Performa: Media Ilmiah Teknik Industri
ISSN 1412-8624 (print) | ISSN 2620-6412 (online)
Vol. 25, No. 2, 2026, Page. 119-129 doi.org/10.20961/performa.v25i2.3246

Time Prediction Using Artificial Neural Networks for the Permutation Flow Shop Problem: Predict-Then-Optimize Framework

Denny Suci Prastiya*[1]
[1] Department of Industrial Engineering, Engineering Faculty, Bakti Indonesia University, Jln Kampus Bumi Cempokosari Cluring, Banyuwangi Regency, East Java, 68482, Indonesia
E-mail: denny@ubibanyuwangi.ac.id[1]

Abstrak

Studi ini memperkenalkan implementasi prediksi-kemudian-optimasi (PTO) sebagai revolusi riset operasional (OR) di bidang teknik industri. Dalam kontribusi ini, penelitian ini mengusulkan model jaringan saraf tiruan (ANN) dengan multi-layer perceptron (MLP) 4-20-4 menggunakan penurunan gradien pelatihan dan pembelajaran dengan momentum dalam proses backpropagation sebagai kerangka prediksi. Sementara itu, untuk model yang dioptimalkan, penelitian ini mengusulkan pemrograman linier bilangan bulat campuran (MILP) untuk masalah penjadwalan flow shop permutasi (PFSP) menggunakan algoritma NEH. Selanjutnya, peneliti menguji parameter momentum, gradient, dan laju pembelajaran untuk memvalidasi kinerja MSE terbaik dalam model ANN. Solusi terbaik dengan akurasi tertinggi digunakan untuk mensimulasikan prediksi penjadwalan. Selain itu, dalam kasus penjadwalan, penelitian ini membandingkan model flow shop FIFO umum berdasarkan kondisi yang ada dengan model PFSP yang diusulkan, menggunakan hasil ANN yang terdiri dari 4 mesin dan 4 pekerjaan, berdasarkan simulasi prediksi ANN. Selanjutnya, penelitian ini mengusulkan model PFSP yang dapat menyusun ulang pekerjaan menggunakan algoritma NEH dengan tujuan meminimalkan makespan. Hasilnya menunjukkan bahwa dalam kerangka PTO, prediksi adaptif dan kondisi optimal dapat mencapai solusi keputusan terbaik. Pada akhirnya, penelitian ini berhasil membuktikan secara praktis dan teoritis bahwa dua topik terbesar di era saat ini, yaitu pembelajaran mesin (ML) dan riset operasional (OR), dapat bekerja sama dalam kerangka model PTO.

***Kata Kunci**: Artificial Neural Network, Backpropagation, Gradient Descent, NEH Algorithm, Permutation Flow Shop Scheduling*

Abstract

This study introduces the implementation of predict-then-optimize (PTO) as a revolution in operational research (OR) in the industrial engineering field. In this contribution, we proposed an artificial neural network (ANN) model with a 4-20-4 multi-layer perceptron (MLP) using training and learning gradient descent with momentum under the backpropagation process as a prediction framework. Meanwhile, for the optimized model, we proposed mixed integer linear programming (MILP) for the permutation flow shop scheduling problem PFSP case using the NEH algorithm. Subsequently, we tested momentum, gradient, and learning rate parameters to validate the best performance of MSE in the ANN model. The best solution with the highest accuracy is used to simulate a scheduling prediction. Moreover, in the scheduling case, we compare the general flow shop FIFO model based on existing conditions with the proposed PFSP model, using the results of ANNs that consist of 4 machines and 4 jobs, based on the ANNs' prediction simulation. Furthermore, we proposed a PFSP model that can re-sequence jobs using the NEH algorithm with aims for minimize makespan. The results show that in the PTO framework, adaptive prediction and optimal conditions can achieve the best decision solution. Ultimately, this study successfully proves, in both practical and theoretical aspects, that the two biggest topics in the current era, that is, machine learning (ML) and operational research (OR), can work together under the PTO framework model.

Keywords: Artificial Neural Network, Backpropagation, Gradient Descent, NEH Algorithm, Permutation Flow Shop Scheduling

1. Introduction

Indonesian manufacturing sectors in the current situation have encountered the highest uncertainty conditions, where production demand is more sensitive to greater competition with the national and international industrial sectors (Badan Pusat Statistik Indonesia, 2026). Collaboration between artificial intelligence (AI) and manufacturing will have an impact on accelerating the performance of technology in the production process (Gao et al., 2024). Meanwhile, several countries have been successfully developing programs for the industrial revolution under AI implementation, such as the United

[1]* Denny Suci Prastiya
Received 1st February 2026; Received in revised form 13th September 2026; Accepted 23th September 2026

States (US), the United Kingdom (UK), Germany, France, Japan, and China, where these countries can achieve increased production quantity, performance quality with the highest precision, and better prediction accuracy in planning (Mao et al., 2019).

Nowadays, business planning in the current era has become more complex with the presence of automation in the production process. On the other hand, the appearance of AI is making business more competitive, which can impact to enhancing demand uncertainty. In automation, production tends to focus on mass-production characteristics, including more machines, system interconnection, and higher energy requirements to run the production line (Xin et al., 2023). Based on these cases, the flow shop has been developing a permutation (prmu) in the flow shop problem (PFSP) to overcome these situations. However, PFSP still needs to find the best parameter settings for determining sequences of jobs that minimize makespan.

The combination of operational research (OR) and machine learning (ML) has the highest popularity in recent research, as proven in (Zhou et al., 2023; Y. Li and Yu, 2025; Tang and Dong, 2024; Liu et al., 2022). In addition, the integration of OR and ML has opened the gate to a research framework known as predict-then-optimize (PTO), where the integration of OR and ML can improve decision-making for planning in a dynamic environment (Grumbach et al., 2024; Ren and Liu, 2024). The PTO framework has been studied in (Nasseri et al., 2025; Anis Lahoud et al., 2025; Prastiya and Wahyuni, 2024), which shows that ML is used as a predictive tool as well as an input parameter for optimization.

Meanwhile, in this research (Zhou et al., 2023), it is shown that the result of permutation flow shop optimization PFSP is used as input for predicting to minimize makespan using deep reinforcement learning (DRL), that shown integration between OR and ML frameworks. Subsequently, integration between OR and ML frameworks was also conducted in this research (Liu et al., 2022; Mu et al., 2024; Lee, 2025; Y. Li and Yu, 2025), which is known as integrated prediction and optimization, where the optimization and prediction processes were conducted simultaneously.

The research by (Lee, 2025) used integrated prediction and optimization to predict machine faults under optimized conditions. This research also shows that the model can dynamically adapt to real-time conditions. In other research, integration between OR and ML is conducted by (Yan et al., 2022), where ML can adjust predictions better in manufacturing scheduling, and an ideal condition has been achieved by utilizing an optimization model.

The framework of predict-then-optimize (PTO), as well as integrating prediction and optimization, are different stage processes but share the same focus of utilizing ML and OR simultaneously. For instance, in the integrating framework, model optimization and machine learning can join to train models to make decisions based on prediction results, and the PTO framework utilizes a two-step process for training parameters in machine learning to minimize error, which can be used as input for an OR optimization model with parameter accuracy.

By utilizing OR and ML, respectively, these models can improve decision-making and future planning under environmental uncertainty. Additionally, the combination of OR and ML as two models can operate in tandem as a revolutionary process in the fundamentals of decision support systems (DSS) that can predict and optimize sequentially.

To address the revolution of AI/ML for improving manufacturing planning decisions, system planning efficiency, and minimizing total completion time on the shop floor. Subsequently, we conducted a case study at a fabrication manufacturing company that produces fixtures, jigs, nuts, bolts, and iron pipes. Where the company is located in West Java, Indonesia. We noticed that the fabrication company has difficulty finding the optimum sequence of jobs that can minimize makespan under strict conditions. Besides that, the fabrication company also showed that machine processing time for each job has different proessing time so that can increase uncertainty in the completion time. In addition to these situations, the company also has a business environment under make-to-order (MTO) with the purpose of satisfying customer lead-time orders. Therefore, the company needs to find robust prediction and a minimum makespan scheduling under decision planning so that the company can provides recomendations waiting time for each job's completion time to customer orders.

To address this problem, we proposed a model using the predict-then-optimize (PTO) framework, where ML and OR optimization can work together. Although this framework has been popularized by (Elmachtoub and Grigas, 2022), we are offering a different case model for ML concepts and OR optimization techniques. The following are the novelties of this research:

First, we use the PTO framework rule using ANNs as an ML model, then an optimization process under the PFSP problem in a manufacturing environment. In addition, we also using two-steps process under the PTO framework that involves predicting for improving parameter and then optimizing parameters to enhance decision performance.

Second, we are offering different algorithms for ML as well as for optimization models. In the ANN model, gradient descent with momentum was applied to improve the parameters of prediction processing time, while in the optimization case, we applied the predicted parameter results for PFSP models and solved using the NEH algorithm to obtain the minimum makespan.

Meanwhile, this research was conducted to deploy the PTO framework to enrich the literature and fill the

gap in how PTO can be implemented in real conditions, as well as to show how a machine learning model can be used as a prediction for input parameters to the optimization model, so that it can create the best planning decisions. Then, this study also uses 2 algorithms in the machine learning process and 1 algorithm in the optimization model.

The artificial neural networks (ANNs) are chosen as a machine learning (ML) method for prediction, while model optimization is used based on permutation flow shop scheduling that is in line with the case study. Subsequently, we analyze the processing time for each machine, job, and job sequence to calculate the total makespan under the existing condition. Furthermore, several studies, including (Gupta et al., 2020), have shown that an artificial neural network is more flexible and more accurate when using optimization parameters.

Furthermore, the following is Table 1. The state of the art proposes a PTO framework using an artificial neural network model as a machine learning model and the permutation flow shop scheduling problem as an optimization problem that functions to enhance decision accuracy while minimizing makespan.

Table 1. State of The Art

No	Study	Framework Machine Learning and Optimization	Machine Learning Model	Algorithm in Machine Learning	Optimization Model Case	Algorithm in Optimization Case	Objective Function
1	(Gupta et al., 2020)	Integrated Prediction and Optimization	Artificial Neural Network	Levenberg-Marquardt (L-M) algorithm	Flow Shop Scheduling Problem	NEH Algorithm	Minimizing Makespan
2	(Liu et al., 2022)	Integrated Prediction and Optimization	Deep reinforcement learning	Double DeepQ-Network algorithm	Dynamic Flexible Job Shop Scheduling	Simulated Annealing	Minimising cumulative tardiness
3	(Zhou et al., 2023)	Integrated Prediction and Optimization	Reinforcement Learning	Proximal Policy Optimization	Permutation Flow Shop Scheduling Problem	NEH and SPT	Minimizing Makespan
4	(Mu et al., 2024)	Integrated Prediction and Optimization	Artificial Neural Network	Backpropagation	Flow Shop Scheduling Problem	Genetic Algorithm	Minimizing Makespan
5	(Zhang and Liu, 2025)	Smart Predict-Then-Optimized	Reinforcement Learning	Policy Gradient Optimization	Resource Assignment	Monte Carlo	Minimize total cost
6	(Nasseri et al., 2025)	Predict-Then-Optimized	XGboost	Gradient Boosted Decision Trees	Resource Assignment	Integer Programming	Maximizes overall profits
7	(X. Li et al., 2026)	Smart Predict-Then-Optimized	Long Short-Term Memory	Stochastic batch gradient descent	Resource Assignment	Mixed-Integer Programming	Minimize total cost
8	This Study	Predict-Then-Optimized	Artificial Neural Network	Gradient with the momentum algorithm	Permutation Flow Shop Scheduling Problem	NEH Algorithm	Minimizing Makespan

2. Research Methods

2.1 Predict-Then-Optimized Framework

Elmachtoub and Grigas (2022) have invented a predict-then-optimize (PTO) framework as an application in optimization, considering a combination of machine learning (ML) and an optimization process. In addition, their framework runs well for two-step processing, where the ML relies on a tool for prediction using the best-performing parameters, and then the optimization executes an operations research case based on the output prediction parameters. In addition, the PTO framework can give optimal decision solutions with respect to the result of prediction. However, the loss function model assumes that the results are completely independent of the optimization process. In other words, the loss function created doesn't factor in the result of the optimization process.

Whereas in “smart” predict-then-optimize (SPO), integrating prediction and optimization within the framework shows a difference in the end process in output decisions. Where the decision result is controlled by the output ML prediction, so optimization is used as a robust parameter for the input model. Meanwhile, the consequences of these models can result in a loss function as a functional integration between the ML and optimization process.

Conversely, we notice that in using the PTO framework, the quality of the output forecast has a significant influence on model optimization, as shown in Figure 1. Therefore, to overcome the highest error value in output prediction, we need to ensure that the initial parameters input into the ML model are robust.

Based on this PTO model paradigm, we aim to provide faster decision-making under uncertain conditions while optimizing resources in the manufacturing scheduling simultaneously.

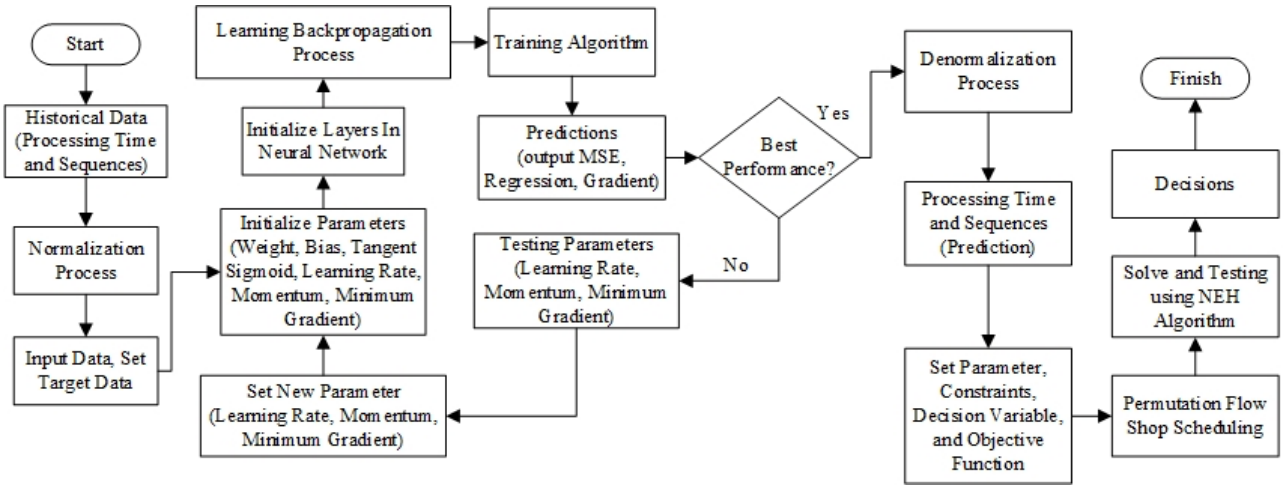

Figure 1: Predict-Then-Optimized Framework

2.2 Normalization and denormalization

In the first step of this study, a normalization process is needed to transform the data processing time on machine i for each job j. Additionally, a normalization process is used to utilize the maximum and minimum processing time for each machine to convert data into the range of 0 to 1. The normalization was conducted to adjust parameter especialy for caculate error in a neural network in non-dimensional units, while enhancing output accuracy. The equations of normalization (1) and denormalization (2) are shown as follows:

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \quad (1)$$

$$x_{actual} = x_{norm}[max(x) - min(x)] + min(x) \quad (2)$$

x_{norm} = data transformation processing time to a binary interval (normalization).

$min(x)$ = Minimum data of processing time (x)

$max(x)$ = Maximum data of processing time (x)

x_{actual} = data transformation output to actual data processing time (denormalization).

2.3 Activation function

In this study, an activation function is applied using the tangent sigmoid to increase performance in the neural network process. The activation function is used in both the hidden layer (m) and the output layer (M). Meanwhile, the range of the tangent sigmoid function is between 1 and -1, where the distributed function is active when the neuron chooses to increase performance while minimizing error $\vec{e}_p$ output $\vec{o}_t$ at time t. Output $\vec{o}_t$ at time t is directed to the training algorithm to optimize weights and biases for each layer.

$$\vec{\phi}^{(m)} = \vec{\phi}^{(M)} = \frac{2}{1 + e^{(-2a_{p,i})}} - 1 \quad (3)$$

The equation for the activation function $\vec{\phi}$ for an active neuron is shown in (3), which is similar for the hidden layer and output layer. Where $a_{p,i}$ is the output from the calculation of weights and bias that occur in the hidden layer and output layer, as shown in equation (5).

2.4 Feed-forward back-propagation network

In this study, a feed-forward back-propagation network (FBNN) is used with two layers between the input and output layers. In a feed-forward network (FNN), input data in hidden layers $\vec{o}_p^{(m-1)}$ lead forward to output data in output layers $\vec{o}_p^{(M)}$. The input data $\vec{x}_p$ and output data $\vec{o}_p$ equivalently represent $\vec{o}_p^{(m-1,)}$ and $\vec{o}_p^{(M)}$ is shown in equation (4). Meanwhile, the net processing calculation in hidden layers m and output layers M uses parameter weights $\boldsymbol{W}$ and bias $\vec{\theta}$.

$$\vec{x}_p \equiv \vec{o}_p^{(m)} \text{ and } \vec{o}_p \equiv \vec{o}_p^{(M)} \quad (4)$$

$$\vec{a}_{p,i}^{(m)} = \left[\boldsymbol{W}^{(m-1)}\right]^T \vec{o}_p^{(m-1)} + \vec{\theta}^{(m)} \text{ and } \vec{a}_{p,i}^{(M)} = \left[\boldsymbol{W}^{(M-1)}\right]^T \vec{o}_p^{(M-1)} + \vec{\theta}^{(M)} \quad (5)$$

In Equation (5), when the activation function is active at time t in neuron i, the tangent sigmoid (tansig) can be applied in the network processing $\vec{a}_p$, so that the output data $\vec{o}_t$ at time t in Equation (6) can be calculated for each layer in the active neuron i.

$$\vec{o}_t = \vec{\phi}^{(m)}\left(\vec{a}_{p,i}^{(m)}\right) \text{ and } \vec{o}_t = \vec{\phi}^{(M)}\left(\vec{a}_{p,i}^{(M)}\right) \quad (6)$$

2.5 Backpropagation algorithm

In multi-layer perceptrons (MLPs), this study applied backpropagation to calculate the error in the output layer that propagates backward through the network to the input layer to adjust weights and biases (Du et al., 2022). This network algorithm is most popular for evaluating error for each layer while minimizing the error (Singh et al., 2022).

To minimize total error, the backpropagation network uses mean squared error (MSE) ε_p for output data $\vec{o}_p$ and target data $\vec{y}_p$ under gradient descent is shown in Equation (7). Where single data p is calculated using MSE, as shown in Equation (8). In the

backpropagation rule, minimizing MSE ε_p is conducted in the training process at time t.

$$\varepsilon = \frac{1}{N}\sum_{p=1}^{N}\varepsilon_p = \frac{1}{2N}\sum_{p=1}^{N}\left\|\vec{y}_p - \vec{o}_p\right\|^2 \quad (7)$$

$$\varepsilon_p = \frac{1}{2}\left\|\vec{y}_p - \vec{o}_p\right\|^2 = \frac{1}{2}\vec{e}_p^T\vec{e}_p \quad (8)$$

Furthermore, a single data error $\vec{e}_p = \vec{y}_p - \vec{o}_p$, where the error for a single data in neuron i is $e_{p,i} = y_{p,i} - o_{p,i}$, which is the processing of the updating error in data p based on output data and target data. However, backpropagation calculates MSE in the hidden layer m using the data weights and biases based on the output layer M in neurons i and j, as shown in Equation (9)

$$\Delta_p w_{ij}^{(m-1)} = -\eta\frac{\partial\varepsilon_p}{\partial w_{ij}^{(m-1)}} \text{ and } \Delta_p\theta_i^{(m)} = -\eta\frac{\partial\varepsilon_p}{\partial\theta_i^{(m)}} \quad (9)$$

Meanwhile, gradient descent for updating weights and bias in the backpropagation process is shown in Equation (10), where the learning rate η is applied to increase performance and minimize error $e_{p,i}$ in data p.

$$\Delta_p w_{ij}^{(M-1)} = \eta e_{p,i}\left(\frac{d\phi_i^{(M)}(a)}{da}\right)_{a=\vec{a}_{p,i}^{(M)}} o_{p,j}^{(M-1)} \text{ and}$$
$$\Delta_p\theta_i^{(M)} = \eta e_{p,i}\left(\frac{d\phi_i^{(M)}(a)}{da}\right)_{a=\vec{a}_{p,i}^{(M)}} 1 \quad (10)$$

In Equation (11), the generalized error term is used with the chain rule to calculate the error p while updating the output derivative under gradient descent.

$$e_{p,i}\left(\frac{d\phi_i^{(M)}(a)}{da}\right)_{a=\vec{a}_{p,i}^{(M)}} \equiv \delta_{p,i}^{(M)} \quad (11)$$

The chain rule also serves as a connection between the output layer and the input layer, moving backward while minimizing error.

$$\Delta_p w_{uv}^{(m-1)} = \eta\delta_{p,u}^{(m)} o_{p,v}^{(m-1)}$$
$$and\ \Delta_p\theta_u^{(m)} = \eta\delta_{p,u}^{(m)} 1 \quad (12)$$

The updated weights and biases in the hidden layer of neuron u are shown in Equation (12). The change in the output layer M, while minimizing error, is backpropagated to the input layer through the generalized error term (chain rule) δ.

Algorithm 1: Feed-forward backpropagation

Input: initialization of random weights ($\boldsymbol{W}^{(m-1)} and\ \boldsymbol{W}^{(M-1)}$), random bias ($\vec{\theta}^{(m)}, \vec{\theta}^{(M)}$), input data $\vec{x}_p$, target data $\vec{y}_p$, tangent sigmoid function ($\vec{\phi}^{(M)}, \vec{\phi}^{(m)}$)

Output: $\Delta_p w_{uv}^{(m-1)}, \Delta_p w_{ij}^{(M-1)}, \Delta_p w_{ij}^{(m-1)}$ and $\Delta_p\theta_u^{(m)}, \Delta_p\theta_i^{(M)}, \Delta_p\theta_i^{(m)}$

Forward propagation

Step 1. Calculate net processing $\vec{a}_{p,i}$ using Equation (5) in layers m and M

Step 2. Calculate net processing $\vec{a}_{p,i}$ using the activation function $\vec{\phi}$ to get the output when neuron i is active in layers m and M, Equation (6)

Backward propagation

Step 3. Calculate error for each data sample p using $e_{p,i} = y_{p,i} - o_{p,i}$ in neuron i

Step 4. Update weight $\Delta_p w_{ij}^{(M-1)}$ and bias $\Delta_p\theta_i^{(M)}$ using Equation (10) in the output layer M

Step 5. Minimizing error $e_{p,i}$ using Equation (11) for neuron i in the output layer M to obtain the generalized error term $\delta_{p,i}^{(M)}$

Step 6. Applying the chain rule using Equation (12) to update the weight $\Delta_p w_{uv}^{(m-1)}$ and bias $\Delta_p\theta_u^{(m)}$ in layer m

Step 7. Calculating the MSE using Equation (8) for each data sample p

Step 8. Minimizing MSE using Equation (9) for each data sample p for weights and bias in the hidden layer using the data output layer.

2.6 Gradient descent with momentum

In this study, a training algorithm is used to update the weights and biases at time t. In this case, gradient descent with momentum was chosen to achieve improved performance in fitting data in a regression model between input data and output data (Chen et al., 2025). Additionally, gradient descent with momentum is also more effective in improving the algorithm's speed in finding the optimal solution (Lapucci et al., 2025). Furthermore, training using gradient descent with momentum is useful to accelerate the gradient that is used to escape from the local to the global optimum solution. Therefore, this study proposed this training algorithm to achieve better accuracy for predicting processing time.

$$\Delta_p\vec{v}(t+1) = \alpha\Delta_p\vec{v}(t) + (1-\alpha)\frac{\partial\varepsilon}{\partial\vec{w}} \quad (13)$$

$$\Delta_p\vec{w}(t+1) = \Delta_p\vec{w}(t) - \eta\Delta_p\vec{v}(t+1) \quad (14)$$

$$\Delta_p\vec{v}(t+1) = \alpha\Delta_p\vec{v}(t) + (1-\alpha)\frac{\partial\varepsilon}{\partial\vec{\theta}} \quad (15)$$

$$\Delta_p\vec{\theta}(t+1) = \Delta_p\vec{\theta}(t) - \eta\Delta_p\vec{v}(t+1) \quad (16)$$

Meanwhile, the gradient-with-momentum model is shown in Equation (14) for weights and in Equation (16) for bias, using velocity in Equations (13) and (15) to accumulate gradients and get smoother updates for weights and bias, as well as accelerate convergence. The initialization velocity in the output layer is $\Delta_p\vec{v}(t) = 0$, where $\Delta_p\vec{v}(t) = 0$ is the change in velocity from the weighted sum of all gradient descent.

Algorithm 2: Training using the gradient with the momentum algorithm

Input: $\vec{w} = (\Delta_p w_{uv}^{(m-1)}, \Delta_p w_{ij}^{(M-1)}, \Delta_p w_{ij}^{(m-1)})$ and $\vec{\theta} = (\Delta_p\theta_u^{(m)}, \Delta_p\theta_i^{(M)}, \Delta_p\theta_i^{(m)})$, η, α data from backpropagation

Output: ε, $\Delta_p\vec{w}(t+1)$, $\Delta_p\vec{\theta}(t+1)$

Step 1. Initialize velocity in the output layer: $v_{ij}^{(m-1)} = 0$

Step 2. Initialize time $t = 0$

Step 3. While the weights and biases have not converged, do:

Step 4. Update velocity $\Delta_p\vec{v}(t)$ for weights $\vec{w}$ and biases $\vec{\theta}$ using equations (13) and (15) at t

Step 5. Update weight $\Delta_p\vec{w}$ and bias $\Delta_p\vec{\theta}$ at $t+1$

Step 6. Resulting weight $\Delta_p\vec{w}$ and bias $\Delta_p\vec{\theta}$ for minimizing MSE ε for data sample p

2.7 Description of Permutation Flow Shop Problem (PFSP)

Meanwhile, this study uses a real problem in manufacturing fabrication with a focus on the part component machine and equipment, especially in jigs and fixtures products under characteristics of make-to-order (MTO). Subsequently, the production process in this manufacturing environment uses a flow shop scheduling environment for several jobs and machines.

Furthermore, this study uses four machines and four jobs with a focus on minimizing makespan.

PFSP with an independent setup time is used, as referred to in research (Belabid et al., 2020; Elissaouy and Allali, 2024), with aims to minimize makespan. In addition, considering this model is also based on the characteristic problem in fabrication manufacturing, where a set of jobs j that consist of a production batch is processed through a set of machines (Hoffmann et al., 2025). All jobs have the same path of operations on all machines that start in the first machine and finish in the last machine, continuing with their cycle. The FSP for real conditions is presented using mixed integer linear programming (MILP) to solve the objective function and constraints as follows:

$$C_{1,\pi_1} = P_{1,\pi_1} + s_1 \quad (17)$$

$$C_{1,\pi_j} = C_{1,\pi_{j-1}} + P_{1,\pi_j} + s_1, \ \ j = 2, \dots, n \quad (18)$$

$$C_{1,\pi_1} = \max\{s_i, C_{i-1,\pi_1}\} + P_{i,\pi_i}, \quad i = 2, \dots, m \quad (19)$$

$$C_{1,\pi_1} = \max\left\{C_{i,\pi_{j-1}} + s_i, C_{i-1,\pi_j}\right\} + P_{i,\pi_j}, \quad i = 2, \dots, m;\ j = 2, \dots, n \quad (20)$$

$$C_{max} = \max_{1 \le j \le n}\left\{C_{m,\pi_j}\right\} \quad (21)$$

Constraint (17) shows the completion time in machine 1 for sequence 1, π_1, which is calculated based on processing time and setup time. Subsequently, constraint (18) shows the next completion time in sequence j using the completion time before, processing time, and setup time in machine i. Constraints (19) and (20) show the determined completion time in machine 1 and sequence 1 to prevent overlapping processes. Finally, Equation (21) shows the completion time for machine m in sequence π_j.

Meanwhile, this study focuses on determining the best sequence that can optimize the makespan C_{max} under the permutation rule (π). Therefore, the objective function is $C_{max}(\pi^*) \le C_{max}(\pi)$, where $C_{max}(\pi^*)$ is the optimal sequence to minimize makespan. To minimize the makespan, the mathematical model is shown as follows:

Table 2. Indices of Permutation Flow Shop Scheduling

Notations	Definition
k, j	Job j is assigned to k position ($k, j = 1,2, \dots, n$)
j	Jobs ($j = 1,2,..,n$)
k	Position ($k = 1,2,..,n$)
i	Machine ($i = 1,2, \dots, m$)
π	Sequence ($\pi = 1,2,..,n$)

Table 3. Parameters of Permutation Flow Shop Scheduling

Notations	Definition
s_i	Set-up time on machine i
P_{i,π_j}	Processing time on machine i in the π_j sequence

Table 4. Decision Variable of Permutation Flow Shop Scheduling

Notations	Definition
$x_{k,j}$	Binary decision for job j is assigned to the k position
$C_{i,k}$	Completion time on machine i at position k

$$minimize\ C_{max} \quad (22)$$

$$\sum_{j=1}^{n} x_{k,j} = 1 \ \ k = 1, \dots, m \quad (23)$$

$$\sum_{k=1}^{n} x_{k,j} = 1 \ \ \ j = 1, \dots, n \quad (24)$$

$$C_{1,1} \ge s_1 + \sum_{j=1}^{n} x_{1,j}\, P_{1,\pi_1}, j = 1, \dots, n \quad (25)$$

$$C_{1,k} \ge s_1 + C_{1,k-1} + \sum_{j=1}^{n} x_{k,j}\, P_{1,\pi_j} \ \ , k = 2, \dots, n \quad (26)$$

$$C_{i,k} \ge C_{i-1,k} + \sum_{j=1}^{n} x_{k,j}\, P_{i,\pi_j} \ \ , i = 1, \dots, m \ \ \ k = 1, \dots, n \quad (27)$$

$$C_{i,k} \ge s_i + C_{i,k-1} + \sum_{j=1}^{n} x_{k,j}\, P_{i,\pi_j}, i = 1, \dots, m \ \ \ k = 1, \dots, n \quad (28)$$

$$C_{i,k} \ge 0, i = 1, \dots, m \ \ \ k = 1, \dots, n \quad (29)$$

$$\begin{cases} x_{k,j} = 0 \\ x_{k,j} = 1 \end{cases} \quad (30)$$

Equation (22) shows that the objective function in this model is to minimize makespan. Constraints (23) and (24) show that each position can only be used for a single job, and jobs are processed for only one position. Meanwhile, Constraint (25) is used to ensure that the completion time of the first job at a position on the first machine has been finished. Constraint (26) is given to ensure that the completion time on the first machine at position $k \ge 2$ is greater than the previous position $k - 1$. Furthermore, the completion time on machine i at position k must be greater than the previous time on machine $i - 1$, as shown in Constraint (27). The completion time on machine i at position k is greater than the last completion time, including setup time and processing time. Completion time must be a positive value in Constraint (28) and a binary decision variable in the constraint. Furthermore, Constraint (29) shows that completion time is a positive value, and Constraint (30) is the assignment for the binary decision variable $x_{k,j}$.

2.8 Nawaz, Enscore, and Ham (NEH) Algorithm

In this study, we use the NEH algorithm to determine the best sequences for the permutation flow shop problem (PFSP). The NEH algorithm is chosen based on research (Cáceres-Gelvez et al., 2026; Puka et al., 2021; Fernandez-Viagas et al., 2022) that shows that NEH can achieve optimal decisions to determine the best sequences to minimize makespan in large flow shop

scheduling. In addition, the characteristic PFSP is also a strict scheduling problem where m machines and n jobs are finite, and each job should go through all m machine with same order. Compared with the Johnson algorithm, which only focuses on general flow shops in small cases, NEH can overcome PFSP with a large number of machines and jobs while determining the best sequences under an independent setup time process. Furthermore, we shown step of the NEH algorithm as follows:

Algorithm 3: NEH Algorithm
Input: P_{i,π_j}, s_i
Output: π_{NEH}
Step 1: Calculate total completion time using $\sum_{i=1}^{m} P_{1,\pi_1} + s_1$
Step 2: Sort total completion time = sort (completion time, descending)
Step 3: Construct sequences = $\pi\ (\pi_1, \pi_2, .., \pi_j)$
For $j = 2$ to n
Step 4: Current makespan on sequences $C_{max}(\pi)$
Step 5: Find the optimal job sequence $\pi\ (\pi_1, \pi_2, \ldots, \pi_j)$
For $P_{1,\pi_j} = 1$ to π_{j+1}
Step 6: Insert for testing = (sequences π in P_{i-1,π_1}, new sequence π, sequences π in P_{i,π_j})
Step 7: Calculate new sequeneces $\pi = C_{max}$(testing, P_{i,π_j}, s_i)
Step 8: If $C_{max}(\pi^*) \leq C_{max}(\pi)$
Step 9: The best sequences are obtained with minimum makespan

3. Results and Discussion

To solve the mathematical model of ANNs and the PFSP problem, we used MATLAB 2016 to execute the model and find the optimum solution. In the first step, we executed ANNs using a backpropagation network type with learning and training data using gradient descent with momentum. In the second step, we conducted a simulation for the 4-20-4 network to determine the optimal allocation of $4 \times 4\ (m \times n)$ jobs and machines. In the final steps, we ran a mathematical model of PFSP under the NEH algorithm using input from the prediction ANNs, with the objective function being to find optimal sequences with independent setup times that minimize makespan.

3.1 Artificial Neural Networks

In the PTO framework, this study utilizes artificial neural networks (ANNs) as a machine learning (ML) tool, which functions as a predection tool for processing time. Meanwhile, we are using a backpropagation network type as used to calculate error during training. In addition, we are using a hidden layer $m - 1$ and an output layer $M - 1$ as a multi-layer perceptron (MLP). Furthermore, the weights and biases in this model are determined based on random initialization that follows the normalization range of 0-1 for the input processing time. Subsequently, the training algorithm using gradient descent with momentum is used to update weights and biases. Figure 2 shows an ANN network from the input layer to the output layer as shown as follows:

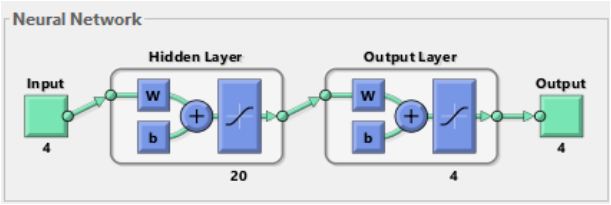

Figure 2: Multi-layer perceptron (MLP) with 4-20-4 Layers

Based on Figure 2, the data input in ANNs is 4 processing time machines for 4 jobs, as well as job sequences that consist of 1-2-3-4 in the input layer. Meanwhile, in the hidden layer, we use 20 neurons that lead to the output layer under the backpropagation process. Subsequently, the number of neurons in the output layer is adjusted based on the input data, which is the 224 historical processing time data we are using. The 224 historical processing time data where the type of product which is jigs. Meanwhile, we are using $4 \times 4\ (m \times n)$ for the jigs production process under 14 historical data points.

Table 5. Results of ANNs 4-20-4

	Job 1	Job 2	Job 3	Job 4
Machine 1	0.32201	0.23617	0.34923	0.32958
Machine 2	0.081793	0.29897	0.1736	0.27341
Machine 3	0.61879	0.79027	0.83748	0.79603
Machine 4	0.020733	1.45E-01	1.48E-01	0.13636

The input to the ANNs is the normalized processing time for each job using Equation (2). Subsequently, Algorithm 2 uses a normalized function to calculate model speed during training. So that faster convergence can be achieved. Furthermore, the result of the ANN model shown in Table 5 needs a denormalization process in Equation (3) to convert processing time to real units, using 5 minutes as the minimum processing time and 36 minutes as the maximum processing time.

The result of predicted vs actual processing time is shown in Table 6. The predicted processing time was obtained using simulate network with the best-performing parameter, as shown in Table 8, based on testing parameters using 10 replications. In addition, the setup time for each processing time is shown in Table 7. The setup time consists of setting the workpiece and tool.

Table 6. Predicted vs Actual Processing Time

	Predicted Processing Time Result				Actual Processing Time			
	J1	J2	J3	J4	J1	J2	J3	J4
M1 (minutes)	15	12	16	15	15	16	16	15
M2 (minutes)	8	14	10	13	9	17	18	15
M3 (minutes)	24	29	31	30	25	32	27	29
M4 (minutes)	6	9	10	9	5	9	5	6

Table 7. Setup time for machine i and job j

	Job 1	Job 2	Job 3	Job 4
Setup 1 (minutes)	2	3	3	4
Setup 2 (minutes)	3	2	4	2

Setup 3 (minutes)	5	6	7	6
Setup 4 (minutes)	2	2	2	2

3.2 Testing Parameters

To update weights $\boldsymbol{W}$ and biases $\vec{\theta}$, we use gradient descent with a momentum algorithm with a learning rate and momentum. This algorithm can successfully achieve the optimum solution with the minimum gradient (target). However, to test the best performance of the parameter learning rate and momentum, we proposed a sensitivity analysis to check that the model can obtain the highest regression in input and output, as well as the minimum MSE in the backpropagation process. We conducted changes in input parameters η and α that increased gradually, aiming to test gradient, MSE, and regression, while we also focused on checking the change in iteration and validation process.

Table 8. Change the input data parameters

Input Parameters			Output				
Learning rate η	Momentum α	Min_gradient	Average of MSE	Variance of MSE	Average of Regression R	Variance of R	Replication
0.01	0.91	1.00E-05	0.00949	6.87E-04	0.93298	2.03E-03	10
0.02	0.92		0.00827	5.06E-04	0.93720	1.45E-03	10
0.03	0.93		0.00850	3.14E-04	0.93876	0.00E+00	10
0.04	0.94		0.00847	5.44E-04	0.93894	2.06E-04	10
0.05	0.95		0.00815	8.49E-04	0.94013	8.75E-04	10
0.06	0.96		0.00796	6.93E-04	0.94155	4.51E-04	10
0.07	0.97		0.00804	8.67E-04	0.94192	1.11E-16	10
0.08	0.98		0.00774	4.83E-04	0.94240	3.16E-04	10

Table 7. The change in input data parameters shows the best performance of the learning rate and momentum with min_gradient for testing the best parameters. The best parameters were selected based on the lowest MSE and the highest R. Meanwhile, the optimal parameters were selected using a learning rate of 0.08 and a momentum of 0.98. Subsequently, the optimal learning rate and momentum parameters will be used for the ANN model to predict processing time under 4-20-4 layers. To ensure the result is under control, we also applied a control input for each weight $\boldsymbol{W}$ and bias $\vec{\theta}$ under seed control of random generalization. In addition, we used 2000 epochs, which were fixed for the learning algorithm during processing.

3.3 Permutation Flow Shop Scheduling Problem

The general flow shop in fabrication manufacturing conditions in this problem has been shown in equations 17-20, where the total makespan obtained is 160 minutes. The sequences of machines used in fabrication are the grinding machine 1, drilling process 2, milling machine 3, and lathe machine 4, and production tool machining for jigs and fixtures based on customer order specifications. We also conducted different observations on processing time and setup time; hence, setup time was only used for the optimization model to enhance the accuracy performance of prediction data ANNs. In addition, the manufacturing process can still change demand are order following the customer plan. However, we observe that manufacturing has 4 jobs with the highest order-to-fulfillment demand. Meanwhile, the first-in-first-out (FIFO) scheduling process has been implemented using predicted data, with sequences of jobs being 1-2-3-4 following the real manufacturing fabrication problem.

Table 9. Scheduling Performance Tested

Job × machine	NEH Algorithm	SPT	LPT	Classical PSFP
4 x 4	160	162	160	161
5 x 4	188	192	190	188
6 x 4	224	228	226	228
7 x 4	255	259	257	257
8 x 4	291	295	293	295
9 x 4	326	330	326	326
10 x 4	354	361	359	358

Table 9. Presents the results of comparisons between the NEH algorithm and heuristics such as shortest processing time (SPT) and longest processing time (LPT), as well as classical PSFP scheduling. The result shows that the NEH algorithm provides the best performance solutions based on the maximum completion time (makespan).

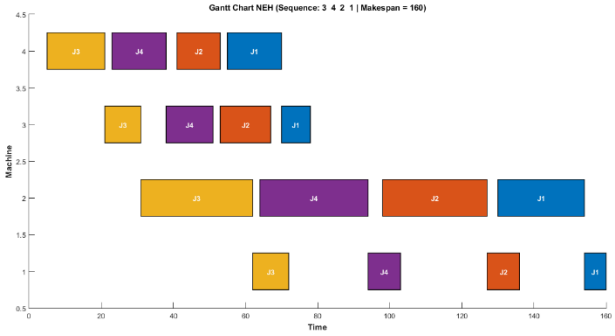

Figure 3: Permutation Flow Shop- Independent Setup Time Using NEH Algorithm

As can be seen, Figure 3 shows the Gantt chart for permutation flow shop scheduling based on the optimal sequence π^*. By considering the NEH algorithm, the best sequences have been obtained that can be

achieved to minimize makespan: $C_{max}(\pi^*) \leq C_{max}(\pi)$. However, we notice that on machine 3, the longest processing time creates a bottleneck. This bottleneck occurs because the milling machine is moving slowly with attention to detail. Besides that, during the process, the milling machine tool needs to reset the depth of cut, which can create the longest processing time. Nevertheless, the change of routing cannot be allowed in a permutation flow shop manufacturing environment because of interdependent machine processes.

Furthermore, to guarantee that our model can be solved using heuristics in a classical permutation flow shop without an algorithm, we tested enumerations of scheduling sequences $\pi!$ on our jobs and machines.

Table 10. Enumerations of scheduling sequences and makespan in 24 permutations

Sequences π_j	Makespan C_{max}	Sequences π_j	Makespan C_{max}
1-2-3-4	161	3-1-2-4	163
1-2-4-3	162	3-1-4-2	163
1-3-2-4	161	3-2-1-4	163
1-3-4-2	161	3-2-4-1	160
1-4-2-3	162	3-4-1-2	163
1-4-3-2	161	3-4-2-1	160
2-1-3-4	164	4-1-2-3	168
2-1-4-3	165	4-1-3-2	167
2-3-1-4	164	4-2-1-3	168
2-3-4-1	161	4-2-3-1	164
2-4-1-3	165	4-3-1-2	167
2-4-3-1	161	4-3-2-1	164

Table 10 presents the NEH algorithm given a minimum makespan of 160 minutes with job sequences of 3-4-2-1. Although other minimum results can be obtained, such as 3-2-4-1 but NEH algorithm can give an advantage in solution search speed and consistency.

Furthermore, the result of our model under the PTO framework can be a basis for a decision in predicting machine processing time, but it is also used to re-evaluate sequences for planning scheduling problems.

3.4 Managerial Implication

This research can serve as proof of how ML and OR can simultaneously achieve optimum solutions and better performance in decision planning under a predict-then-optimize framework. In addition, this research also broadens the horizons of implementation of ML and OR optimization in a two-step process for operational planning in manufacturing. Moreover, the development of ML prediction can support technical knowledge for production managers or supervisors to make better decisions for scheduling and sequencing on the shop floor. Utilizing the models, maintaining processing time is an important aspect. We notice that the waiting time from machine 2 to machine 3 creates a bottleneck, wherefore improved these process can significant reducing bottleneck and makespan time. The manager can prevent these conditions by using our prediction model for the next job process before execution. In addition, by utilizing prediction output, manufacturing fabrication can use an optimization process to estimate re-sequencing jobs to evaluate scheduling plans.

Furthermore, stakeholders in manufacturing fabrication can consider our model under the PTO framework to improve decisions for daily operational production scheduling plans. In manufacturing fabrication, the settlement of production flow strictly in the production process definitely reduces the complexity of the job process. Besides that, this study was also used as a model to project lead time accurately.

4. Conclusion

This study has been sucessfully implement PTO framework using ANNs as an ML model and PFSP optimization as an OR application. In the ANN model, we used a 4-20-4 layer under the backpropagation process, with learning and training using gradient descent with momentum. In addition, we also tested the learning rate and momentum with gradually increasing values to obtain the best performance in MSE and R regression. Subsequently, the best parameters are used for a prediction simulation to estimate processing time and sequences. Meanwhile, we successfully establish a robust model of PFSP using the NEH algorithm to determine the optimal sequence of jobs that minimizes makespan.

We conclude with practical considerations using ML and OR optimization under the PTO framework; better decisions can be made. In addition, this study also provides the stage of implementation of ML using ANNs and OR optimization based on the permutation flow shop scheduling problem in a production case.

However, we notice that the model is limited to achieve optimal performance of decision. Therefore, we propose developing a regret loss function $\mathcal{L}_{spo}$ for a decision model. In addition, we recommend further research to develop “smart predict-then-optimize” (SPO) as a framework that can present a regret loss function $\mathcal{L}_{spo}$ as a decision estimate for operational planning in manufacturing conditions. Meanwhile, ANNs can improve performance output data by adding more hidden layers as a deep learning process to handle complex data inputs. Subsequently, dynamic optimization in job shop and flow shop scheduling cases can be considered to overcome complex manufacturing systems that are adjusted to the demand plan. Additionally, we also recommend the second framework of integrating ML and OR, as the basis prediction data has been optimized.

References

Anis Lahoud, A., Khan, A. S., Schaffernicht, E., Trincavelli, M., & Stork, J. A. (2025). Predict-and-Optimize Techniques for Data-Driven Optimization Problems: A Review. *Neural Processing Letters*, *57*(2), 1–28. https://doi.org/10.1007/s11063-025-11746-w

Badan Pusat Statistik. (2026). BPS: Manufacturing Products Support the Continued Growth of Non-Oil

and Gas Exports. https://www.bps.go.id/en/news/2026/03/03/871/bps--manufacturing-products-support-the-continued-growth-of-non-oil-and-gas-exports.html

Belabid, J., Aqil, S., & Allali, K. (2020). Solving Permutation Flow Shop Scheduling Problem with Sequence-Independent Setup Time. *Journal of Applied Mathematics*, *2020*. https://doi.org/10.1155/2020/7132469

Cáceres-Gelvez, S., Dang, T. H., & Letchford, A. N. (2026). MIP-based local search for permutation flowshop scheduling with makespan objective. *Computers and Operations Research*, *186*(February 2025), 107309. https://doi.org/10.1016/j.cor.2025.107309

Chen, X., Yang, H., Zhang, H., & Wong, C. U. I. (2025). Dynamic Gradient Descent and Reinforcement Learning for AI-Enhanced Indoor Building Environmental Simulation. *Buildings*, *15*(12), 1–25. https://doi.org/10.3390/buildings15122044

Du, K. L., Leung, C. S., Mow, W. H., & Swamy, M. N. S. (2022). Perceptron: Learning, Generalization, Model Selection, Fault Tolerance, and Role in the Deep Learning Era. *Mathematics*, *10*(24), 1–46. https://doi.org/10.3390/math10244730

Elissaouy, O., & Allali, K. (2024). Minimizing the Maximum Tardiness for a Permutation Flow Shop Problem Under the Constraint of Sequence Independent Setup Time. *RAIRO - Operations Research*, *58*(1), 373–395. https://doi.org/10.1051/ro/2024001

Elmachtoub, A. N., & Grigas, P. (2022). Smart "Predict, then Optimize." *Management Science*, *68*(1), 9–26. https://doi.org/10.1287/mnsc.2020.3922

Fernandez-Viagas, V., Sanchez-Mediano, L., Angulo-Cortes, A., Gomez-Medina, D., & Molina-Pariente, J. M. (2022). The Permutation Flow Shop Scheduling Problem with Human Resources: MILP Models, Decoding Procedures, NEH-Based Heuristics, and an Iterated Greedy Algorithm. *Mathematics*, *10*(19). https://doi.org/10.3390/math10193446

Gao, R. X., Krüger, J., Merklein, M., Möhring, H. C., & Váncza, J. (2024). Artificial Intelligence in manufacturing: State of the art, perspectives, and future directions. *CIRP Annals*, *73*(2), 723–749. https://doi.org/10.1016/j.cirp.2024.04.101

Grumbach, F., Müller, A., Reusch, P., & Trojahn, S. (2024). Robust-stable scheduling in dynamic flow shops based on deep reinforcement learning. *Journal of Intelligent Manufacturing*, *35*(2), 667–686. https://doi.org/10.1007/s10845-022-02069-x

Gupta, J. N. D., Majumder, A., & Laha, D. (2020). Flowshop scheduling with artificial neural networks. *Journal of the Operational Research Society*, *71*(10), 1619–1637. https://doi.org/10.1080/01605682.2019.1621220

Hoffmann, J., Neufeld, J. S., & Buscher, U. (2025). Customer order scheduling in a permutation flow shop environment. *Operations Research Perspectives*, *15*(November). https://doi.org/10.1016/j.orp.2025.100362

Lapucci, M., Liuzzi, G., Lucidi, S., Pucci, D., & Sciandrone, M. (2025). A globally convergent gradient method with momentum. *Computational Optimization and Applications*. https://doi.org/10.1007/s10589-025-00741-5

Lee, H. (2025). Fault-Resilient Manufacturing Scheduling with Deep Learning and Constraint Solvers. *Applied Sciences (Switzerland)*, *15*(4), 1–24. https://doi.org/10.3390/app15041771

Li, X., Yu, J., Wang, X., & Lu, H. (2026). Balanced Smart Predict-Then-Optimize Framework for Container Yard Intelligent Retrofit Decision-Making. *Journal of Advanced Transportation*, *2026*(1), 1–20. https://doi.org/10.1155/atr/8856441

Li, Y., & Yu, C. (2025). Flexible Job Shop Scheduling with Job Precedence Constraints: A Deep Reinforcement Learning Approach. *Journal of Manufacturing and Materials Processing*, *9*(7), 1–21. https://doi.org/10.3390/jmmp9070216

Liu, R., Piplani, R., & Toro, C. (2022). Deep reinforcement learning for dynamic scheduling of a flexible job shop. *International Journal of Production Research*, *60*(13), 4049–4069. https://doi.org/10.1080/00207543.2022.2058432

Mao, S., Wang, B., Tang, Y., & Qian, F. (2019). Opportunities and Challenges of Artificial Intelligence for Green Manufacturing in the Process Industry. *Engineering*, *5*(6), 995–1002. https://doi.org/10.1016/j.eng.2019.08.013

Mu, H., Wang, Z., Chen, J., Zhang, G., Wang, S., & Zhang, F. (2024). A Flow Shop Scheduling Method Based on Dual BP Neural Networks with Multi-Layer Topology Feature Parameters. *Systems*, *12*(9), 1–18. https://doi.org/10.3390/systems12090339

Nasseri, M., Falatouri, T., Brandtner, P., Darbanian, F., & Mirshahi, S. (2025). Enhancing Resource Assignment Efficiency in Service Industry: A Predict-then-Optimize Approach with XGBoost. *Procedia Computer Science*, *253*, 644–653. https://doi.org/10.1016/j.procs.2025.01.126

Prastiya, D. S., & Wahyuni, R. S. (2024). Artificial Neural Network Model For Optimization of Forecasting Material Inventory. *Jurnal Teknik Industri*, *25*(2), 173–188. https://doi.org/10.22219/jtiumm.vol25.no2.173-188

Puka, R., Duda, J., Stawowy, A., & Skalna, I. (2021). N-NEH+ algorithm for solving permutation flow shop problems. *Computers and Operations Research*, *132*(October 2020). https://doi.org/10.1016/j.cor.2021.105296

Ren, F., & Liu, H. (2024). Dynamic scheduling for flexible job shop based on MachineRank algorithm and reinforcement learning. *Scientific Reports*, *14*(1), 1–17. https://doi.org/10.1038/s41598-024-79593-8

Singh, A., Kushwaha, S., Alarfaj, M., & Singh, M. (2022). Comprehensive Overview of Backpropagation Algorithm for Digital Image Denoising. *Electronics (Switzerland)*, *11*(10). https://doi.org/10.3390/electronics11101590

Tang, H., & Dong, J. (2024). Solving Flexible Job-Shop Scheduling Problem with Heterogeneous Graph Neural Network Based on Relation and Deep Reinforcement Learning. *Machines*, *12*(8). https://doi.org/10.3390/machines12080584

Xin, X., Jiang, Q., Li, C., Li, S., & Chen, K. (2023). Permutation flow shop energy-efficient scheduling with a position-based learning effect. *International Journal of Production Research*, *61*(2), 382–409. https://doi.org/10.1080/00207543.2021.2008041

Yan, Q., Wu, W., & Wang, H. (2022). Deep Reinforcement Learning for Distributed Flow Shop Scheduling with Flexible Maintenance. *Machines*, *10*(3). https://doi.org/10.3390/machines10030210

Zhang, H., & Liu, H. (2025). Real-Time Power System Optimization Under Typhoon Weather Using the Smart "Predict, Then Optimize" Framework. *Energies*, *18*(3). https://doi.org/10.3390/en18030615

Zhou, T., Luo, L., Ji, S., & He, Y. (2023). A Reinforcement Learning Approach to Robust Scheduling of Permutation Flow Shop. *Biomimetics*, *8*(6). https://doi.org/10.3390/biomimetics8060478