CIRRELT
1971 – 2021

CIRRELT-2021-18

Fleet Sizing of Healthcare Automated Guided Vehicles

Imadeddine Aziez
Jean-François Côté
Leandro C. Coelho

April 2021

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2021-005

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Fleet Sizing of Healthcare Automated Guided Vehicles

Imadeddine Aziez, Jean-François Côté*, Leandro C. Coelho

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, Québec, Canada

Abstract. Transportation activities within hospitals are becoming more complex due to the expansion of the number of materials and equipment used such as an increased use of disposable items. 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. Each request is composed of several tasks consisting of moving material from a pickup point to a delivery point using different types of carts. Operation synchronization among the tasks of the same request and movement synchronization with respect to AGVs and carts are major challenges of the adressed problem. In this paper, we describe, model, and solve the FSRPS-AGV. We introduce a mathematical formulation and propose a powerful matheuristic. The algorithm is 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. Finally, we provide managerial insights with respect to the number of AGVs and carts that should be acquired by our industrial partner.

Keywords: automated guided vehicles, healthcare logistics, fleet sizing problem.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Council of Canada (NSERC) under grants 2015-04893 and 2019-00094. We thank Compute Canada for providing high-performance parallel computing facilities. We thank an associate editor for their valuable suggestions on an earlier version of the paper.

---

* Corresponding author: Jean-Francois.Cote@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2021

1 Introduction

Healthcare systems around the world are facing many challenges such as increasing costs of treatment, growing eldery 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, considerably savings can be obtained by improving clinical practices by better managing supplies, labour, 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 efficiency and productivity of 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].

The design of AGV systems requires taking into consideration a set of tactical and operational issues: system design, dispatching, routing, scheduling, fleet sizing, and flow path design [Vis, 2006]. In this paper, we study the fleet sizing and routing problem with synchronization for AGVs with dynamic demands (FSRPS-AGV). In this problem, each one of the daily requests is composed of at least two tasks, and each task is defined by a pickup-delivery pair. For instance, an AGV performing a request with two tasks starts by driving to the pickup point of the first task to pickup a cart of a given type. Then, the AGV drives to the delivery point to deliver the cart. Subsequently, this delivery point is the origin of the next task of the AGV. We assume that the tasks of the same request can be served by multiple AGVs, hence, a synchronization challenge is raised in this case with respect to the temporal coordination of AGVs serving the tasks of the same request. 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 windows. Moreover, a comparison between two AGV fleet exploitation strategies is conducted in Section 5.3.3: 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. In contrast, the tasks of the same request must be served using the same cart. Consequently, one may observe that this problem is charecterized by two fleets with different operating modes. Therefore, another synchronization challenge is related to the spatial coordination of AGVs and carts, meaning that the carts must always move with AGVs, called movement synchronization. Finally, it is also 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 including: food carts, waste carts, cage carts, pharmacy carts, and other 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. Our 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 high uncertainty 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 Figure 1, we show an example derived from the real-data provided by our partner, in which a food delivery request is composed of two tasks: the first task consists of picking up a food cart from the food preparation section and delivering it to a given client inside the hospital, while the second task consists of picking up the same cart (with food remains) from the client and delivering it to the food soil department where it will be cleaned before it is ready to be sent again to the food preparation area by another task. In this example, an empty AGV must be sent from the depot to perform the first task of this request, then the same or another AGV is responsible for performing the second task. When no more tasks exist, the AGV drives back to the depot for storage and charging for example. In this example, it is clear how synchronization plays an important role in satisfying these tasks. A time window (TW) is associated with each task, and more than one AGV may be used to fulfill the tasks of a given request.

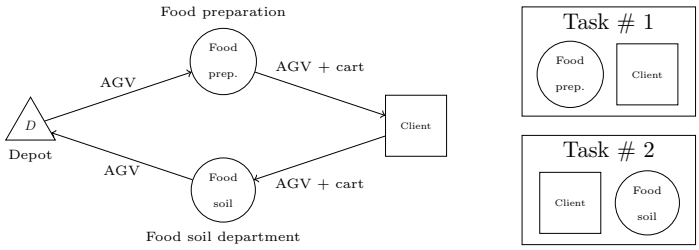


Figure 1: An example of satisfying a request with two tasks using AGVs

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, then based on these routes, an extended min-flow problem 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 is devoted to the literature review. Section 3 provides a formal description and a mathematical formulation for the problem. Section 4 presents an overview of our matheuristic. Detailed computational experiments are presented in Section 5 and conclusions follow in Section 6.

2 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 then 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 [Kim and Bae, 2004, Chen et al., 2020], warehouses [Boysen et al., 2019], and it has recently become more popular in the healthcare industry [Bačík et al., 2017, Chikul et al., 2017]. For healthcare facilities, AGVs are used to handle a variety of materials (samples, food, medicines, 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], particle swarm optimization [Ai and Kachitvichyanukul, 2009], parallel neighborhood descent [Subramanian et al., 2010], tabu embedded simulated annealing [Şahin et al., 2013], and a population-based metaheuristic of [Cherkesly et al., 2015]. 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, Cherkesly et al., 2014, Veenstra et al., 2017], and a set partitioning-based algorithm [Baldacci et al., 2011].

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. The problem was solved heuristically using ALNS where real-time decisions concerned the acceptance of the new requests, not about their service. This was done by generating multiple scenarios and solving them during the execution. 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) [Derigs et al., 2013, Parragh and Cordeau, 2017]. In the TTRP which was orginally 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 VRPs with multiple synchronization constraints (VRPMSs, e.g., Fink et al. [2019], Lahyani et al. [2015]). Synchronization requirements may also appear in many real-world VRPs such as delivery and installation routing problems [Ali et al., 2021]. A classification of different types of synchronization in the VRPMSs is given in Drexl [2012]. The FSRPS-AGV covers all three synchronization types among the five introduced by Drexl [2012], namely, task synchronization, operation synchronization, and movement synchronization.

AGV fleet sizing is an important problem which received considerable attention in the literature. Egbelu [1987] proposed four analytical models to estimate the AGVs fleet size in an 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. They concluded that analytical models either under-estimate or over-estimate the vehicle fleet size required in a system. 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 task. Other analytical approaches estimating the required number of AGVs include: queuing models [Tanchoco et al., 1987, Talbot, 2003, Choobineh et al., 2012], 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 [Yifei et al., 2010, Chang et al., 2014a,b, Pjevcevic et al., 2017].

In light of the reviewed literature relative to the AGV fleet size, we observe that the double fleet size kind of problems have not yet been studied in the literature despite their practicality. AGV systems where AGVs must be coupled with a specific cart in order to perform a request are increasingly used in various industries such as in healthcare industry [Pedan et al., 2017]. The high investment cost of carts in some AGV systems such as the ones used in hospitals leads to the need of investigating the AGV and cart fleet size simultaneously to lower the total investment cost of the entire AGV system. One of the main challenges raised by this problem is related to the need to coordinate the synchronization of an AGV to a cart in order to satisfy a given request.

3 Problem description and 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.

In this problem each node can represent either a pickup or a delivery point, however, the graph can be similarly defined with one node for a full task (pickup and delivery). Let N be the set of task nodes, the problem can be represented by a graph $G = (V, A)$ in which the set of vertices $V = N \cup \{0, n+1\}$ where the starting and ending depots are the same and are defined by nodes 0 and $n+1$. Let $R = \{r_1, \ldots, r_k\}$ be the set of requests to be routed such that $k \leq n$. Each request $r \in R$ is represented by a set of transportation tasks. The number of tasks per request is variable, and in our application it typically ranges between two and five. Let $r(i)$ be the request associated with node $i \in N$. Precedence constraints must be respected, meaning that the pickup node is visited earlier than the delivery node. Moreover, precedence constraints must also be respected when performing the tasks of each request $r \in R$ such that they are completed in a specific order. We consider a fleet of homogeneous AGVs sufficiently large to serve all the requests such that $K = \{1, \ldots, k\}$ 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 $H = \{1, \ldots, h\}$. For each cart type $h \in H$, a sufficiently large number of carts is available to serve the requests which use a cart of type h. Each cart type h is associated with a cost, say, the price of purchase of one cart of type h. Note that 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 $A = V \times V$ minus arcs that lead to infeasible solutions. 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 vertex $i \in V$ 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. AGVs can depart from the depot at a_0 and must return not later than b_{n+1}. Furthermore, a travel time $t_{ij} \geq 0$ is associated with each arc $(i, j) \in A$. The triangle inequality holds for travel times as the distances represent shortest paths.

The set of omitted arcs is divided into two sets: the set of cart only omitted arcs, and the set of AGV and cart omitted arcs. For simplicity, we assume two requests $r_1, r_2 \in R$ such that $r_1 = \{i_1, i_2, i_3\}$, $r_2 = \{i_4, i_5, i_6\}$ are the sets of nodes of r_1 and r_2, respectively. The set of cart only omitted arcs includes: arcs from nodes i_1, i_2 to nodes of r_2, arcs between nodes i_4, i_5 and nodes of r_1, arcs from node i_3 to nodes i_5, i_6, arcs between node i_6 and nodes i_2, i_3, arcs from nodes i_1, i_2, i_4, i_5 to the end depot, and arcs from the start depot to nodes i_2, i_3, i_5, i_6. In the case where requests r_1 and r_2 must be served by different types of carts, all arcs between r_1 and r_2 and vice-versa are omitted. The set of AGV and cart omitted arcs includes: arcs violating TW constraints and arcs violating precedence constraints among the nodes of the same request such as the arc (i_2, i_1) for request r_1. 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 travel time, 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 cost of the cart used to serve node j, and β representing the cost of one AGV (price of purchase of one AGV). 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 starting time of service at node $i \in V$. The model can then be formulated as follows:

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

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

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

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

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

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

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

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

$$a_i \leq S_i \leq b_i \qquad i \in V \qquad (9)$$
$$x_{ij}, y_{ij} \in \{0,1\} \qquad (i,j) \in A \qquad (10)$$
$$S_i \geq 0 \qquad i \in V. \qquad (11)$$

The objective function (1) minimizes the overall cost of carts, AGVs, and the travel time. Constraints (2)–(5) are degree constraints requiring each node to be visited exactly once by an AGV and a cart. Constraints (6)–(9) guarantee schedule feasibility with respect to TWs. Here, M is a big number and can be set to the maximum return time to the depot b_{n+1}. Note that constraints (6) also handle synchronization requirements with regard to temporal aspect among the nodes of the same request, and they preserve the precedence order of nodes in r. 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.

4 Matheuristic algorithm for the FSRPS-AGV

This section describes the algorithm proposed to solve the FSRPS-AGV, presenting its general dynamic framework. The main features of the heuristic applied to optimize the number of AGVs are described in detail in Section 4.1. The mathematical formulation used to optimize the number of carts which is based on the classical formulation of the min-flow problem is presented in Section 4.2.

Our algorithm is based on a periodic reoptimization of the routes whenever a new request arises. A static problem is solved with the latest available data upon the arrival of new requests following an "AGV First, Cart Second" approach, meaning that it starts by optimizing AGV routes, then optimizing the number of carts. 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.

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. To minimize the number of AGVs in the initial solution, we set a high insertion cost on moves within an AGV that has not been used. More AGVs can be used during the execution of ALNS if the need arises. More formally, an additional AGV is used if one or more requests cannot be inserted into the solution for

certain number of ALNS iterations. In the *while* loop from lines 2–6, we repeatedly reoptimize the previously planned AGV routes by solving a static problem with the latest available data upon the arrival of new requests. At any given moment, previously assigned requests which have not yet been served are also re-planned when reoptimizing the routes. In contrast, each previously assigned request which has already been served or that has an AGV currently driving towards it is fixed in its position of the solution, as this decision can no longer be changed. This corresponds to some fixed variables in the mathematical model or a partial solution S that it is not subject to any changes. Therefore, any insertion operation must be performed after the last fixed node of a given route, and only non-fixed requests can be removed from the solution when executing ALNS.

ALNS is executed at line 4 in order to reoptimize the routes and obtain the number of required AGVs (see Section 4.1). Then, the extended min-flow problem is solved to optimize the number of carts in the solution (line 5, Section 4.2). Note that solving the extended min-flow problem implies updating the starting time of service at non-fixed nodes. This dynamic procedure is repeated until there are no new requests not served. 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: $ExecuteALNS(S)$
5: Solve $Min\_Flow(S)$ to minimize the number of carts
6: **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 5, we explain the large real-life instances we obtained for a 24h period, and how we assess the performance of this framework with respect to its sensitivity to different parameters.

4.1 ALNS framework to minimize the 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 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].

4.1.1 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.

4.1.2 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, we generate the set of feasible insertion moves and their 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. The set of feasible insertion moves of all configurations is then used to generate the variables m_{ij}^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, the set L_r of configurations of request r, i_f as the first node of a configuration, and i_l as the last node of a configuration. The cost of inserting node i from configuration l after node j is denoted c_{ij}^l. The problem can be mathematically formulated with variables: m_{ij}^l equal to 1 if node i from configuration l is inserted after node j, 0 otherwise; and S_i indicating the starting time of service at node $i \in V$. The MIP can then be formulated as follows:

$$\min \sum_{l \in L_r} \sum_{i \in r} \sum_{j \in I \setminus \{n+1\}} c_{ij}^l m_{ij}^k \tag{12}$$

$$\text{s.t.} \sum_{l \in L_r} \sum_{j \in I} m_{ij}^l = 1 \qquad i \in r \tag{13}$$

$$S_{i_f} \geq S_j + s_j + t_{ji_f} - M\left(1 - \sum_{l \in L_r} m^l_{i_f j}\right) \qquad i_f \in r, j \in I \qquad (14)$$

$$S_k \geq S_{i_l} + s_{i_l} + t_{i_l k} - M\left(1 - \sum_{l \in L_r} m^l_{i_l j}\right) \qquad j,k \in I, k \succ j, i_l \in r \qquad (15)$$

$$S_k \geq S_j + s_j + t_{jk} - M \sum_{i \in r} \sum_{l \in L_r} m^l_{ij} \qquad j,k \in I, k \succ j \qquad (16)$$

$$S_i \geq S_j + s_j + t_{ji} \qquad i,j \in r, i \succ j \qquad (17)$$

$$S_k \geq S_j + s_j + t_{jk} \qquad j,k \in r', r' \in I_R, k \succ j \qquad (18)$$

$$a_i \leq S_i \leq b_i \qquad i \in V \qquad (19)$$

$$m^l_{ij} \in \{0,1\} \qquad i \in r, j \in I, l \in L_r \qquad (20)$$

$$S_i \geq 0 \qquad i \in V. \qquad (21)$$

The objective function (12) minimizes the overall insertion cost of request r. Constraints (13) ensure that only one configuration is used to insert each node $i \in r$. Constraints (14)–(19) guarantee schedule feasibility with respect to time windows. Constraints (17) and (18) 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 (20) and (21) 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.

4.2 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 AGV routes previously defined. 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 $U = \{(i,j) \in V \times V |\ i$ 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.

This model is an extension of the classical formulation of the min-flow problem by adding schedule feasibility constraints with respect to time windows. A new graph $\bar{G} = (V, \bar{A})$, where the set of arcs is $\bar{A} \subseteq A_c$ minus omitted arcs. In addition to the set of cart only omitted arcs presented in Section 3, 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 first node of r and j is the last 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 the starting 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 4.1, the objective function of ALNS minimizes the total travel time of AGVs, thus, the starting time of service of AGVs at each node is naturally set to its earliest value. It is then possible to manipulate the starting 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 starting time of service at node $i \in V$. The model can then be formulated as follows:

$$\min \sum_{j \in N} y_{0j} \tag{22}$$

$$\text{s.t.} \sum_{j \in \bar{A}^+(i)} y_{ij} = 1 \qquad i \in N \tag{23}$$

$$\sum_{j \in \bar{A}^-(i)} y_{ji} = 1 \qquad i \in N \tag{24}$$

$$S_j \geq S_i + s_i + t_{ij} - M(1 - y_{ij}) \qquad (i,j) \in \bar{A}, (r_i, r_j) \in W \tag{25}$$

$$S_j \geq S_i + s_i + t_{ij} \qquad i,j \in r, r \in R, j \succ i \tag{26}$$

$$S_j \geq S_i + s_i + t_{ij} \qquad (i,j) \in U \tag{27}$$

$$a_i \leq S_i \leq b_i \qquad i \in V \tag{28}$$

$$y_{ij} \in \{0,1\} \qquad (i,j) \in \bar{A} \tag{29}$$

$$S_i \geq 0 \qquad i \in V. \tag{30}$$

The objective function (22) minimizes the number of carts departing from the start depot. Constraints (23) and (24) are degree constraints requiring each node to be visited exactly once. Constraints (25)–(28) guarantee schedule feasibility with respect to time windows. Note that constraints (26) 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 (29) and (30) 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}.

5 Computational experiments

This section describes the computational experiments. These have been performed on a computer equipped with a 2.1 gigahertz 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 5.1 provides a detailed description of the real dataset used and introduces the characteristics of the problem instances. Parameter setting is discussed in Section 5.2. The results of detailed and extensive computational experiments are then presented in Section 5.3.

5.1 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 we report the number of requests and the average time needed to perform a request which is the difference between the TW upper bound of the last task of the request and the TW lower bound of its first task. 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 minutes before the TW lower bound of its first task. In terms of frequency of occurrence, we have: *High* frequency requests occuring four times a week, *Medium* frequency requests occuring three times a week, and *Low* requests occuring twice a week. In terms of TW intervals, on a typical day, we distinguish two types of requests: requests with *fixed 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 *flexible TW*. In the dataset, the TW of the first type of requests is enlarged by 5 minutes, while the one of the second type is enlarged by 30 minutes. Finally, as stated in Section 1, 10 types of carts are used to serve request, namely: 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. Figure 2 depicts the number of requests per category based on the different characteristics of the requests in the dataset. Figure 3 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.

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

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 between 0 and 1 is generated, if the generated value is less than or equal

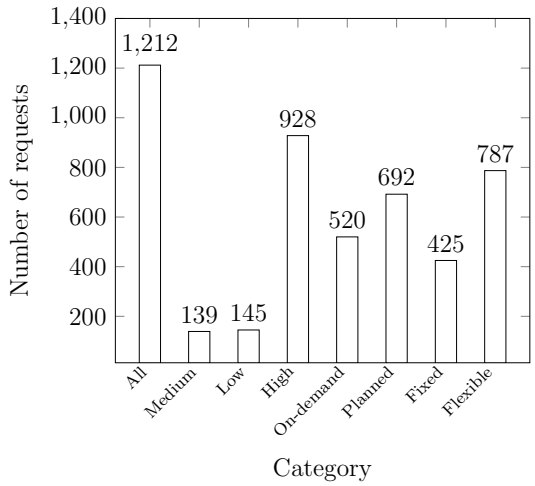


Figure 2: Number of requests per category

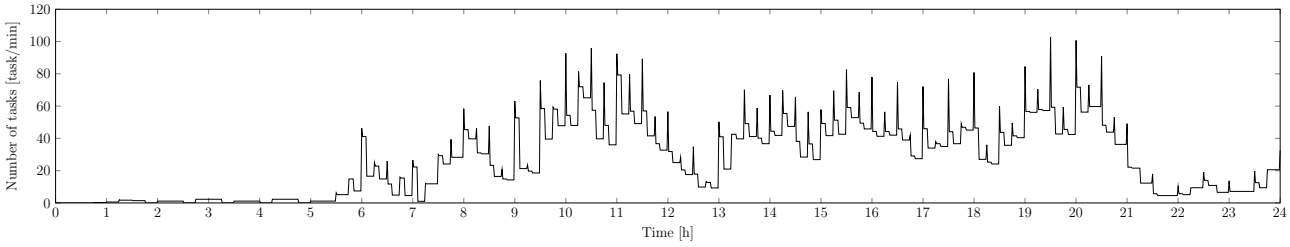


Figure 3: Number of tasks over time during the day

$FrequencyOfOccurrence/7$ then it is added to the instance, otherwise we pass to the next request. Our instances have an average degree of dynamism of 42%.

5.2 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 minutes. 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 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 seconds which is somewhat far from the maximum runtime per event which is set to 300 seconds. This can be explained by the fact that the runtime per event of most events is much less than 300 seconds, however, in specific moments during the day, the runtime per event can exceed 200 seconds such as in the case of optimizing the routes of planned requests at the start of the day due to a the high number of requests, this can also happen during peak activity moments where optimization runtime may exceed the average runtime per event.

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

5.3 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.

5.3.1 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 this 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 average number of events per instance category. 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 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 the exact algorithm for all instance categories. The matheuristic outperforms exact algorithm in terms of number of AGVs for large instances with 400 requests. The average runtime per event increases significantly by increasing the size of the problems solved by the exact algorithm, while in the case of the matheuristic the average runtime per event still increases but less significantly than the exact method. Despite the travel times being large, 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

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

		Exact algorithm					Matheuristic							
# Req.	# Eve.	# 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

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 15000 of iterations in order to optimize the travel time. We then run a non-restricted ALNS (similar to the one described in Section 4.1) 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 can be 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 extended min-flow problem 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, 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.* reports the same metric but only for the instances solved to optimality by the exact algorithm. The optimality gap is calculated as before. 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 outperforms the exact method. 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 also fast: the average total runtime goes from 154.5 to 741.9 seconds for all instance categories, which is one order of magnitude faster than the exact algorithm.

One can also compare 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. This is justified by the holistic character of the static problem which assumes the

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

	Exact algorithm					Matheuristic								
# Req.	Sol.	# AGV	# Carts	Travel t. (min)	Time (s)	# AGV		# Carts		Travel t. (min)		Gap (%)	Gap sol. (%)	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	154.5
200	3	112	1057	3670.8	5416.9	111	111	1062	1063.0	3999.7	4057.9	1.61	1.31	348.6
300	1	159	1560	5459.2	6940.1	157	157	1568	1573.0	5995.9	6069.1	0.99	1.39	546.2
400	0	228	2043	7308.8	7191.9	203	203	2051	2054.6	7895.0	8006.6	4.15	-	741.9

knowledge of all input information at the start of time horizon. The results of the exact algorithm shows that the gap in terms number of AGVs between the dynamic and static problems increases by increasing the number of requests per instance. This mainly due to an increasing difficulty to achieve high quality solutions for larger instances when solving the static problem, whereas, in the dynamic problem, a small and easier static problem is solved upon the arrival of new requests. 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.

We highlight the fact that the future requests are not known does not make the problem any more difficult. Indeed, when an oracle is used and a fully static problem is solved (i.e., all future requests are known at once, when the optimization is performed), the results of Table 4 indicate the optimal number of AVGs and carts; this is in contrast to the results of Table 3 where the same instances are solved with dynamic information arrival; note that for instances with up to 300 requests, the difference is of only 2 AVGs and zero carts (we exclude instances with 400 requests here because those were not generally solved to optimality). This indicates that dynamism does not make the problem more challenging requiring more resources. 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 sized. This is further demonstrated by our sensitivity analyses when we vary the size of the time windows, which cause the number of AVGs and carts to decrease considerably.

5.3.2 Results of the dynamic optimization using real-size instances

We now present the results of the matheuristic algorithm presented in Section 4 using the 40 real-size instances. For simplicity, the instances are divided into two subsets whose number of requests is between [526,629] and [629,685]. Table 5 reports the detailed results. For each subset, we report the average number of events (# Eve.), the average number of AGVs (# AGV), the average number of carts (# Carts), and the average travel time (Travel t.). Note that *Best* denotes the value corresponding to the best solution and *Avg* is the average value of five replications. We also report the average runtime per event (T/E) in the last column.

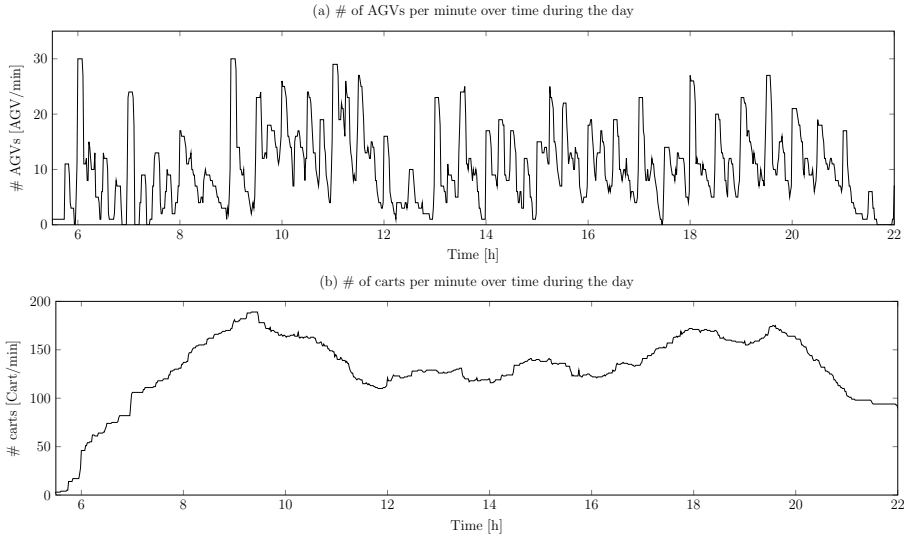
The results in Table 5 show that our matheuristic is very fast and robust in solving large instances of the FSRPS-AGV, as the total average runtime per event is 135.4 seconds and the best values are very close to the averages. The number of AGVs corresponding to

the best solution varies between 25 and 36 for all instances, with a total average of 29.6, while the number of carts varies between 288 and 344 with a total average of 317.8. 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. Figure 4 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).

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

		# AGV		# Carts		Travel t. (min)		
# req.	# Eve.	Best	Avg	Best	Avg	Best	Avg	T/E (s)
[526,629]	53.4	29.1	29.3	310.9	310.8	1261.5	1268.0	132.8
[629,685]	54.2	30.2	30.5	325.1	325.1	1316.9	1324.8	138.1
Tot avg.	53.8	29.6	29.9	317.8	317.8	1288.5	1295.7	135.4

Figure 4: Number of AGVs and carts in use per minute over time during the day for a real-size instance



5.3.3 Sensitivity analysis

In this section, we conduct sensitivity analysis by varying the TW intervals, AGV fleet exploitation strategy, AGV speed, and the release time of requests. We use the 40 real-size instances described in Section 5.1 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 Fixed and the Flexible 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 Fixed TW requests and the second one is the TW interval of Flexible TW requests. Table 6 and Figure 5 show the sensitivity analysis with respect to TWs. Table 6 reports a summary of the results for

all TW configurations where the average number of Fixed TW and Flexible TW requests are 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 Fixed TW requests has a significant impact in decreasing the number of AGVs. For example, varying the Fixed TW from 3 to 10 min while keeping Flexible TW constant at 15 min leads the number of AGVs to decrease from 1266 to 908. Moreover, the change in the Flexible TW causes a significant decrease of the number of carts, as it can be observed in case of 3 min TW interval for Fixed TW requests: the number of carts decreases from 13019 to 11833. Figure 5 shows that larger TWs help reduce the peaks of activity during the day.

Tables 7 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 4 against a restricted matheuristic. 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 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 minutes 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 8 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.

In terms of release time, we used five different values: $\mathcal{R} = \{5, 8, 12, 20, 60\}$ minutes. This analysis showed that the release time has a limited impact on the quality of the solution. Minor savings can be achieved by increasing the release time.

Table 6: Sensitivity analysis on TWs

		Flexible (392.3)					
		15 min		30 min		45 min	
		# AGV	# Carts	# AGV	# Carts	# AGV	# Carts
Fixed (237.3)	3 min	1266	13019	1265	12706	1263	11833
	5 min	1194	13019	1186	12719	1182	11830
	10 min	908	13019	754	12716	734	12033

Figure 5: Number of AGVs in use per minute over time during the day in the case of different TWs

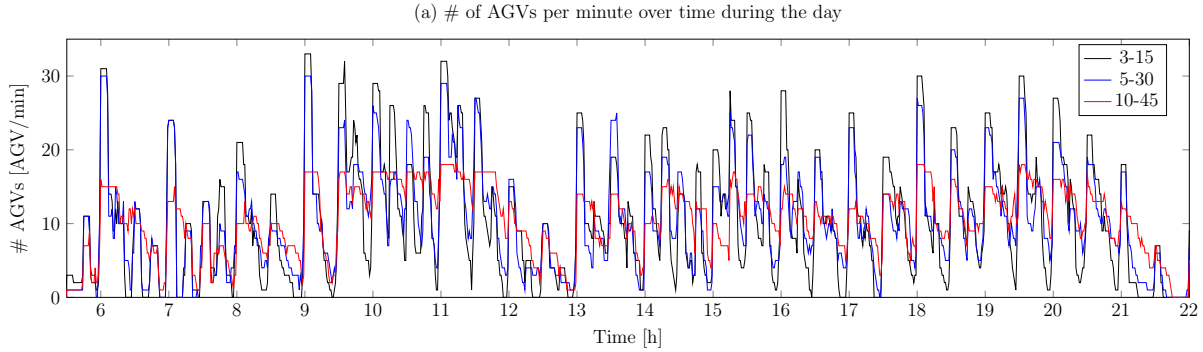


Table 7: Sensitivity analysis on AGV fleet exploitation strategy

	# AGV		# Carts		Travel t. (min)		
Strategy	Best	Avg	Best	Avg	Best	Avg	T/E (s)
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 8: Sensitivity analysis on AGV speed

	# AGV		# Carts		Travel t. (min)		T/E (s)
Speed (m/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

5.3.4 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 5.3.2. Tables 9 and 10 depict summary statistics relative to the number of AGVs and carts, where for each table we report the number of AGVs (# AGVs) or the interval of number of carts (# Carts) and their corresponding: number of instances (# Inst.), the cumulated number of instances (Cumul. # inst.), the percentage of feasible instances (Perc. (%)) for a given number of AGVs or an interval of number of carts. Deciding the number of AGVs and carts to acquire by our partner based on the results of Tables 9 and 10 depends 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. The level of lateness should not be high considering that adding few minutes to TWs drastically reduces the number of AGVs/carts as shown by the results in Table 6. 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 recommandation.

Table 9: Summary statistics relative to the number of AGVs

# AGVs	# Inst.	Cumul. # inst.	Perc. (%)
25	1	1	2.5
26	2	3	7.5
27	5	8	20
28	4	12	30
29	8	20	50
30	7	27	67.5
31	7	34	85
32	1	35	87.5
33	2	37	92.5
34	1	38	95
35	1	39	97.5
36	1	40	100

6 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

Table 10: Summary statistics relative to the number of carts

# Carts	# Inst.	Cumul. # inst.	Perc. (%)
]285,290]	1	1	2.5
]290,295]	1	2	5
]295,300]	1	3	7.5
]300,305]	3	6	15
]305,310]	6	12	30
]310,315]	3	15	37.5
]315,320]	10	25	62.5
]320,325]	7	32	80
]325,330]	1	33	82.5
]330,335]	4	37	92.5
]335,340]	2	39	97.5
]340,345]	1	40	100

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, where the performance of our matheuristic in solving large size instances and the effects of the instances' parameters on the solution are highlighted. They also present findings relative to the static version of the problem. For all small instances solved to optimality by the exact methods, near-optimal solutions were obtained by our matheuristic. The results of large instances show that our matheuristic is fast, very robust and it can handle dynamism of demand very well. Finally, the results of the sensitivity analysis show that TW intervals and AGV speed have a significant impact on the number of AGVs and carts. The results of the sensitivity analysis with respect to AGV fleet exploitation strategies show that the strategy with synchronization (M-AGV) is overall more advantageous especially when looking at its effect on the investment cost.

Although our study was conducted in a specific context which is healthcare industry, our methodology can be easily applied to different contexts where AGVs and carts are used to handle transportation operations such as in industrial contexts and within warehouses. Moreover, this work could be extended by considering more real-life challenges such as battery charging constraints and congestion. If extended, the presented matheuristic can be embedded in a general tool to build an efficient real-time routing and scheduling tool for AGVs and carts.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2015-04893 and 2019-00094. We thank Compute

Canada for providing high-performance parallel computing facilities. We thank an associate editor for their valuable suggestions on an earlier version of the paper.

References

T.J. Ai and V. Kachitvichyanukul. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36 (5):1693–1702, 2009.

O. Ali, J.F. Côté, and L.C. Coelho. Models and algorithms for the delivery and installation routing problem. *European Journal of Operational Research*, 291(1):162–177, 2021.

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

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

N. Azi, M. Gendreau, and J.-Y. Potvin. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, 199(1):103–112, 2012.

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

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

G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

N. Boysen, R. De Koster, and F. Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411, 2019.

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

K.-H. Chang, Y.-H. Huang, and S.-P. Yang. Vehicle fleet sizing for automated material handling systems to minimize cost subject to time constraints. *IIE Transactions*, 46 (3):301–312, 2014b.

I.M. Chao. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51, 2002.

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

M. Cherkesly, G. Desaulniers, and G. Laporte. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, 49(4):752–766, 2014.

M. Cherkesly, G. Desaulniers, and G. Laporte. A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading. *Computers & Operations Research*, 62:23–35, 2015.

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

F.F. Choobineh, A. Asef-Vaziri, and X. Huang. Fleet sizing of automated guided vehicles: a linear programming approach based on closed queuing networks. *International Journal of Production Research*, 50(12):3222–3235, 2012.

U. Derigs, M. Pullmann, and U. Vogel. Truck and trailer routing – problems, heuristics and computational experience. *Computers & Operations Research*, 40(2):536–546, 2013.

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

M. Drexl. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227(2):275–283, 2013.

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

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

M. Fink, G. Desaulniers, M. Frey, F. Kiermaier, R. Kolisch, and F. Soumis. Column generation for vehicle routing problems with multiple synchronization constraints. *European Journal of Operational Research*, 272(2):699–711, 2019.

K.H. Kim and J.W. Bae. A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science*, 38(2):224–234, 2004.

R. Lahyani, M. Khemakhem, and F. Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015.

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

B. Mahadevan and T.T. Narendran. Estimation of number of AGVs for an FMS: an analytical model. *The International Journal of Production Research*, 31(7):1655–1670, 1993.

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

S. Mitrović-Minić and G. Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7): 635–655, 2004.

T. Nishi, Y. Hiranaka, and I.E. Grossmann. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers & Operations Research*, 38(5):876–888, 2011.

G. Ozbaygin and M.W.P. Savelsbergh. An iterative re-optimization framework for the dynamic vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 128:207–235, 2019.

S.N. Parragh and J.-F. Cordeau. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28–44, 2017.

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

V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

D. Pjevcevic, M. Nikolic, N. Vidic, and K. Vukadinovic. Data envelopment analysis of AGV fleet sizing at a port container terminal. *International Journal of Production Research*, 55(14):4021–4034, 2017.

S. Rajotia, K. Shanker, and J.L. Batra. Determination of optimal AGV fleet size for an FMS. *International Journal of Production Research*, 36(5):1177–1198, 1998.

U. Ritzinger, J. Puchinger, and R.F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

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

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

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

M. Şahin, G. Çavuşlar, T. Öncan, G. Şahin, and D.T. Aksu. An efficient heuristic for the multi-vehicle one-to-one pickup and delivery problem with split loads. *Transportation Research Part C: Emerging Technologies*, 27:169–188, 2013.

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

D. Sinriech and J. Tanchoco. An economic model for determining AGV fleet size. *International Journal of Production Research*, 30(6):1255–1268, 1992.

A. Subramanian, L. M. A. Drummond, C. Bentes, L.S. Ochi, and R. Farias. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911, 2010.

L. Talbot. *Design and performance analysis of multistation automated guided vehicle systems*. PhD thesis, UCL-Université Catholique de Louvain, 2003.

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

M. Veenstra, M. Cherkesly, G. Desaulniers, and G. Laporte. The pickup and delivery problem with time windows and handling operations. *Computers & Operations Research*, 77:127–140, 2017.

J.G. Villegas, C. Prins, C. Prodhon, A.L. Medaglia, and N. Velasco. A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230 (2):231–244, 2013.

I.F.A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006.

I.F.A. Vis, R. De Koster, K.J. Roodbergen, and L.W. Peeters. Determination of the number of automated guided vehicles required at a semi-automated container terminal. *Journal of the Operational Research Society*, 52(4):409–417, 2001.

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

K. Vivaldini, L.F. Rocha, N.J. Martarelli, M. Becker, and A.P. Moreira. Integrated tasks assignment and routing for the estimation of the optimal number of AGVs. *The International Journal of Advanced Manufacturing Technology*, 82(1-4):719–736, 2016.

T. Yifei, C. Junruo, L. Meihong, L. Xianxi, and F. Yali. 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*, volume 9, pages 432–435. IEEE, 2010.