Integer Linear Programming for Automated Guided Vehicles Path Planning in Container Terminals

Karim Terfasse, Ghassen Cherif, Marie-José Huguet

Integer Linear Programming for Automated Guided Vehicles Path Planning in Container Terminals

Karim Terfasse[1], Ghassen Cherif[1,2] and Marie-José Huguet[1,3]

***Abstract*— Port automation has emerged as a transformative solution enabling seaport management to efficiently handle the escalating volume of operations, driven in part by the advancements in low-cost long-distance maritime transport. In this context, the use of Automated Guided Vehicles for the loading and unloading of ships creates a need to optimize the conflict-free routing of these vehicles, in order to guarantee the safe transportation of goods within automated container terminals. In this work, we propose an Integer Linear Programming model to address the problem of collision-free path planning of Automated Guided Vehicles in container terminals where each vehicle is associated to one routing request. Numerical experiments are performed to evaluate the proposed method's performances. Conclusions are drawn regarding its efficiency, and future avenues of research to address the problem are proposed.**

I. INTRODUCTION

Automating port operations is key to improve its competitiveness regarding the challenge of handling the expanding amounts of port operations due to the lucrative aspect of maritime logistics. In this work, we focus on improving the loading and unloading of ships, through the automation of container transport within a specific type of container terminals. For this objective, a specific category of autonomous vehicles known as Automated Guided Vehicles (AGVs) is employed. Effective utilization of these resources necessitates a careful planning of their movements, ensuring both safety and efficiency. This leads to the challenge of devising collision-free path planning strategies for AGVs for automated container terminals.

The collision-free routing of autonomous vehicles has been extensively studied in the literature, using various approaches. Classic shortest path algorithms, adapted to treat collisions, have been proposed to address the problem in different environments [1], [2]. The graph-like architectures commonly employed in the study of this problem make approaches of this type highly convenient. However, they have been demonstrated to be most effective for small instances, which involve a relatively limited number of vehicles in compact networks. Another category of approaches consists of leveraging Integer Programming (IP) or Mixed Integer Programming (MIP) to address the problem in various environments, including typical vehicle traffic setups [3], [4] and automated seaport container terminals [5], [6]. Here, the problem is modeled as an IP or MIP problem and usually solved using standard solvers. Decomposition methods have also been proposed to treat the solving aspect of this type of approaches [7]. While these solution methods are widely used, they also face scalability challenges when applied to large instances of the studied problem. Other approaches proposed in the literature include the use of Petri Nets [8], [9] which provide a compact modeling of the architectures considered for conflict-free AGV routing problems. Additionally, multi-objective optimization methods have also been suggested to treat the problem in small to medium-sized container terminals [10], and architecture-specific heuristics are commonly developed for particular setups of the problem, like in [11] where four different heuristics are proposed to treat the problem in the context of automated warehousing.

In conclusion, exact and heuristic methods have been proposed to address the problem of collision-free routing of autonomous vehicles in various environments. While most of these methods are very efficient when solving small instances of the considered problem, scalability challenges are faced with the increase of the number of vehicles involved, and the size of the architecture considered.

The main contributions of this work lie in the following. First, we propose an Integer Linear Programming (ILP) model adapted from the ones formulated in [5] and [6]. Then, we perform an extensive experimental study that aims to evaluate the model's performance and validate its results, as we believe that the tests presented in the aforementioned articles are not sufficiently representative.

The paper is organized as follows. Section II presents a description of the studied problem. In Section III, an ILP model is detailed as a solution method for the studied problem. Numerical experiments and results are presented in Section IV, followed by conclusions in the final section.

II. PROBLEM DEFINITION

We consider the same container terminal presented in [5], [6], [8], which is organized into blocks and crossroads, connecting the docks of the seaport to a storage area. An example of this type of structure is shown in Fig. 1, where storage areas at the top are connected to docking stations at the bottom, through blocks represented in blue, and crossroads represented in orange. A Terminal Operating System (TOS) provides loading and unloading orders, and a service provider is responsible for assigning container transport missions to the AGVs, to fulfill these orders. Essentially, a mission is defined as a travel order from a block within a docking station to a block within the storage area, or vice versa. These missions must be carried out safely,

[1] are members of the ROC team within the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS), University of Toulouse, 31013, Toulouse, France {kterfasse,gcherif,huguet}@laas.fr

[2] is with Paul Sabatier University, 31062 Toulouse, France

[3] is with INSA Toulouse, 31400 Toulouse, France

following a path provided by a routing engine, ensuring that collisions between the AGVs are avoided. A collision is defined by the presence of two or more AGVs on the same block or crossroad at the same time. The objective is to find a set of collision-free routes, which consist of an alternating succession of blocks and crossroads, that minimize the total time needed to perform all the assigned missions. The network and problem exhibit additional features, outlined below:

- A crossroad is a connection node between several blocks.
- A block is connected, at most, to two crossroads.
- Two blocks/crossroads cannot be directly connected.
- Each block and crossroad has a crossing duration.
- Each AGV (mission) has an origin and a destination that are necessarily blocks of the storage or docking areas.
- Only one AGV can be in a block or a crossroad at a time.
- An AGV cannot cross a block or a crossroad twice for a given mission.
- An AGV can wait only in a block

In this work, we focus on a particular case of the problem addressed in [5] and [6], where a unique mission is assigned for each AGV. We propose to call it the one-mission-per-AGV problem, denoted by `1MA`. In this setup, each AGV fulfills a unique mission defined by an origin block and a destination block, following collision-free paths provided by the routing engine. The purpose of this consideration is to provide a basis for solving the more general problem of multiple-missions-per-AGV in a sequential manner, using ILP-based heuristic methods.

III. Integer Linear Programming for the 1MA Problem

In this section, we propose an ILP model adapted from the two models presented in [5] and [6].

A. Network Topology Notations

We consider an environment where AGVs can circulate in a network consisting of connected **blocks** and **crossroads**. In the following, we describe these two types of components, as well as the relationship that connects them.

Blocks Blocks are the main component of the network that AGVs use to travel from the origin to the destination of their assigned missions. We denote by $\mathcal{B} = \{b_1, b_2, \ldots, b_B\} = \{b_i\}_{i\in[1,\ldots,B]}$ the set of blocks with B being its cardinal number. Each block $b_i \in \mathcal{B}$ has a crossing duration $d_i^b \in \mathbb{N}^+$. We also define ${}^{A}\mathcal{B}_i$ the set of blocks accessible from block b_i and $\overleftarrow{{}^{A}}\mathcal{B}_i$ the set of blocks from which block b_i could be accessed. The AGVs are allowed to wait in blocks to avoid collisions.

Crossroads Crossroads are the second components of the network that connect blocks to each other. Crossing from one block to another implies the use of a crossroad. We denote by $\mathcal{C} = \{c_1, c_2, \ldots, c_C\} = \{c_p\}_{p\in[1,\ldots,C]}$ the set of crossroads, with C being its cardinal number. Each crossroad $c_p \in \mathcal{C}$ has a crossing duration $d_p^c \in \mathbb{N}^+$. We also define $\mathcal{C}^i$ the set of crossroads accessible from block b_i, ${}^{A}\mathcal{B}_i^p$ the set of blocks accessible from block b_i through crossroad c_p and $\overleftarrow{{}^{A}}\mathcal{B}_i^p$ the set of blocks from which block b_i could be accessed through crossroad c_p. Additionally, for each block $b_i \in \mathcal{B}$ and each block $b_j \in {}^{A}\mathcal{B}_i$, there exists a unique crossroad c_p such that $b_j \in {}^{A}\mathcal{B}_i^p$. This crossroad is denoted ${}^{i}c^{j}$. Unlike blocks, the AGVs are not allowed to wait in crossroads to avoid collisions.

B. Missions and AGV

A mission consists of transporting a container from a block in the docking area to a block in the storage area or vice versa. A mission m is defined by its origin block denoted s_m, and its destination block denoted t_m. We denote $\mathcal{M}$ the set of missions, $\mathcal{T}$ the set of destination blocks, and $\mathcal{S}$ the set of origin blocks. In the `1MA` problem, only one mission is assigned to a given AGV. The set of AGVs is denoted $\mathcal{A} = \{a_1, a_2, \ldots, a_A\} = \{a_m\}_{m\in[1,\ldots,A]}$ with A being its cardinal number.

C. Decision variables

For the proposed ILP, we define the following decision variables :

- $X_m^{i,j}$: binary variable that identifies if AGV a_m moved from block b_i to block b_j.
- Y_m^i : binary variable that identifies if AGV a_m crossed the block b_i.
- Z_m^p : binary variable that identifies if AGV a_m crossed the crossroad c_p.
- α_m^i and β_m^i: integer variables that define the arrival time of AGV a_m to block b_i and its departure time from b_i, respectively.
- ϕ_m^p and ψ_m^p: integer variables that define the arrival time of AGV a_m to crossroad c_p and its departure time from c_p, respectively.
- ϵ_m^i : integer variable that defines the waiting time by AGV a_m in block b_i.
- $B_{m,n}^i$ and $C_{m,n}^p$: binary variable that defines the crossing order in the block b_i (resp. crossroad c_p) for each pair a_m and a_n of AGVs.

The main difference between our set of variables and that of the models presented in [5] and [6] is that, here, the arrival and departure from a block or a crossroad are defined for an AGV as a whole, whereas, in the other studies, they are defined for both the front and the back of the AGV. This eliminates two types of variables, which subsequently creates differences between the formulated constraints.

D. The objective function

We consider the same objective function as the one presented in [6], which aims at minimizing the total time required to perform this set of missions. It takes into consideration, for each AGV, three types of durations involved in our problem ; the block crossing duration, the crossroad crossing duration and the waiting time at blocks. Despite

minimizing the same quantity, our formulation of the objective function utilizes different variables from those employed in [6]. This is presented as follows :

$$\min \sum_{a_m \in \mathcal{A}} \sum_{b_i \in \mathcal{B}} d_i^b Y_m^i + \sum_{a_m \in \mathcal{A}} \sum_{c_p \in \mathcal{C}} d_p^c Z_m^p + \sum_{a_m \in \mathcal{A}} \sum_{b_i \in \mathcal{B} \setminus (\mathcal{T} \cup \mathcal{S})} \epsilon_m^i \tag{1}$$

E. The constraints

The constraints of the proposed ILP model are presented through the following blocks:

Pre-computations. Some binary variables are trivially null. Constraint (2) indicates that $X_m^{i,j}$ is null if block b_j is not accessible from block b_i. Constraint (3) indicates that a block b_i can never be accessed from itself:

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \notin {}^A\mathcal{B}_i, \quad X_m^{i,j} = 0 \tag{2}$$

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \quad X_m^{i,i} = 0 \tag{3}$$

Origins and destinations. Each AGV has to leave its assigned mission's origin. Constraint (4) indicates that AGV a_m leaves s_m one time, and one time only, and cannot access it again.

$$\forall\, a_m \in \mathcal{A}, \sum_{b_j \in \mathcal{B} \setminus \{s_m\}} (X_m^{s_m j} - X_m^{j s_m}) = 1 \tag{4}$$

Similarly, constraint (5) indicates that AGV a_m enters t_m one time, and one time only, and does not leave it.

$$\forall\, a_m \in \mathcal{A}, \sum_{b_j \in \mathcal{B} \setminus \{t_m\}} (X_m^{t_m j} - X_m^{j t_m}) = -1 \tag{5}$$

Constraints 4 and 5 were each initially separated into two distinct constraints in [5], but were reformulated as presented here by the same authors in [6]. We have opted to select the more compact version for integration into our ILP model. Additionally, the Y_m^i binary variable that identifies the crossing of a block b_i by AGV a_m needs to be set to 1 for both the origin and destination blocks of all missions, since it is trivial that AGVs cross these blocks when performing their assigned missions.

$$\forall\, a_m \in \mathcal{A}, \quad Y_m^{s_m} = 1 \tag{6}$$

$$\forall\, a_m \in \mathcal{A}, \quad Y_m^{t_m} = 1 \tag{7}$$

It is important to note that constraints (6) and (7) are redundant, and their intended purpose could be achieved and verified through the combination of constraints (4), (5), (9) and (10). However, it is always best practice to set trivial and known values as initial constraints in mathematical models, as this usually helps improve solving efficiency.

Routing constraints. Constraint (8) indicates that AGV a_m can only move to a unique accessible block from its current block b_i. The use of the inequality allows all variables $X_m^{i,j}$ to be null in case the AGV does not visit block b_i.

$$\forall\, a_m \in \mathcal{A}, \quad \forall\, b_i \in \mathcal{B}, \sum_{b_j \in \mathcal{B} \setminus \{b_i\}} X_m^{i,j} \leq 1 \tag{8}$$

Furthermore, an AGV a_m moving from block b_i to block b_j means that a_m has used both blocks, which translates to :

$$\forall\, a_m \in \mathcal{A}, \quad \forall\, b_i \in \mathcal{B} \setminus \{t_m\}, \sum_{b_j \in {}^A\mathcal{B}_i} X_m^{i,j} = Y_m^i \tag{9}$$

$$\forall\, a_m \in \mathcal{A}, \quad \forall\, b_j \in \mathcal{B} \setminus \{s_m\}, \sum_{b_i \in \mathcal{B}} X_m^{i,j} = Y_m^j \tag{10}$$

Indeed, if AGV a_m moves from b_i to b_j, the sum in the left-hand sides of the equations in (9) and (10) will be equal to 1. This equality imposes the same value on the Y_m^i and Y_m^j variables.

A similar constraint is formulated for the use of crossroad ${}^ic^j$ that links blocks b_i and b_j :

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^A\mathcal{B}_i, \quad X_m^{i,j} \leq Z_m^{{}^ic^j} \tag{11}$$

The constraint is formulated with an inequality, which is justified by the fact that the crossroad ${}^ic^j$ could also be the link between another pair of blocks b_k and b_l (${}^ic^j = {}^kc^l$). In this case, using an equality constraint would impose the crossing from b_i to b_j when ${}^ic^j$ is used for the unrelated crossing from b_k to b_l.

Additionally, an AGV can only cross a block in a unique direction for a given mission, meaning that U-turns are forbidden in the considered topology. This constraint is formulated as follows:

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^A\mathcal{B}_i, \quad X_m^{i,j} + X_m^{j,i} \leq 1 \tag{12}$$

Finally, constraint (13) makes sure that an AGV a_m leaves the block b_j once it has accessed it. It also verifies that if a_m leaves the block b_i, it should have previously accessed it.

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B} \setminus \{s_m, t_m\}, \sum_{b_j \in \mathcal{B} \setminus \{b_i\}} (X_m^{i,j} - X_m^{j,i}) = 0 \tag{13}$$

In reality, (13) is an equality constraint between sums $\sum_{b_j \in \mathcal{B} \setminus \{b_i\}} X_m^{i,j}$ and $\sum_{b_j \in \mathcal{B} \setminus \{b_i\}} X_m^{j,i}$. The first sum being equal to 1 means there is a block b_j such that $X_m^{i,j} = 1$. The equality constraint imposes that AGV a_m leaves b_j in that case. In case one of the sums is null, the other is too, meaning that a_m does not visit b_j.

The following two sets of constraints define block and crossroad occupation time intervals. Since we proposed to use 5 time variables instead of the 7 defined in [5] and [6], most of the constraints are modified to fit the new variable choice.

Block occupation time intervals. Constraint (14) defines occupation time intervals of blocks crossed by AGVs. If AGV a_m crosses block b_j, its departure date is equal to its arrival date, to which we add the crossing duration of b_j and the parking time of AGV a_m on b_j if any.

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^A\mathcal{B}_i \setminus \{s_m\}, \\ (X_m^{i,j} = 1) \Rightarrow \beta_m^j = \alpha_m^j + \epsilon_m^j + d_j^b \tag{14}$$

This constraint is linearized using big M constraints (15) and (16).

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^{A}\mathcal{B}_i \setminus \{s_m\}, \\ \beta_j - \alpha_j - \epsilon_j - d^b_j \leq M(1 - X^{i,j}_m) \quad (15)$$

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^{A}\mathcal{B}_i \setminus \{s_m\}, \\ \alpha_j + \epsilon_j + d^b_j - \beta_j \leq M(1 - X^{i,j}_m) \quad (16)$$

When $X^{i,j}_m = 1$, the right-hand sides of both inequalities are equal to 0, which corresponds to the intended result of the implication in (14). However, when $X^{i,j}_m = 0$, both inequalities are trivial and don't affect solution feasibility, considering that we take a sufficiently great constant value M. This linearization method will be used for all constraints of this type presented in this model.

Now, as the origin blocks are excluded from constraint (14), it is important to add a constraint that verifies this condition for this type of blocks.

$$\forall\, a_m \in \mathcal{A}, \beta^{s_m}_m = \alpha^{s_m}_m + \epsilon^{s_m}_m + d^b_{s_m} \quad (17)$$

Additionally, the arrival date of a_m to s_m is trivially set to 0.

$$\forall\, a_m \in \mathcal{A}, \alpha^{s_m}_m = 0 \quad (18)$$

Following the same logic as in constraints (6) and (7), we can set the occupation time intervals of unused blocks as initial constraints to potentially improve solving efficiency. It is done by establishing the following implication between the value of Y^i_m and the values of α^i_m and ϵ^i_m :

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \quad (Y^i_m = 0) \Rightarrow \begin{cases} \alpha^i_m = 0 \\ \epsilon^i_m = 0 \end{cases} \quad (19)$$

To finish off this block of constraints, it is important to guarantee the logical succession of crossed blocks regarding occupation time intervals. This constraint is formulated as follows :

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^{A}\mathcal{B}_i, \\ (X^{i,j}_m = 1) \Rightarrow \alpha^j_m = \beta^i_m + d^c_{i_{c^j}} \quad (20)$$

Crossroad occupation time intervals. We start off by formulating the constraint that sets the arrival date of an AGV to a crossroad, which should be equal to the departure time from the previously crossed block :

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^{A}\mathcal{B}_i, \quad (X^{i,j}_m = 1) \Rightarrow \phi^{i_{c^j}}_m = \beta^i_m \quad (21)$$

Once the arrival time to the crossroad is defined, we need to formulate a constraint to calculate the departure time from said crossroad. This is achieved through the following implication :

$$\forall\, a_m \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, b_j \in {}^{A}\mathcal{B}_i, \\ (X^{i,j}_m = 1) \Rightarrow \psi^{i_{c^j}}_m = \phi^{i_{c^j}}_m + d^c_{i_{c^j}} \quad (22)$$

Ultimately, these last two constraints determine the occupation time intervals of each crossroad by the AGVs performing a set of given missions. However, these intervals could be pre-computed for unused crossroads in order to potentially improve solving efficiency. For that, we formulate the following constraint :

$$\forall\, a_m \in \mathcal{A}, \forall\, c_p \in \mathcal{C}, \quad (Z^p_m = 0) \Rightarrow \phi^p_m = 0 \quad (23)$$

Safety constraints This final block of constraints allows the routing engine to generate paths for the AGVs, while maintaining safety by eliminating all potential collisions. First, we need to formulate two constraints that determine the crossing order of a block (resp. crossroad) by a pair of AGVs, using the $B^i_{m,n}$ (resp. $C^p_{m,n}$) variables. We remind that $B^i_{m,n} = 1$ (resp. $C^p_{m,n} = 1$) means that AGV a_m crosses block b_i (resp. crossroad c_p) before AGV a_n:

$$\forall\, a_m \neq a_n \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \quad B^i_{m,n} + B^i_{n,m} = 1 \quad (24)$$

$$\forall\, a_m \neq a_n \in \mathcal{A}, \forall\, c_p \in \mathcal{C}, \quad C^p_{m,n} + C^p_{n,m} = 1 \quad (25)$$

These two constraints are used to make sure that, for every pair of AGVs using the same block (resp. crossroad), one and only one vehicle will cross before the other. Then, we impose the constraint that will prevent collisions between AGVs using the same block while performing their assigned missions. This is formulated as follows :

$$\forall\, a_m \neq a_n \in \mathcal{A}, \forall\, b_i \in \mathcal{B}, \forall\, c_p \in \mathcal{C}^i, \beta^i_m + \omega \sum_{j \in \overleftarrow{{}^{A}\mathcal{B}^p_i}} X^{j,i}_n \\ \leq \alpha^i_n + M.(4 - Y^i_m - Y^i_n - B^i_{m,n} - \sum_{j \in {}^{A}\mathcal{B}^p_i} X^{i,j}_m) \quad (26)$$

The purpose of the constraint is to make sure that, in the returned solution, if two AGVs use the same block, the departure date of one of them is never equal to the arrival date of the other. To better understand the formulation, it could be seen as a big M constraint that imposes the aforementioned date difference when three conditions are met :

(i) A pair of AGVs (a_m, a_n) use the same block b_i which translates to $Y^i_m = Y^i_n = 1$.
(ii) AGV a_m crosses block b_i before AGV a_n which translates to $B^i_{m,n} = 1$.
(iii) AGV a_m is leaving block b_i while AGV a_n is entering the same block, which translates to $\sum_{j \in {}^{A}\mathcal{B}^p_i} X^{i,j}_m = \sum_{j \in \overleftarrow{{}^{A}\mathcal{B}^p_i}} X^{j,i}_n = 1$.

When all these conditions are verified, the constraint is equivalent to $\beta^i_m + \omega \leq \alpha^i_n$, with ω being a strictly positive value that behaves similarly to M as detailed earlier.

We formulate the same collision avoidance constraint for crossroads, following exactly the same logic :

$$\forall\, a_m \neq a_n \in \mathcal{A}, \forall\, c_p \in \mathcal{C}, \\ \psi^p_m + \omega \leq \phi^p_n + M.(3 - Z^p_m - Z^p_n - C^p_{m,n}) \quad (27)$$

A complete ILP model was presented. It provides, for each AGV performing a unique assigned mission, a succession

of blocks and crossroads that should be followed to travel from the mission's origin to its destination. Additionally, the solved model returns, for each AGV, the arrival and departure dates to and from all the blocks and crossroads that constitute its route, and any potential parking durations in a block in order to avoid collisions. In the next section, we propose to perform numerical experiments using the formulated model and present an analysis of the provided solutions.

IV. Numerical experiments [†]

A. The FACT and FACT2 networks

For our experimental study, we consider the FACT and the FACT2 networks proposed in [6]. The FACT network (Fig. 1) is composed of 101 blocks linked through 36 crossroads. We also note that the network contains 6 docking stations at the bottom and 4 storage areas at the top. In the FACT2 network, two similar FACT networks are placed side by side. However, there is no detail in [6] on how to connect the two FACT networks, so we choose to add 3 additional blocks to link the external crossroads. Thus, the FACT2 network contains 205 blocks linked through 72 crossroads, 12 docking stations and 8 storage areas. In our experiments, the crossing durations are discrete values randomly chosen between 10 and 15 seconds for blocks and between 3 and 8 seconds for crossroads.

B. Experiment description

1) Instance generation: In order to test the performance of the ILP model on the FACT2 network, we create an instance generator that works as follows. To generate a mission, a free block is firstly chosen within a docking station/storage area. It represents the origin block of the mission. Then, a destination block is randomly and exclusively chosen within the four nearest docking stations/storage areas at the opposite side of the origin block. This restriction is justified by the fact that the service provider would make the same realistic choices when assigning transportation orders to the AGVs. Additionally, we restrict the instances to contain a maximum of two (resp. three) mission origin blocks from the same docking station/storage area, for examples involving up to 16 AGVs (resp. examples involving 18 AGVs). This is done in order to provide a realistic distribution of the missions within the FACT2 network. The proposed instance generator is used to create two types of instances, *X_u* and *X_d*, where *X* represents the number of AGVs, *u* indicates that the vehicles are following the same *unique* direction between the docking stations and the storage areas or vice versa, and *d* indicates that missions are performed following both directions between the two. For our experimental study, we generate 50 instances of each type *X_u* and *X_d*, with $X \in \{4, 8, 10, 12, 16, 18\}$. This is done in order to enhance the genericity and the quality of the numerical experiments performed in [5] and [6]. The previous studies did not comprehensively test the model across a wide range of instances, including those with 10, 12, and 18 AGVs.

[†]The networks and the instances used for the experiments can be found at: https://gitlab.laas.fr/roc/ghassen-cherif/integer-linear-programming-for-automated-guided-vehicles-path-planning-in-container-terminals

TABLE I
Results of the proposed ILP model

Instances	% Opt	Avg Cost	Avg Runtime (s)
4_u	100%	482.7	9.81
4_d	100%	498.4	12.03
8_u	82%	902.3	388.39
8_d	86%	1019.8	234.93
10_u	52%	1167.8	739.30
10_d	62%	1288.9	1123.33
12_u	6%	1487.9	1824.06
12_d	12%	1392.3	1429.17
16_u	0%	2078.3	3600
16_d	0%	2235.8	3600
18_u	0%	2405.8	3600
18_d	0%	2627.1	3600

2) Experimental setup: The computational experiments are conducted on a computer cluster composed of 11 nodes, totaling 444 cores and 2 TB of RAM, with 32 GB of RAM allocated for this experimental study. The various instances are solved using IBM CPLEX version 22.1.1, accessed via the DOcplex modeling library for optimization using Python 3.8.10. The runtime limit is set to one hour. For each set of 50 instances of the same type, we track the percentage of instances solved to optimality, their average objective value and their average runtime. It is important to note that the aim here is to assess the maximum number of AGVs the problem can be solved for, rather than to compare the quality of the obtained solutions to those of [5] and [6].

C. Results and analysis

The results of our experimental campaign are shown in Table I. The first column indicates the instance sets, while the second presents the percentage of the 50 instances solved to optimality. The third and fourth columns give, respectively, the average cost and the average runtime (over the instances solved to optimality for instances with 4, 8, 10 and 12 AGVs). For instances involving 16 AGVs or more, the average cost is calculated over the feasible solutions obtained, as no instance was optimally solved. Their results are separated from the rest of the table using a dashed line. For instances of both types *X_u* and *X_d* involving 4 AGVs, the solver is able to optimally solve 100% of the considered examples. However, this percentage gradually diminishes as the number of AGVs increases, reaching only 6% (resp. 12%) for the 12_u (resp. 12_d) instance set, while no instance involving 16 or 18 AGVs was optimally solved. We note that *X_d* instances present, on average, a 31.02% advantage over *X_u* instances regarding the %Opt metric. It's also important to highlight that if an optimal solution isn't obtained, it means that either the instance remains unsolved or a feasible solution is provided within the maximum runtime for instances with up to 18 AGVs. In terms of average runtime, we observe that it gradually increases according to the number of AGVs, going from 9.81 s (resp. 12.03 s) for instances 4_u (resp. 4_d) to 1824.06 s (resp. 1429.17) for instances 12_u (resp. 12_d), with no discernible advantage for an instance type over the other. In

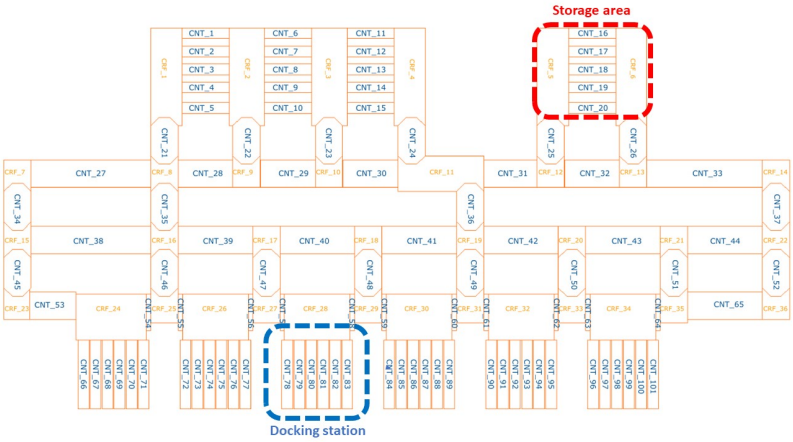

Fig. 1. The FACT network [6]

comparison to the numerical experiments conducted in [6] and [5], our experimental campaign utilized a larger set of instances. This expansion adds depth and significance to the results obtained from the numerical study. Additionally, we were able to provide solutions for instances involving up to 18 AGVs, whereas no solutions are provided for instances involving more than 8 AGVs in [6], when using the same solution method. Based on these results, we can conclude that the proposed ILP approach is very efficient when solving smaller instances of the 1MA problem, but is increasingly challenged as instances' sizes are increased. As seen earlier in the introduction section, this struggle is very common when using IP and MIP approaches to solve combinatorial optimization problems, usually due to the complexity and combinatorial explosion aspects that characterize this type of problem.

V. Conclusion

Automating seaport operations has been a major lever for improving their efficiency and competitiveness. This is particularly true for the loading and unloading of ships, where autonomous vehicles are increasingly being used for transport operations within container terminals. In this work, we proposed an ILP approach for the optimal collision-free path planning of AGVs in container terminals organized in blocks and crossroads. Thorough numerical experiments involving various instances of the studied problem showed that, while the method is efficient in solving small instances, it struggles to scale for larger ones. Based on this study, several avenues of research can be explored. These may involve investigating decomposition methods to tackle the combinatorial nature of the problem, devising heuristic techniques for handling larger instances, or exploring alternative approaches such as constraint programming. In our future works, these methods will be implemented and comprehensive comparative analyses of their effectiveness will be carried out. Another research perspective would be to investigate the more realistic problem of multiple missions per AGV. For that, we intend to explore heuristic methods where the problem is decomposed into multiple instances of the 1MA problem, which will be sequentially solved using the studied ILP model.

References

[1] D. Liu, X. Wu, A. Kulatunga, and G. Dissanayake, "Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*. IEEE, 2006, pp. 1–6.

[2] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26 022–26 035, 2018.

[3] S. A. Fayazi and A. Vahidi, "Mixed-Integer Linear Programming for Optimal Scheduling of Autonomous Vehicle Intersection Crossing," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 287–299, 2018.

[4] T. Miyamoto and K. Inoue, "Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems," *Computers & Industrial Engineering*, vol. 91, pp. 1–9, 2016.

[5] N. Danloup, B. Trouillet, T. Bourdeaud'huy, and A. Toguyéni, "Optimisation de tournées de véhicules dans un environnement portuaire," in *GOL 2018 - International Conference on Logistics Operations Management, Apr 2018, Le Havre, France.*, 2018.

[6] N. Danloup, B. Trouillet, A. Toguyéni, and T. Bourdeaud'Huy, "Gestion d'une flotte de véhicules automatisés dans un environnement portuaire," in *MSR 2019-12ème Colloque sur la Modélisation des Systèmes Réactifs, Nov 2019, Angers, France*, 2019.

[7] 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*, vol. 38, no. 5, pp. 876–888, 2011.

[8] G. Cherif, B. Trouillet, and A. K. Toguyeni, "Modeling and routing problems of automated port using T-TPN and Beam search," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1. IEEE, 2022, pp. 1201–1206.

[9] T. Nishi and Y. Tanaka, "Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 5, pp. 1230–1243, 2012.

[10] M. Harrison, Z. Yang, T. Nguyen, S. Kavakeb, J. Wang, and S. Bonsall, "A TOPSIS method for vehicle route selection in seaports—a real case analysis of a container terminal in North West Europe," in *2015 International Conference on Transportation Information and Safety (ICTIS)*. IEEE, 2015, pp. 599–606.

[11] M. Croucamp and J. Grobler, "Metaheuristics for the robot part sequencing and allocation problem with collision avoidance," in *Progress in Artificial Intelligence: 20th EPIA Conference on Artificial Intelligence, EPIA 2021, Virtual Event, September 7–9, 2021, Proceedings 20*. Springer, 2021, pp. 469–481.