




Article

Comprehensive MILP Formulation and Solution for Simultaneous Scheduling of Machines and AGVs in a Partitioned Flexible Manufacturing System

Cheng Zhuang ¹ , Jingbo Qu ¹ , Tianyu Wang ¹, Liyong Lin ², Youyi Bi ¹  and Mian Li ^{3,*} 

¹ UM-SJTU Joint Institute, Shanghai Jiao Tong University, Shanghai 200240, China; mike.z.c@sjtu.edu.cn (C.Z.); qujingbo@sjtu.edu.cn (J.Q.); gunnerwang27@sjtu.edu.cn (T.W.); youyi.bi@sjtu.edu.cn (Y.B.)

² Contemporary Amperex Technology Co., Ltd., Fujian 352100, China; linly02@catl.com

³ Global Institute of Future Technology, Shanghai Jiao Tong University, Shanghai 200240, China

* Correspondence: mianli@sjtu.edu.cn

Abstract: This paper proposes a comprehensive Mixed-Integer Linear Programming (MILP) formulation for the simultaneous scheduling of machines and Automated Guided Vehicles (AGVs) within a partitioned Flexible Manufacturing System (FMS). The main objective is to numerically optimize the simultaneous scheduling of machines and AGVs while considering various workshop layouts and operational constraints. Three different workshop layouts are analyzed, with varying numbers of machines in partitioned workshop areas A and B, to evaluate the performance and effectiveness of the proposed model. The model is tested in multiple scenarios that combine different layouts with varying numbers of workpieces, followed by an extension to consider dynamic initial conditions in a more generalized MILP framework. Results demonstrate that the proposed MILP formulation efficiently generates globally optimal solutions and consistently outperforms a greedy algorithm enhanced by A*-inspired heuristics. Although computationally intensive for large scenarios, the MILP's optimal results serve as an exact benchmark for evaluating faster heuristic methods. In addition, the study provides practical insight into the integration of AGVs in modern manufacturing systems, paving the way for more flexible and efficient production planning. The findings of this research are expected to contribute to the development of advanced scheduling strategies in automated manufacturing systems.

Keywords: FMS; MILP; simultaneous scheduling; production optimization; task allocation; greedy algorithm



Academic Editors: Xianjian Jin and Xin Xia

Received: 12 May 2025

Revised: 5 June 2025

Accepted: 11 June 2025

Published: 13 June 2025

Citation: Zhuang, C.; Qu, J.; Wang, T.; Lin, L.; Bi, Y.; Li, M.

Comprehensive MILP Formulation and Solution for Simultaneous Scheduling of Machines and AGVs in a Partitioned Flexible Manufacturing System. *Machines* **2025**, *13*, 519. <https://doi.org/10.3390/machines13060519>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The manufacturing industry faces increasing pressure to respond to dynamic market conditions characterized by fluctuating consumer demand and evolving product specifications. Manufacturers must rapidly adapt to product variations, new market trends, and increasingly complex customer requirements. As a result, manufacturing systems must evolve to emphasize flexibility, adaptability, and rapid response.

Flexible Manufacturing Systems represent an effective solution to these challenges, providing the necessary flexibility to accommodate changing demands and diverse product variations. Unlike traditional manufacturing systems, FMSs do not rely on dedicated machines; instead, machines are designed to be multi-functional and can perform multiple operations. This flexibility allows FMSs to rapidly adjust to product changes and ensure efficient resource utilization. Despite their advantages, the implementation of FMSs is usually

associated with a higher capital investment due to the need for multi-functional machinery and flexible transportation systems. Effective management of these complicated systems requires strategic planning, efficient scheduling, and optimized operational strategies.

Extensive research has been conducted on FMS scheduling, with numerous methods proposed to optimize machine and AGV operations. Most existing studies focus on machine scheduling or AGV scheduling independently. The simultaneous scheduling (SS) of both remains unexplored despite its potential to improve overall system efficiency.

In recent decades, several papers [1–9] have been published on the solution of SS problems, using approximate solution methods or exact methods.

Among approximate optimization strategies, Abdelmaguid et al. devised a hybrid genetic algorithm in which a deterministic assignment rule dispatched transport tasks [1]. Hurink and Knust reformulated the problem as a disjunctive graph model and investigated it through a tabu-search method, although their study considered only a single vehicle [10]. Adopting a task-oriented viewpoint, Deroussi et al. proposed an innovative chromosome representation and fortified local search with three complementary metaheuristics [11]. Gnanavel Babu et al. applied differential evolution to refine the operation vector and incorporated a repair operator to restore individuals who violated the precedence relations [12]. Lacomme et al. constructed a non-oriented disjunctive graph and incorporated two specialized heuristics within a memetic framework to determine carrier selection and task sequencing, respectively [13]. Extending the graph-based paradigm, Zhang et al. introduced processing, transport, and buffer nodes, coupling this structure with a modified shifting bottleneck heuristic to dispatch AGV operations [14]. Zheng et al. designed a two-layer encoding for tabu-search that persistently delivered state-of-the-art solutions [15]. Using Colored Petri nets, Baruwa and Piera crafted a composite heuristic capable of completing resource allocations [3]. Nouri et al. presented a multi-agent hybrid in which a genetic algorithm performed global exploration while agents guided the direction of a tabu-search phase, thus intensifying search efficiency [16]. Gao et al. used a hybrid genetic algorithm with a large neighborhood search (GA-LNS) to optimize the routing of heterogeneous AGV fleets in the routing of green vehicles. This method minimized transportation distance and energy consumption in FMS [6].

Approximate solution methods exhibit three critical limitations in partitioned FMS scheduling scenarios. First, their heuristic rules and stochastic exploration mechanisms inherently lack an optimality guarantee when handling dynamic routing constraints across partitioned workshops. Second, performance consistency degrades significantly for novel problem configurations, particularly in multi-shop coordination scenarios where prior benchmarks remain unavailable. The absence of known optimal solutions further complicates the quantitative evaluation of the quality of the solution. Third, metaheuristics demonstrate computational unpredictability that makes them unsuitable for real-time industrial applications requiring deterministic response time.

These fundamental limitations underscore the necessity for exact mathematical programming approaches. Unlike heuristic-based methods, mathematical optimization frameworks establish rigorous performance baselines through provably optimal solutions while systematically encoding complex spatial-temporal constraints. The deterministic methodology proves particularly valuable for validating emerging scheduling scenarios in a partitioned FMS environment. The earliest exact treatment of the SS problem is due to Khayat et al., who devised both a mixed integer programming (MIP) formulation and a constraint programming model [17]. Caumond et al. later focused on an FMS with a single AGV by means of a MILP that explicitly handles loading, unloading, and limited machine buffers [18]. To explore shop-floor flexibility, Çemal, et al. developed MILPs for the flexible job-shop scheduling problem (FJSP) and its extension with process plan flexibility [19].

Zheng, et al. dealt with the SS problem by formulating a joint machine–AGV MIP in which the first and last trips of each vehicle are modeled separately, although the scheme remains tractable only for small instances [15]. Huang redesigned the index sets within a similar MILP, markedly accelerating the exact solution of SS cases [20]. Fontes and Homayouni employed the GUROBI solver to shrink run times for SS, although their model does not output an explicit AGV transport task order [9]. Abderrahim, et al. embedded a variable-neighborhood search inside an MIP and could certify optimal SS plans for part of the test bed [21]. Yao, et al. decomposed the job shop with mobile robots into processing order, transport assignment, and transport sequencing, then recast the three coupled decisions in a novel MILP that directly targets SS [22]. Boccia, et al. incorporated battery depletion and recharging into SS by pairing a MILP with a three-stage metaheuristic to minimize makespan for multiple AGVs [8]; these researchers later fused a MILP with a genetic algorithm to downsize the AGV fleet while respecting identical energy limits, still within an SS framework [23]. Addressing energy-aware SS in flexible job shops, Meng, et al. proposed a bi-objective MILP that balances makespan and power consumption [24], whereas Liu, et al. offered two MILPs for integrated process planning and scheduling (IPPS), covering both fully and semi-flexible SS scenarios [25]. For hybrid flow shops, Amirteimoori, et al. coupled an MILP with a two-stage parallel heuristic to generate collision-free SS solutions for moderate problem sizes [26]. Guo, et al. developed an MILP to handle conflict-free routing in multi-AGV systems with partial charging capabilities. Their model manages route conflicts efficiently in constrained layouts, providing precise scheduling solutions for small instances [27]. Wang, et al. propose four different modeling approaches based on MILP (sequence-based, position-based, time-based, and adjacent sequence-based) to address the Three-Stage Remanufacturing System Scheduling Problem (3T-RSSP) [28]. However, these models focus primarily on job sequencing and machine assignment without considering AGV routing or transportation logistics.

Through the aforementioned research, it becomes evident that existing models often target simplified workshop production scenarios and make numerous idealized assumptions (neglecting transportation time, unlimited machine buffer) to reduce model complexity, thereby facilitating easier solving and improving solution efficiency. Moreover, most existing studies on FMS scheduling consider a single, unpartitioned workshop and assume static initial conditions, whereas, in reality, most factories have different steps of a production line distributed across multiple workshops. The concept of a partitioned FMS—where the shop floor is divided into distinct areas requiring dedicated inter-area material transport—introduces additional complexity that has not been comprehensively addressed in previous scheduling models. Furthermore, real-world manufacturing often involves dynamic initial conditions, such as machines already processing jobs or jobs carried over from prior shifts, which are rarely modeled in classical MILP formulations. Motivated by these gaps, this paper aims to address this limitation by providing a comprehensive MILP mathematical modeling of common real-world industrial production scenarios and obtaining globally optimal solutions through numerical calculation.

To the best of our knowledge, this is the first work to consider an FMS divided into separate workshop areas within a single MILP scheduling framework, incorporating inter-area transport requirements. Furthermore, our model is extended to handle dynamic initial conditions (ongoing jobs and occupied machines at the start of the schedule), thereby increasing its practical applicability to rolling-horizon production planning. In this study, we present the formulation of the MILP model and demonstrate its application across various scenarios. We also compare the MILP's performance with that of a heuristic (greedy algorithm enhanced by A*-like search) to illustrate the benefits and limitations of exact optimization in this context. This model is of greater significance for guiding

the improvement of production efficiency in real factories. Due to its alignment with real production scenarios, the solutions of this model can theoretically be directly applied to digital simulation platforms for optimization. This facilitates the seamless transition of real-world scenarios to simulation environments, enabling better integration with digital twins and rendering digital production scenarios more lifelike.

In summary, the key contributions of this work are: (1) a comprehensive MILP model for integrated machine–AGV scheduling in a partitioned FMS environment, capturing realistic layout constraints; (2) an extension of the model to accommodate dynamic initial conditions (partial job completion and machine occupation from previous operations), which is rarely addressed in the literature; and (3) a thorough evaluation against a heuristic approach, establishing the MILP model’s optimal solutions as a benchmark to assess and guide the development of faster scheduling heuristics.

The remaining sections of this paper are organized as follows. Section 2 provides the related background, such as scenario description, for the joint production and transportation scheduling problem in Flexible Manufacturing Systems. In Section 3, the problem is described in detail, and after introducing the used notation, the mixed-integer linear programming model is developed. The results are discussed in Section 4, followed by the conclusions and possible future works in Section 5.

2. Background

2.1. Flexible Manufacturing System (FMS)

Flexible Manufacturing Systems (FMSs) are automated production systems designed to handle a variety of products with minimal manual intervention, offering high flexibility in response to varying production demands. FMSs typically integrate multiple machines and automated transport systems, all coordinated by a centralized computer control system, allowing rapid adjustments to changes in the types, volume, and process sequences of the product [29]. FMSs enable rapid changeovers, efficient resource utilization, and reduced lead times, allowing manufacturers to respond quickly to market demands. These advantages make FMSs ideal for industries that require customization and responsiveness, such as automotive and electronics manufacturing.

The core of an FMS lies in its ability to efficiently manage different production tasks by dynamically allocating resources, such as machines and transport vehicles, to handle variations in the product flow. For example, an FMS can adapt to different product designs and volumes without the need for extensive retooling, thus reducing lead times and production costs [30]. However, the complexity of scheduling and coordinating resources in FMSs, especially with the introduction of Automated Guided Vehicles (AGVs), presents significant challenges that require sophisticated scheduling algorithms and real-time optimization techniques [2].

2.2. Partitioned FMS

Partitioned Flexible Manufacturing Systems aim to improve the modularity and manageability of the entire manufacturing system by dividing it into several independent manufacturing units or regions. This division is usually based on factors such as product families, process similarity, or physical location [31]. Performing simultaneous scheduling of machines and AGVs in a partitioned FMS, compared to a traditional centralized FMS, introduces new challenges and considerations.

First, partitioning limits the activity range of AGVs. In a centralized FMS, AGVs can generally move freely throughout the system, while in a partitioned FMS, AGVs may be restricted to specific partitions or need to follow specific protocols to move between

different partitions. This requires considering the transfer time and availability of AGVs between different partitions in the scheduling model [32].

Second, there are material flow dependencies between partitions. Semi-finished products produced in one partition may need to be transported to another partition for subsequent processing. This cross-partition material flow needs to be coordinated during the scheduling process to avoid a production bottleneck in one partition that affects the normal operation of other partitions. Therefore, the scheduling model needs to be able to handle inter-partition material transportation requests and priorities.

In addition, scheduling decisions within each partition will also be affected by other partitions. For example, the processing completion time of a machine in one partition will directly affect the time when the workpiece needs to be transported to the next partition. Therefore, a coordination mechanism must be developed so that scheduling decisions within each partition can be consistent with the overall goals of the system.

2.3. Mixed-Integer Linear Programming

Mixed-Integer Linear Programming (MILP) is a group of mathematical optimization formulations widely used for problems that require decision variables to be both continuous and discrete. MILP models are particularly valuable in areas like production scheduling, logistics, and finance, where exact solutions can produce optimal resource allocation and sequencing. The MILP model is defined as an optimization problem where both the objective function and constraints are linear, and some variables are constrained to take integer values [33].

A standard MILP formulation is given as follows:

$$\text{minimize } Z = \sum_{j \in J} c_j x_j \quad (1)$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \leq b_i, \quad \forall i \in I \quad (2)$$

$$x_j \in \mathbb{Z} \quad \text{or} \quad x_j \in \mathbb{R}, \quad \forall j \in J \quad (3)$$

where:

- Z is the objective function to be minimized,
- c_j are the coefficients for each decision variable x_j in the objective function,
- a_{ij} represents the coefficients in the constraint matrix,
- b_i are constants defining the bounds of each constraint, and
- I and J represent the sets of constraints and decision variables, respectively.

This structure allows MILP to model complex systems in which binary or integer constraints reflect practical restrictions on resources, tasks, or operational states [34,35]. MILP models are precise and can provide optimal solutions for small- to medium-sized instances. However, they are computationally intensive for larger instances, often requiring heuristic or metaheuristic approaches to approximate solutions within a reasonable time frame [35].

2.4. Greedy Algorithm

The Greedy Algorithm is an iterative approach used to solve optimization problems by making a sequence of locally optimal choices at each step, aiming to reach an optimal solution. This algorithm is particularly effective for problems where each local decision does not affect future decisions, making it efficient in terms of time complexity. Greedy algorithms have been commonly applied to problems such as minimum spanning trees,

shortest paths, and scheduling [36]. For example, in the context of the Knapsack Problem, a greedy approach would involve selecting items based on the highest value-to-weight ratio until the knapsack reaches its capacity. A general illustration of the Greedy Algorithm is shown in Figure 1.

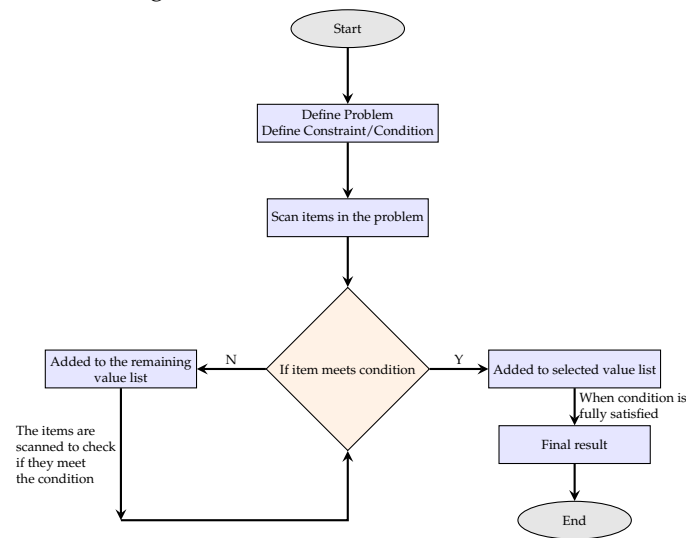


Figure 1. Flowchart illustrating the Greedy Algorithm process in solving a problem based on constraints and conditions.

The greedy algorithm is widely used due to its simplicity and computational efficiency, making it suitable for a variety of problems, particularly those that satisfy the greedy choice property and optimal substructure [37]. However, despite its efficiency, the greedy approach does not always guarantee an optimal solution, as its focus on local decisions can result in suboptimal solutions at the global level [38]. In this paper, the greedy algorithm will be used as a benchmark to illustrate the effectiveness of the proposed MILP model.

3. Problem Definition and Formulation

For an FMS, multiple production tasks and AGVs compete for limited resources such as space and equipment. Different resource allocations result in different completion times. Optimization of the scheduling problem in an FMS requires careful consideration of various factors. It involves determining the optimal order in which the workpieces are processed, assigning tasks to the AGVs in an efficient manner, and planning the AGV paths to minimize traveling time and resource conflicts. By addressing these challenges and considering process constraints and resource limitations, this research aims to achieve the shortest possible completion time for manufacturing tasks. By minimizing the processing completion time, the proposed approach can enhance the overall efficiency and productivity of the system.

In this study, a simplified scenario and a general scenario are used to capture the different layers of operational complexity encountered in partitioned flexible manufacturing systems. Both scenarios share the same job sets, number of machines, and AGVs. Their difference is the shop-floor state at the beginning of optimization. In the simplified scenario, every machine and buffer are empty at time zero, so production begins from an idle state. In the general scenario, several machines may already be processing residual jobs carried over from a previous batch, yielding a random initial occupation pattern. This contrast isolates the influence of initial conditions and allows the model to be evaluated both for first-time scheduling and for rescheduling after an unexpected interruption without altering the underlying problem scale.

3.1. Simplified Workshop Scenario

As shown in Figure 2, the simplified workshop scenario is divided into two distinct regions, each region being assigned an AGV for task scheduling. Region 1 comprises a loading station S, processing area A, and processing area B, while Region 2 comprises processing area B and an unloading station E.

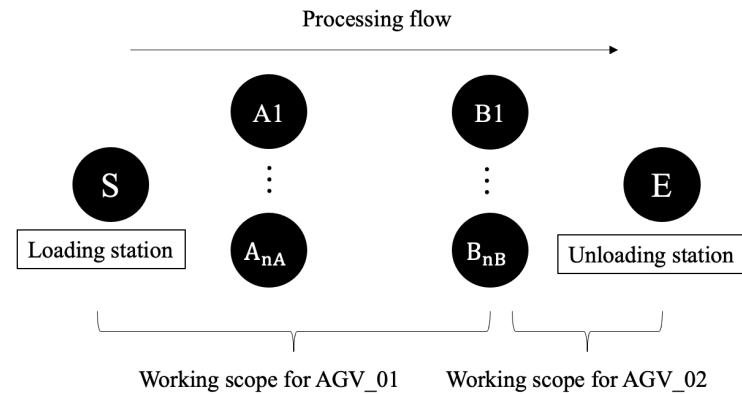


Figure 2. Layout of simplified workshop.

The objective of AGV scheduling in this scenario is to transport all materials from the loading station through processing areas A and B and to deliver them to the unloading station. In this scheduling scenario, all workpieces are initially stored at the loading station. Each workpiece is then transported to any available machine in Area A for execution of the first processing operation, followed by transportation to any available machine in Area B for the second processing operation. Upon completion of the processing operations, the workpiece is transported to the unloading station, where the entire scheduling process is concluded.

There are several important considerations in this scenario:

1. Processing areas do not have buffer zones associated with individual machines. This means that materials cannot be temporarily stored within the processing areas and that a continuous flow of materials is required to maintain efficient operations.
2. Processing area B is located on the boundary between Region 1 and Region 2. It is imperative to ensure that machines in this area do not allow simultaneous occupancy by two AGVs. This constraint is crucial to avoid conflicts and potential disruptions in the workflow.

These considerations highlight the need for careful planning and scheduling in the workshop setting. The absence of buffer zones in the processing areas, which is very common due to the limited space in the workshop, requires effective coordination and management of the material flow to minimize delays and optimize production efficiency. Additionally, the constraint at the boundary between regions underscores the importance of proper sequencing and allocation of tasks to AGVs, ensuring that conflicts are avoided and the overall system performance is maximized.

3.1.1. MILP Formulation

This section constructs the MILP model and presents the following notation, parameters, and decision variables to facilitate the description of this model.

However, prior to model development, it is necessary to define the conditions and limitations considered in the model. Thus, the following conditions are defined for model development:

- All AGVs have unit-load capacity.
- Each machine processes only one product at any given moment.

- AGVs and machines operate continuously without failure.
- There are no traffic problems, collisions, deadlocks, or conflicts.
- Each job has a fixed setup time that is not sequence-dependent. This fixed setup time is included in the job's total processing time
- The vehicles maintain a constant velocity and can only move forward.
- The distances between machines, as well as the distances from the L/U to the machines, are predefined and known.
- AGVs remain at L/U until they receive dispatch instructions.

Notation

i, j	job indices, $i, j \in N = \{1, 2, \dots, n\}$, where n is the total number of jobs.
p	indices for operations, $p \in P = \{1, 2\}$
O_{ip}	the p th operation for job i
k, l	indices for machines, $k, l \in K = S \cup A \cup B \cup E$. $S = \{0\}$, $A = \{1, 2, \dots, nA\}$, $B = \{nA + 1, \dots, nA + nB + 1\}$, $E = \{nA + nB + 2\}$, where nA is the total number of machines in area A, nB is the total number of machines in area B.
Ta_{ip}	the loaded task of AGV01 transporting job i to perform O_{ip} .
Tae_{ip}	AGV01's unloaded trip to pick up job i for completing Ta_{ip}
Tb_i	the loaded task of AGV02 transporting job i from B to E.
Tbe_i	AGV02's unloaded trip to pick up job i for completing Tb_i

Parameters

t_{up} / t_{down}	upload/download time
t_A	processing time in area A (setup time included)
t_B	processing time in area B (setup time included)
$t_{k,l}$	traveling time from machine k to machine l
M	a large positive number

Variables

All the following variables are decision variables in the MILP model. The time-related variables (start times and completion times, beginning with ST and CT) are continuous decision variables representing scheduling timestamps (in seconds). The binary variables (e.g., those beginning with x , f , l , and v) are 0–1 decision variables indicating assignment and sequencing decisions.

$STae_{ip}$	start time of Tae_{ip} (AGV1's empty trip to pick up job i for task Ta_{ip}).
$CTae_{ip}$	completion time of Tae_{ip} .
STa_{ip}	start time of Ta_{ip} (AGV1's loaded trip transporting job i for operation O_{ip}).
CTa_{ip}	completion time of Ta_{ip} .
$STbe_i$	start time of Tbe_i (AGV2's empty trip to pick up job i for task Tb_i).
$CTbe_i$	completion time of Tbe_i .
STb_i	start time of Tb_i (AGV2's loaded trip transporting job i from Area B to the unloading station).
CTb_i	completion time of Tb_i .
xO_{ipk}	$\begin{cases} 1 & \text{if } O_{ip} \text{ is processed on machine } k, \text{ where } k \in A \cup B \\ 0 & \text{otherwise} \end{cases}$
fa_{ip}	$\begin{cases} 1 & \text{if } Ta_{ip} \text{ is AGV01's first transportation task} \\ 0 & \text{otherwise} \end{cases}$

$$\begin{aligned}
la_{ip} & \begin{cases} 1 & \text{if } Ta_{ip} \text{ is AGV01's last transportation task} \\ 0 & \text{otherwise} \end{cases} \\
fb_i & \begin{cases} 1 & \text{if } Tb_i \text{ is AGV02's first transportation task} \\ 0 & \text{otherwise} \end{cases} \\
lb_i & \begin{cases} 1 & \text{if } Tb_i \text{ is AGV02's last transportation task} \\ 0 & \text{otherwise} \end{cases} \\
va_{ipjq} & \begin{cases} 1 & \text{if } Ta_{jq} \text{ immediately follows } Ta_{ip} \text{ on AGV01, where } p, q \in P \\ 0 & \text{otherwise} \end{cases} \\
vb_{ij} & \begin{cases} 1 & \text{if } Tb_j \text{ immediately follows } Tb_i \text{ on AGV02} \\ 0 & \text{otherwise} \end{cases} \\
xa_{ijk} & \begin{cases} 1 & \text{if job } j \text{ is processed after job } i \text{ on machine } k, \text{ where } k \in A \\ 0 & \text{otherwise} \end{cases} \\
xb_{ijl} & \begin{cases} 1 & \text{if job } j \text{ is processed after job } i \text{ on machine } l, \text{ where } l \in B \\ 0 & \text{otherwise} \end{cases} \\
C_{max} & \text{makespan}
\end{aligned}$$

The MILP model for the simplified workshop scenario is given below.

3.1.2. Objective Function

$$\text{Minimize } C_{max} \quad (4)$$

The objective function stated in Equation (4) is to minimize the makespan.

3.1.3. Constraints

The above objective function is subject to several different categories of constraints.

Resource Constraints

These constraints govern the allocation of machines and AGVs, ensuring that each resource can be assigned only to a single task at a time. In other words, no individual workpiece can occupy multiple machines in the same region, and each AGV is restricted to exactly one initial task and one terminal task. In doing so, the formulation captures the limitation in the real world that machines and vehicles cannot engage in parallel usage for the same job.

$$\sum_{k \in A} xO_{ik} = \sum_{l \in B} xO_{il} = 1, \sum_{k \in B} xO_{ik} = \sum_{l \in A} xO_{il} = 0. \quad \forall i \in N \quad (5)$$

$$\sum_{i \in N} fa_{i1} = \sum_{i \in N} la_{i2} = \sum_{i \in N} fb_i = \sum_{i \in N} lb_i = 1 \quad (6)$$

$$\sum_{i \in N} fa_{i2} = \sum_{i \in N} la_{i1} = 0 \quad (7)$$

Constraint (5) ensures that in each region, each workpiece can be assigned to only one machine. Constraints (6) and (7) ensure that each AGV can have only one first task and one last task. If an AGV were allowed multiple first or last tasks, it would imply that the same vehicle is attempting to initiate or conclude multiple transport operations simultaneously, which is impossible in real-world scenarios. By restricting each AGV to a single first and last task, the model enforces a coherent, continuous chain of operations. Each vehicle has a

unique starting point and a unique endpoint, thus ensuring logical consistency and better reflecting how AGVs are actually utilized on the shop floor.

Precedence Constraints

Precedence constraints enforce the logical ordering of tasks along an AGV's route. They stipulate that if a specific transportation task is not the first (or last), it must be immediately preceded (or succeeded) by another task. Consequently, the operational sequence of each AGV is made continuous, and there are no gaps or overlaps, corresponding to the practical requirement that every transport operation must be part of a coherent path from start to finish.

$$fa_{i1} + \sum_{j \in N} \sum_{q \in P} va_{jq i1} = 1, \forall i \in N \quad (8)$$

$$fa_{i2} + va_{i1 i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{jq i2} = 1, \forall i \in N \quad (9)$$

$$la_{i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{i2 jq} = 1, \forall i \in N \quad (10)$$

$$la_{i1} + va_{i1 i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{i1 jq} = 1, \forall i \in N \quad (11)$$

$$fb_i + \sum_{j \in N \setminus \{i\}} vb_{ji} = 1, \forall i \in N \quad (12)$$

$$lb_i + \sum_{j \in N \setminus \{i\}} vb_{ij} = 1, \forall i \in N \quad (13)$$

Constraints (8) and (9) indicate that if Ta_{ip} is not the first task, it must be immediately preceded by another task. Constraints (10) and (11) indicate that if Ta_{ip} is not the final task, it must be immediately followed by another task. Constraint (12) indicates that if Tb_i is not the first task, it must be immediately preceded by another task. Constraint (13) indicates that if Tb_i is not the final task, it must be immediately followed by another task.

Timing and Synchronization Constraints

The following constraints handle the timing aspects of the schedule, ensuring synchronization between different operations and transportation tasks. They also ensure that tasks are completed within their specified time windows.

$$Cmax \geq CTb_i \quad \forall i \in N \quad (14)$$

$$STa_{i1} \geq CTae_{i1} + t_{down} \quad (15)$$

$$CTa_{i1} \geq STa_{i1} + \sum_{k \in A} xO_{i1k} t_{0,k} + t_{up} \quad (16)$$

$$STa_{i2} = \max(CTae_{i2}, CTa_{i1} + t_A) + t_{down} \quad (17)$$

$$CTa_{i2} \geq STa_{i2} + \sum_{k \in A} \sum_{l \in B} xO_{i1k} xO_{i2l} t_{k,l} + t_{up} \quad (18)$$

$$STb_i = \max(CTbe_i, CTa_{i2} + t_B) + t_{down} \quad (19)$$

$$CTb_i \geq STb_i + \sum_{l \in B} xO_{i2l} t_{l,40} + t_{up} \quad (20)$$

$$CTae_{i1} \geq STae_{i1} + \sum_{j \in N} \sum_{q \in P} \sum_{c \in A \cup B} va_{jq i1} xO_{jqc} t_{c,S} \quad \forall i \in N \quad (21)$$

$$CTae_{i2} \geq STae_{i2} + \sum_{j \in N} \sum_{q \in P} \sum_{c \in A \cup B} \sum_{k \in A} va_{jq i2} xO_{jqc} xO_{i1k} t_{c,k} \quad \forall i \in N \quad (22)$$

$$CTbe_i \geq STbe_i + \sum_{l \in B} xO_{i2l} t_{E,l} \quad (23)$$

$$STae_{jq} + M(1 - va_{ipjq}) \geq CTa_{ip} \quad \forall p, q \in P \quad (24)$$

$$STbe_j + M(1 - vb_{ij}) \geq CTb_i \quad (25)$$

Constraint (14) describes that the makespan should not be less than the completion time of the last operation of any workpiece.

Constraint (15) indicates that the start time ST_{ai1} for the loaded trip of AGV1 from station S to Area A must equal the completion time CT_{aei1} of its previous movement unloaded to S plus unloading time t_{down} . In essence, the AGV cannot begin transporting a new workpiece from S to A until it has fully arrived at S (while unloaded) and finished the necessary loading operations.

Constraint (16) indicates that the completion time of the AGV1 load task from S to A is its start time plus the transport time and the time taken to load the material on the machine in Area A .

The start time of the AGV1 load task from A to B is influenced by two factors: the time when the unloaded AGV1 arrives at the machine in Area A , where the material is located, and the completion time of the material processing. If AGV1 arrives first but the material has not yet been processed, AGV1 must wait until the material processing is completed and then begin the loading task. Conversely, if the material processing is completed before AGV1 arrives, the start time is determined by AGV1's arrival. This physical interpretation is encapsulated by Constraint (17). Constraint (18) is similar to Constraint (16) except that the trip is from A to B . Constraint (19) is for the start time of the AGV2 load task and its physical meaning is similar to Constraint (17). Constraint (20) indicates that the completion time of the AGV2 load task is its start time plus the transportation time and loading time. Constraints (21) and (22) each indicate that the time for unloaded AGV1 to arrive at the workpiece location is equal to the start time of the unloading task plus the traveling time. In addition, the starting point of the unloading task is the endpoint of the previous transport task. Constraint (23) ensures that the time in which AGV2 arrives unloaded at the material location is equal to the start time of the unloaded task plus the traveling time. The starting point of the unloaded task for AGV2 can only be D . Constraint (24) indicates that if the transport task Ta_{ip} is transported by AGV1 before Ta_{jq} , the start time of the unloaded task Tae_{jq} must be later than the end time of Ta_{ip} . Constraint (25) indicates that if the transportation task Tb_i is transported by AGV2 before Tb_j , the start time of the unloaded task Tbe_j must be later than the end time of Tb_i .

Capacity and Buffer Constraints

The following constraints address the limitations of the buffer space at the loading/unloading stations and machines. They ensure that buffer capacities are not exceeded, which could otherwise lead to production delays or stoppages.

$$STae_{j1} + M(1 - xA_{ijk}) \geq CTa_{i2}, \quad \forall i, j \in N \quad \forall k \in A \quad (26)$$

$$CTa_{j2} - t_{down} + M(1 - xB_{ijl}) \geq STb_i, \quad \forall i, j \in N \quad \forall l \in B \quad (27)$$

Constraint (26) indicates that if J_i occupies M_k in Area A before J_j , AGV1 can start transporting J_j only after J_i has been moved to Area B . Constraint (27) indicates that if J_i occupies M_k in Area B before J_j , the arrival time of AGV1 transporting J_j must be later than the time when J_i starts being transported by AGV2.

Flow Constraints

Flow constraints ensure that the movement of materials or workpieces through the system follows a logical and efficient path. They prevent unnecessary back-tracking or detours, optimizing the transportation routes.

$$va_{ipjq} + va_{jqip} \leq 1, \quad \forall i, j \in N \quad \forall p, q \in P \quad (28)$$

$$vb_{ij} + vb_{ji} \leq 1, \quad \forall i, j \in N \quad (29)$$

$$xA_{ijk} + xA_{jik} = xO_{ik}xO_{jk}, \quad \forall i, j \in N, i \neq j \quad \forall k \in A \quad (30)$$

$$xB_{ijl} + xB_{jil} = xO_{il}xO_{jl}, \quad \forall i, j \in N, i \neq j \quad \forall l \in B \quad (31)$$

Constraints (28) and (29) ensure that each transportation task is not executed more than once. Constraint (30) and (31) indicates that if J_i and J_j are assigned to be processed at the same machine, a sequential order must be established.

Initial Condition Constraints

Constraint (32) is an initial condition constraint specifying the starting conditions of the system.

$$STae_{11} = STbe_1 = 0 \quad (32)$$

3.2. General Workshop Scenario

The model discussed above considers a simplified scenario, assuming that all machines are initially unoccupied. However, real-world conditions are rarely so ideal. For example, in the case of a system shutdown and restart, the recalculation must begin from the current machine occupation state. The general workshop scenario extends the simplified scenario by considering initial machine states, where certain machines may already be occupied with jobs from a previous batch. This requires adjusting the index ranges for jobs in progress, ensuring that the system can handle random initial conditions. However, these modifications primarily involve setting initial conditions and adjusting constraints for jobs already in the system (e.g., modifying the index range for jobs that are still in progress). The complexity and solution time of this general scenario, when applied to similar-sized job sets, do not differ significantly from the simplified model. This ensures that the proposed model can handle dynamic scheduling situations (e.g., system interruptions) without introducing substantial computational overhead while also laying the groundwork for incremental computation. The detailed modifications to the general model, compared to the simplified model, are as follows.

3.2.1. Notation Modification

$[i, j]$ indices for jobs, $i, j \in N = N_{\text{preA}} \cup N_{\text{preB}} \cup N_{\text{now}}$, where N_{now} refers to the set of job indices in the current batch, N_{preA} refers to the set of remaining job indices from the previous batch in Area A, and N_{preB} refers to the set of remaining job indices from the previous batch in Area B.

3.2.2. Constraints Modification

Machine k is considered occupied when job i has t_{rest} seconds remaining to complete its processing on machine k . To describe this state, we need to set several constraints on the parameters.

$$xO_{ipk} = 1 \quad (33)$$

Constraint (33) thus fixes the decision variable xO_{ipk} to 1 for the specific job i (and its corresponding operation p) that is still being processed on machine k (with remaining

time t_{rest}) at the start of the new scheduling horizon. In other words, this ensures that job i is definitively assigned to continue processing on machine k in the current schedule, reflecting the carry-over occupation of that machine. This condition is applied only for jobs carried over from a previous batch (i.e., $i \in N_{preA}$ or $i \in N_{preB}$, depending on which partition the job was in).

If $p = 1$, at this point Ta_{i1} has been completed, so

$$STae_{i1} = CTae_{i1} = STa_{i1} = CTa_{i1} = 0 \quad (34)$$

$$STae_{i2} = \max(CTae_{i2}, t_{rest}) + t_{down} \quad \forall i \in N_{preA} \quad (35)$$

$$fa_{i1} = la_{i1} = 0 \quad (36)$$

$$va_{i1jq} = va_{jq i1} = 0 \quad \forall j \in N, q \in P. \quad (37)$$

If $p = 2$, at this point, both Ta_{i1} and Ta_{i2} have been completed, which means the impact of these two transportation tasks on the results should be eliminated. So we have:

$$STae_{i1} = CTae_{i1} = STa_{i1} = CTa_{i1} = STae_{i2} = CTae_{i2} = STa_{i2} = CTa_{i2} = 0 \quad (38)$$

$$STbe_i = \max(CTbe_i, t_{rest}) + t_{down} \quad \forall i \in N_{preB} \quad (39)$$

$$fa_{i1} = la_{i1} = fa_{i2} = la_{i2} = 0 \quad (40)$$

$$va_{i1jq} = va_{jq i1} = va_{i2jq} = va_{jq i2} = 0 \quad \forall j \in N, q \in P \quad (41)$$

Consequently, parts of the original constraints need to be modified as well because they cannot be applied to all jobs. For example, if the job does not belong to the current batch, obviously, it has already finished the transportation task from the loading station to area A, which is Ta_{i1} .

$$fa_{i1} + \sum_{j \in N} \sum_{q \in P} va_{jq i1} = 1, \forall i \in N_{now} \quad (42)$$

$$fa_{i2} + va_{i1i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{jq i2} = 1, \forall i \in N \setminus N_{preB} \quad (43)$$

$$la_{i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{i2jq} = 1, \forall i \in N \setminus N_{preB} \quad (44)$$

$$la_{i1} + va_{i1i2} + \sum_{j \in N \setminus \{i\}} \sum_{q \in P} va_{i1jq} = 1, \forall i \in N_{now} \quad (45)$$

$$STa_{i1} = CTae_{i1} + t_{down} \quad \forall i \in N_{now} \quad (46)$$

$$CTa_{i2} = STa_{i2} + \sum_{k \in A} \sum_{l \in B} x_{O_{i1k}} x_{O_{i2l}} t_{k,l} + t_{up} \quad \forall i \in N \setminus N_{preB} \quad (47)$$

$$STb_i = \max(CTbe_i, CTa_{i2} + t_B) + t_{down} \quad \forall i \in N \setminus N_{preB} \quad (48)$$

4. Case Study

To evaluate the effectiveness and performance of the proposed MILP model, three different workshop layouts are designed, as illustrated in Figure 3. **Layout 1** consists of one processing machine in Area A and one in Area B ($n_A = 1$, $n_B = 1$). In **Layout 2**, Area A contains two machines ($n_A = 2$) while Area B includes three machines ($n_B = 3$). Finally, **Layout 3** features five machines in Area A ($n_A = 5$) and four machines in Area B ($n_B = 4$). The traveling time matrices between the machines and the loading/unloading stations for these layouts are provided in Table 1. These layouts facilitate a systematic analysis of the model performance under increasing levels of complexity.

Table 1. Traveling time matrices for Layouts 1, 2, and 3.

Layout 1 ($n_A = 1, n_B = 1$)				
Layout 1	S	A1	B1	E
S	0	100	200	300
A1	100	0	100	200
B1	200	100	0	100
E	300	200	100	0

Layout 2 ($n_A = 2, n_B = 3$)							
Layout 2	S	A1	A2	B1	B2	B3	E
S	0	100	110	200	210	220	300
A1	100	0	10	100	110	120	200
A2	110	10	0	110	120	130	210
B1	200	100	110	0	10	20	100
B2	210	110	120	10	0	10	110
B3	220	120	130	20	10	0	120
E	300	200	210	100	110	120	0

Layout 3 ($n_A = 5, n_B = 4$)											
Layout 3	S	A1	A2	A3	A4	A5	B1	B2	B3	B4	E
S	0	100	110	120	130	140	200	210	220	230	300
A1	100	0	10	20	30	40	100	110	120	130	200
A2	110	10	0	10	20	30	110	120	130	140	210
A3	120	20	10	0	10	20	120	130	140	150	220
A4	130	30	20	10	0	10	130	140	150	160	230
A5	140	40	30	20	10	0	140	150	160	170	240
B1	200	100	110	120	130	140	0	10	20	30	100
B2	210	110	120	130	140	150	10	0	10	20	110
B3	220	120	130	140	150	160	20	10	0	10	120
B4	230	130	140	150	160	170	30	20	10	0	130
E	300	200	210	220	230	240	100	110	120	130	0

To further test the simplified MILP model, nine different scenarios are generated, each combining various workshop layouts and workpiece counts, as detailed in Table 2. For example, in Scenario 4, a total of five workpieces are processed using two machines in both Areas A and B.

For the general MILP model, we introduce six additional scenarios to consider different initial conditions within the workshop (Table 3). For example, in Scenario 10, five jobs are present, one already in progress on Machine 1, expected to complete after 50 s.

Table 2. Parameters for the scenarios.

No. of Scenario	No. of Layout	n
1	1	5
2	1	10
3	1	20
4	2	5
5	2	10
6	2	3
7	3	5
8	3	7
9	3	8

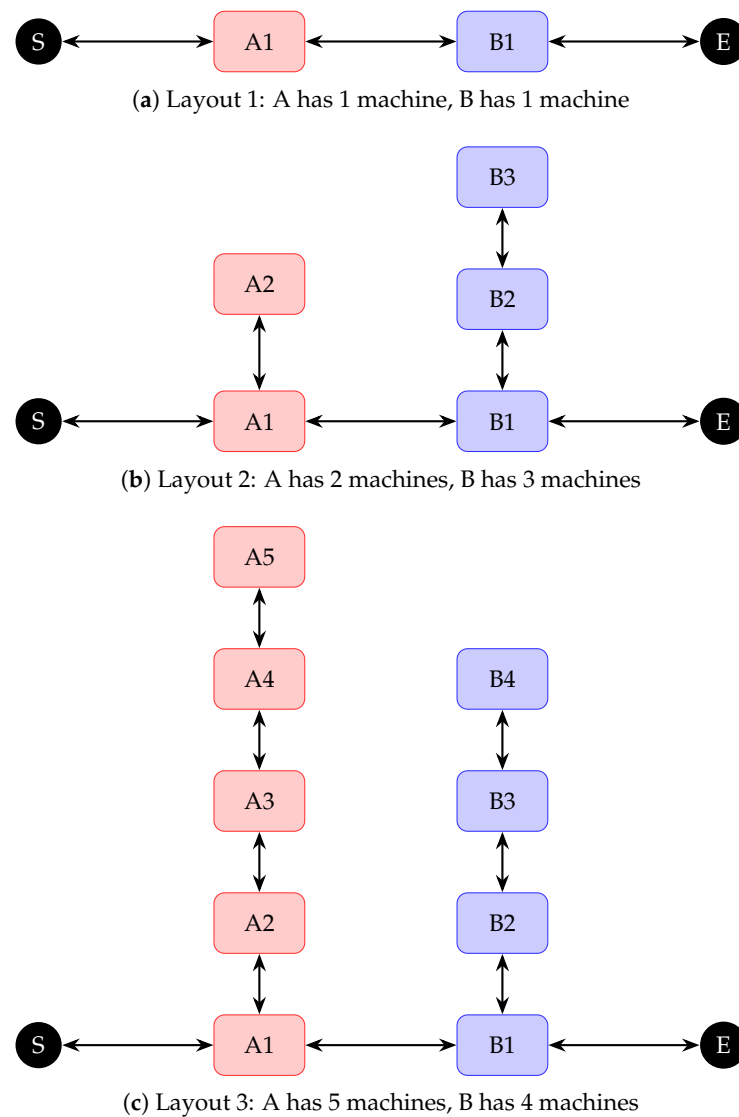


Figure 3. Configuration of Layout 1–3. This diagram illustrates the arrangement of machines in Areas A and B for Layout 1–3.

Table 3. Parameters for the added scenarios.

No. of Scenario	No. of Layout	No. of Occupied Machine	Rest Processing Time (s)	n
10	1	1	50	5
11	1	2	100	5
12	1	1 & 2	200 & 300	5
13	3	1	50	5
14	3	6	100	5
15	3	1 & 6	200 & 300	5

4.1. MILP Solution Based on CPLEX Solver

CPLEX, developed by IBM, is a highly regarded optimization software widely recognized for solving complex MILP problems. Renowned for its robustness and efficiency, CPLEX has powerful algorithms and advanced computational capabilities, making it an authoritative tool for addressing various optimization challenges. Using CPLEX, this study ensures the use of a proven and respected tool, thereby enhancing the credibility and reliability of the results obtained (we still use a small-scale scenario to prove the correctness of the results obtained by CPLEX).

Based on the capabilities of the CPLEX solver, optimal solutions are obtained for various problems of different sizes in Tables 2 and 3, demonstrating the model's ability to handle different scales of scheduling challenges.

The problems are solved using IBM ILOG CPLEX Optimization Studio 22.1.0.0 on a computer with an Intel Core i7-8750H CPU @2.2 GHz.

4.2. Greedy Algorithm for Comparison

For the sake of comparison, an improved greedy algorithm is developed to incorporate the design principles of the A* algorithm to formulate a loss function for AGV scheduling planning:

$$f(n) = g(n) + h(n) \quad (49)$$

In this context, $g(n)$ represents the total time taken by the AGV from the start of task execution until the completion of the current task, while $h(n)$ estimates the remaining time for the AGV to complete its tasks and reach the unloading station. This heuristic is more sophisticated than a simple distance-based function. In our implementation, $h(n)$ takes into account not only the direct travel distance but also factors like machine availability and the potential waiting time. Specifically, the heuristic considers the time required for AGVs to move between stations (e.g., from Area A to B), as well as the processing times that may vary based on machine availability and the dynamic state of the system. By factoring in these elements, the heuristic attempts to predict the time required for completing all remaining tasks, allowing the algorithm to make more informed decisions at each step. By comparing the values of the loss function of various decision schemes, the optimal current decision can be identified, thereby improving the greedy algorithm.

The greedy algorithm is used in this scheduling scenario to optimize the assignment of transportation tasks between multiple AGVs across different nodes. The core idea is to choose the locally optimal solution at each step, selecting the transportation task with the minimum cost to execute. This algorithm is efficient and suitable for online scheduling, and decision-making for multiple AGVs is simulated in parallel using multi-threading.

Figure 4 illustrates the AGV task scheduling scenario based on the greedy algorithm. AGV-01 operates between Nodes S, A, and B, while AGV-02 operates between Nodes B and E, with tasks dynamically assigned based on the local optimization of the greedy algorithm.

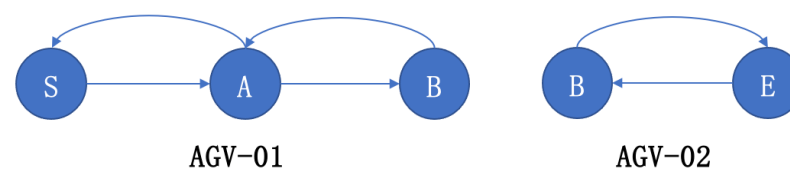


Figure 4. AGV task scheduling scenario based on greedy algorithm.

4.2.1. Node State Definitions

- **Node S:** Number and sequence of parts waiting to be picked up.
- **Node A:** Status of nA storage slots and the remaining processing time for each part.
- **Node B:** Status of nB storage slots and the remaining processing time for each part.
- **Node E:** Loading status of AGV.

4.2.2. Node Transition Rules

- **S to A:** The AGV picks up the highest priority part from Node S and transports it to Node A.
- **A to S:** If loaded, the AGV places the part in an available slot at Node A, then returns to Node S. If empty, it directly returns to Node S.

- **A to B:** If loaded, the AGV places the part in Node B's available slot. If empty, it picks up the part with the shortest remaining waiting time and transports it to Node B.
- **B to A:** The AGV places the part in an available slot at Node B and then returns to Node A.
- **B to E:** The AGV places the part in Node E if loaded or fetches the part with the shortest remaining waiting time from Node B and transports it to Node E.
- **E to B:** If loaded, the AGV places the part in an available slot at Node E and then returns to Node B. If empty, it directly returns to Node B.

4.2.3. Cost Definition

The cost of each transition is defined by the time consumed, which includes the pickup and placement time, the transportation time, and the waiting time. The greedy algorithm always selects the transition with the lowest cost at each decision point.

4.2.4. Collaboration Mechanism

Each AGV performs tasks independently, but they share the state information of Node B to avoid conflicts. The final completion status is jointly determined through information sharing and optimal decision-making.

4.2.5. Detailed Procedure of the Algorithm

The procedure for solving the problem is described in the pseudo-code shown in Algorithm 1.

4.3. Results Comparison

4.3.1. Comparison Between Simplified MILP and Greedy Algorithm

Table 4 summarizes the results in terms of makespan and computational time for both the simplified MILP model and the greedy algorithm in nine scenarios. For several smaller instances (Scenarios 1–3 and 6), two approaches yield identical or nearly identical makespans. In these cases, their computational times remain fairly low and do not differ dramatically, indicating that a simple MILP model can still handle moderate-scale problems efficiently.

In more demanding scenarios (e.g., Scenarios 4, 5, 7–9), the simplified MILP method achieves strictly lower makespans than the greedy approach, sometimes by a notable margin. This improvement in solution quality, however, comes with a considerable increase in computational time. For instance, Scenario 7 shows only a marginal improvement in objective value when using the MILP, yet the solution time is orders of magnitude higher than the heuristic. Moreover, in Scenario 9, the MILP required 91,178.93 s (over 25 h) to find the optimal solution. This run time is clearly impractical for real-time scheduling. Such a stark difference underscores how the exhaustive search and exact optimization inherent in MILP can become computationally expensive as the complexity of the problem increases. On the other hand, the MILP's global optimum is valuable as a benchmark: it quantifies the maximum possible improvement. Therefore, we emphasize that the MILP formulation is best suited for offline planning or small instances rather than live control of a manufacturing system. As shown in our experiments, the MILP can solve small to moderate instances fairly quickly. For example, scenarios with up to roughly 10 jobs and 3 machines per partition were solved in a reasonable time (typically seconds). In contrast, solution time grows very rapidly beyond this size. For larger instances (like Scenario 9), heuristic methods are necessary to obtain timely solutions.

Algorithm 1 Greedy Search for AGV Task Scheduling.

```

1: Initialize the production world with a grid for storage locations at stations A and B
2: Set initial transportation times and parameters
3: Initialize state variables for the current parts and AGVs
4: procedure AGV1_TASK_SCHEDULING
5:   AGV-1 starts at source node S
6:   while there are remaining parts to process do
7:     Pick up a part from source
8:     Transition to station A and place the part in an available slot
9:     Update time and state for AGV-1
10:    if there is a part ready at station A then
11:      Transport the part to station B and place it in an available slot
12:    end if
13:  end while
14: end procedure
15: procedure AGV2_TASK_SCHEDULING
16:   AGV-2 starts at station D
17:   while there is a part ready at station B do
18:     Fetch the part from station B
19:     Transport it to station D and unload
20:     Update state for AGV-2
21:   end while
22:   Wait for AGV-1 if required
23: end procedure
24: Calculate the cost for each slot in the storage grid
25: Periodically check if all parts have been transported and processed
26: Start threads for AGV-1 and AGV-2 to operate concurrently
27: while tasks are not finished do
28:   Execute both AGV tasks concurrently
29: end while

```

The MILP model is consistently capable of outperforming the greedy approach in terms of finding optimal solutions. In contrast, the greedy algorithm can rapidly generate solutions but may fail to capture nuanced interdependencies, particularly when there are multiple machines, AGVs, or intricate precedence requirements. Consequently, it can produce suboptimal schedules for more complex scenarios.

In summary, while the simplified MILP formulation guarantees optimality, it can incur dramatic computational overheads for certain problem sizes. The greedy algorithm, with its heuristic nature, offers a much faster runtime but cannot ensure the best-possible makespan. Deciding which method to employ depends on the scale of the instance and whether the application prioritizes speedy results over absolute optimality.

4.3.2. Comparison Between General MILP Method and Greedy Algorithm

Table 5 compares the performance of the general MILP approach and the greedy algorithm in six different scenarios (10–15), reporting both makespan and computational time. From a makespan standpoint, the general MILP model consistently achieves equal or better solutions than the greedy approach, although its computation time can be higher, particularly for larger instances.

Table 4. Comparison between the simplified MILP method and the greedy algorithm.

No. of Scenario	Makespan(s) (MILP)	Makespan(s) (Greedy)	Computational Time (s) (MILP)	Computational Time (s) (Greedy)
1	8064	8064	0.11	1.13
2	15,504	15,504	0.98	2.15
3	30,384	30,384	19.31	4.19
4	5088	5616	3.00	1.22
5	8592	9364	1858.96	2.26
6	3600	3884	0.17	0.82
7	3208	3248	16.47	1.34
8	4068	5020	6225.52	1.75
9	4536	5324	91,178.93	1.95

In **Scenarios 10, 11, and 12**, the MILP method achieves makespans of 6980, 6576, and 5592 s, respectively, while the greedy algorithm produces 6980, 6672, and 5882 s. Notably, in Scenario 10, there is no difference in makespan (both are 6980 s), yet the MILP approach computes more quickly (0.45 s vs. 1.14 s), suggesting that for such mid-scale problems, the MILP solver can efficiently reach an optimal outcome. In Scenarios 11 and 12, the MILP model outperforms the greedy approach by reducing the makespan by 96 and 290 s, respectively, and requires less than one second of runtime (0.19 s and 0.13 s) as compared to about one second for the greedy algorithm (1.13 s and 1.12 s). These observations point to the ability of the MILP model to balance the quality of solutions and efficiency in moderately sized problems.

By contrast, in **Scenarios 13 and 14**, as the problem becomes more complex, the computational time of the MILP model rises to 142.29 s and 7.82 s, while the greedy algorithm remains near 1.14 s. However, the MILP method still achieves notably lower makespans (2984 s vs. 3188 s and 2904 s vs. 2964 s, respectively), indicating that even under increased complexity, the MILP model can deliver superior scheduling results if the user can afford the additional computational time.

Finally, in **Scenario 15**, the MILP solver requires 5.51 s, compared to 1.13 s for the greedy approach. Despite the difference in runtime, the MILP solution further reduces the makespan to 2700 s relative to 2960 s under the greedy strategy. This result underscores that, for cases of the intermediate size, the MILP model continues to exhibit strong performance in optimizing schedules while demanding a moderate and often acceptable level of computational effort.

Overall, these findings demonstrate that the general MILP model consistently outperforms the greedy algorithm in makespan minimization. Although the MILP approach may require more time, especially for larger-scale scenarios, it delivers better-quality solutions. In contrast, the fast execution of the greedy algorithm makes it suitable for highly dynamic or real-time applications but at the cost of suboptimal schedules. Thus, the final choice between these methods depends on the user's specific needs for the quality of the solution versus responsiveness.

Table 5. Comparison between general MILP method and greedy algorithm.

No. of Scenario	Makespan(s) (General MILP)	Makespan(s) (Greedy)	Computational Time (s) (General MILP)	Computational Time (s) (Greedy)
10	6980	6980	0.45	1.14
11	6576	6672	0.19	1.13
12	5592	5882	0.13	1.12
13	2984	3188	142.29	1.14
14	2904	2964	7.82	1.14
15	2700	2960	5.51	1.13

5. Conclusions

This study presents a comprehensive MILP model for the simultaneous scheduling of machines and AGVs in a partitioned FMS. By addressing the complex interplay between machine operations and AGV movements, the proposed model aims to optimize the overall scheduling process, thus minimizing the total completion time and improving the efficiency of the system.

The MILP model is solved using IBM's CPLEX, a highly regarded optimization solver known for its robustness and accuracy. The results obtained from CPLEX demonstrate the effectiveness of the model in achieving globally optimal solutions for small- to medium-sized scheduling problems. Furthermore, comparison analysis with an improved greedy algorithm provides valuable insight into the trade-offs between the solution quality and computational efficiency. Although the MILP model consistently ensures optimal solutions, the greedy algorithm offers a pragmatic alternative for larger-scale problems where computational resources may be limited.

This research makes several contributions to the field of manufacturing optimization. First, the MILP model incorporates realistic production constraints, such as transportation times and multi-workshop coordination, making it more applicable to real-world industrial scenarios. Second, by integrating the scheduling of machines and AGVs, the model provides a holistic approach that can significantly improve operational performance in FMS environments. Lastly, the use of CPLEX underscores the importance of using advanced optimization tools to solve complex scheduling problems, thereby enhancing the credibility and reliability of the results.

Future work will focus on extending the model to handle even larger and more complex scheduling problems, possibly by integrating other heuristic or metaheuristic approaches to further reduce computational times while maintaining solution quality. In addition, the current model assumes no traffic collisions, deadlocks, or conflicts between AGVs. While this assumption is reasonable for an initial study, omitting collisions means that travel times may be underestimated in highly congested settings. In future work, AGV conflicts could be handled by adding constraints or specialized rules (e.g., defining reservation tables or path-priority rules) to the model. Furthermore, exploring the application of the model in different industrial settings and integrating it with digital simulation platforms could provide deeper insights into its practical utility and impact on production efficiency. Last but not least, while the comparison in this study focuses on makespan and computational time, we recognize the importance of additional production indicators, such as AGV idle rate and machine utilization, in evaluating scheduling strategies. AGV idle rate measures the proportion of time that AGVs are not in active use, while machine utilization tracks the percentage of time machines are engaged in production. We plan to include these metrics to provide a more comprehensive evaluation of the MILP and greedy algorithms, highlighting the trade-offs between solution speed and resource optimization.

Author Contributions: Conceptualization, C.Z., L.L. and M.L.; methodology, C.Z.; software, C.Z., J.Q. and T.W.; validation, C.Z.; investigation, C.Z., J.Q., T.W. and Y.B.; writing—original draft preparation, C.Z.; writing—review and editing, M.L.; visualization, C.Z.; supervision, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All the data are available in the manuscript.

Conflicts of Interest: The author Liyong Lin was employed by the company Contemporary Amperex Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Abdelmaguid, T.F.; Nassef, A.O.; Kamal, B.A.; Hassan, M.F. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Res.* **2004**, *42*, 267–281. [\[CrossRef\]](#)
2. Li, J.; Cheng, W.; Lai, K.K.; Ram, B. Multi-AGV flexible manufacturing cell scheduling considering charging. *Mathematics* **2022**, *10*, 3417. [\[CrossRef\]](#)
3. Baruwa, O.T.; Piera, M.A. A coloured Petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Res.* **2016**, *54*, 4773–4792. [\[CrossRef\]](#)
4. Dabwan, A.; Kaid, H.; Al-Ahmari, A.; Alqahtani, K.N.; Ameen, W. An Internet-of-Things-Based Dynamic Scheduling Optimization Method for Unreliable Flexible Manufacturing Systems under Complex Operational Conditions. *Machines* **2024**, *12*, 192. [\[CrossRef\]](#)
5. Zhang, S.; Du, H.; Borucki, S.; Jin, S.; Hou, T.; Li, Z. Dual resource constrained flexible job shop scheduling based on improved quantum genetic algorithm. *Machines* **2021**, *9*, 108. [\[CrossRef\]](#)
6. Gao, J.; Zheng, X.; Gao, F.; Tong, X.; Han, Q. Heterogeneous multitype fleet green vehicle path planning of automated guided vehicle with time windows in flexible manufacturing system. *Machines* **2022**, *10*, 197. [\[CrossRef\]](#)
7. Wu, M.; Yang, D.; Zhou, B.; Yang, Z.; Liu, T.; Li, L.; Wang, Z.; Hu, K. Adaptive population nsga-iii with dual control strategy for flexible job shop scheduling problem with the consideration of energy consumption and weight. *Machines* **2021**, *9*, 344. [\[CrossRef\]](#)
8. Boccia, M.; Masone, A.; Sterle, C.; Murino, T. The parallel AGV scheduling problem with battery constraints: A new formulation and a matheuristic approach. *Eur. J. Oper. Res.* **2023**, *307*, 590–603. [\[CrossRef\]](#)
9. Fontes, D.B.M.; Homayouni, S.M. Joint production and transportation scheduling in flexible manufacturing systems. *J. Glob. Optim.* **2019**, *74*, 879–908. [\[CrossRef\]](#)
10. Hurink, J.; Knust, S. Tabu search algorithms for job-shop problems with a single transport robot. *Eur. J. Oper. Res.* **2005**, *162*, 99–111. [\[CrossRef\]](#)
11. Deroussi, L.; Gourgand, M.; Tchernev, N. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Res.* **2008**, *46*, 2143–2164. [\[CrossRef\]](#)
12. Gnanavel Babu, A.; Jerald, J.; Noorul Haq, A.; Muthu Luxmi, V.; Vigneswaralu, T. Scheduling of machines and automated guided vehicles in FMS using differential evolution. *Int. J. Prod. Res.* **2010**, *48*, 4683–4699. [\[CrossRef\]](#)
13. Lacomme, P.; Larabi, M.; Tchernev, N. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *Int. J. Prod. Econ.* **2013**, *143*, 24–34. [\[CrossRef\]](#)
14. Zhang, Q.; Manier, H.; Manier, M.A. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *Int. J. Prod. Res.* **2014**, *52*, 985–1002. [\[CrossRef\]](#)
15. Zheng, Y.; Xiao, Y.; Seo, Y. A tabu search algorithm for simultaneous machine/AGV scheduling problem. *Int. J. Prod. Res.* **2014**, *52*, 5748–5763. [\[CrossRef\]](#)
16. Nouri, H.E.; Driss, O.B.; Ghédira, K. Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment. *Appl. Intell.* **2016**, *45*, 808–828. [\[CrossRef\]](#)
17. El Khayat, G.; Langevin, A.; Riopel, D. Integrated production and material handling scheduling using mathematical programming and constraint programming. *Eur. J. Oper. Res.* **2006**, *175*, 1818–1832. [\[CrossRef\]](#)
18. Caumond, A.; Lacomme, P.; Moukrim, A.; Tchernev, N. An MILP for scheduling problems in an FMS with one vehicle. *Eur. J. Oper. Res.* **2009**, *199*, 706–722. [\[CrossRef\]](#)
19. Özgüven, C.; Özbakır, L.; Yavuz, Y. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Appl. Math. Model.* **2010**, *34*, 1539–1548. [\[CrossRef\]](#)
20. Huang, S. Optimization of Job Shop Scheduling with Material Handling by Automated Guided Vehicle. Ph.D. Thesis, Iowa State University, Ames, IA, USA, 2018.
21. Abderrahim, M.; Bekrar, A.; Trentesaux, D.; Aissani, N.; Bouamrane, K. Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints. *Optim. Lett.* **2022**, *16*, 255–280. [\[CrossRef\]](#)
22. Yao, Y.J.; Liu, Q.H.; Li, X.Y.; Gao, L. A novel MILP model for job shop scheduling problem with mobile robots. *Robot. -Comput.-Integr. Manuf.* **2023**, *81*, 102506. [\[CrossRef\]](#)
23. Boccia, M.; Mancuso, A.; Masone, A.; Murino, T.; Sterle, C. Exact and heuristic solution approaches for the multi-objective AGV scheduling problem with battery constraints. *Transp. Res. Procedia* **2024**, *78*, 369–376. [\[CrossRef\]](#)
24. Meng, L.; Zhang, C.; Zhang, B.; Gao, K.; Ren, Y.; Sang, H. MILP modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times. *Swarm Evol. Comput.* **2023**, *82*, 101374. [\[CrossRef\]](#)
25. Liu, Q.; Li, X.; Gao, L.; Fan, J. Two novel MILP models with different flexibilities for solving integrated process planning and scheduling problems. *J. Oper. Res. Soc.* **2023**, *74*, 1955–1967. [\[CrossRef\]](#)
26. Amirteimoori, A.; Mohamed, M.; Kia, R. Simultaneous scheduling of jobs and transporters in a hybrid flow shop with collision-free transporter routing: A novel parallel heuristic. *Comput. Oper. Res.* **2024**, *168*, 106687. [\[CrossRef\]](#)

27. Guo, Y.; Yao, J.; Su, R.; Ling, K.; Han, B.S.; Wong, H.Y.A. A MILP Model For Conflict-Free Routing Problem With Partial Charging In Multiple-AGV System. In Proceedings of the 2024 IEEE 18th International Conference on Control & Automation (ICCA), Reykjavik, Iceland, 18–21 June 2024; pp. 749–754.
28. Wang, W.; Tian, G.; Luo, M.; Zhang, H.; Yuan, G.; Niu, K. More mixed-integer linear programming models for solving three-stage remanufacturing system scheduling problem. *Comput. Ind. Eng.* **2024**, *194*, 110379. [[CrossRef](#)]
29. Groover, M.P. *Automation, Production Systems, and Computer-Integrated Manufacturing*; Pearson: London, UK, 2015.
30. Javaid, M.; Haleem, A.; Singh, R.P.; Suman, R. Enabling flexible manufacturing system (FMS) through the applications of industry 4.0 technologies. *Internet Things Cyber-Phys. Syst.* **2022**, *2*, 49–62. [[CrossRef](#)]
31. Reddy, S.N.; Ramamurthy, V.; Rao, P.K.; Lalitha, P.M. Integrated scheduling of machines, AGVs and tools in multi-machine FMS using crow search algorithm. *Int. J. Comput. Integr. Manuf.* **2019**, *32*, 1117–1133.
32. Liu, Q.; Wang, N.; Li, J.; Ma, T.; Li, F.; Gao, Z. Research on Flexible Job Shop Scheduling Optimization Based on Segmented AGV. *CMES-Comput. Model. Eng. Sci.* **2023**, *134*, 2073–2091. [[CrossRef](#)]
33. Nemhauser, G.L.; Wolsey, L.A. *Integer and Combinatorial Optimization*; John Wiley & Sons: Hoboken, NJ, USA, 1999.
34. Wolsey, L.A. *Integer Programming*; John Wiley & Sons: Hoboken, NJ, USA, 1998.
35. Bertsimas, D.; Weismantel, R. *Optimization over Integers*; Dynamic Ideas: Waltham, MA, USA, 2005.
36. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2009.
37. Dasgupta, S.; Papadimitriou, C.H.; Vazirani, U. *Algorithms*; McGraw-Hill: New York, NY, USA, 2006.
38. Kleinberg, J.; Tardos, E. *Algorithm Design*; Pearson Education: London, UK, 2006.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.