the objective function of the best final solution obtained by the matheuristic, and *BestEx* is the best lower bound on the objective function of the final solution obtained by the exact algorithm.

Table 3 shows that overall, our matheuristic achieves high-quality solutions when compared to the exact algorithm, especially in terms of the number of AGVs and carts, while average travel times corresponding to the best solutions obtained by our matheuristic is higher than average travel time of the exact algorithm for all instance categories. For instances with 400 requests, the exact method finds 214 AGVs, whereas, the matheuristic finds 203.6 on average. This can be justified by the fact that the maximum runtime per event is set to 5 min, hence, in some events, the exact method cannot find the optimal solution within 5 min and it ends up returning the best integer solution which affects the quality of the solution provided by the exact method. In addition, the average runtime per event increases significantly when the size of the problems solved by the exact algorithm is increased, while in the case of the matheuristic, the average runtime per event increases but less significantly than in the exact method. Based on the previous discussion, one may conclude that our matheuristic is better positioned to solve real-size instances. The average gap for instances with 400 requests (negative) reinforces the last conclusion. Despite the travel times being larger, they all respect the constraints of the problem and can be justified by the effort to have fewer AGVs, which was the main cost driver.

We also compare the results of the static problem where we assume that all input information is known beforehand and solve a static problem. The exact model is given a time limit of two hours. Heuristically, the static problem is solved in three steps: in step 1, we use a restricted ALNS to find the minimum number of AGVs ensuring that the tasks of each request must be served by the same AGV, i.e., ignoring potential synchronization opportunities. Different insertion operators are used in this case, namely regret-2 and regret-k with and without noise where k is the number of AGVs. The restricted ALNS is executed at the beginning using an initial number of available AGVs, then, we repeatedly remove one AGV and run ALNS again while this procedure remains feasible. If it fails to find a feasible solution with all requests served, we stop and the number of AGVs found in the last feasible solution is the minimum number of AGVs of the restricted problem. The number of iterations used each time an AGV is removed in step 1 is 500 iterations. In step 2, we run the restricted ALNS again but with 15 000 of iterations in order to optimize the travel time. We then run a non-restricted ALNS (similar to the one described in Section 5.1) starting with the best solution obtained so far to further reduce the number of AGVs by removing one AGV and trying to assign its requests somewhere else using a low number of iterations (50 iterations). This process is repeated 10 times. We then re-run the non-restricted ALNS for 1000 iterations to further optimize the travel time. Finally, in step 3, we use the best solution found in step 2 to solve the mathematical model presented in Section 5.2 in order to optimize the number of carts.

Table 4 presents a comparison between the exact algorithm and the matheuristic using small size instances. The information reported in this table is similar to Table 3 with the following minor differences. For the exact algorithm, we report the number of instances solved to optimality (Sol.), and the average total runtime (Time) is reported instead of the average of the average runtime per event. For the matheuristic, the average total runtime (Time) is reported in the last column. We also report the average optimality gap with respect to the objective function per category named *Gap*, whereas, *Gap sol.* (*Gap not.*) reports the same metric but only for the instances solved (not solved) to optimality by the exact algorithm. The optimality gap is calculated as before. Note that for the instances not solved to optimality by the exact algorithm, the reported values (number of AGVs, carts, and travel time) correspond to the best integer solution found when the time limit is reached, however, the best lower bound obtained at the end of the execution time is used to calculate the optimality gap.

Table 4 shows that the exact algorithm solves to optimality 12 out of 40 small instances. One can also observe that our matheuristic provides near-optimal solutions, the average optimality gap with respect to the objective function for the instances solved to optimality by the static algorithm varies between 0.87% and 1.39% on average. The same number of AGVs is found by our matheuristic for instances with 100 requests, whereas, for the remaining instance categories, the matheuristic finds lower number of AGVs. In terms of number of carts and average travel time, the matheuristic finds slightly larger values due to the fact that fewer AGVs are available, which makes the overall objective better. In addition to providing near-optimal solutions, our matheuristic is

Table 4
Comparison between the exact algorithm and the matheuristic for the static problem.

# Req.	Exact algorithm					Matheuristic									
	Sol.	# AGV	# Carts	Travel t. (min)	Time (s)	# AGV		# Carts		Travel t. (min)		Gap (%)	Gap sol. (%)	Gap not. (%)	Time (s)
						Best	Avg	Best	Avg	Best	Avg				
100	8	64	550	1997.4	1510.6	64	64	551	555.4	2111.8	2125.0	0.96	0.87	1.29	154.5
200	3	112	1057	3670.8	5416.9	111	111	1062	1063.0	3999.7	4057.9	1.61	1.31	1.74	348.6
300	1	159	1560	5459.2	6940.1	157	157	1568	1573.0	5995.9	6069.1	0.99	1.39	0.95	546.2
400	0	228	2043	7308.8	7191.9	203	203	2051	2054.6	7895.0	8006.6	4.15	–	4.15	741.9

Table 5
Results of the dynamic optimization using real-size instances.

# req.	# Eve.	# AGV		# Carts		Travel t. (min)		T/E (s)
		Best	Avg	Best	Avg	Best	Avg	
[526, 629]	53.4	582	586.0	6218	6219.8	25230.6	25359.3	132.8
[629, 658]	54.2	604	609.0	6501	6501.6	26338.0	26496.9	138.1

also fast: the average total runtime is between 154.5 to 741.9 s for all instance categories, which is one order of magnitude faster than the exact algorithm.

One can also highlight the impact of dynamism on small problems by comparing the results of the dynamic and static problems. The results of the matheuristic in Tables 3 and 4 show that the impact of dynamism on the number of AGVs and carts is limited, whereas, it is more significant when it comes to travel time which is justified by the holistic character of the static problem which assumes the knowledge of all input information at the start of the time horizon. On the other hand, the results of the exact algorithm show that the difference in the number of AGVs between the dynamic and static problems increases with the size of the instances. This is mainly due to an increase in the complexity for larger instances which yields difficult problems to solve exactly, hence, achieving optimal/high-quality solutions when solving the static problem is computationally difficult; whereas, in the dynamic problem, a small and easier static problem is solved upon the arrival of new requests, hence, achieving optimal/high-quality solutions of the sub-problems is easier. This can also be observed when comparing the number of completed instances in the dynamic problem to the number of instances solved to optimality in the static problem: for instances with 100 and 200 requests, the number of completed instances is 9 and 4, versus 8 and 3 instances solved to optimality in the static version.

Overall, the previous discussion aims to highlight the fact that the future requests being not known does not make small instances require more AGVs and carts. In fact, this is due to the very tight time windows; even with the full knowledge of future requests, the algorithm cannot find a better resource utilization in order to decrease the fleet size. It is also due to the size of the problems solved in this case as an increase in the operational difficulty caused by an increase in the size of the problem (real-size instances) may require more AGVs and carts to handle the dynamic setting.

6.3.2. Results of the dynamic optimization using real-size instances

We now present the results of the matheuristic algorithm presented in Section 5 using the 40 real-size instances. For simplicity, the instances are divided into two subsets whose number of requests is between [526,629] and [629,658]. Table 5 reports the detailed results. For each subset, in the column # Eve. we report the average number of events per day (24 h). Then we report the sum of the values corresponding to the best solutions ($Best$) and the sum of the average values (Avg) of: the number of AGVs, the number of carts, and the travel time. The last column reports the average runtime per event (T/E) in seconds.

The results in Table 5 show that our matheuristic is very fast and robust in solving large instances of the FSRPS-AGV, as the total average runtime per event is 135.4 s and the sum of the best values are very close to the sum of the average values. An increase of all the values is observed when comparing the results of the subsets 1 and 2, mainly due to a higher number of requests in the instances of subset 2.

Fig. 5 depicts the number of AGVs and carts in use per minute over time during the day for a real-size instance. The figure shows that the vast majority of activities are performed during day time between 5 am and 20 pm. Several peaks of activity occur during the day such as the peaks related to food service (breakfast, lunch, and dinner). One can also highlight that the use of AGVs and carts is different. AGVs are used briefly and very often, which explains the peaks, whereas, the carts are used throughout the time window of a request, which explains the relative absence of recurring peaks.

6.3.3. Sensitivity analysis

In this section, we conduct sensitivity analysis by varying the TW intervals, AGV fleet exploitation strategy, and AGV speed. We use the 40 real-size instances described in Section 6.1 by changing their parameters to assess how the parameters affect the peak usage (required number) of AGVs and carts.

In terms of TWs, we test different values of TW intervals for the Tight and the Loose TW requests. Nine configurations are tested: 3–15, 3–30, 3–45, 5–15, 5–30, 5–45, 10–15, 10–30, 10–45, where the first number is the TW interval of Tight TW requests and the second one is the TW interval of Loose TW requests. Table 6 and Fig. 6 show the sensitivity analysis with respect to TWs. Table 6

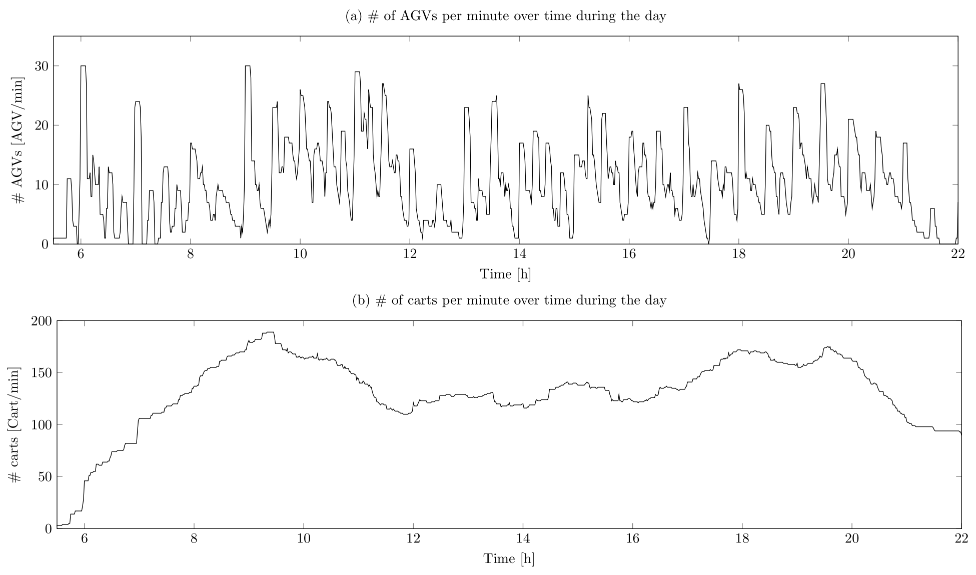

Fig. 5. Number of AGVs and carts in use per minute over time during the day for a real-size instance.

reports a summary of the results for all TW configurations in the dynamic setting. The average number of Tight TW/Loose TW requests for the real-size instances is shown in parenthesis. In general, this analysis shows that larger TWs result in an important and significant decrease of the number of AGVs and carts. In a managerial perspective, this shows the importance of the TW factor in lowering the cost of investment while taking into consideration the necessity of ensuring a good service level. By looking deeper into the results, one can observe that the change in TW intervals corresponding to Tight TW requests has a significant impact in decreasing the sum of the number of AGVs in all the scenarios when varying the Loose TW. For example, varying the Tight TW from 3 to 10 min while keeping Loose TW constant at 15 min leads the sum of the number of AGVs to decrease from 1267 to 908. Moreover, changing the Loose TW while keeping Tight TW constant at 10 min yields a significant decrease of the sum of the number of AGVs and carts. It is also important to highlight that fewer AGVs may increase the number of carts in some scenarios such as the case when Tight TW changes from 5 to 10 min at Loose TW set to 15 min: the sum of the number of AGVs decreases from 1193 to 908 while the sum of the number of carts increases from 12 703 to 13 019. Fig. 6 shows that larger TWs help reducing the peaks of activity during the day. Table 7 reports a summary of the results for all TW configurations in the static setting. The results show that the same remarks for the dynamic case apply to the static case, meaning that larger TWs yield a considerable reduction of the number of AGVs and carts in the system. One can also highlight the impact of dynamism on real-size problems. The results in Tables 6 and 7 show that in general, more resources must be allocated in the dynamic setting due to a higher operational difficulty of managing a dynamic system in real-life which extends the insights from Section 6.3.1 where only small problems are solved. However, the impact of dynamism is still somewhat limited when the TWs are tight; this is in contrast to its impact when the TWs become larger where the difference in the number of AGVs is more important between the dynamic and the static settings.

Tables 8 reports the results relative to the AGV fleet exploitation strategy, namely if only one AGV satisfies a full request (1-AGV) or if many AGVs are synchronized to serve the tasks of a request (M-AGV). In the sensitivity analysis with respect to AGV fleet exploitation strategy, we compare the matheuristic which is described in Section 5 against a restricted matheuristic. The restricted matheuristic follows the same logic of the matheuristic in Section 5 as described in Algorithm 1, the only difference is in the ALNS part where the tasks of each request are served by the same AGV, which means that synchronization opportunities are not explored. Furthermore, classical insertion operators are used in the ALNS part of the restricted matheuristic. Note that the ALNS algorithm of the restricted matheuristic is similar to the ALNS that is used in step 1 of the algorithm designed to solve the static problem which is described in Section 6.3.1. This analysis clearly shows that the solution achieved by M-AGV is overall significantly better than that procured by 1-AGV when comparing the *Best* of the sum of number of AGVs and travel time. In terms of *Best* sum of number of AGVs, M-AGV reports 1186 versus 1200 AGVs for 1-AGV, and in terms of *Best* sum of travel time, M-AGV reports 51568.6 versus 51796.4 min for 1-AGV. However, 1-AGV is slightly better than M-AGV in terms of number of carts: this can be explained by the fact that using more AGVs increases the possibility of exchanging carts among the routes which then leads to decreasing the number of carts. From an investment cost perspective, and given the fact that the price of one AGV is almost seven times that of the most expensive cart (according to our industrial partner), one can conclude that M-AGV strategy is overall more advantageous as it has a positive net effect on investment cost. It also confirms the added value of synchronization.

In terms of AGV speed, we test four different values starting from 1.2, 1.5, 1.8, and 2.1 m/s (meters per second). Table 9 shows that a significant decrease in the number of AGVs and carts can be achieved by increasing the speed of AGVs. Therefore, from a managerial perspective, the speed of AGVs in the system can be another important factor to take into consideration while trying to reduce the cost of investment. However, the flexibility with respect to the AGV speed in this context is somewhat limited due

Table 6
Sensitivity analysis on TWs (dynamic problem).

		Loose (392.3)					
		15 min		30 min		45 min	
		# AGV	# Carts	# AGV	# Carts	# AGV	# Carts
Tight (237.3)	3 min	1267	13024	1266	13032	1266	13015
	5 min	1193	12703	1186	12719	1183	12700
	10 min	908	13019	754	12716	739	12183

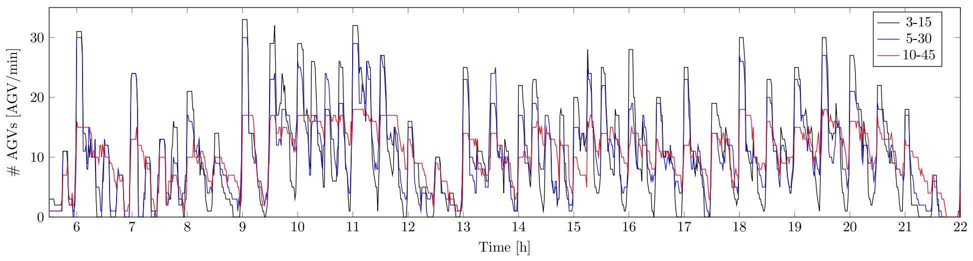

Fig. 6. Number of AGVs in use per minute over time during the day in the case of different TWs.

Table 7
Sensitivity analysis on TWs (static problem).

		Loose (392.3)					
		15 min		30 min		45 min	
		# AGV	# Carts	# AGV	# Carts	# AGV	# Carts
Tight (237.3)	3 min	1262	13000	1262	13008	1262	13004
	5 min	1179	12687	1179	12675	1179	12684
	10 min	875	12986	700	12681	673	12164

Table 8
Sensitivity analysis on AGV fleet exploitation strategy.

Strategy	# AGV		# Carts		Travel t. (min)		T/E (s)
	Best	Avg	Best	Avg	Best	Avg	
1-AGV	1200	1218.4	12675	12691.4	51796.4	51745.4	125.1
M-AGV	1186	1195.0	12719	12718.4	51568.6	51856.2	135.4

Table 9
Sensitivity analysis on AGV speed.

Speed (m/s)	# AGV		# Carts		Travel t. (min)		T/E (s)
	Best	Avg	Best	Avg	Best	Avg	
1.2	1263	1271.6	12728	12744.4	64251.4	64675.0	138.4
1.5	1186	1195.0	12719	12718.4	51568.6	51856.2	135.4
1.8	1112	1124.6	12697	12709.4	29431.4	29514.8	143.9
2.1	1063	1064.4	12684	12705.2	24482.9	24682.4	143.0

to safety concerns that may be raised, especially in the case where AGVs share transportation infrastructure (corridors, roads) with humans.

6.3.4. Managerial insights

In this section, we discuss managerial insights with respect to the number of AGVs and carts that should be acquired by our industrial partner, based on the results using the parameters of Section 6.3.2 (default scenario). The number of AGVs and carts to acquire depends on two main factors: the budget allocated to acquire them and the risk tolerance to a potential violation of TWs. If the number of AGVs and carts is not sufficient, some tasks risk being served a few minutes later. Our analysis indicates that this delay should not be high considering that enlarging the TWs by a few minutes drastically reduces the number of AGVs/carts as shown by the results in Table 6. The average number of AGVs and carts in the case of the default scenario is: 29.6 and 317.8, respectively. Furthermore, for 95% of the instances the number of AGVs is less than or equal to 34, and the number of carts is less than or equal to 338. Hence, this gives an estimation of the number of AGVs and carts needed to cover 95% of daily demand scenarios. Extra percentage of AGVs and carts should be acquired as well to account for breakdowns, damages, and high activity days and events. This percentage can be defined based on the experience of other healthcare institutions or it can also depend on the manufacturer's recommendation.

7. Conclusion

In this paper, we studied the FSRPS-AGV based on a real-life application. This problem investigates an important tactical decision in the context of a healthcare environment. The addressed problem opens the door to a new class of problems which can be described as multi-fleet sizing and routing problems under multiple synchronization constraints. We described the problem and introduced a mathematical formulation. A powerful matheuristic algorithm was designed for the problem. The algorithm is based on a fast and efficient dynamic reoptimization of the routes upon the arrival of new requests. We also solve a static version of the problem which assumes the knowledge of all requests at the start of the day. Computational experiments were conducted using small and large size instances generated based on real-data from a large hospital. Computational results present a comparison of our matheuristic against the exact method and highlight the performance of our matheuristic in solving large size instances. They also present findings relative to the static version of the problem, and the results of the sensitivity analysis.

A key finding is that the tight time windows of the requests make the static problem very similar to the dynamic one, indicating a potential use of the algorithm in real-life. We demonstrate that with a small degree of flexibility (slightly larger time windows and less conservative speed estimations), significant performance improvements are possible, and important savings can be obtained with a much smaller fleet of the expensive AGVs, at the expense of a slightly larger number of carts, which are much cheaper. However, it is important to highlight that the flexibility with respect to AGV speed is limited due to safety concerns. Finally, we provide managerial insights with respect to the number of AGVs and carts that should be acquired by our industrial partner. We highlighted the fact that deciding the number of AGVs and carts to acquire by our partner based on our results depends mainly on two main factors: the budget allocated to acquire them and the risk tolerance to a potential violation of TWs in case where the number of AGVs and/or carts is not sufficient to serve all the requests at a given moment of the time horizon. Moreover, we point out that extra percentage of AGVs and carts should be acquired as well to account for breakdowns, damages, and high activity days and events.

The approach proposed in this paper can be successfully applied to support decision making during preliminary design of automated transportation systems in hospitals and to estimate their future performance, especially in the case where expensive and time consuming simulations are not economically justifiable. Moreover, this work could be extended by considering more real-life challenges such as battery charging constraints, resource capacity constraints, and congestion.

CRediT authorship contribution statement

Imadeddine Aziez: Conceptualization, Software, Formal analysis, Writing – original draft. **Jean-François Côté:** Conceptualization, Writing – review & editing, Supervision. **Leandro C. Coelho:** Conceptualization, Writing – review & editing, Supervision.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2019-00094 and 2021-04037. We thank Compute Canada for providing high-performance parallel computing facilities. We thank an associate editor and three anonymous referees for their valuable comments on an earlier version of the paper. This work was also partially supported by grants from the Key Program of National Natural Science Foundation of China (NSFC) and Fonds de recherche du Québec – Société et culture (FRQSC) Joint Project (NSFC No. 72061127002, FRQSC No. 295837).

References

Ai, T.J., Kachitvichyanukul, V., 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. 36 (5), 1693–1702.

Ali, O., Côté, J.F., Coelho, L.C., 2021. Models and algorithms for the delivery and installation routing problem. European J. Oper. Res. 291 (1), 162–177.

Arifin, R., Egbelu, P.J., 2000. Determination of vehicle requirements in automated guided vehicle systems: a statistical approach. Prod. Plan. Control 11 (3), 258–270.

Arslan, A.M., Agatz, N., Kroon, L., Zuidwijk, R., 2019. Crowdsourced delivery – A dynamic pickup and delivery problem with ad hoc drivers. Transp. Sci. 53 (1), 222–235.

Azi, N., Gendreau, M., Potvin, J.-Y., 2012. A dynamic vehicle routing problem with multiple delivery routes. Ann. Oper. Res. 199 (1), 103–112.

Bačík, J., Ďurovskỳ, F., Biroš, M., Kyslan, K., Perduková, D., Padmanaban, S., 2017. Pathfinder–development of automated guided vehicle for hospital logistics. IEEE Access 5, 26892–26900.

Baldacci, R., Bartolini, E., Mingozzi, A., 2011. An exact algorithm for the pickup and delivery problem with time windows. Oper. Res. 59 (2), 414–426.

Benzidia, S., Ageron, B., Bentahar, O., Husson, J., 2019. Investigating automation and AGV in healthcare logistics: a case study based approach. Int. J. Logist. Res. Appl. 22 (3), 273–293.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2010. Dynamic pickup and delivery problems. European J. Oper. Res. 202 (1), 8–15.

Bhosekar, A., Ekşioğlu, S., Işık, T., Allen, R., 2021a. A discrete event simulation model for coordinating inventory management and material handling in hospitals. Ann. Oper. Res. 1–28.

Bhosekar, A., Işık, T., Ekşioğlu, S., Gilstrap, K., Allen, R., 2021b. Simulation-optimization of automated material handling systems in a healthcare facility. IISE Trans. Healthcare Syst. Eng. 1–22.

Boysen, N., De Koster, R., Weidinger, F., 2019. Warehousing in the e-commerce era: A survey. European J. Oper. Res. 277 (2), 396–411.

Chang, K., Chang, A., Kuo, C., 2014. A simulation-based framework for multi-objective vehicle fleet sizing of automated material handling systems: an empirical study. J. Simul. 8 (4), 271–280.

Chao, I.M., 2002. A tabu search method for the truck and trailer routing problem. Comput. Oper. Res. 29 (1), 33–51.

Chen, W.A., De Koster, R.B., Gong, Y., 2021a. Performance evaluation of automated medicine delivery systems. Transp. Res. E 147, 102242.

Chen, C., Demir, E., Huang, Y., Qiu, R., 2021b. The adoption of self-driving delivery robots in last mile logistics. Transp. Res. E 146, 102214.

Chen, X., He, S., Zhang, Y., Tong, L.C., Shang, P., Zhou, X., 2020. Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework. Transp. Res. C 114, 241–271.

Chikul, M., Maw, H.Y., Soong, Y.K., 2017. Technology in healthcare: A case study of healthcare supply chain management models in a general hospital in Singapore. J. Hosp. Adm. 6 (6), 63–70.

Choobineh, F.F., Asef-Vaziri, A., Huang, X., 2012. Fleet sizing of automated guided vehicles: a linear programming approach based on closed queuing networks. Int. J. Prod. Res. 50 (12), 3222–3235.

Drexl, M., 2012. Synchronization in vehicle routing – a survey of VRPs with multiple synchronization constraints. Transp. Sci. 46 (3), 297–316.

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. European J. Oper. Res. 227 (2), 275–283.

Drexl, M., 2014. Branch-and-cut algorithms for the vehicle routing problem with trailers and transshipments. Networks 63 (1), 119–133.

Egbelu, P.J., 1987. The use of non-simulation approaches in estimating vehicle requirements in an automated guided vehicle based transport system. Mater. Flow 4 (1), 17–32.

Fink, M., Desaulniers, G., Frey, M., Kiermaier, F., Kolisch, R., Soumis, F., 2019. Column generation for vehicle routing problems with multiple synchronization constraints. European J. Oper. Res. 272 (2), 699–711.

Fragapane, G.I., Bertnum, A.B., Hvolby, H., Strandhagen, J.O., 2018. Material distribution and transportation in a norwegian hospital: a case study. IFAC-Papersonline 51 (11), 352–357.

Fragapane, G.I., Hvolby, H., Sgarbossa, F., Strandhagen, J.O., 2020. Autonomous mobile robots in hospital logistics. In: IFIP International Conference on Advances in Production Management Systems. Springer, pp. 672–679.

Ganesharajah, T., Hall, N.G., Sriskandarajah, C., 1998. Design and operational issues in AGV-served manufacturing systems. Ann. Oper. Res. 76, 109–154.

Hamzheei, M., Farahani, R.Z., Rashidi-Bajgan, H., 2013. An ant colony-based algorithm for finding the shortest bidirectional path for automated guided vehicles in a block layout. Int. J. Adv. Manuf. Technol. 64 (1–4), 399–409.

Katevas, N., 2001. Mobile Robotics in Healthcare, vol. 7. IOS Press, Amsterdam, Netherlands.

Le-Anh, T., De Koster, R., 2006. A review of design and control of automated guided vehicle systems. European J. Oper. Res. 171 (1), 1–23.

Lund, K., Madsen, O.B., Rygaard, J.M., 1996. Vehicle routing problems with varying degrees of dynamism. Technical report, IMM Institute of Mathematical Modelling.

Mahadevan, B., Narendran, T.T., 1993. Estimation of number of AGVs for an FMS: an analytical model. Int. J. Prod. Res. 31 (7), 1655–1670.

Maxwell, W.L., Muckstadt, J.A., 1982. Design of automatic guided vehicle systems. IIE Trans. 14 (2), 114–124.

Mitrović-Minić, S., Laporte, G., 2004. Waiting strategies for the dynamic pickup and delivery problem with time windows. Transp. Res. B 38 (7), 635–655.

Nishi, T., Hiranaka, Y., Grossmann, I.E., 2011. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. Comput. Oper. Res. 38 (5), 876–888.

Ozbaygin, G., Savelsbergh, M.W.P., 2019. An iterative re-optimization framework for the dynamic vehicle routing problem with roaming delivery locations. Transp. Res. B 128, 207–235.

Parragh, S.N., Cordeau, J.-F., 2017. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. Comput. Oper. Res. 83, 28–44.

Pedan, M., Gregor, M., Plinta, D., 2017. Implementation of automated guided vehicle system in healthcare facility. Procedia Eng. 192, 665–670.

Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L., 2013. A review of dynamic vehicle routing problems. European J. Oper. Res. 225 (1), 1–11.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Rajotia, S., Shanker, K., Batra, J.L., 1998. Determination of optimal AGV fleet size for an FMS. Int. J. Prod. Res. 36 (5), 1177–1198.

Regnier-Coudert, O., McCall, J., Ayodele, M., Anderson, S., 2016. Truck and trailer scheduling in a real world, dynamic and heterogeneous context. Transp. Res. E 93, 389–408.

Ritzinger, U., Puchinger, J., Hartl, R.F., 2016. A survey on dynamic and stochastic vehicle routing problems. Int. J. Prod. Res. 54 (1), 215–231.

Ropke, S., Cordeau, J.-F., 2009. Branch and cut and price for the pickup and delivery problem with time windows. Transp. Sci. 43 (3), 267–286.

Ropke, S., Cordeau, J.-F., Laporte, G., 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49 (4), 258–272.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Rossetti, M.D., Felder, R.A., Kumar, A., 2000. Simulation of robotic courier deliveries in hospital distribution services. Health Care Manag. Sci. 3 (3), 201–213.

Rossetti, M.D., Selandari, F., 2001. Multi-objective analysis of hospital delivery systems. Comput. Ind. Eng. 41 (3), 309–333.

Şahin, M., Çavuşlar, G., Öncan, T., Şahin, G., Aksu, D.T., 2013. An efficient heuristic for the multi-vehicle one-to-one pickup and delivery problem with split loads. Transp. Res. C 27, 169–188.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 417–431.

Sinriech, D., Tanchoco, J., 1992. An economic model for determining AGV fleet size. Int. J. Prod. Res. 30 (6), 1255–1268.

Subramanian, A., Drummond, L.M.A., Bentes, C., Ochi, L.S., Farias, R., 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. 37 (11), 1899–1911.

Sun, P., Veelenturf, L.P., Hewitt, M., Van Woensel, T., 2020. Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. Transp. Res. E 138, 101942.

Sun, W., Yu, Y., Wang, J., 2019. Heterogeneous vehicle pickup and delivery problems: Formulation and exact solution. Transp. Res. E 125, 181–202.

Talbot, L., 2003. Design and performance analysis of multistation automated guided vehicle systems (Ph.D. thesis). UCL-Université Catholique de Louvain.

Tanchoco, J., Egbelu, P.J., Taghaboni, F., 1987. Determination of the total number of vehicles in an AGV-based material transport system. Mater. Flow 4 (1), 33–51.

Veenstra, M., Cherkesly, M., Desaulniers, G., Laporte, G., 2017. The pickup and delivery problem with time windows and handling operations. Comput. Oper. Res. 77, 127–140.

Villegas, J.G., Prins, C., Prodhon, C., Medaglia, A.L., Velasco, N., 2013. A matheuristic for the truck and trailer routing problem. European J. Oper. Res. 230 (2), 231–244.

Vis, I.F.A., 2006. Survey of research in the design and control of automated guided vehicle systems. European J. Oper. Res. 170 (3), 677–709.

Vis, I.F.A., De Koster, R., Roodbergen, K.J., Peeters, L.W., 2001. Determination of the number of automated guided vehicles required at a semi-automated container terminal. J. Oper. Res. Soc. 52 (4), 409–417.

Vis, I.F.A., de Koster, R., Savelsbergh, M.W.P., 2005. Minimum vehicle fleet size under time-window constraints at a container terminal. Transp. Sci. 39 (2), 249–260.

Vivaldini, K., Rocha, L.F., Martarelli, N.J., Becker, M., Moreira, A.P., 2016. Integrated tasks assignment and routing for the estimation of the optimal number of AGVs. Int. J. Adv. Manuf. Technol. 82 (1–4), 719–736.

Yifei, T., Junruo, C., Meihong, L., Xianxi, L., Yali, F., 2010. An estimate and simulation approach to determining the automated guided vehicle fleet size in FMS. In: 2010 3rd International Conference on Computer Science and Information Technology, vol. 9. IEEE, pp. 432–435.