PESWS-LGBM: A Hybrid Swarm Optimization and LightGBM Framework for Real-Time Production Scheduling in Smart Factories

Minghui Du[1*], Kexin Wang[1], Zhang Ran[2]
[1]School of Information Engineering, Xinxiang Vocational and Technical College, Xinxiang, Hennan,453006, China
[2]School of Architecture, Xinxiang Vocational and Technical College, Xinxiang, Hennan,453006, China
E-mail: taecherdu@163.com, wkx1102@126.com, 787415627@qq.com
*Corresponding author

Keywords: production scheduling, optimization, smart factories, hybrid algorithm, AI

Received: June 16, 2025

The smart factory sector has been a frontrunner in adopting machine learning (ML) technologies to enhance production systems and enable real-time decision-making. However, challenges remain in translating raw sensor data into actionable scheduling strategies for fully autonomous operations. To address this issue, this research proposes a novel hybrid framework, the production-based Elephant Swarm Water Search-Driven Light Gradient Boosting Machine (PESWS-LGBM), which integrates metaheuristic optimization with predictive modeling for real-time production scheduling. The model not only identifies key process parameters (e.g., vibration, power usage, cycle time) but also dynamically generates optimized scheduling decisions, including task sequencing, job-to-machine assignments, and time allocations. These decisions are guided by a dual-stage process in which the LGBM component predicts scheduling performance, and the PESWS component refines scheduling configurations to minimize makespan, reduce job tardiness, and maximize machine utilization. The dataset based on real-time industrial sensor data was preprocessed using noise filtering, missing value imputation, and feature scaling. Experimental results show that PESWS-LGBM significantly improves scheduling outcomes, lowering downtime and material loss while increasing Overall Equipment Effectiveness (OEE). The proposed model achieved strong performance metrics, including accuracy (0.96), Precision (0.91), recall (0.98), and F1-score (0.94). These findings validate the effectiveness of hybrid intelligent systems in enabling adaptive scheduling and improving operational efficiency in smart manufacturing environments.

Povzetek: Za prilagodljivo razporejanje v pametnih tovarnah je predlagan hibrid PESWS-LGBM, kjer LGBM napoveduje učinkovitost urnikov, PESWS pa optimizira zaporedje opravil, dodelitve strojev in časovne okvire.

1 Introduction

The rapid evolution of Industry 4.0 has transformed the manufacturing landscape by emphasizing smart factories with interconnected systems, real-time data collection, and advanced automation. In such an environment, production scheduling has become a crucial enabling function for utilizing resources, maintaining continuous operations, and ensuring timely deliveries. Traditional scheduling methods typically rooted in static heuristics or a rigid optimization framework struggle to keep pace with the more volatile and complex contemporary industrial operational conditions [1]. Smart factories utilize a network of embedded sensors, IoT devices, and Cyber-Physical Systems (CPS) that capture endless amounts of process-level data. This data provides a real-time snapshot of machine availability, energy usage, labor utilization, material flow, and unforeseen disruptions like machine failures or supply chain delays. Consequently, scheduling is no longer a one-time act but is a continuous act that requires dynamic and adaptive responses to an ever-evolving production landscape [2]. The Machine Learning (ML) revolution provides a solution to the complexities of scheduling. ML can detect patterns, learn from the past, and predict future goals to make intelligent systems that can adjust schedules in real time, autonomously. Unlike standard optimization methods, ML-driven scheduling systems can accommodate high-dimensional data and non-linear dependencies to be flexible and robust when problems arise [3]. Artificial intelligence (AI) enhances smart manufacturing by enabling predictive maintenance and automated quality control. These advancements reduce downtime and improve

efficiency, supporting real-time production scheduling. AI-driven strategies like deep reinforcement learning boost scheduling flexibility and adaptability [4]. These systems leverage the predictive capabilities of machine learning and the search capabilities of metaheuristics to generate executable, high-quality schedules. Specifically, these models provide a way to optimize the sequencing of tasks, job-to-machine assignments, and timeslot assignments, while also taking feedback from the shop floor to continuously assess and refine schedules [5]. Intelligent mechanical manufacturing supports rapid market response through AI-driven fault prediction and anomaly detection. These technologies enhance scheduling accuracy and adaptability in smart factory environments [6]. Real-time scheduling plays a crucial role in environments where delayed task execution leads to inefficiencies or obsolescence. This highlights the need for adaptive scheduling mechanisms that respond promptly to dynamic operational conditions in smart factories [7]. As DTs are not bound to a prescribed plan as ERP systems do, they provide higher velocity definitions and execution accuracy of scheduling under uncertainty [8]. Ultimately, the combination of intelligent models, real-time sensor data, and digital twin platforms has transformed planning and scheduling in production, moving from reactive scheduling to actively optimizing or adapting. Increasing complexity in manufacturing paired with ML, metaheuristics, and virtual modeling provides a compelling route to self-organizing factories and resilient operations [9].

Research questions

RQ1: Can the proposed PESWS-LGBM hybrid framework effectively improve production scheduling accuracy and overall equipment effectiveness (OEE) in smart factory environments using real-time industrial data?

RQ2: How does the PESWS-LGBM model compare to existing optimization methods (e.g., DT-DQL, MM-PSO, RL) in terms of computational efficiency, adaptability, and performance metrics such as accuracy, precision, recall, and F1-score?

The contribution of the study is as follows,

- A novel hybrid framework is created to improve the efficiency of the production scheduling system. The study presents the Production-based Elephant Swarm Water Search Driven Light Gradient Boosting Machine (PESWS-LGBM) to identify the key process factors, which open the door to adaptive scheduling and workflow modifications.
- This proposed methodology enhances production efficiency by processing data in real-time and making appropriate decisions.
- Pre-processing steps such as noise filtering, missing value imputation, and feature scaling were used to clean and standardise the dataset, which was derived from real-time industrial data.
- By lowering downtime, allowing proactive modifications, increasing overall equipment effectiveness (OEE), and minimising material loss.

2 Related work

According to the theoretical point of view, scheduling is best understood as a problem-solving exercise in which the production planner dealing with improbability, bottlenecks, and vibrant events. This optimisation problem is hard and separates it from the area of manufacturing planning and control systems [10]. The optimisation community agrees that precise production scheduling is a notoriously difficult problem, with the majority of cases being deemed NP-hard. Not to add that it differs significantly from the typical planning challenges [11]. Scheduling uses accurate data from the shop floor to create schedules in contrast to production planning, this uses aggregate data and presents plans on extended or temporary time range [12]. Scheduling computational methods, in contrast to planning optimisation techniques, aims to minimise time-based performance metrics which are susceptible to numerous distribution boundaries rather than the total cost. Optimal approaches become intractable when the degree of uncertainty rises with decreasing time horizons. Theoretical scheduling problems have prompted the proposal of several estimates and optimisation methods [13]. The reduced processing costs of estimation methods make them more suitable for application on actual manufacturing floors. In the last 20 years, a new trend has developed: using DSSs for full-scale production planning. Software for enterprise resource planning (ERP) or material requirements planning (MRP) can benefit from these solutions as supplemental tools [14]. By integrating state-of-the-art operational research methods, they hope to create distribution networks or shop floor roadmaps. Although these methods are sometimes seen as practical implementations of operational research theories supported by IT and can be useful tools for shop floor control, they are rarely effective when used in practice. To counteract ERP systems' flaws, a lot of scientific effort has gone into developing individualised decision support systems (DSSs) for planning. Table 1 represents related works.

Table 1: List of related studies

Ref	Objectives	Methods	Data Types Used	Complexity	Performance Metrics	Limitation
[15]	This research examines multi-project planning in the assembling procedure for ETO components.	Machine learning	Real-world flexible job-shop data	Medium; tested on 10–12 parts and 8 machines	Total overdue time by ~15%; machine utilization by 10–12%	Connecting efficient procedures with manufacturing tools, as well as process scheduling.
[16]	The study intends to deploy a method of optimization in smart manufacturing systems for co-optimizing the functioning of several manufacturing facilities to meet demands for different commodities.	Optimization technique	Mixed industrial datasets	Not evaluated for scheduling; higher computation due to deep nets	N/A (focus is interpretability)	The scheduling consequences are extremely dependent on the componentseparation decisions.
[17]	In this research, a genetic algorithm is employed to create production times for an enterprise that is adaptable.	Internet of things	NA	NA	N/A	Optimizing large-scale production schedules with numerous variables and constraints can be computationally expensive, especially in real-time.
[18]	This research developed an optimization framework for improving enterprises' manufacturing schedules.	Optimization algorithm	Sensor streams, process logs	Scalable but not real-time optimized	N/A (control-focused)	As the size and complexity of production systems increase, the computational requirements of hybrid algorithms can become prohibitively expensive
[19]	In this research, the genetic algorithm is employed to locate the optimal scheduling approach for the complicated job-shop organization issue, with the number of delayed jobs, overall overdue duration, task completion time, thorough load rate, and the highest load rate of the equipment serving as arranging algorithm effectiveness indications.	Deep learning	Synthetic & small-scale real factory data	Low–medium; single-machine N ≤ 50 jobs	Total weighted lead time by 8–12%, tardiness by 10%	Smart factories often involve complex production systems with multiple variables, constraints, and uncertainties, making it challenging to develop effective optimization algorithms
[20]	In this research offer a hybrid computational architecture and its incorporation into a precision production scheduling platform used in a Greek metal forming industry.	DT technology	Simulated DT sensor streams	Scales to small lots; case study on rolling mill	RL-DQN achieved ~95% of optimal baseline within <1000 episodes	The success of any optimization algorithm depends on the availability of accurate and complete data.
[21]	It examines trends in AI-based planning systems and some of the often used AI methodologies.	Artificial intelligence	Benchmark flow-shop instances + energy use	High; tested up to 20 machines, 100 jobs	Makespan by ~18%, energy by ~22%	The effectiveness of hybrid algorithms relies heavily on high-quality and real-time data, which can be difficult to obtain in practice
[22]	The purpose of the research is to examine current scientific research on the planning challenge, taking into account the use of the DT strategy and the ZDM paradigm to accomplish self-management while reducing or eliminating the requirement for human interaction.	ZDM model	Simulated production planning stream	Medium; training requires ≈1000 episodes	Throughput by 12%, lead time by ~9%	Accurately modeling complex production processes, including interdependencies between operations, machine capacities, and resource constraints, can be challenging.
[23]	According to this research, the more common strategies for solving scheduling issues are optimizing particle swarms, neural network training, and reinforcement learning.	Particle swarm optimization	Sensor health data, machine logs	Medium; tested on simulated 4-machine shop	Machine downtime by ~15%, tardiness by ~10%	Integrating hybrid algorithms with existing production planning and control systems can be complex and require significant investment in infrastructure and training.

3 Materials and methods

Each record corresponds to a snapshot of machine activity, characterized by various sensor readings and performance indicators. These raw features were subsequently transformed into scheduling-relevant instances using preprocessing techniques. Table 2 below summarizes the key features present in the dataset, along with their descriptions and relevance to production scheduling. These features were later aggregated into synthetic job instances based on time-window segmentation, enabling the simulation of job-shop scheduling problems.

Table 2: Dataset features

Feature Name	Description
timestamp	Real-time data capture in minute-level intervals
Machine id	Unique identifier for each machine
Machine type	Type of production equipment
temperature	Machine temperature (°C)
Vibration level	Vibration intensity reflects wear or misalignment
Power consumption	Energy usage in kilowatts (kW)
pressure	System pressure in pascals (Pa)
Material flow rate	Speed of material movement through the system
Cycle time	Duration of one production cycle (seconds)
Error rate	Proportion of defective products in the cycle
downtime	Machine inactivity time (minutes)
Maintenance flag	Indicates if maintenance is required (0 = No, 1 = Yes)
Efficiency score	Overall production efficiency (0 to 100)
Production status	Binary indicator of efficiency (0 = Inefficient, 1 = Efficient)

3.2 Preprocessing

Preprocessing is the process of cleaning and standardizing raw data for analysis. Techniques like noise filtering, missing value imputation, and feature scaling enhance data quality, ensuring accuracy and consistency in ML models. The raw sensor data were segmented into fixed time windows to form individual job instances, each associated with a specific machine and operational snapshot. Statistical aggregation techniques (e.g., mean, max) were applied to features such as temperature, vibration, and cycle time. These processed job records were then labeled based on production status and used to simulate scheduling tasks for model training.

i. Noise filtering

This noise filtering is applied to the input dataset to enhance accuracy in production system optimization. A Gaussian filter is utilized to reduce sensor noise, mitigating disruptions in data readings. By adjusting standard deviation and mean values, the method minimizes distortions, ensuring reliable data for real-time decision-making and workflow adjustments. These are represented in equations (1) and (2).

$$I^G(i,j) = I' \times (i,j) = I'\left(\frac{1}{2\pi\sigma^2} e^{\frac{(-i^2-j^2)}{2\sigma^2}}\right) \quad (1)$$

$$\sigma^2 = E(X^2) - [E(x)]^2 \quad (2)$$

In these particular situations, the $I^G(i,j)$ represents the filtered sensor data after the application of the Gaussian noise filtering method. Then σ^2 represents noisy sensor data variance. The parameters $iandj$ are the displacements from the origin in the x and$y$ directions, respectively. $E(X^2)$is the value of the sensor data measured at a point. The filtering method used by the model facilitates data enhancement, which leads to superior accuracy when making real-time decisions and adapting workflow systems.

ii. Feature scaling

Industrial sensor data varies in units and magnitudes, requiring feature scaling for balanced ML contributions. Common techniques include normalization (Min-Max Scaling) and standardization (Z-score Scaling), to standardize data ranges and distributions effectively.

Min-Max Scaling normalization adjusts data to a fixed range [0,1] using equation (3). This ensures uniform feature contribution.

$$Q' = \frac{Q - Q_{min}}{Q_{max} - Q_{min}} \quad (3)$$

Z-score Standardization (Scaling) – Converts data into a normalized distribution with an average of 0 and distribution range of 1, as shown in equation (4).

$$Q' = \frac{Q - \mu}{\sigma} \quad (4)$$

Where, normalized data point is represented as (Q'), original value of the variable as (Q), mean of the dataset as (μ), and standard deviation of the dataset as (σ).

iii. Missing value imputation

The system is required to manage incomplete industrial real-time AI-driven production scheduling system sensor data caused by transmission errors or sensor errors. Missing data can affect ML model accuracy and decision-making. Techniques like mean/median imputation replace missing values with dataset averages, interpolation estimates values based on trends, and regression imputation predicts missing values using statistical models, ensuring reliable and consistent data analysis. These preprocessing steps improve data quality, reduce errors, and ensure that the ML model receives clean, structured, and reliable input for accurate predictions and decision-making.

3.3 Hybrid model - *Production Based Elephant Swarm Water Search (PESWS)- Light Gradient Boosting Machine (LGBM)*

The PESWS-LGBM model integrates ML and metaheuristic optimization for AI-driven production optimization. This model predicts critical process parameters and fine-tunes workflow using adaptive exploration-exploitation mechanisms, ensuring real-time scheduling adjustments, resource efficiency, and improved OEE in smart factories.

The PESWS-LGBM model incorporates predictive learning and metaheuristic-based optimization into the scheduling module of the real-time production planning and scheduling framework. In this model, the PESWS algorithm designs comprehensive schedules by solving three main aspects of scheduling: (1) sequence of tasks is the order in which jobs are executed, (2) assignment to machines, and (3) timing of work start and finish for each task. The model optimizes scheduling outputs to achieve low makespan, low tardiness of jobs, and high machine usability, thereby improving Overall Equipment Effectiveness (OEE).

The PESWS-LGBM model is designed to generate full production schedules by integrating predictive learning with metaheuristic optimization. Specifically, it maps real-time input features such as temperature, vibration, cycle time, and energy usage to key scheduling outputs, including Task sequencing (the execution order of jobs), Job-to-machine assignments, Time window allocations (start and end times), and resource availability constraints. This mapping enables the system to optimize scheduling objectives such as minimizing makespan, reducing job tardiness, lowering downtime, and maximizing machine utilization. The framework works in a dual-stage fashion: LGBM first predicts performance outcomes based on input configurations, and PESWS then iteratively searches the solution space for optimal schedule configurations guided by those predictions.

The LGBM portion of the PESWS-LGBM model is treated as a multi-class classifier, where the target labels are some categories of scheduling outcome. Labeling comes from the performance class of the scheduling configuration. Class 0 shown in (2.1) represents Low efficiency (high makespan, high downtime); Class 1 represents Moderate efficiency; Class 2 represents High efficiency (low makespan, low tardiness, high OEE). For each candidate schedule produced by PESWS, LGBM evaluates the candidate scheduling configuration and classifies it as one of the potential performance classes (0, 1, or 2). This classification is leveraged as a multi-class fitness score for the optimization. Through iterations, PESWS will evolve candidate schedules into the "high-efficiency" class (class 2). Dataset had many IoT-driven attributes. LGBM used specified importance ranking where it was able to select important feature attributes such as vibration level, cycle time, power consumption, and downtime. These features deliver PESWS optimizers parameters which will directly govern job sequencing, machine assignment, tracking priority. Further, this integration confirms that we are using scheduling modes that take into account data derived operational attributes.

Light Gradient Boosting Machine (LGBM)

LGBM is an enhanced-performance, gradient-boosting framework based on ML. It is designed to improve efficiency and scalability compared to traditional boosting algorithms like XGBoost. LGBM enhances training speed by using histogram-based algorithms, reducing memory consumption, and employing a leaf-wise growth strategy with depth constraints to prevent overfitting.

LGBM optimizes decision tree learning with four key techniques. ML converts continuous features into discrete bins, reducing memory usage and speeding up computations. Gradient-Based One-Side Sampling (GOSS) retains steep-gradient instances while randomly sampling minimal-gradient ones, improving efficiency without significant accuracy loss. Exclusive Feature Bundling (EFB) groups mutually exclusive features, reducing dimensionality and memory consumption. The leaf-wise growth strategy selects the leaf with the highest information gain for expansion, improving model accuracy, though a depth constraint is needed to prevent overfitting. These optimizations make LGBM highly efficient for large datasets.

Training dataset is given by,$W = \{(W_j, z_j)\}_{j=1}^{n}$. The goal is to find an approximate function $\hat{e}(w)$that closely estimates $e^*(w)$to minimize expected values of specific loss operations $(z, e(w))$, The objective function mathematical representation is shown in equation (5).

$$\hat{e}(w) \underset{e}{\arg\, min} F_{z,W} K(z, e(w)) \tag{5}$$

This function minimizes the expected loss operation to optimize predictions. Equation (6) represents the additive learning formula.

$$e_S(W) = \sum_{s=1}^{S} e_s(W) \tag{6}$$

LGBM builds a model as a sum of regression trees. Leaf weight optimization is represented in equation (7).

$$\omega_i^* = -\frac{\sum_{j \in J_i} h_j}{\sum_{j \in J_i} g_j + \lambda} \tag{7}$$

Here, $h_j and g_j$are the primary and secondary gradients of the loss operation, optimizing the tree's leaf nodes. Tree structure optimization (Extreme Values of Γ_S^*) is denoted by equation (8).

$$\Gamma_S^* = -\frac{1}{2}\sum_{i=1}^{I}\frac{(\sum_{j\in J_i} h_j)^2}{\sum_{j\in J_i} g_j+\lambda} \quad (8)$$

Equation (9) determines the best feature split during tree growth.

$$H = \frac{1}{2}\left(\frac{(\sum_{j\in J_i} h_j)^2}{\sum_{j\in J_i} g_j+\lambda} + \frac{(\sum_{j\in J_q} h_j)^2}{\sum_{j\in J_q} g_j+\lambda} + \frac{(\sum_{j\in J} h_j)^2}{\sum_{j\in J} g_j+\lambda}\right) \quad (9)$$

Where, final model after$S$ boosting iterations is $e_S(W)$, $e_s(W)$ is the weak learner at iteration (s), optimal leaf weight is (ω_i^*), regularization parameter (controls complexity) is (λ), optimized tree structure value is (Γ_S^*), set of samples in the parent node is (J), set of samples in the right child node is (J_i), and gain function, measures the quality of a split (H).

LGBM's efficient structure enables fast training and accurate predictions, making it ideal for large-scale ML tasks. The hybrid algorithm for production scheduling system optimization improves training speed, reducing memory consumption, by utilizing a leaf-wise growth strategy. It efficiently predicts critical process parameters, enabling real-time adaptive scheduling, workflow adjustments, and enhanced overall production effectiveness. In the context of scheduling, LGBM is not merely a predictor of feature importance. Rather, it evaluates the predicted performance of candidate scheduling configurations. These predictions serve as fitness scores that guide the PESWS optimizer in refining the schedule. Each sample in the training set represents a synthesized job-machine-time instance, constructed from the preprocessed sensor data. Thus, LGBM functions as a surrogate performance evaluator that estimates the expected quality of scheduling decisions proposed by PESWS.

Production Based Elephant Swarm Water Search (PESWS)

The PESWS algorithm is employed as a metaheuristic optimization approach to fine-tune process parameters within the AI-enabled production system. The algorithm mimics the collective intelligence and social behavior of elephant swarms, optimizing the system by minimizing downtime, material waste, and inefficiencies. The PESWS algorithm is an innovative version that was inspired by the herd behavior of elephants and their water-searching behavior. Compared to traditional swarm optimizers, PESWS weighs exploration and exploitation differently, with inertia weight adjustment and movement that is context aware, exploiting both local and global optimal values. It is helpful to visualize "production units" as elephant herds. Each group of elephants is allowed to iteratively update scheduling parameters based on performance feedback. This structure enhances convergence speed and robustness to local optima while allowing better processing of high-dimensional scheduling variables. Keep in mind that PESWS is not simply a tuning optimization, but rather a structurally distinct metaheuristic for adaptive real-time production scheduling.

Elephant Groups as Process Solutions

In the optimization framework, each elephant group represents a potential solution for process optimization, adapting production parameters dynamically to enhance efficiency and minimize resource consumption. Each elephant group represents a potential configuration of production parameters, such as scheduling, resource allocation, and workflow adjustments.

The position of an elephant group in the search space is defined by equation (10).

$$U_{j,c}^s = U_{j1,}U_{j2,} \dots \dots \dots U_{jc} \quad (10)$$

Where, $U_{j,c}^s$ represents the **d-dimensional** position of the j^{th} **elephant group** at iteration s.

Each elephant group encodes a candidate scheduling solution, where position vectors represent a complete job-machine-time mapping. The algorithm evaluates each group based on key scheduling KPIs such as makespan, average tardiness, and idle time. The best-performing configurations are selected and evolved across iterations.

Velocity Update Based on Best Found Solutions

The velocity of each elephant group is adjusted using local and global best solutions, ensuring adaptive learning that refines production strategies for optimal scheduling and reduced operational inefficiencies. The velocity of each elephant group is updated to optimize decision-making based on local and global search strategies represented in equation (11).

$$U_{j,c}^{s+1} = U_{j,c}^s * \omega^s + \varepsilon \odot (H_{best,c}^s - W_{j,c}^s) \quad (11)$$

$$U_{j,c}^{s+1} = U_{j,c}^s * \omega^s + \varepsilon \odot (H_{besti,c}^s - W_{j,c}^s) \quad (12)$$

Where the global best solution (optimal production setting) is $(H_{best,c}^s)$, the local best solution (best found by a specific elephant group) is $(H_{besti,c}^s)$, the inertia weight for balancing exploration and exploitation is (ω^s), if $rand \leq o$ [for global search], the random factor affecting movement direction is (ε) .

Objective Function

The objective of the PESWS optimization process is to generate a schedule that minimizes overall makespan, reduces machine idle time, and satisfies job precedence and

resource constraints. To achieve this, we define a multi-objective loss function using Equation (13):

$$L = \alpha.Makespan + \beta.Total_{Tardiness} + \lambda.Constraint\_Violation\_Penalty \quad (13)$$

Where:

- Makespan is the total completion time of all jobs,
- Total Tardiness is the cumulative delay beyond job due times.
- Constraint Violation Penalty accounts for overlap, sequence violation, or machine overloading.
- α, β, and λ are tunable weights that balance these objectives.

Tie PESWS to Scheduling Elements

In the context of production scheduling, each elephant group's position vector Uj,cU_{j,c}Uj,c represents a complete scheduling configuration, which includes Job sequencing, Job-to-machine assignments, and Start and end times for each task.

The velocity update equations guide how the current schedule (position) is modified in the search space. For example, a position shift may correspond to reordering jobs, reallocating a task to a different machine, or adjusting its time slot. The global and local best solutions are determined based on the fitness value calculated from the scheduling objective function $\{L\}$. This ensures that updates are directed toward more efficient and constraint-satisfying schedules.

Penalty Formulation

Constraint violations such as two jobs overlapping on the same machine or violating precedence rules are penalized using Equation (14):

$$Penalty_{Overlap} = \sum_{j,i} \delta_{ji} . D \quad (14)$$

Where δ_{ji} if Job j and Job i overlap on the same machine, and D is a fixed penalty constant. These penalty terms are included in the total loss function to guide the optimization toward feasible schedules.

Updating Production Parameters

The algorithm continuously updates critical production parameters based on optimized solutions, enabling real-time workflow adjustments in environments to minimize downtime, improve efficiency, and enhance overall process performance. The new production system configuration is updated and is denoted by equation (15).

$$\omega^s = \omega_{max} - \left\{\frac{\omega_{max}-\omega_{min}}{s_{max}}\right\} \times s \quad (15)$$

This process iteratively refines ML models to predict optimal production workflow. PESWS updates it **dynamically** using equations (16) and (17).

$$W_{j,c}^{s+1} = U_{j,c}^{s+1} + U_{j,c}^{s} \quad (16)$$

$$o(s) = o_{max} - \left(\frac{o_{max}-o_{min}}{s_{max}}\right) * s \quad (17)$$

Where, $o_{max} = 1$, $o_{min} = 0$, $s_{max}$is is the maximum iteration count.

This approach ensures a stronger global search initially and a more refined local search as iterations progress.

Velocity and Position Updates in PESWS

Velocity and position update in PESWS remains the same; **manage search adaptability and**new production system, which is stated by following equations (18) and (19).

$$W_{j,c}^{s+1} = U_{j,c}^{s} * \omega^s + \epsilon \odot (H_{best,c}^{s} - W_{j,c)}^{s} \quad (18)$$

$$W_{j,c}^{s+1} = U_{j,c}^{s+1} + U_{j,c}^{s} \quad (19)$$

Where, ($W_{j,c}^{s+1}$) - Updated position at time ($s+1$), Current position at time - (s), Inertia weight controlling exploration-exploitation balance - (ω^s), Updated velocity used for position update - ($U_{j,c}^{s+1}$), Current velocity at time - (s).

Figure 2 compares velocity and position updates in **(a) PSO**(Particle Swarm Optimization) and **(b) PESWS**. In **PSO**, particle movement is influenced by **current velocity**, **particle memory (**$H_{besti,c}^{s}$**)**, and **swarm memory (**$H_{best,c}^{s}$**)**, ensuring global and local search. In P**ESWS**, an adaptive mechanism adjusts movement based on a probability factor($rand$), and enhances the exploration-exploitation balance, preventing premature convergence. It improves

production efficiency by fine-tuning workflow parameters more effectively than static PESWS.

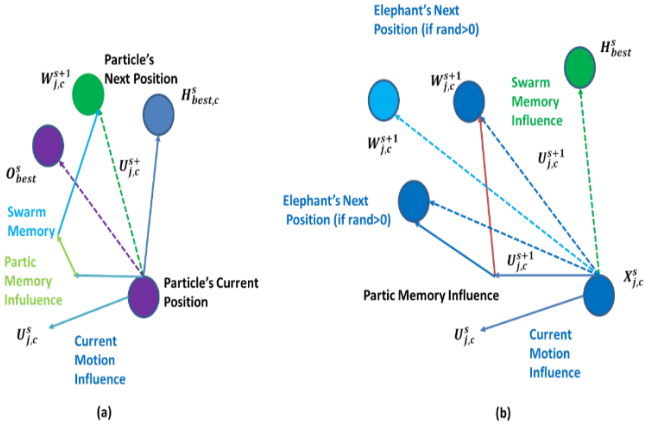

Figure 2: Visualization of velocity and position updates in (a) PSO (b) PESWS approach.

The PESWS-LGBM framework optimizes complete production schedules by predicting and refining job start times, task sequences, and machine assignments. LGBM estimates scheduling KPIs such as expected downtime, machine load, and makespan based on real-time input data, and PESWS adjusts scheduling configurations to minimize delays and maximize efficiency. The LGBM is the automated performance assessor. For each schedule produced with PESWS, the LGBM model predicts important scheduling KPIs. These predictions are taken as the fitness function within the PESWS optimization loop, meaning that the PESWS logic does not suffer the costs of simulated evaluations and hence can more quickly converge on an optimal schedule. Thus, PESWS generates and decides scheduling decisions, and LGBM will provide data-driven performance feedback on those scheduling decisions. Adaptive scheduling is the aspect of the model that allows for updates to the schedule results based on current impact features from the production context. The input features can be real-time sensor inputs such as vibration, temperature, and cycle time. The PESWS-LGBM framework continually assesses the performance of the schedule it produced using the predicted KPIs and dynamically re-optimizes the ordering of tasks, job-to-machine assignments, and time buckets. The nature of the re-optimization loop allows for real-time adjustment to schedules at the occurrence of disturbances, such as machine failure and load imbalance. Algorithm 1 shows the steps involved in production by integrating PESWS-based parameter tuning with LGBM, enhancing predictive accuracy and adaptive scheduling for efficiency. This approach ensures that the PESWS-LGBM model outputs executable production schedules, not alone priority scores or predictions. These schedules are structured in a job–machine–time matrix format and are validated in a digital twin environment to ensure feasibility and responsiveness. The framework dynamically adapts to changing input features, allowing real-time rescheduling and proactive adjustment of job flows.

Algorithm 1: PESWS-LGBM for Production Scheduling Optimization

Input:
- *Dataset D (production data with scheduling targets)*
- *Population Size = 30*
- *Max Iterations = 100*
- *ω_max = 0.9 (initial inertia weight)*
- *s_max = 0.5 (maximum step size)*
- *β = 0.4 (influence of water source)*
- *Number of Clans = 3*

Output:
- *Optimized LGBM model with best parameters*
- *Improved production scheduling outcome*

Step 1: Initialize Population

For each elephant i in population (i = 1 to 30):
- *Randomly initialize position Xi = [x1, x2, x3, x4] ∈ [0,1]*
- *Set velocity Vi = 0*
- *Evaluate fitness Fi = objective(Xi)*

- Store global best position X_best with minimum Fi
- Set water_density = 0.7

Step 2: Optimization Loop (Repeat for 100 iterations)

For iteration t = 1 to 100:
For each elephant i:
- *Update velocity:*
  *Vi = ω * Vi + β * rand(0,1) * (X_best - Xi)*
- *Update position:*
  Xi = Xi + Vi
- *Apply bounds: Xi ∈ [0,1]*
- *Evaluate new fitness Fi*
- *Update X_best if Fi improves*

- Divide elephants into 3 clans (10 elephants each)
For each clan:
- *Identify clan leader (best Fi in clan)*
- *For each non-leader elephant:*
  - *Move toward leader:*
    *Xi = Xi + rand(0,1) * (leader - Xi)*
- *Replace worst member with random solution*

- Optionally update ω:
*ω = ω_max * (1 - t / max_iterations)*

Step 3: Extract Best Hyperparameters

From X_best = [x1, x2, x3, x4]:
*- num_leaves = 20 + int(x1 * 100) → e.g., 20 to 120*

*- learning_rate = 0.01 + x2 * 0.19 → e.g., 0.01 to 0.2*

*- max_depth = 3 + int(x3 * 10) → e.g., 3 to 13*

*- min_data_in_leaf = 10 + int(x4 * 90) → e.g., 10 to 100*

Step 4: Train LGBM Model

- Train LightGBM using optimized hyperparameters on dataset D

- Evaluate performance (e.g., accuracy, OEE, runtime, etc.)

Return:

- Best LGBM model

- Best production scheduling metrics

End

3.4 Digital twin construction

A virtual replica of the physical production environment was created using the real-time dataset that contains sensor-derived parameters such as temperature, vibration, cycle time, error rate, and machines' operational states. The data are constantly synchronized between the physical system and the virtual model. The DT models machine behaviors and production line states under different scheduling strategies. It is effectively a sandbox environment for the PESWS optimizer to explore many scheduling options prior to real-world execution. To assess the adaptiveness and performance of the PESWS-LGBM model, it is compared against static scheduling methods using First-Come-First-Serve (FCFS), with a definition of fixed job priorities. The comparison is different from the benchmark because the PESWS-LGBM model dynamically adapts schedules during the simulation runs, similar to how sensor feedback changes.

Twin Pairing: The virtual twin communicates with the physical system through data feedback loops. The changes in the physical sensors are reflected in the twin, and the optimized schedules determined by the PESWS-LGBM model in the twin are pushed back to the physical system for execution in real-time.

The final output of the PESWS-LGBM model is a fully structured production schedule represented in matrix form. In this matrix, each row corresponds to a job or task, while each column represents a specific machine. The individual cells within the matrix contain detailed scheduling information, including the assigned machine, start time, and end time for each job. This structured format facilitates the execution of scheduling decisions in real manufacturing environments. To better interpret and validate these outputs, the matrix was visualized as a Gantt chart or job-shop timeline within the digital twin simulation. These visual representations enabled the evaluation of the model's feasibility, responsiveness, and performance in terms of resource utilization, task sequencing, and adherence to time constraints. As shown in Table 3, the PESWS algorithm dynamically tunes the hyperparameters of the LightGBM model.

Table 3: LightGBM Hyperparameters Tuned by PESWS

Hyperparameter	**Symbol**	**Search Range**	**Description**	**Mapping from X_best (xi ∈ [0, 1])**
Number of Leaves	num leaves	20 – 120	Controls complexity of individual trees. Higher values allow more splits.	Num leaves = int(20 + xi × 100)
Learning Rate	Learning rate	0.01 – 0.2	Shrink's contribution of each tree; lower values need more trees.	Learning rate = 0.01 + xi × 0.19
Maximum Tree Depth	Max depth	3 – 13	Limits the maximum depth of trees to control overfitting.	Max depth = int(3 + xi × 10)
Min Data in Leaf	Min data in leaf	10 – 100	Minimum number of data points per leaf. Controls tree growth.	Min data in leaf = int(10 + xi × 90)
Feature Fraction	Feature fraction	0.6 – 1.0	Fraction of features used per tree (random subset for each tree).	Feature fraction = 0.6 + xi × 0.4 (optional)
Bagging Fraction	Bagging fraction	0.5 – 1.0	Fraction of data used for each boosting round (subsampling).	Bagging fraction = 0.5 + xi × 0.5 (optional)
Lambda L2	Lambda l2	0 – 10	L2 regularization term to avoid overfitting.	lambda_l2 = xi × 10

4 Results and discussion

This section describes the experimental design and performance evaluation of the PESWS-LGBM approach. It details the performance measures, and comparison analysis, identifying the model's effectiveness in optimizing hybrid model-driven factory structures.

4.1 Experimental setup

The research was conducted on a Windows 10 laptop equipped with an Intel i5 processor and 16 GB of RAM. All experiments were implemented in Python 3.12.1, using well-established libraries for machine learning and optimization. To ensure reproducibility and fairness, each model was executed over 10 independent runs, and a fixed random seed was applied to control for variability across experiments. Performance metrics (accuracy, precision, recall, F1-score, makespan, tardiness, flow time, and machine utilization) were reported as mean values, and statistical significance tests were applied to verify that the improvements of the PESWS-LGBM model over baseline methods were not due to random variation.

The ROC-AUC curve (Figure 3) demonstrates the strong classification performance of the PESWS-LGBM model, achieving an AUC of 0.97. This high value indicates excellent discrimination between scheduled and unscheduled production events. The curve's robustness further confirms the model's reliability, even under class imbalance, by effectively balancing true positive and false positive rates across thresholds.

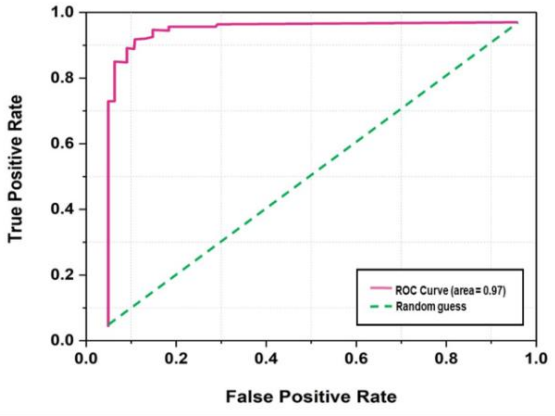

Figure 3: ROC-AUC Curve

Figure 4 illustrates the comparative performance of the PESWS and PESWS-LGBM (proposed) models across three scheduling difficulty levels Low, Medium, and Moderate using four key metrics: makespan (Figure 4 (a)), tardiness (Figure 4 (b)), flow time (Figure 4 (c)), and machine utilization (Figure 4 (d)). Across all difficulty levels, the PESWS-LGBM model consistently outperforms the baseline PESWS model. It achieves lower makespan and average tardiness, indicating more efficient job sequencing and reduced delay. Similarly, flow time per job is shorter under PESWS-LGBM, demonstrating improved throughput. Notably, machine utilization is also higher for the proposed model, reflecting better resource allocation and reduced idle time. This overall improvement confirms the efficacy of the hybrid PESWS-LGBM approach in dynamic production scheduling scenarios.

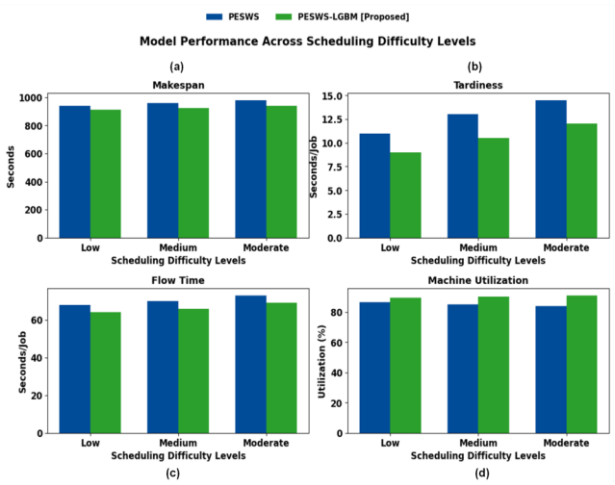

Figure 4: Scheduling Difficulty Levels across Metrics: (a) makespan, (b) tardiness, (c) flow time, and (d) machine utilization.

Certain performance metrics are analyzed that include accuracy, recall, precision, and F1-score to evaluate the PESWS-LGBM model. Accuracy evaluates the proportion of correct predictions, reflecting overall model effectiveness. Precision assesses the ratio of True Positives (T_p) to all predicted positives, minimizing False Positives(F_p). Recall evaluates the model's ability to identify critical process parameters. The F1-score, a harmonic mean of precision and recall, ensures balanced performance, particularly for imbalanced manufacturing datasets. These metrics are crucial for validating the efficiency of the PESWS-LGBM optimization framework, with their equations (18 to 21) detailed in the following sections.

$$Accuracy = \frac{T_p+T_N}{T_p+T_N+F_p+F_p} \tag{18}$$

$$Precision = \frac{T_P}{T_P+F_p} \tag{19}$$

$$Recall = \frac{T_P}{T_P+F_p} \tag{20}$$

$$F1\ score\ =\ 2\ \times \frac{Precision \times Recall}{Precision + Recall} \quad (21)$$

Where, T_N- True Negatives,F_N – False Negatives.

The proposed method PESWS-LGBM is compared with the existing methods Digital Twin- Deep Q learning (DT-DQL) [20], Multi-objective Memetic–Particle Swarm Optimization (MM-PSO) [21], and Reinforcement Learning (RL) [22]. Table 4 and Figure 5 show the comparison results based on accuracy and precision.

Table 4: Comparison results based on accuracy and precision

Methods	**Accuracy**	**Precision**
DT-DQL [20]	0.92	0.85
MM-PSO [21]	0.93	0.87
RL [22]	0.95	0.89
PESWS-LGBM [Proposed]	**0.96**	**0.91**

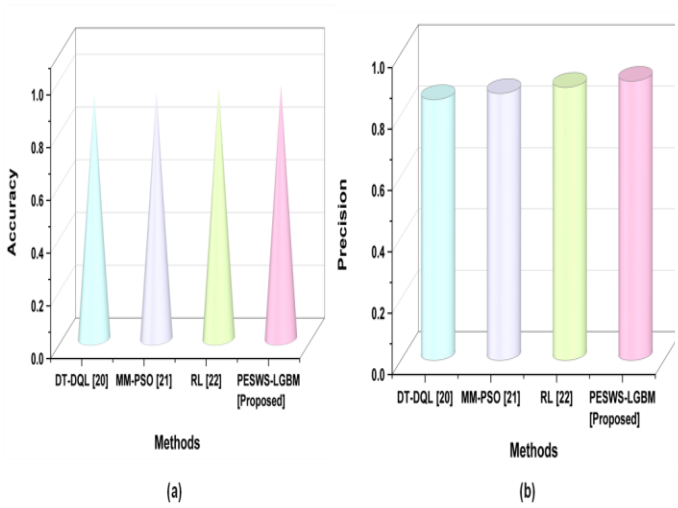

Figure 5: Visual Representations of (a) Accuracy and (b) Precision

Table 4and Figure 5 compares the performance of DT-DQL, MM-PSO, RL, and the proposed ESWS-LGBM based on accuracy and precision. DT-DQL achieves 0.92 accuracy and 0.85 precision, while MM-PSO improves to 0.93 accuracy and 0.87 precision. RL further enhances accuracy to 0.95 and precision to 0.89. The proposed PESWS-LGBM outperforms all, achieving 0.96 accuracy and 0.91 precision, demonstrating superior predictive performance for optimizing smart factories through DT-driven real-time process adjustments. Table 5 and Figure 6 demonstrate the comparison results of recall and F1 score.

Table 5: Comparison Results based on Recall and F1-Score

Methods	**Recall**	**F1-Score**
DT-DQL [20]	0.88	0.87
MM-PSO [21]	0.96	0.90
RL [22]	0.97	0.92
PESWS-LGBM [Proposed]	**0.98**	**0.94**

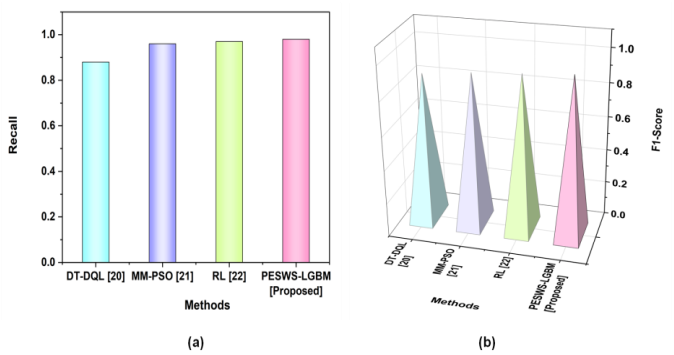

Figure 6: Visual Representation of (a) Recall and (b) F1-score

Table 6and Figure 6 present a comparison of recall and F1-score across DT-DQL, MM-PSO, RL, and the proposed PESWS-LGBM. DT-DQL achieves 0.88 recall and 0.87 F1-score, while MM-PSO improves to 0.96 recall and 0.90 F1-score. RL further enhances recall to 0.97 and F1-score to 0.92. The proposed ESWS-LGBM method outperforms other methods, achieving 0.98 recall and 0.94 F1-score, proving superior performance in accurately detecting critical process parameters and ensuring optimized smart factories. This research emphasizes the role of hybrid technology in driving productivity and efficiency, reinforcing its widespread adoption in the manufacturing sector.

Table 6 presents a sample schedule generated by the PESWS-LGBM model, illustrating how it optimizes job allocation across machines while minimizing delays. The table demonstrates improved timing (e.g., reduced initial delays) and production outcomes (e.g., early or on-time completion). This instance-level output bridges the gap between the proposed methodology and its practical impact in real-world scheduling.

Table 6: Example schedule generated by PESWS-LGBM

Job ID	Assigned Machine	Start Time	End Time	Initial Delay	Optimized Delay	Status
J1	M2	09:00	09:40	8 min	2 min	Completed Early
J2	M1	09:10	09:45	5 min	1 min	On-Time
J3	M3	09:45	10:30	12 min	3 min	Completed Late
J4	M1	10:00	10:30	6 min	2 min	On-Time
J5	M2	10:15	10:50	7 min	2 min	Completed Early

Table 7 presents the ablation study conducted to analyze the individual and combined contributions of different components in the proposed PESWS-LGBM pipeline. The baseline LGBM-only configuration (without PESWS or any preprocessing) achieves an accuracy of 0.90, indicating the strong standalone predictive capability of the LGBM model. However, when PESWS is integrated without proper data preprocessing, accuracy slightly drops (0.87-0.88), likely due to unmitigated noise and missing data.

The entire proposed configuration with noise filtering, feature scaling, missing values imputation, and PESWS-LGBM achieved the greatest accuracy of 0.96 with a notable improvement. This suggests that PESWS improves LGBM's alignment with scheduling when preprocessing is adequately handled. It is evident that preprocessing is a critical enabling function in unleashing the power of PESWS and, together, the combination of PESWS and LGBM provides better outcomes than either separately. This study shows that the best performance was not only from using LGBM, but from the integrated pathways of all components of the pipeline, particularly PESWS and data preprocessing.

Table 7: Ablation study comparing the effect of individual and combined components

Configuration	Noise Filtering	Feature Scaling	Missing Value Imputation	PESWS	LGBM	Accuracy
Noise Filtering + PESWS-LGBM	✓	✗	✗	✓	✓	0.87
Noise Filtering + Feature Scaling + PESWS-LGBM	✓	✓	✗	✓	✓	0.88
LGBM-only	✗	✗	✗	✗	✓	0.90
Noise Filtering + Feature Scaling + Missing Value Imputation + PESWS-LGBM (Proposed)	✓	✓	✓	✓	✓	**0.96**

To assure reproducibility, performed 5-fold stratified cross-validation (Table 8), to achieve a balanced class distribution within each fold. The metrics reported (accuracy, precision, recall, F1-score) are averaged across five independent folds, with a fixed random seed to maintain the same level of variability across studies.

Table 8: 5-Fold Cross-Validation Results (LightGBM)

Fold	Accuracy	Precision	Recall	F1 Score
Fold 1	0.9150	0.9020	0.9293	0.9154
Fold 2	0.9050	0.9082	0.8990	0.9036
Fold 3	0.9150	0.8879	0.9500	0.9179
Fold 4	0.9450	0.9406	0.9500	0.9453
Fold 5	0.9250	0.9048	0.9500	0.9268
Average	**0.9210**	**0.9087**	**0.9357**	**0.9218**

4.2 Discussion

The literature has recently researched various optimization strategies within smart manufacturing. Author of [18] demonstrated the feasibility of controlling smart factories

through a cloud-based IoT framework, which facilitated data acquisition in real-time but did not include a scheduling optimization or adaptive learning component. According to the research [21], a hybrid PSO and Q-learning method for energy-efficient flow-shop scheduling showed significant improvements in energy savings and makespan reduction but had high computational complexity with larger problems. In [23], the authors proposed a dynamic scheduling model driven by the health of machines that relied on real-time sensors utilized predictive maintenance to assign jobs, although this study focused on small shop floor interventions. Compared to these approaches, the proposed PESWS-LGBM framework achieves higher performance across multiple metrics while also reducing downtime and enhancing Overall Equipment Effectiveness (OEE). Its integration of LightGBM for critical feature prediction and PESWS for adaptive optimization offers greater scalability, faster training, and better generalizability. Unlike prior models, it enables real-time, noise-resilient, and flexible scheduling, validating its effectiveness for dynamic smart factory environments.

4.3 Challenges and implications of research

Integrating the PESWS-LGBM model into a practical smart factory scenario brings forth numerous ethical and practical implications. In human-in-the-loop situations, maintaining operator trust in an AI-informed scheduling system will be essential. Operator rejection or misinterpretation of recommendations may occur if operators lose confidence due to over-reliance on full automation without any process transparency. Ensuring explainability and letting human operators take over when appropriate could minimize this risk.

Sensor reliability is also a significant issue. When sensors have faults or drift, incorrect data inputs lead to poor model performance and reduced data quality through time. Careful calibration of the sensors and fault detections of sensor performance are required. The dynamic nature of the industrial environment means that the model will precipitate new conditions that occur with changing production performance and equipment behavior. Data privacy and system security are also concerns when deploying systems of this type. Although this solution enables adaptive scheduling to benefit productivity and efficiency, this framework needs to consider automation using and relying on new forms of autonomy, human oversight, system durability, and longevity.

5 Conclusion

The effectiveness of a hybrid algorithm for optimizing production scheduling systems was analyzed. Real-time data was collected from industrial sensors, capturing essential production metrics such as temperature, vibration, power consumption, and material flow rate. Preprocessing techniques, including noise filtering, missing value imputation, and feature scaling, ensure data quality and standardization. By leveraging the ESWS-LGBM method to predict critical process parameters, adaptive scheduling and workflow adjustments were enabled. Results demonstrated that the proposed model outperformed existing methods, achieving 0.96 accuracy, 0.91 precision, 0.98 recall, and an F1-score of 0.94, demonstrating superior predictive capabilities. The integration of artificial intelligence technology enables real-time simulations, further improving operational efficiency. These results confirm that PESWS-LGBM significantly enhances OEE, reinforcing its potential for widespread adoption in smart factories to drive productivity, cost efficiency, and intelligent automation. Despite its effectiveness, the proposed model faces challenges such as high computational requirements, dependency on high-quality sensor data, and integration complexities in diverse industrial settings. Future research can explore lightweight models, hybrid optimization techniques, and enhanced digital twin and machine learning integration for improved scalability, real-time adaptability, and broader industrial applications.

References

[1] Jalal, A.M.; Toso, E.A.V.; Morabito, R. Integrated Approaches for Logistics Network Planning: A Systematic Literature Review. Int. J. Prod. Res. 2022, 60, 5697–5725. DOI:10.1080/00207543.2021.1963875

[2] Thenarasu, M.; Rameshkumar, K.; Rousseau, J.; Anbuudayasankar, S.P. Development and Analysis of Priority Decision Rules Using MCDM Approach for a Flexible Job Shop Scheduling: A Simulation Study. Simul. Model. Pract. Theory 2022, 114, 102416. https://doi.org/10.1016/j.simpat.2021.102416

[3] Wu, J.; Huang, Z.; Hu, Z.; Lv, C. Toward Human-in-the-Loop AI: Enhancing Deep Reinforcement Learning via Real-Time Human Guidance for Autonomous Driving. Engineering 2023, 21, 75–91. https://doi.org/10.1016/j.eng.2022.05.017

[4] Zhao, Y. Optimization of Mechanical Manufacturing Processes via Deep Reinforcement Learning-Based Scheduling Models. Informatica 2025, 49. https://doi.org/10.31449/inf.v49i14.7204

[5] Hossain, Q.; Yasmin, F.; Biswas, T.R.; Asha, N.B. Integration of Big Data Analytics in Management Information Systems for Business Intelligence. Saudi J. Bus. Manag. Stud. 2022, 9, 192–203. DOI:10.36348/sjbms. 2024.v09i09.002

[6] Wang, H. Innovative Application of Intelligent Mechanical Manufacturing Based on Self-Supervised Learning and Graph Neural Network Fusion Optimization. Informatica 2025, 49. DOI: https://doi.org/10.31449/inf.v49i17.7595

[7] Xue, W.; Luo, Q.; Ni, L.M. Real-Time Action Scheduling in Pervasive Computing. Informatica 2011, 35.

[8] Yang, S.; Xu, Z. Intelligent Scheduling and Reconfiguration via Deep Reinforcement Learning in Smart Factories. Int. J. Prod. Res. 2022, 60, 4936–4953. DOI:10.1080/00207543.2021.1943037

[9] van der Beek, T.; Souravlias, D.; van Essen, J.; Pruyn, J.; Aardal, K. Hybrid Differential Evolution Algorithm for the Resource Constrained Project Scheduling Problem with a Flexible Project Structure and Consumption and Production of Resources. Eur. J. Oper. Res. 2024, 313, 92–111. DOI: 10.1016/j.ejor.2023.07.043

[10] Kasapidis, G.A.; Paraskevopoulos, D.C.; Mourtos, I.; Repoussis, P.P. A Unified Solution Framework for Flexible Job Shop Scheduling Problems with Multiple Resource Constraints. Eur. J. Oper. Res. 2024, 320, 479–495. DOI: 10.1016/j.ejor.2024.08.010

[11] Rohaninejad, M.; Tavakkoli-Moghaddam, R.; Vahedi-Nouri, B.; Hanzálek, Z.; Shirazian, S. A Hybrid Learning-Based Meta-Heuristic Algorithm for Scheduling of an Additive Manufacturing System Consisting of Parallel SLM Machines. Int. J. Prod. Res. 2022, 60, 6205–6225. DOI:10.1080/00207543.2021.1987550

[12] Luo, X.; Hu, Y.; Yu, X. Design and Application of Scheduling Algorithm Based on Multi-Objective and Multi-Constraint. Manuf. Technol. Mach. Tool 2022, 04, 159–164. DOI:10.1109/ACCESS.2023.3285710

[13] Hong, L.; Wang, Y.; Nan, K.; Tian, H. Scheduling Optimization Study of Timed Petri Net Models for Flexible Manufacturing Systems. Mach. Des. Manuf. 2022, 04, 262–265, 269. DOI:10.1109/CCDC62350.2024.10587856

[14] Huang, Z. Research on Flexible Workshop Dynamic Real-Time Scheduling Based on Hybrid Genetic Algorithm. North China Inst. Aerosp. Eng. 2022. [Thesis] https://doi.org/10.3390/app14062586

[15] Bao, B.; Duan, Z.; Xu, N.; Zhang, H.; Luo, Y.; Wang, W.; Yu, X.; Luo, Y.; Liu, X. A New Algorithm of the Scheduling of a Flexible Manufacturing System Based on Genetic Algorithm. Manuf. Rev. 2023, 10, 11. https://doi.org/10.1051/mfreview/2023010

[16] Li, X.; Xiong, H.; Li, X.; Wu, X.; Zhang, X.; Liu, J.; Dou, D. Interpretable Deep Learning: Interpretation, Interpretability, Trustworthiness, and Beyond. Knowl. Inf. Syst. 2022, 64, 3197–3234. DOI:10.1007/s10115-022-01756-8

[17] Matsuo, Y.; LeCun, Y.; Sahani, M.; Precup, D.; Silver, D.; Sugiyama, M.; Morimoto, J. Deep Learning, Reinforcement Learning, and World Models. Neural Netw. 2022, 152, 267–275. DOI: 10.1016/j.neunet.2022.03.037

[18] Khan, S.I.; Kaur, C.; Al Ansari, M.S.; Muda, I.; Borda, R.F.C.; Bala, B.K. Implementation of Cloud-Based IoT Technology in Manufacturing Industry for Smart Control of Manufacturing Process. Int. J. Interact. Des. Manuf. 2023, 1–13. DOI:10.1007/s12008-023-01366-w

[19] Sang, Y.-W.; Wang, J.-Q.; Sterna, M.; Błażewicz, J. Single Machine Scheduling with Due Date Assignment to Minimize the Total Weighted Lead Time Penalty and Late Work. Omega 2023, 121, 102923. DOI: 10.1016/j.omega.2023.102923

[20] Müller-Zhang, Z.; Kuhn, T.; Antonino, P.O. Towards Live Decision-Making for Service-Based Production: Integrated Process Planning and Scheduling with Digital Twins and Deep-Q-Learning. Comput. Ind. 2023, 149, 103933. DOI: 10.1016/j.compind.2023.103933

[21] Zhang, W.; Li, C.; Gen, M.; Yang, W.; Zhang, G. A Multiobjective Memetic Algorithm with Particle Swarm Optimization and Q-Learning-Based Local Search for Energy-Efficient Distributed Heterogeneous Hybrid Flow-Shop Scheduling Problem. Expert Syst. Appl. 2024, 237, 121570. DOI:10.1007/s10845-023-02227-9

[22] Esteso, A.; Peidro, D.; Mula, J.; Díaz-Madroñero, M. Reinforcement Learning Applied to Production Planning and Control. Int. J. Prod. Res. 2022, 61, 5772–5789. https://doi.org/10.1080/00207543.2022.2104180

[23] Yin, L.; Zhang, W.; Zhou, T. Machine Health-Driven Dynamic Scheduling of Hybrid Jobs for Flexible Manufacturing Shop. Int. J. Precis. Eng. Manuf. 2023, 24, 797–812. DOI:10.1007/s12541-023-00784-w