Future Technology
Open Access Journal
ISSN 2832-0379

November 2025| Volume 04 | Issue 04 | Pages 282-295

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.4.4.23

Future Publishing LLC

Article

Integrated scheduling of jobs, tools, and AGVs in FMS with non-identical machines using a recurrent neural network

Swapnil More*, Naveen Kumar

Mechanical Engineering, Sir Padampat Singhania University, Udaipur, India

ARTICLE INFO

Article history:
Received 12 July 2025
Received in revised form
25 August 2025
Accepted 09 September 2025

Keywords:
Scheduling, Recurrent Neural Network (RNN), Flexible manufacturing system, Automated guided vehicles

*Corresponding author
Email address:
swapnil.more@spsu.ac.in

DOI: 10.55670/fpll.futech.4.4.23

ABSTRACT

In a flexible manufacturing system (FMS), scheduling jobs and tools across non-identical machines, integrating automated guided vehicles (AGVs), and considering multi-objective functions, constitutes a significant obstacle for typical mathematical optimization techniques. Herein, we consider scheduling jobs, tools, and AGVs in an FMS that consists of three non-identical machines. The multi-objective functions targeted are tooling cost minimization and makespan reduction. The non-identical machines' processing rates are specified in the ratio of 1:1.2:1.4. Each of the tools (T1, T2, and T3) is available in a single mode, with T1 being more expensive than T2, which is more expensive than T3. To address such a complex optimization problem, we use a Recurrent Neural Network (RNN) and an Improved version to obtain near-optimum solutions and evaluate such algorithms' comparative performance. The average computation time to determine the optimal sequence was reduced from 10.33 minutes to 6.24 minutes (for a 4-job problem) as we employed the Improved RNN algorithm instead of the RNN algorithm.

1. Introduction

Flexible Manufacturing Systems (FMS) represent an advanced manufacturing paradigm characterized by adaptability to varying production tasks and dynamic operational requirements. In such environments, jobs with heterogeneous processing demands are scheduled on non-identical machines, and integrating Automated Guided Vehicles (AGVs) for intra-facility material transport introduces an additional layer of complexity. Scheduling in FMS becomes a multi-objective optimization problem, typically aiming to minimize performance metrics such as makespan (i.e., the total time to complete all jobs) and tooling cost. Flexible Manufacturing Systems (FMS) represent a cornerstone of modern Industry 4.0 environments due to their adaptability, efficiency, and ability to handle diverse production requirements. However, the scheduling of jobs in such systems is particularly challenging when non-identical machine speeds, single-copy tool constraints, and Automated Guided Vehicle (AGV) coordination are jointly considered. These interdependent factors transform scheduling into a multi-objective, combinatorial optimization problem, where minimizing makespan must be balanced against tooling cost and system resource utilization. Metaheuristics have been applied to flexible manufacturing systems (FMS) scheduling for a long time. Early studies employed Multi-Objective Simulated Annealing (MOSA), which emphasized the importance of weight vectors in balancing multi-criteria optimization [1]. Building upon evolutionary paradigms, a Constraint-Based Genetic Algorithm (CBGA) introduced novel operators that efficiently solved machine loading problems with reduced computational effort [2]. Subsequently, an Adaptive Genetic Algorithm (AGA) was designed for integrated job and Automated Guided Vehicle (AGV) scheduling, achieving superior performance in minimizing penalty costs and improving utilization compared with conventional Gas [3]. Similarly, a genetic algorithm (GA) approach, implemented as a spreadsheet add-in, was used to solve the complex problem of simultaneously scheduling machines and automated guided vehicles (AGVs) within flexible manufacturing systems (FMS) to minimize the total completion time, also known as the makespan [4]. A more recent advancement is the Modified Genetic Algorithm (MGA), which incorporates a three-parent crossover and mutation operator to improve population diversity and avoid premature convergence. By leveraging the Giffler–Thompson procedure, MGA generated active feasible schedules and demonstrated superior makespan performance across various problem sizes. Although computation time increased with larger instances, MGA consistently achieved optimal or near-optimal solutions [5].

Abbreviations

FMS	Flexible Manufacturing System
AGV	Automated Guided Vehicle
MOSA	Multi-Objective Simulated Annealing
CBGA	Constraint-Based Genetic Algorithm
AGA	Adaptive Genetic Algorithm
MGA	Modified Genetic Algorithm
DE	Differential Evolution
WGA	Weighted genetic algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PLS	Pareto Local Search
PSO	Particle Swarm Optimization
VNS	Variable Neighborhood Search
ILS	Iterated Local Search
MILP	Mixed Integer Linear Programming
KCSA	Knowledge-Based Cuckoo Search Algorithm
RL	Reinforcement learning
HMAPPO	Hybrid Multi-Agent Proximal Policy Optimization
MARL	Multi-Agent Reinforcement Learning

Other evolutionary techniques include the Differential Evolution (DE) algorithm for solving the scheduling problem and the optimal sequences of machines and AGVs, specifically to minimize makespan [6]. Multi-objective algorithms, including the weighted genetic algorithm (WGA), non-dominated sorting genetic algorithm (NSGA-II), and Pareto local search (PLS), were used to model and solve a broad class of scheduling problems, including flexible job shop and flow shop scheduling problems [7]. Hybrid approaches have been extensively explored to capture interdependencies in FMS scheduling. An effective hybrid multi-objective genetic algorithm for scheduling machines and AGVs in FMS was developed to optimize multiple conflicting objectives like makespan, mean flow time, and mean tardiness [8]. An integrated scheduling model with an improved particle swarm optimization (PSO) algorithm was developed, which uses the makespan of jobs as the optimization objective and the utilization ratios of machines and AGVs as evaluation factors [9].

Other notable hybrid methods include a novel control strategy for avoiding deadlock and collisions in zone-controlled AGVS, using coloured Petri nets to model the dynamics of AGVS, and implementing the control strategy [10]. A novel multi-objective formulation for AGV scheduling in cyclic flexible flow shops was introduced and evaluated using efficient local search heuristics, Variable Neighborhood Search (VNS), and Iterated Local Search (ILS) with different job scheduling rules, and provides a detailed performance analysis using the hypervolume indicator, highlighting the VNS approach as generally preferable due to its speed [11].

Parallel to metaheuristics, exact optimization methods have also been developed. A new mixed integer linear programming (MILP) model was proposed for the simultaneous scheduling of machines and Automated Guided Vehicles (AGVs) in FMS environments [12]. Further, a multi-objective mixed-integer linear programming model was formulated to formulate a scheduling problem with pickup and delivery in a matrix manufacturing workshop with multi-variety and small-batch production, aiming to maximize customer satisfaction while minimizing distribution cost [13]. While MILP guarantees optimality, scalability remains a significant limitation for larger scheduling instances. Bio-inspired algorithms have enriched FMS scheduling research with novel solution strategies. A novel knowledge-based cuckoo search (KCSA) algorithm designed to tackle complex flexible job shop scheduling problems. By integrating reinforcement learning and hybrid heuristics, KCSA provides a robust and efficient solution that adaptively controls parameters and guides the search process, leading to superior performance [14]. With the rising complexity of dynamic FMS environments, reinforcement learning (RL) has emerged as a promising paradigm. A Hybrid Multi-Agent Proximal Policy Optimization (HMAPPO) framework was developed for real-time scheduling in dynamic partial no-wait flexible job shops, achieving improved job completion rates, reduced tardiness, and superior Pareto trade-offs compared with MOPSO, NSGA-II, and dispatching rules [15]. Extending this direction, Multi-Agent Reinforcement Learning (MARL) was applied for joint job and AGV scheduling, with tailored action decoding and reward structures. MARL achieved over 10% performance improvement compared to traditional RL and genetic programming approaches, while exhibiting robustness against disturbances [16].

Recently, neural-network-based methods have been explored to address scalability and adaptability in FMS scheduling. A Recurrent Neural Network (RNN)–Genetic Algorithm (GA) framework was proposed for non-identical machine scheduling, with a focus on minimizing makespan. Comparative experiments across three RNN models demonstrated that RNN Phase 3 achieved 99.8% accuracy with an average computational time of 4.02 minutes, outperforming GA in scalability and effectiveness across varying job sizes. The algorithm calculates the number of possibilities to find the optimal solution by applying an equation [17]. This indicates the growing potential of RNN-based frameworks for integrated and adaptive scheduling. The above shows that while metaheuristics and hybrid approaches provide efficient approximations, they often lack adaptability in dynamic environments. Exact MILP formulations guarantee optimality but are computationally infeasible for large-scale problems. Recent advances in deep reinforcement learning and multi-agent systems demonstrate strong potential for real-time, adaptive, and disturbance-resilient scheduling. However, integrating machine, tool, and AGV scheduling under multi-objective constraints remains underexplored, providing the key motivation for the present work. This research proposes a Recurrent Neural Network (RNN) and an Improved RNN framework for integrated job, tool, and AGV scheduling to address these limitations. Unlike traditional metaheuristics, RNNs inherently capture sequential dependencies and can learn non-linear interrelationships across operations, making them particularly suitable for dynamic scheduling problems. The Improved RNN introduces a probabilistic filtering mechanism that prunes low-value machine–job combinations, thereby reducing computational complexity without compromising solution quality. The key objectives of this work are:

- To develop an RNN-based framework for integrated scheduling jobs, tools, and AGVs in FMS with non-identical machines.
- To propose and evaluate an improved RNN algorithm that minimizes makespan and tooling costs with reduced computational overhead.

This contribution enhances the use of AI-driven methods in manufacturing scheduling while offering a scalable framework for intelligent, data-driven production systems.

The research paper contributes to the collective scheduling of Jobs, tools, and AGVs in an FMS involving non-identical machines. The problem has not been addressed in the available literature. The major innovation is establishing an Improved Recurrent Neural Network (RNN) that utilizes probabilistic filtering to minimize computational overhead while maintaining solution quality. The direction is to balance the makespan and tooling costs with realizable single-copy tool and AGV constraints.

2. Problem formulation

The Flexible Manufacturing System (FMS) under consideration consists of non-identical machines, each capable of processing a variety of jobs. However, job processing is constrained by the capabilities of machines and the availability of specific tools required to complete individual operations. The non-identical nature of the machines is reflected in their processing speed, defined in a ratio of 1:1.2:1.4 for machines M1, M2, and M3, respectively. This depicts that the non-identical machine M1 has a faster processing speed than the non-identical machine M2, which in turn has a faster processing speed than M3. It consists of 4 jobs, three tool types, and 2 AGVs. Each non-identical machine can perform all operations; each job comprises multiple operations requiring a specific tool type. Only one copy of each tool type is available, reflecting the high tooling cost constraint. No two machines can use the same tool simultaneously. Two Automated Guided Vehicles (AGVs), AGV1 and AGV2, are responsible for transporting jobs from the Load/Unload (L/U) station to the designated machines and returning for the next job.

3. Assumptions

- The operation sequence, required tools, and processing times are predetermined.
- The tool requirements and operation sequences vary across jobs.
- A job must be completed entirely on a single machine, with all its operations performed on one visit.
- No job preemption is allowed; once an operation starts, it must be completed without interruption.
- Machines and tools can handle only one job at a time.
- Tools are centrally managed and become available shortly after completion of an operation (minimal tool return time).
- AGV routing and wait times are considered while ensuring no overlap in job transportation.
- AGVs are assumed to be identical in efficiency and can transport only one job at a time.

Each job must be allocated to a single machine. Hence, the objective is to schedule job (j_i) to machine (m_j) while utilizing the required tools (t_k), preventing tool conflicts. This involves coordinating the processing of J jobs across M non-identical machines while utilizing a shared set of T tool types for various operations. The aim is to establish an integrated schedule for the jobs in a flexible manufacturing system with non-identical machines, in an optimal sequence, by minimizing the makespan and tooling cost.

4. DATA set generation

This study employed a synthetically generated dataset because publicly available real-world datasets for integrated FMS scheduling (involving non-identical machines, single-copy tools, and AGV coordination) are extremely limited. Industrial data are often proprietary and not openly shared due to confidentiality constraints. To ensure that the synthetic dataset is representative of realistic shop-floor conditions, the following considerations were applied during its construction: (i) machine processing times were drawn from uniform distributions (1–20 units) and then scaled according to machine speed ratios of 1:1.2:1.4, which reflect typical non-identical machine performance differences reported in literature; (ii) AGV travel times were sampled from the range of 1–15 units, consistent with reported intra-shop transport durations; and (iii) tool usage was restricted to single copies, with cost multipliers aligned to relative tooling costs found in manufacturing practice. Most FMS scheduling studies rely on synthetic datasets because real-world data are rarely accessible due to industrial confidentiality.

4.1 Creation of the processing time data and tool allocation dataset

Figure 1 illustrates the algorithm developed for estimating the processing time of machines and assigning appropriate tools to each operation. The data generation procedure is structured to simulate 500 distinct job records, each composed of multiple operations with associated parameters. To initiate the process, a loop iterates over 500 job records. The number of operations for each job is determined probabilistically: 75% of the jobs contain three operations, 20% contain two operations, and the remaining 5% consist of a single operation. This count is stored in a variable denoted as N. A corresponding number of operations is generated based on the value of N. Each operation's processing time is randomly sampled from a uniform integer distribution between 1 and 20 and stored in the variable Dops. Simultaneously, each operation is randomly assigned a tool, and the tool type is stored in the variable Tops.

A critical constraint is imposed to ensure that tool assignments are unique within each job; no tool is repeated across operations belonging to the same job. Placeholder values marked as 'X' are appended in cases where the number of operations is fewer than three, thereby standardizing the record length. Additionally, to simulate processing across different machines, the generated processing times are scaled by multiplication factors of 1.2 and 1.4 and stored in separate rows, thereby modeling variability in machine performance. Each job record comprises four such jobs, and a unique hash is computed for traceability and sorting purposes. The finalized dataset is sorted based on this hash and exported in a structured format to a CSV file titled 'Outcome_ProcessingTime.csv'. Table 1 provides a representation of the generated dataset.

4.2 Creation of the AGV travel duration dataset

Figure 2 presents the algorithm synthesizing the Automated Guided Vehicle (AGV) travel duration dataset. The algorithm operates through an iterative process over 500 synthetic records, each simulating potential AGV travel durations across four distinct locations: M1, M2, M3, and L/U (Load/Unload station). For each iteration, a 4×4 matrix represents the travel durations between every pair of source and destination locations. Each element within the matrix denotes the time required for the AGV to traverse from one location to another. The travel duration values are generated using a uniformly distributed random integer from 1 to 15, thereby modeling realistic variability in transit times due to environmental or operational factors.

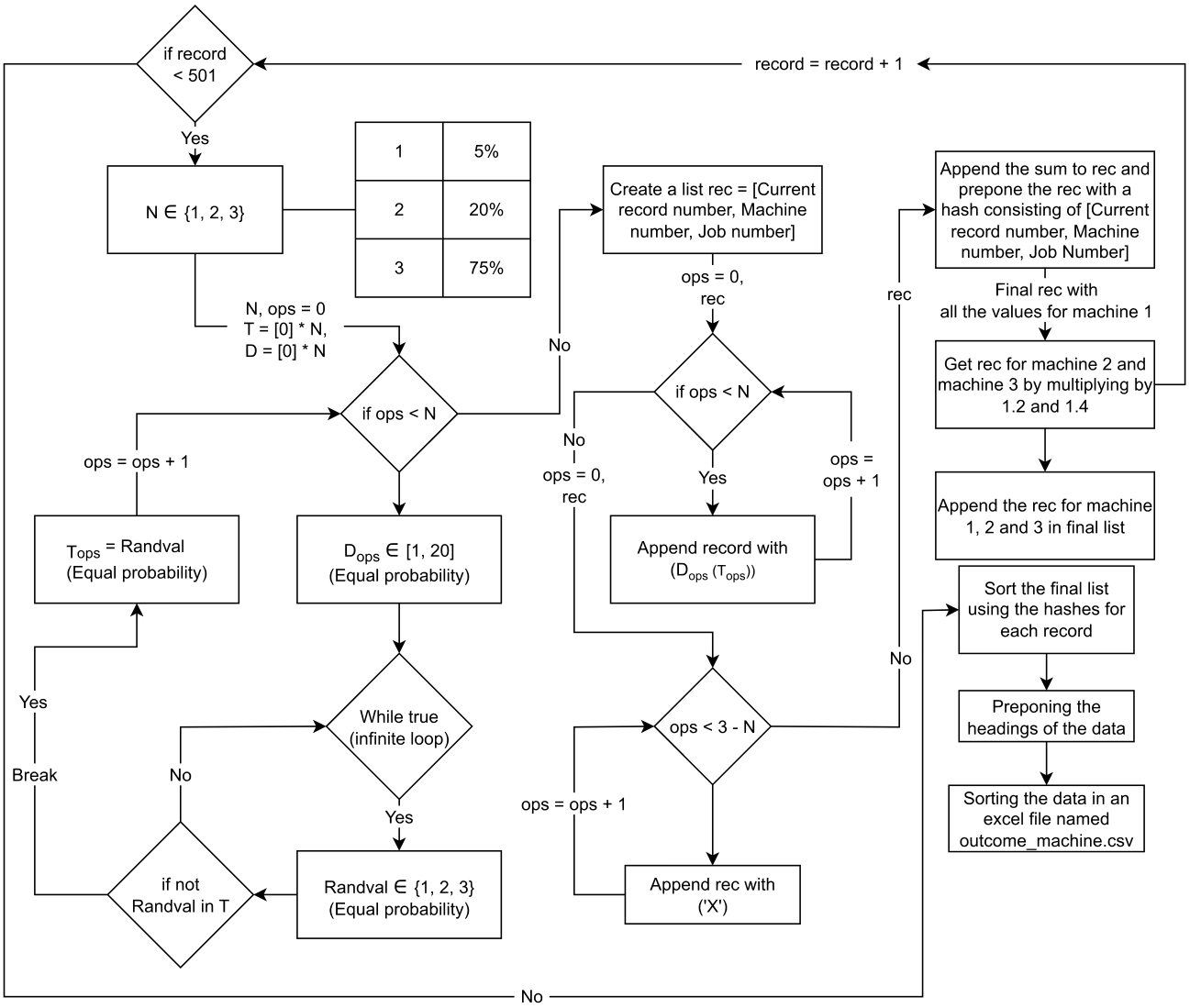

Figure 1. Creation of the processing time data and tool allocation data

Table 1. Processing time and tool allocation data

Hash	Record number	Machine number	Job number	Operation - 1	Operation - 2	Operation - 3	Total
2011	20	1	1	10(3)	19(2)	X	29
2012	20	1	2	15(2)	X	X	15
2013	20	1	3	7(3)	8(2)	4(1)	19
2014	20	1	4	3(1)	14(2)	15(3)	32
2021	20	2	1	12.0(3)	22.8(2)	X	34.8
2022	20	2	2	18.0(2)	X	X	18
2023	20	2	3	8.4(3)	9.6(2)	4.8(1)	22.8
2024	20	2	4	3.6(1)	16.8(2)	18.0(3)	38.4
2031	20	3	1	14.0(3)	26.6(2)	X	40.6
2032	20	3	2	21.0(2)	X	X	21
2033	20	3	3	9.8(3)	11.2(2)	5.6(1)	26.6
2034	20	3	4	4.2(1)	19.6(2)	21.0(3)	44.8

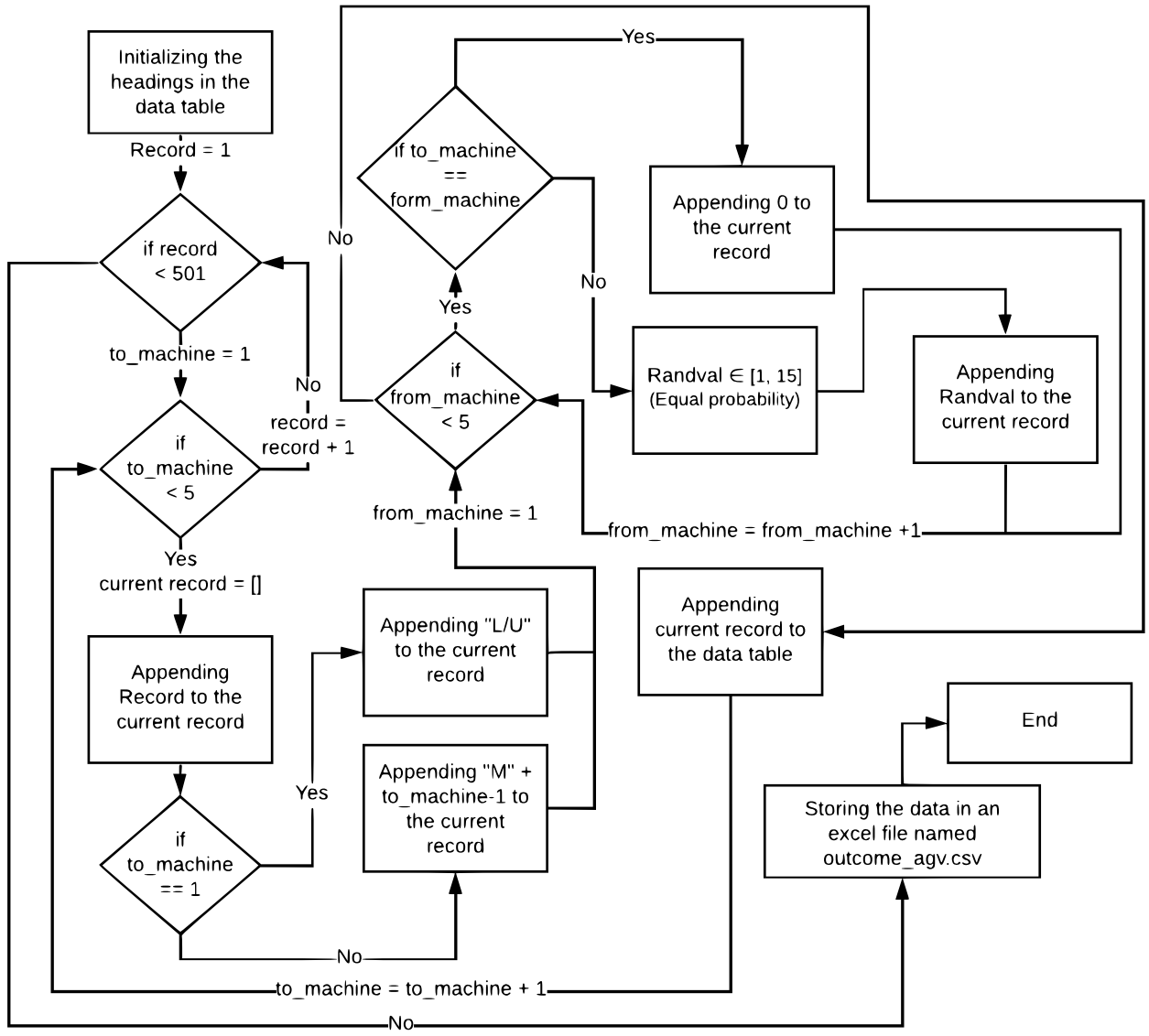

Figure 2. Creation of the AGV travel duration dataset

However, when the source and destination locations are identical, the corresponding matrix entry is assigned a value of 0, representing a null travel time. After completing all iterations, the resulting dataset, comprising 500 such matrices, is compiled and stored in a structured file named 'Outcome_AGVDuration.csv'. Table 2 provides an illustrative excerpt of the generated dataset, capturing the temporal dynamics of AGV mobility across manufacturing system locations.

Table 2. AGV travel duration data

Record number		L/U	M1	M2	M3
20	L/U	0	5	8	11
20	M1	11	0	13	11
20	M2	15	14	0	11
20	M3	15	5	14	0

Single-job transport capacity is considered here. It is also typical in shop floors where AGVs are designed to carry pallets, fixtures, or part bins individually, ensuring safety and preventing tool/job damage during transit. While this modeling choice simplifies AGV scheduling, it still captures the essential bottleneck behavior of transport resources under contention. We acknowledge that more advanced AGVs with heterogeneous speeds, multi-load capabilities, and dynamic routing are increasingly being introduced in smart factories. Incorporating such heterogeneity would increase the scalability of the scheduling model and remain an important direction for future research. Nonetheless, by focusing on the identical single-load AGV case, this work addresses a realistic baseline scenario widely applicable in current manufacturing systems while keeping the scheduling model computationally tractable.

4.3 Cost Data Generation

Figure 3 outlines the algorithm employed for generating cost data, wherein the cost associated with processing each operation is computed based on three primary factors: the

processing time, the tool assigned to the operation, and the machine on which it is executed. The cost estimation model adopts a parametric approach, wherein the processing time is multiplied by a specific cost multiplier, contingent on the tool–machine combination. These multiplier values are referenced from Table 3.

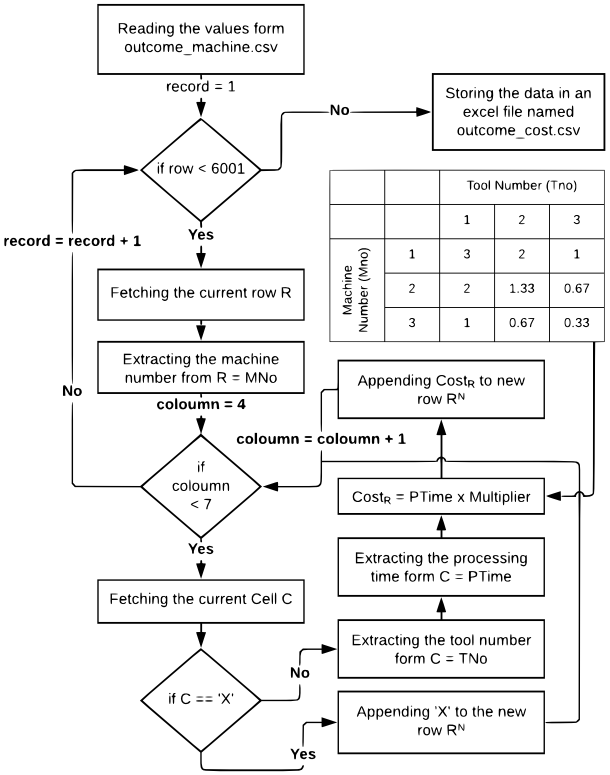

Figure 3. Cost data generation

Table 3. Multiplier Matrix for the cost calculation

	M1	M2	M3
T1	3	2	1
T2	2	1.33	0.67
T3	1	0.67	0.33

The cost data generation process utilizes input from the previously generated dataset, 'Outcome_ProcessingTime.csv', which consists of 500 unique input configurations. Each configuration encapsulates the processing times of four jobs, evaluated on three non-identical machines, resulting in 12 rows per configuration, and 6000 rows overall. The algorithm is initialized by setting a row counter to 1 and extracting the corresponding machine number from the current row. For each row, the fourth column onward is iteratively accessed to retrieve operation-level data. The value in each column is checked: if the value is not equal to 'X', it is parsed to extract the tool identifier and the associated processing time. The cost for the operation is then computed from Eq (1):

$$cost = (Processing\ time)\{(Multiplier)_{(tool,machine)}\} \quad (1)$$

The multiplier is determined in Table 3 based on the specific tool–machine pairing.

Suppose the value in the column is found to be 'X', indicating a placeholder for a non-existent operation. In that case, the cost is recorded as 'X' to maintain structural consistency in the dataset. The algorithm proceeds iteratively across all three possible operations for each job, repeating this process for all 6000 rows. The resulting cost values are systematically compiled and stored in the output file 'Outcome_Cost.csv', ensuring alignment with the format and integrity of the associated processing time and tool allocation datasets. Once each operation's cost is calculated, it is stored in a file, 'Outcome_Cost.csv', as shown in Table 4.

Table 4. Cost data

Hash	Record number	Machine number	Job number	Operation - 1	Operation - 2	Operation - 3
2011	20	1	1	10	38	X
2012	20	1	2	30	X	X
2013	20	1	3	7	16	12
2014	20	1	4	9	28	15
2021	20	2	1	8.04	30.324	X
2022	20	2	2	23.94	X	X
2023	20	2	3	5.628	12.768	9.6
2024	20	2	4	7.2	22.344	12.06
2031	20	3	1	4.62	17.822	X
2032	20	3	2	14.07	X	X
2033	20	3	3	3.234	7.504	5.6
2034	20	3	4	4.2	13.132	6.93

5. Proposed method and implementation

5.1 RNN algorithm

Figure 4 illustrates the proposed Recurrent Neural Network (RNN) algorithm for optimal job sequencing and machine allocation in a flexible manufacturing environment. A distinctive feature of the algorithm is the deferred assignment of jobs to machines. Unlike conventional scheduling methods that initiate machine-job bindings at the outset, this algorithm postpones such decisions until the final phase. This strategy significantly reduces the solution space by focusing on optimal job sequences, each initiated with a unique job assignment. Four placeholder variables—P1, P2, P3, and P4—are introduced, each representing the position of a specific job in the scheduling sequence to facilitate the generation of candidate sequences. All feasible sequences are generated by assigning distinct jobs to these placeholders, and each resulting sequence is evaluated to determine its scheduling efficiency. The sequence yielding the minimum cumulative score is identified as the optimal solution. The algorithm constructs a triangular compatibility matrix that captures pairwise job compatibility scores. These scores are derived from inter-job conflicts, specifically downtime overlap and AGV collision delays.

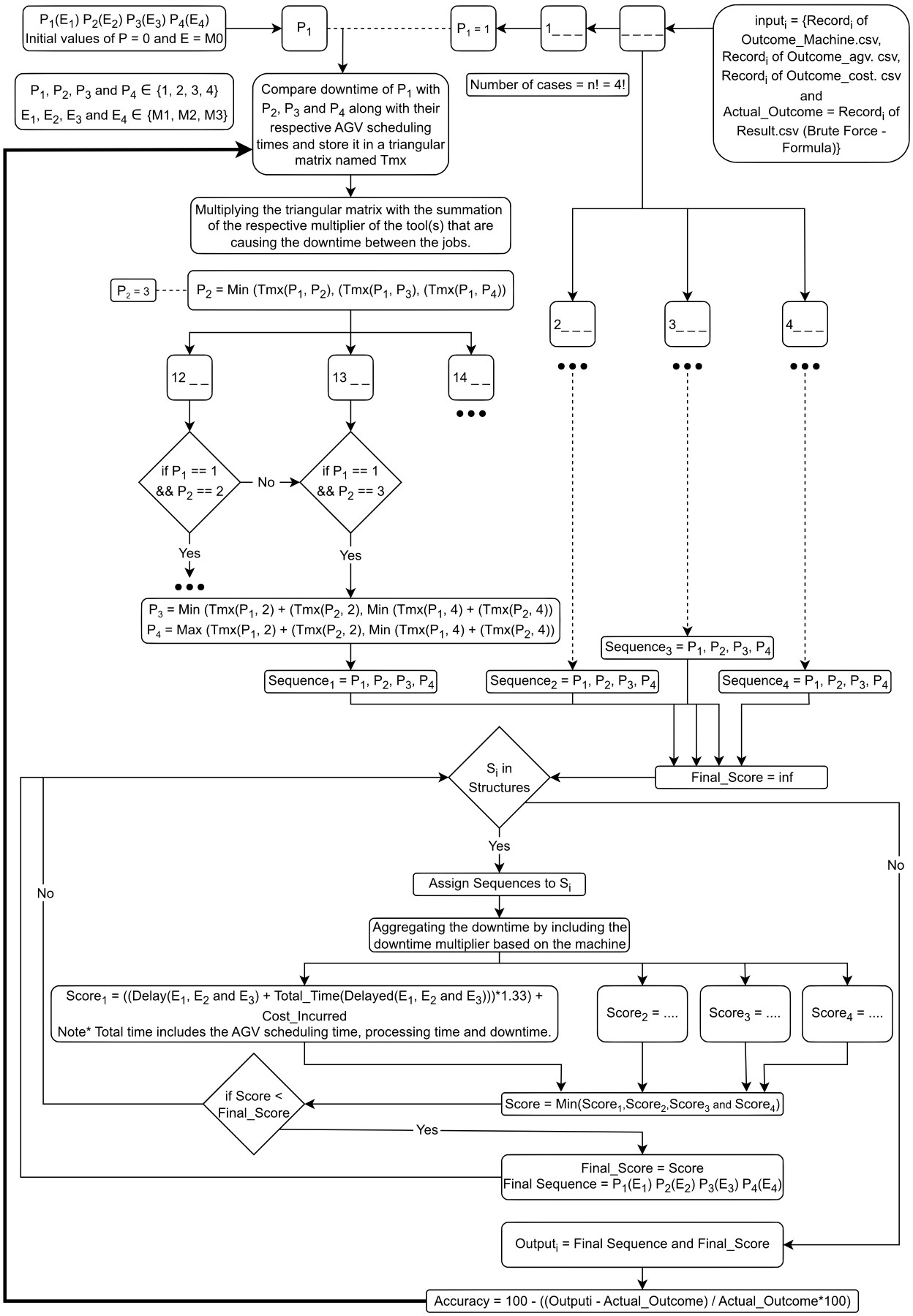

Figure 4. RNN algorithm

They are further weighted using tool-based collision multipliers in the ratio of 3:2:1, based on the severity or likelihood of collision induced by the tools involved. Following the initialization phase, the algorithm explores the solution space through a branching mechanism. The first layer of branches includes four subtrees, each corresponding to a unique assignment of one job to P1. Each of these subtrees further branches into three nodes for possible P2 assignments. The remaining job positions, P3 and P4, are determined based on maximum compatibility with previously assigned jobs, resulting in four complete sequences per iteration. For each generated sequence, the algorithm evaluates all viable machine-job combinations using the combinatorial formula in Eq (2)[17]:

$$number\ of\ combinations = \binom{n}{r} + 1 \quad (2)$$

where:
n – number of jobs,
r – (number of machines – 1)

Each sequence and machine assignment combination is subjected to a simulation process, in which downtimes and AGV conflicts are re-evaluated using the same collision weightings (3:2:1), now applied based on the machines involved in potential conflicts. Each simulation run is scored based on a composite metric integrating makespan and processing cost. Specifically, the combined objective function (score) is computed from Eq (3):

$$combined\ objective\ function = [(makespan)(1.33) + Cost] \quad (3)$$

1.33 represents the average cost multiplier derived from the tool–machine cost matrix (Table 3). Upon completion of all simulations, the sequence with the lowest aggregate score is selected as the optimal job-machine schedule. The accuracy of the result is computed by evaluating the deviation between the predicted and actual optimal values. This error metric is then propagated to the next iteration of the RNN, enabling it to refine predictions through sequential learning and backpropagation.

5.2 Improved RNN algorithm

The Improved Recurrent Neural Network (RNN) algorithm retains the foundational structure of the RNN model shown in Figure 4, described earlier. However, it introduces a significant enhancement in its final decision-making phase. The major improvement is in minimizing combinations for machine allocation, considerably lowering the overall computational complexity. The RNN algorithm attempts all possible job-to-machine assignments, as suggested in the combinatorial formula in Eq (2). The Improved Recurrent Neural Network selectively excludes part of such combinations using a probabilistic filtering technique. That filter is based on patterns of observed frequency in the model's training process. In more detail, the algorithm monitors the frequency of specific machine-job assignment combinations seen within optimal sequences within earlier training runs. Combinations that are infrequently observed or are not chosen in optimal results are considered non-contributory and are therefore excluded from subsequent simulations. The pruning strategy based on statistics ensures that only the statistically most beneficial combinations are retained for simulation, without redundant computations at the cost of solution quality. The Improved RNN algorithm thus exhibits a remarkable acceleration in computation time, enabling rapid convergence while maintaining competitive performance in terms of makespan and cost optimization.

6. Testing

A Leave-One-Out Cross Validation (LOOCV) technique is employed to determine the proposed algorithms' performance and generalization ability. In this approach, each of the 500 generated records is used once as a test set, while the remaining 499 records form the training set. This process is repeated 500 times until every record has been tested exactly once. LOOCV is a recognized validation technique, particularly effective for small- to medium-sized datasets, ensuring that all available data are utilized for training and testing while minimizing sampling bias. LOOCV provides a stricter and more exhaustive evaluation, yielding a more reliable estimate of generalization performance. Given the dataset size and the objective of capturing variability across all 500 records, LOOCV was considered appropriate for this study.

7. Results and analysis

The RNN Algorithm used in this manuscript is the updated version of the RNN phase 3 algorithm, where AGV was not considered, and makespan was the only objective function to minimize. So the RNN phase 3 algorithm was previously compared with the Genetic algorithm, which shows that the RNN phase 3 algorithm outperforms the genetic algorithm in terms of computational time. In the current manuscript, we have incorporated AGV into the RNN algorithm, and a combined objective function is considered to minimize both makespan and cost, taking into account the tool type and machine.

7.1 Scalability and computational efficiency

To thoroughly analyze the computational efficiency of the algorithms, their scalability was assessed by examining the effect of the number of jobs on the total processing time for both the RNN and the Improved RNN algorithms, using three machines, three tool types, and 2 AGVs. As illustrated in Figure 5, the RNN consumes significantly more computational time. This is primarily due to its exhaustive enumeration of all possible machine-job pairings for each potential sequence of jobs. In stark contrast, the Improved RNN algorithm demonstrates superior performance.

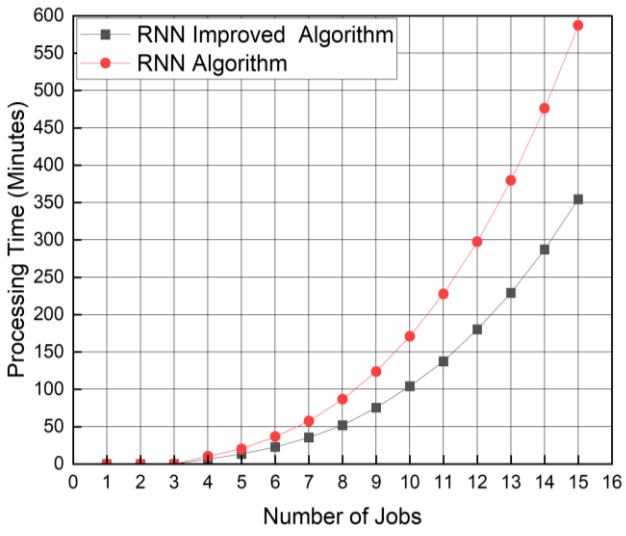

Figure 5. Scalability of algorithms

It strategically leverages its probabilistic filtering mechanism to reduce the number of combinations that need to be evaluated. This targeted reduction results in a considerably slower rate of increase in processing time as the job count rises, showcasing its enhanced scalability under increasing scheduling complexity. Figure 5 provides compelling visual evidence of the Improved RNN's primary performance advantage. The graph clearly shows that as the number of jobs increases, the processing time for the Improved RNN grows at a much slower rate compared to the RNN. This directly supports the claim of superior scalability and quantifies the significant computational efficiency gains achieved by the probabilistic filtering technique. This capability is crucial for handling larger and more complex real-world FMS problems efficiently.

7.2 Objective function aggregation and trade-off impact

The algorithm employs a combined objective function to balance makespan and processing cost during schedule evaluation. These two components are integrated using equal weighting, as expressed in Eq (3). To ascertain the significance of such an aggregation method, experiments were carried out in which optimization was carried out with only cost or makespan as the evaluation criterion. The outcome was that for all cases, omissions of either of these terms resulted in a worsening of total performance, especially with increased jobs. This outcome decisively confirms that a fairly balanced objective function is required for effective scheduling.

The loss of performance that occurs when makespan or cost is individually optimized spotlights one of the most important features of real-world FMS operation: an overall optimal schedule must balance several conflicting objectives. In our manufacturing system, makespan and tool cost are often in conflict. For example, minimizing makespan would likely involve using faster, but more costly, machines or tools (e.g., Tool 1 is costlier than Tool 3). Minimizing tool cost may result in slower operation, as it requires waiting for cheaper tools or machines, thereby increasing makespan. The single-copy tool limitation further exacerbates conflicting objectives. If the algorithm optimizes only for makespan, it would likely result in continuous machine usage, potentially with frequent and costly tool changes. If it is individually optimizing for cost, it would likely result in substantial idle times while machines wait for a less costly tool to arrive.

The noted decline in performance is more than a statistical fluctuation and indicates a failure to formulate an operationally feasible schedule. A schedule with minimal makespan but economically unsustainable tooling costs, or vice-versa, holds limited value for a manufacturer. The combined objective function, therefore, forces the algorithm to find a Pareto-optimal or near-Pareto-optimal solution that balances these critical business objectives, such as time-to-market and cost-efficiency. This approach reflects a more realistic and desirable outcome for industrial operations, ensuring that the generated schedules are technically feasible, economically sound, and aligned with overall business goals. Figure 6 provides empirical evidence directly supporting the argument for multi-objective optimization. It visually demonstrates the performance penalty incurred when makespan and cost are optimized in isolation, particularly as the number of jobs increases. This figure is crucial for validating the multi-objective approach, as it quantifies the benefit of simultaneously considering both time and cost, a critical aspect of practical FMS operations.

Figure 6. Importance of minimizing the makespan and cost

7.3 Machine utilization

Machine utilization is defined as the percentage of time a machine is actively engaged in job processing, and it is inversely related to system downtime and delay. It is calculated using Eq (4):

$$Machine\ Utilization\ \% = \left(\frac{pt}{pt + dt + dl}\right) 100 \quad (4)$$

Where:
pt – Processing Time of all the Jobs,
dt – Downtime of all the Jobs,
dl – Delay of all the Jobs

Figure 7 illustrates the machine utilization percentages across simulations. A general decreasing trend in utilization is observed as the job count increases. This decline is attributed to greater interdependencies and heightened tool contention within the FMS. As more jobs are introduced, the probability of machines idling while awaiting tool release or the completion of operations on other machines significantly increases, leading to more unproductive time.

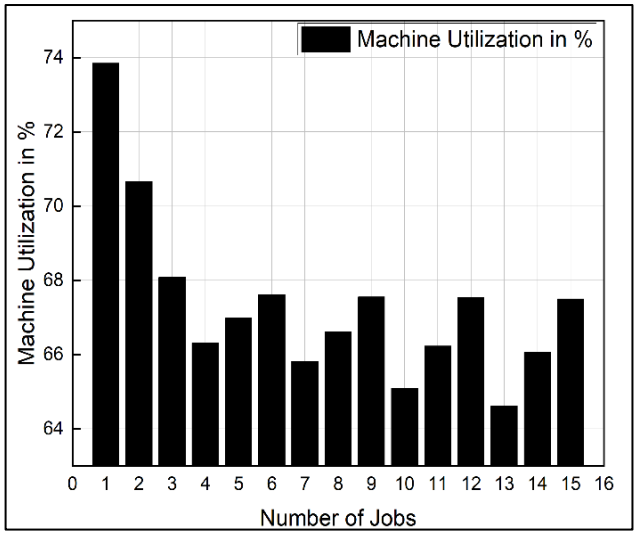

Figure 7. Machine utilization

However, the graph also reveals periodic increases in utilization at multiples of three jobs. These temporary improvements are attributed to reductions in delay, suggesting that at specific job counts, the scheduling algorithm might be more effective at minimizing waiting times, resulting in a transient boost in machine activity. Across all simulations, the average machine utilization was approximately 67%.

7.4 Machine wastage

Machine wastage, directly proportional to downtime, further complements the utilization analysis. It is calculated using Eq (5):

$$Machine\ Wastage\ \% = \left\{1 - \left(\frac{pt}{pt + dt}\right)\right\} 100 \quad (5)$$

Where:
pt – Processing Time of all the Jobs,
dt – Downtime of all the Jobs.

Figure 8 depicts the corresponding machine wastage. As anticipated, given the inverse relationship with utilization, wastage increases as the job count rises. This trend underscores the critical need for improved scheduling strategies, especially in high-load scenarios, to mitigate the negative impact of idle times and delays on overall system efficiency. The corresponding machine wastage across all simulations was observed at 33%. The observed trends in machine utilization and wastage reveal FMS's inherent fragility and bottleneck-prone nature, particularly when operating with single-copy tools. The decreasing utilization and increasing wastage with higher job counts directly stem from the "greater interdependencies and tool contention." In such systems, as more jobs compete for limited resources (tools, machines, AGVs), the likelihood of a machine being idle while waiting for a critical tool or an AGV increases significantly, leading to increased downtime and delays. This highlights the single-copy tool constraint as a primary bottleneck. Even with sophisticated scheduling, if a critical tool is constantly in demand, machines will inevitably experience idle time. The "periodic increases at multiples of three jobs" in utilization are particularly noteworthy.

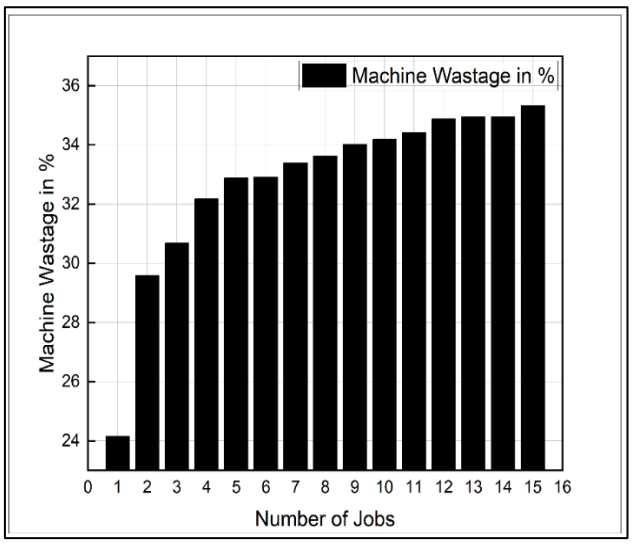

Figure 8. Machine wastage

Given the system's configuration of three machines and three tool types, these might represent specific job batch sizes or configurations where the algorithm can achieve a temporary, more harmonious or balanced distribution of jobs across the available machines and tools. This could lead to a brief reduction in contention and improved flow, suggesting that certain structural properties of the FMS might create specific job loading scenarios more amenable to efficient scheduling. These trends imply that simply increasing the job load in an FMS with tight resource constraints will lead to diminishing returns in efficiency. This has profound implications for capacity planning and for FMS design. Although intelligent scheduling algorithms, such as the Improved RNN, can offset inefficiencies, they cannot compensate for structural bottlenecks resulting from a lack of resources. Once again, the value of a holistic approach lies in the fact that system design and intelligent scheduling reinforce each other.

7.5 Example problem evaluation

To evaluate and compare proposed scheduling algorithm performances, a typical example problem was created with four jobs, three non-identical machines, three single-copy tools, and two Automated Guided Vehicles (AGVs). The configuration represents a typical Flexible Manufacturing System (FMS), for which routing complexity and resource limitation play significant roles in determining scheduling choices. The related data of our example problem—job-operation-tool requirements, AGV traveling times, and processing costs—are represented in Table 5 (Job-operation tool matrix, for example problem), Table 6 (Travel time matrix for the two automated guided vehicles (AGVs) for the example problem), and Table 7 (Cost data for example problem). The corresponding feasible schedule and the final output metrics are summarized in Table 8 (Output obtained for the example problem) and Table 9 (Feasible schedule for the example problem). Table 5 shows the Job-operation tool data for the example problem. Here we have three machines, three tool types, and four jobs, each with a certain number of operations. This data set is generated using the data generation process shown in Figure 1.

Table 5. Job-operation tool data, for the example problem

Machine	Job	Operation No.			Total
M	J	1	2	3	
1	1	3(1)	10(3)	X	13
1	2	16(1)	8(3)	12(2)	36
1	3	16(2)	X	X	16
1	4	18(2)	7(3)	7(1)	32
2	1	3.6(1)	12.0(3)	X	15.6
2	2	19.2(1)	9.6(3)	14.4(2)	43.2
2	3	19.2(2)	X	X	19.2
2	4	21.6(2)	8.4(3)	8.4(1)	38.4
3	1	4.2(1)	14.0(3)	X	18.2
3	2	22.4(1)	11.2(3)	16.8(2)	50.4
3	3	22.4(2)	X	X	22.4
3	4	25.2(2)	9.8(3)	9.8(1)	44.8

Note: The value in parentheses denotes the tool type used to process an operation.

Table 6 shows the Travel time matrix for the two automated guided vehicles (AGVs) for the example problem. Here we have 2 AGVs and the duration of their movement from the load-unload station to different machines and vice versa. This data set is generated using the data generation process shown in Figure 2. Table 7 shows the Cost data for the example problem. This data set is generated using the data generation process shown in Figure 3. Table 8 shows the Output obtained for the example problem. It represents the Output sequence for the example problem, its makespan value, cost value, and combined objective function. Table 9 shows the Feasible schedule for the example problem. In this scenario, the execution of the schedule begins with AGV 1 and AGV 2 transporting Jobs 2 and 4 to Machines 1 and 3, respectively, per the travel time matrix in Table 6. Subsequently, AGV 1 returns to carry Job 1 to Machine 2.

Table 6. Travel time matrix for the two automated guided vehicles (AGVs) for the example problem

	L/U	M1	M2	M3
L/U	0	15	7	15
M1	10	0	4	6
M2	9	14	0	12
M3	12	11	15	0

Table 7. Cost data for example problem

Machine Number	Job Number	Operation - 1	Operation - 2	Operation - 3
1	1	9	10	X
1	2	48	8	24
1	3	32	X	X
1	4	36	7	21
2	1	7.2	8.04	X
2	2	38.4	6.432	19.152
2	3	25.536	X	X
2	4	28.728	5.628	16.8
3	1	4.2	4.62	X
3	2	22.4	3.696	11.256
3	3	15.008	X	X
3	4	16.88	3.23	9.8

Table 8. Output obtained for the example problem

Sequence	Makespan	Cost	Combined Objective
J2(M1),J4(M3),J1(M2),J3(M1)	70.6	165.39	259.288

The allocation of operations is intricately influenced by tool availability and inter-job dependencies, particularly due to the single-copy nature of each tool type. Machine 2 initiates the first operation of Job 1 using Tool 1. However, it encounters a delay before proceeding to the second operation because Tool 3 is simultaneously used by Machine 1 for Job 2. Similarly, Machine 1 experiences delays while waiting for Tool 2, which is allocated to Machine 3 for Job 2's initial operation. These tool-contention scenarios necessitate dynamic rescheduling and highlight the complexity of the scheduling environment. As tools are released and available, AGVs are rerouted to transport remaining jobs, such as Job 3, which is delivered to Machine 1 by AGV 2. The proposed algorithm continuously adapts the sequence of operations to reduce conflicts, downtime, and delays. Its decision-making capability ensures minimal overlapping of tool usage and optimizes machine assignments. These interactions are visualized in Figure 9, which presents a Gantt chart of the complete schedule. The chart utilizes color-coded segments: blue to denote AGV travel time, yellow for active job processing, orange for machine downtime, and green for delays caused by tool unavailability or inter-job waiting. This visualization provides a clear overview of resource contention and scheduling efficiency.

Quantitative performance evaluation of this example problem indicates a makespan of 70.6 units. The total processing cost, incorporating tool-machine-specific multipliers from Table 3, amounts to 165.39 units. Combining these two objectives using a weighted function—where makespan is weighted by a factor of 1.33—yields a final combined objective value of 259.29 units. Additionally, system resource metrics were assessed: the machine utilization rate stands at 75.03%, while machine wastage, attributed to idle times and delays, is recorded at 12.05%. These results validate the proposed algorithm's effectiveness in generating high-quality, feasible schedules under complex constraints. The model adeptly coordinates tool usage and AGV logistics, while minimizing time and cost objectives. Moreover, it demonstrates robustness in handling resource dependencies and bottlenecks, typical challenges in real-world FMS environments. Table 10 indicates the mean computational time of all the designed algorithms.

Table 10. Mean computational time of all designed algorithms

Sr. No	Algorithm	Mean Computational Time
1	RNN	10.33
2	Improved RNN	6.24

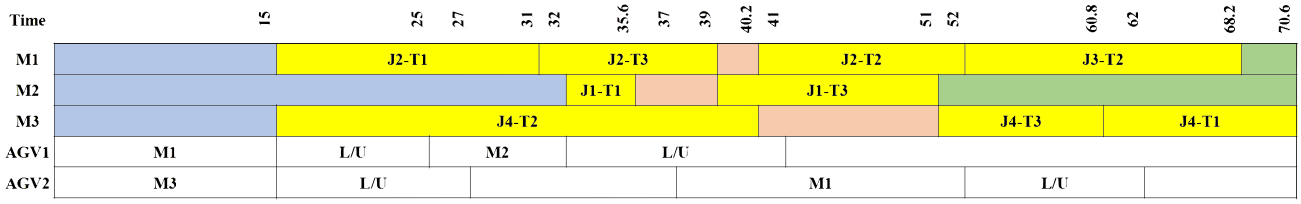

Figure 9. Gantt chart representing the schedule for the example problem

Table 9. Feasible schedule for the example problem

Time	M1			M2			M3			Job Processed			AGV 1	AGV 2
	Tool 1	Tool 2	Tool 3	Tool 1	Tool 2	Tool 3	Tool 1	Tool 2	Tool 3	M1	M2	M3		
1													M1	M3
15													M1	M3
16	1							1		J2		J4	L/U	L/U
25	10							10		J2		J4	L/U	L/U
26	11							11		J2		J4	M2	L/U
27	12							12		J2		J4	M2	L/U
31	16							16		J2		J4	M2	
32			1					17		J2		J4	M2	
33			2	1				18		J2	J1	J4	L/U	
36			5	3.6		0.4		21		J2	J1	J4	L/U	
38			7			2.4		23		J2	J1	J4	L/U	M1
39			8			3.4		24		J2	J1	J4	L/U	M1
40		1				1		25		J2	J1	J4	L/U	M1
41		1.2 0.8				2		25.2	0.8	J2	J1	J4	L/U	M1
42		1.8				3			1.8	J2	J1	J4		M1
43		2.8				4			2.8	J2	J1	J4		M1
44		3.8				5			3.8	J2	J1	J4		M1
45		4.8				6			4.8	J2	J1	J4		M1
46		5.8				7			5.8	J2	J1	J4		M1
47		6.8				8			6.8	J2	J1	J4		M1
48		7.8				9			7.8	J2	J1	J4		M1
49		8.8				10			8.8	J2	J1	J4		M1
50		9.8				11			9.8	J2	J1	J4		M1
51		10.8				12			10.8	J2	J1	J4		M1
52		11.8				1			1	J2		J4		M1
53		12 0.8				2			2	J2 J3		J4		L/U
54		1.8				3			3	J3		J4		L/U
60		7.8				9			9	J3		J4		L/U
61		8.8				10	0.2		9.8	J3		J4		L/U
62		9.8				11	1.2			J3		J4		
68		15.8				17	7.2			J3		J4		
69		16 0.8				18	8.2			J3		J4		
70		1.8				19	9.2					J4		
70.6		2.4				19.6	9.8					J4		

It was observed that the RNN algorithm obtained a mean computational time of 10.33 minutes as a mathematical equation was used, which reduced the number of input configurations for a given number of jobs to get the output sequence by not binding the machines and the jobs at the initial stage of the code. The Improved RNN algorithm obtained a mean computational time of 6.24 minutes, as the combinations that are infrequently observed or are not chosen in optimal results are considered non-contributory and are therefore excluded from subsequent simulations.

7.6 Comparative analysis with existing scheduling approaches

FMS scheduling has seen extensive research, with numerous metaheuristic and evolutionary methods proposed to tackle its inherent complexities. The introduction section provides a comprehensive review of these approaches, including Multi-Objective Simulated Annealing (MOSA), Constraint-Based Genetic Algorithm (CBGA), Adaptive Genetic Algorithm (AGA), Modified Genetic Algorithm (MGA), Differential Evolution (DE), NSGA-II and Pareto Local Search (PLS), various hybrid evolutionary algorithms, Particle Swarm Optimization (PSO), methods incorporating Colored Timed Petri Nets (TCPN), Variable Neighborhood Search (VNS), Mixed Integer Linear Programming (MILP) models, Knowledge-Based Cuckoo Search Algorithm (KCSA), GRASP, Whale Optimization Algorithm (WOA), Discrete Artificial Bee Colony (DABC), Hybrid Multi-Agent Proximal Policy Optimization (HMAPPO), Multi-Agent Reinforcement Learning (MARL), and Recurrent Neural Network (RNN).

Despite this breadth of prior work, a critical research gap was identified: "limited work has addressed scheduling in FMS with non-identical machines and single-copy tools under makespan and tooling cost minimization, while also considering AGV coordination". This highlights a specific, highly integrated problem set that the current RNN-based approach aims to bridge. The proposed RNN approach, particularly the Improved RNN, offers distinct advantages and novel contributions compared to these existing methods. The study's findings indicate that the proposed methods "achieved better performance than traditional metaheuristics in complex FMS scenarios". This suggests a superior capability to navigate the intricate interdependencies and dynamic conflicts inherent in systems with non-identical machines, single-copy tools, and AGV coordination. Many traditional metaheuristics, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Differential Evolution (DE), are powerful search algorithms. They typically operate by defining a solution representation, a fitness function to evaluate solution quality, and operators to explore the search space. Their effectiveness largely depends on how well the complex problem constraints and objectives are explicitly encoded into this fitness function and these operators. For problems with highly dynamic, interdependent, and non-linear constraints, like single-copy tool contention where a tool's availability depends on its current user and the sequence of operations, or AGV routing interacting with machine schedules, explicitly defining these relationships and their impact on makespan and cost can be highly challenging. This often leads to simplified models or heuristics that might miss global optima or struggle with real-world dynamism.

In contrast, RNNs are designed to process sequential data and learn long-term dependencies. In scheduling, decisions are inherently sequential and interdependent: assigning a job to a machine impacts tool availability, which affects other machines, and AGV movements. An RNN can learn these complex, temporal, and non-linear relationships directly from data. It does not require explicit rules for every potential tool conflict or AGV routing scenario; instead, it learns the patterns that lead to efficient conflict resolution and resource utilization across the entire sequence of operations. This enables RNNs to develop an implicit understanding of the system's dynamics and bottlenecks during their training process, thereby moving beyond human-engineered explicit rules and fitness functions. For example, the probabilistic filtering in the Improved RNN is a prime illustration of this learned intelligence, where the algorithm implicitly understands which combinations are unproductive based on observed frequencies. This enables more adaptive and nuanced decision-making. Furthermore, the "complexity" in FMS often arises from the integrated nature of multiple interacting subsystems (machines, tools, AGVs). Various metaheuristic approaches can partition the problem or loosen specific constraints for greater tractability, but often at the cost of a holistic view. In contrast, the RNN approach, by integrating data from different sources, maintains the holistic view necessary to achieve truly optimal combined schedules. This implies a less rigid and more robust system for dealing with highly complex and dynamic scheduling problems, unlike many standard metaheuristic deployments, where explicit representation of complicated, non-linear system behaviors might hinder.

8. Conclusion

This study successfully addressed the complex and dynamic problem of aligning tool and job schedules in Flexible Manufacturing Systems (FMS), especially those involving non-homogeneous machines and the necessary integration of Automated Guided Vehicles (AGVs). The objective was to minimize overall makespan and tooling cost, which is crucial for optimizing operational effectiveness in contemporary manufacturing environments. The study did this by comparing a Recurrent Neural Network (RNN) algorithm with an Enhanced RNN algorithm. The study showed that, while the RNN algorithm was effective, it was accompanied by higher computational requirements owing to its deep exploration of possible scheduling choices. In stark contrast, however, the Enhanced RNN algorithm demonstrated superior performance by significantly reducing processing time. For a standard 4-job case, it achieved optimal scheduling solutions in less than 6 minutes and 30 seconds, representing a 39.6% reduction in computation time from that of the RNN. This enhanced efficiency can primarily be attributed to its innovative probabilistic filtering mechanism, which efficiently removes low-impact machine-job pairings throughout the simulation phase, thereby demonstrating superior computational efficiency and scalability. The efficacy of a multi-objective optimization approach was rigorously proven. Simulations verified that optimizing with equal weight to both makespan and tooling cost gave superior overall scheduling performance compared to optimizing individually for either of those objectives alone. This underscores the importance of a holistic FMS scheduling approach that considers the natural trade-offs between cost and time. Resource usage metrics reflected about 67% average machine usage across all simulation runs, with a related machine wastage of 33%. These numbers reflect difficulties of interdependencies and delays that are part of complex FMS environments, even with prescient scheduling. In conclusion, the Improved RNN algorithm represents a promising direction for developing

scalable, cost-effective, and time-efficient scheduling solutions for intelligent manufacturing systems. Its ability to effectively manage complex resource interdependencies and optimize multiple objectives simultaneously lays a robust groundwork for future, more adaptive and intelligent FMS operations.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically with regard to authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with policies on research ethics. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere.

Data availability statement

The manuscript contains all the data. However, more data will be available upon request from the authors.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] T. Loukil, J. Teghem, and D. Tuyttens, "Solving multi-objective production scheduling problems using metaheuristics," *Eur. J. Oper. Res.*, vol. 161, no. 1, pp. 42–61, 2005, doi: 10.1016/j.ejor.2003.08.029.

[2] A. Kumar, Prakash, M. K. Tiwari, R. Shankar, and A. Baveja, "Solving machine-loading problem of a flexible manufacturing system with constraint-based genetic algorithm," *Eur. J. Oper. Res.*, vol. 175, no. 2, pp. 1043–1069, Dec. 2006, doi: 10.1016/j.ejor.2005.06.025.

[3] J. Jerald, P. Asokan, R. Saravanan, and A. D. C. Rani, "Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 29, no. 5, pp. 584–589, 2006, doi: 10.1007/BF02729112.

[4] I. A. Chaudhry, S. Mahmood, and M. Shami, "Simultaneous scheduling of machines and automated guided vehicles in flexible manufacturing systems using genetic algorithms," *J. Cent. South Univ.*, vol. 18, no. 5, pp. 1473–1486, 2011, doi: 10.1007/s11771-011-0863-7.

[5] N. Kumar, P. Chandna, and D. Joshi, "Integrated scheduling of part and tool in a flexible manufacturing system using modified genetic algorithm," *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, pp. 1596–1607, Nov. 2017, doi: 10.1007/s13198-017-0633-5.

[6] A. Gnanavel Babu, J. Jerald, A. Noorul Haq, V. Muthu Luxmi, and T. P. Vigneswaralu, "Scheduling of machines and automated guided vehicles in FMS using differential evolution," *Int. J. Prod. Res.*, vol. 48, no. 16, pp. 4683–4699, 2010, doi: 10.1080/00207540903049407.

[7] G. Mejía and J. Pereira, "Multiobjective scheduling algorithm for flexible manufacturing systems with Petri nets," *J. Manuf. Syst.*, vol. 54, pp. 272–284, 2020, doi: 10.1016/j.jmsy.2020.01.003.

[8] B. S. P. Reddy and C. S. P. Rao, "A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS," *Int. J. Adv. Manuf. Technol.*, vol. 31, no. 5, pp. 602–613, 2006, doi: 10.1007/s00170-005-0223-6.

[9] F. Zhang and J. Li, "An improved particle swarm optimization algorithm for integrated scheduling model in AGV-served manufacturing systems," *J. Adv. Manuf. Syst.*, vol. 17, no. 03, pp. 375–390, 2018, doi: 10.1142/S0219686718500221.

[10] M. Dotoli and M. P. Fanti, "Coloured timed Petri net model for real-time control of automated guided vehicle systems," *Int. J. Prod. Res.*, vol. 42, no. 9, pp. 1787–1814, 2004, doi: 10.1080/00207540410001661364.

[11] M. Gutjahr, H. Kellerer, and S. N. Parragh, "Heuristic approaches for scheduling jobs and vehicles in a cyclic flexible manufacturing system," *Procedia Comput. Sci.*, vol. 180, pp. 825–832, 2021, doi: 10.1016/j.procs.2021.01.332.

[12] D. B. M. M. Fontes and S. M. Homayouni, "Joint production and transportation scheduling in flexible manufacturing systems," *J. Glob. Optim.*, vol. 74, no. 4, pp. 879–908, 2019, doi: 10.1007/s10898-018-0681-7

[13] W.-Q. Zou, Q.-K. Pan, and L. Wang, "An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery," *Knowl.-Based Syst.*, vol. 218, p. 106881, 2021, doi: 10.1016/j.knosys.2021.106881.

[14] Z. Cao, C. Lin, and M. Zhou, "A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 56–69, 2019, doi: 10.1109/TASE.2019.2945717.

[15] S. Luo, L. Zhang, and Y. Fan, "Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3020–3038, Oct. 2022, doi: 10.1109/TASE.2021.3104716.

[16] Y. Li *et al.*, "Real-Time Scheduling for Flexible Job Shop With AGVs Using Multiagent Reinforcement Learning and Efficient Action Decoding," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 55, no. 3, pp. 2120–2132, 2025, doi: 10.1109/TSMC.2024.3520381.

[17] S. More and N. Kumar, "NONIDENTICAL MACHINE SCHEDULING IN FLEXIBLE MANUFACTURING SYSTEM USING RECURRENT NEURAL NETWORK AND GENETIC ALGORITHM," *Acad. J. Manuf. Eng.*, vol. 23, no. 2, p. 126, 2025, doi: 10.5281/zenodo.15862840.