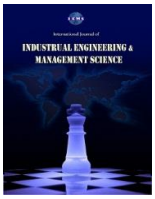
University of Hormozgan

International Journal of Industrial Engineering & Management Science

journal homepage: www.ijiems.com



Balancing and Sequencing in Mixed-Model Assembly Lines Using Deep Neural Networks and Reinforcement Learning

Fahimeh Tanhaie [a,*]

[a] *Faculty of basic science and Engineering, Kosar university of Bojnord*

ARTICLE INFO

Article history:
Received: 2025-11-09
Received in revised form: 2026-02-30
Accepted: 2021-10-05

Keywords:

MMAL
Balancing and Sequencing
DLN
RL

ABSTRACT

This research provides a new angle to solving the problem of balancing and sequencing hybrid assembly lines by leveraging the power of artificial intelligence. Our mixed-model assembly line environment involves using Deep Neural Network (DNN) and Reinforcement Learning (RL)in order to consider both balancing and task sequencing properly. Production history data such as task times, total number of stations and relations between tasks were obtained and modeled. A Deep Q-Network based reinforcement learning model was used to learn the optimal task sequence and assign tasks to workstations in real-time and subject to the least makespan and in line with maximum line efficiency. Similarly, DNNs were used for predicting the time it will take to process tasks and the shifting of tasks among stations. Numerical evaluation on real world data shows that the new approach effective reduces idling time, minimizes waiting time of task and enhances the production flow. A comparison with more conventional optimization algorithms like Geometric Algorithms and Simulated Annealing also emphasizes the compound gain in terms of optimization time and near optimal solution that is possible with the machine learning based approach.

1. Introduction

The issue of assembly lines in the twenty-first-century manufacturing companies cannot be overstated given the increased competition in manufacturing lines. mixed-model assembly lines assembly lines that can accommodate the production of varied types of cars at the same time are of great importance to industries that are experiencing pressure from consumers for more specialized and flexible cars. These lines involve constructing different variants of the product on the same line so that there should not be any distortion in the flow of production so, both line balancing and task sequencing are important here. Line balancing aims to divide activities into equal segments of work so that time spent in different work stations is nearly the same while task sequencing involves the determination of the most appropriate sequence to be followed in executing the tasks. Together, the optimal management of these two aspects is important for achieving the greatest possible performance while incurring the lowest possible expenses.

The mixed-model assembly line balancing problem (MMALBP) aims to assign tasks to workstations while satisfying precedence constraints and minimizing performance measures such as makespan, idle time, and workload imbalance (Boysen et al., 2012; McMullen & Frazier, 1997). MMALBP is particularly important in modern manufacturing systems due to increasing product variety, demand uncertainty, and the need for high production flexibility (Razali et al., 2019). Although metaheuristic approaches such as genetic algorithms, simulated annealing, and tabu search have been widely applied to MMALBP, they often suffer from high computational cost, sensitivity to parameter tuning, and limited adaptability to dynamic production environments (Tanhaie et al., 2017;

Defersha & Mohebalizadehgashti, 2018). Further, these classical methods are in general static and do not incorporate the dynamic and stochastic nature of the current production systems. Newer approaches of ML like Deep Learning and Reinforcement Learning have appeared as potential new frontiers for the identified challenging optimization problems in manufacturing. DNNs are appreciated to offer an excessive amount of accuracy in terms of outcome assessment by analyzing historical data, but RL involving DQN enables the model to learn and assess the best approach when in a dynamic interaction environment. These presented methods of machine learning aim at possibly surpassing the traditional optimization techniques used in manufacturing since they provide adaptive decisions throughout the production process to fit the complexities of mixed-model assembly lines.

In this paper, we introduce IMML-BA that combines the potential of DNNs for predicting the required balancing effort with the ability of RL to manage the sequencing in hybrid assembly lines. From this, we derive production timelines for task processing times and apply a DQN-based RL algorithm as a mechanism for task assignment and scheduling. The objectives of this research are twofold: This allowed for the optimization of two primary objectives: (1) to minimize the total makespan (defined in Section 3.1); that is, the total time required to produce all the items in the network and (2) to increase the general line effectiveness by decreasing idle times and critical non-identical lines.

Unlike existing reinforcement learning–based and hybrid approaches that primarily focus on either balancing or sequencing in an offline or static manner, this study introduces a fully integrated, data-driven framework that jointly addresses both problems in real time. The novelty of the proposed approach lies in:

(i) the integration of a deep neural network to predict task processing times from historical production data and dynamically update the system state;
(ii) a Deep Q-Network formulation specifically designed for mixed-model assembly environments to perform real-time task sequencing and workstation assignment under precedence constraints; and
(iii) the explicit coupling of predictive (DNN) and decision-making (DQN) components to enhance adaptability to variability and uncertainty in production conditions.

The remainder of this paper is organized as follows. In Section 2, we provide a review of related work in assembly line balancing and sequencing, focusing on traditional optimization methods and recent applications of machine learning. Section 3 outlines the methodology, including data preprocessing, model architecture, and the RL algorithm used. Section 4 presents the experimental setup and results, followed by a comparative analysis with traditional methods. Finally, Section 5 concludes the paper with key findings and suggestions for future research.

2. Literature Review

2.1. Evolution of Assembly Lines: Single-Model, Multi-Model, and Mixed-Model

The number of assembly line techniques has evolved from the single-model assembly line which deals with the mass production of one model into the multi-model assembly line, and the most recent and general category, the mixed-model assembly line (MMALs). One way of assigning tasks in single-model lines is very clear, with all being aimed at achieving the most efficient for one sort of product. This is as opposed to the single-model assembly lines which endeavored to attain the highest levels of utilization of production time by reducing both idle time and cycle time in a straight- through production system (Boysen et al., 2012). The new development of multi-model assembly lines posed the issue of model mixing on the assembly line that meant that sequencing mechanisms had to be deployed to allow for little or no time between models. Multimodel lines and their sequencing and balancing problems were studied by Kucukkoc et al. their artificial bee colony algorithm to tackle them improves task distribution across variants (Kucukkoc, 2019).

MMALs are even better since they combine the production of multiple product models in one line, without the necessity to interrupt the work to retool. Boysen et al. (2012) posited the challenges of MMALs, highlighted by the fact that sequencing and balancing cannot occur in isolation because of fluctuating product demand and setup time (Boysen et al., 2012). The latest developments further enhance mixed-model assembly line by improving optimization algorithms that are currently available. For instance, Zhang et al. (2024) proposed a procedure called reinforcement learning-based multi-objective evolutionary algorithm designed to optimize mixed-model assembly lines in condition of unknown demand. This approach acts to incorporate task-worker sequence changes that reduce both the product and penalty cost and also serves to improving the system reliability and eliminating unnecessary idle time (Zhang et al., 2024). Along the same line, Tanhaie and Nahavandi (2017) proposed multi-objective artificial bee colony algorithm for line balancing and sequencing integrated for mixed-model assembly lines. This method focuses in reducing variability of material used, makespan and penalties that go hand in hand with late deliveries makes it perfect when it comes to optimization of production flow in real life instances (Tanhaie and Nahavandi, 2017). These developments reflect ongoing efforts to improve flexibility, efficiency, and responsiveness in mixed-model assembly systems, responding to the complexities of modern production environments.

2.2. Balancing and Sequencing in Mixed-Model Assembly Lines

One of the main challenges in mixed-model assembly lines in balancing the workload of tasks between the workstations because this often results in; more system idle time and bottlenecks. On the same note, sequencing tasks for different product models is just as relevant a task as finding the right tasks to execute. Other authors who identified both balancing and sequencing problems in MMALs in parallel include Kara et al. (2010) who used simulated annealing. Kara et al. (2010) showed that incorporating both variables can dramatically cut out idle time together with enhance line throughput. In addition, Janardhanan et al. also investigated balancing and sequencing of robotic MMALs with metaheuristic-based hybrids in 2019. Of these, their research dealt with makespan and robots where the assignment of robots becomes crucial in automated processes (Li et al., 2019).

Defersha and Mohebalizadehgashti (2018) has put forward a two-stage genetic algorithm for balancing and sequencing of mixed-model manual assembly line at the same time. Their strategy included avoiding long workstations and the number of workstations, eliminating repetition, which enforced an efficient balancing and a sequence (Defersha & Mohebalizadehgashti, 2018). Similarly, Zhang et al. (2023) adopted a reinforcement learning based on a multi-objective evolutionary algorithm for a work in progress, on mixed model assembly line where demand was unpredictable. This approach sought to minimize productivity cost and penalty cost of task-worker sequences showing enhanced performance from different other multi-MOP algorithms (Zhang et al., 2023).

2.3. Traditional Optimization Methods for Balancing and Sequencing

In MMALs, heuristic and metaheuristic techniques have been widely used for balancing and sequencing problems over the course of the years. For example, Akpınar et al. (2011) presented a hybrid genetic algorithm to apply the cycle times in MMALs and to determine the workload distributing and scheduling of the stations to eliminate the idle times (Akpınar et al., 2011). Similarly, Zhang et al. (2024) included human-robot collaboration into MMAL balancing. Their model gives the capability of the stations to be controlled fully by human, fully by robots, or a blend of both, which increase flexibility, and task optimization (Zhang et al., 2024).

In later years, the state of the art has also been developed in heuristic and meta heuristic methodologies for solving these coupled optimization problems. To counter this, Guo et al. (2020) provided a mix integer linear programming model combined with gene expressed programming to improve the MMAL solution in the case of fluctuating demand. His approach evaluates the task-worker distribution and the rules for sending workers into the field and observed higher flexibility and performance (Guo et al., 2020). Likewise, Tang et al, (2016) used a simulated annealing approach in solving the balancing and sequencing in MMALs much in the same way as in He and Wang (2011) but considered sequence-dependent set-up times. As their approach outperforms more conventional methods analytically as well as computationally, it is highly applicable in large scale industries (Tang et al., 2016). The last study of both was conducted by Defersha and Mohebalizadehgashti in 2018 in which a hybrid genetic algorithm is used to minimize cost, balance, and sequence tasks in MMALs. It proposed rules that prevented copying of tasks and adjustment of the workstation length, therefore increasing production rates (Defersha & Mohebalizadehgashti, 2018).

2.4. Machine Learning in Balancing and Sequencing for MMALs

Recent advances in deep learning (DL) and reinforcement learning (RL) have enabled data-driven and adaptive decision-making in manufacturing systems, including scheduling, line balancing, and sequencing problems (Ayough et al., 2023; Zhang et al., 2023). RL and DL in particular are capable of self-redesigning the task distribution and the order of their execution in accordance with feedback acquired during the conduction of the examination; as such, they present more versatility in contrast to conventional strategies. Setting a base for the new AI methodologies, Ayough et al. (2023) proposed a model that integrates DNNs and Dragonfly Algorithm (DA). This model enhanced the coordination of tasks in MMALs as well as the order of operations, which resulted in the general decrease of energy use and work duration. In their work, Ayough et al. (2023) showed that DNNs can predict task processing time thereby improving the actual and real-time adjustments to the line.

Tanhaie (2024) used the augmented multi objective particle swarm optimization (AMOPSO) algorithm to solve a mixed model assembly line where human and robot collaboration is being implemented. Their work enhanced sequencing and balancing by adopting adaptive learning techniques that demonstrated the extent of applying machine learning in mixed-model assembly lines with higher variety and complexity (Tanhaie , 2024). Zhang have also proposed a multi-objective evolutionary algorithm using the reinforcement learning which is used in balancing the mixed-model multimanned assembly line with uncertainty in demand (Zhang, 2023). The algorithm combines Q-learning and evolutionary methods to solve the problem of ordering the sequence of performing tasks in the workers in the production environment in order to minimize time intervals between tasks and to ensure high adaptability to fluctuations in production conditions (Zhang et al., 2023).

For the scheduling sequencing and balancing of multi-mixed model assembly lines Saif et al. (2019) proposed multi-objective artificial bee colony algorithm. Their approach was capable of reducing makespan, reducing the variation in material usage and improving delivery times by leveraging with adaptive learning to manage the sequencing of the model (Saif et al., 2019). Li et al. (2023) developed a mathematical model that integrates learning and fatigue effects into the balancing and sequencing of mixed-model assembly lines. Enhancing a genetic algorithm aspect of their model, to the best of my knowledge, their model focuses on task assignments that suit tasks with an eye on the human figure such as operator fatigue. This shows how these human-oriented factors can enhance production performance within a mixed model layout (Eslamipoor & Nobari, 2023).

2.5. Hybrid Approaches Combining Machine Learning and Heuristics

This has led to the use of integration of the machine learning techniques and the traditional optimization approaches in the recent past. Taha (2022) developed a two-stage heuristic algorithm to combine a genetic algorithm and a procedure for handling assembly line balancing in MMALs with multi-manned workstations.Çil et al. (2020) proposed a novel approach that aims to balance multisting workstation assembly lines using a hybrid PSO algorithm in conjunction with a constructive heuristic to accomplish more of the task than traditional methods (Taha, 2022).ignificantly improved task allocation, workstation management, and overall production efficiency compared to traditional methods (Taha, 2022). Çil et al. (2020) introduced a hybrid method that combines particle swarm optimization (PSO) with a constructive heuristic for balancing assembly lines with multi-manned workstations. This strategy greatly reduced the workstation costs and resource consumptions and has found to be better that tabu search and cuckoo search algorithms in solution quality and computation time (Çil et al. 2020).

Recently, Zhou and Huang (2024) have proposed a new hybrid QPSO approach for dynamic part-feeder and scheduling in MMALs. By adopting both opposition-based learning and variable neighborhood search mechanisms, this study achieved a lower energy consumption and better inventory management (Zhou & Huang, 2024). While balancing /sequencing for mixed-model assembly lines becomes increasingly

challenging owing to increased line complexity, machine learning presents new solutions. While genetic algorithms and ant colony optimization, which are standard, have delivered good solutions, incorporating adaptability of machine learning methods, particularly reinforcement learning, and deep neural networks is revolutionizing the optimization of MMALs. Such changes not only optimise efficiency but also make assembly line dynamic to respond to dynamic product market demands.

3. Objective and Methodological Framework

The proposed methodology is motivated by the complementary roles of prediction and decision-making in mixed-model assembly lines. Accurate estimation of task processing times is essential for effective balancing, yet such information is often uncertain and variable in real production environments. Deep neural networks are well suited to capture nonlinear relationships in historical production data and provide reliable time predictions. On the other hand, task sequencing and assignment require adaptive, sequential decision-making under changing system states, which is naturally modeled using reinforcement learning. By combining DNN-based prediction with DQN-based control, the proposed framework enables informed and adaptive real-time decisions that cannot be achieved by standalone optimization or purely reactive RL approaches. In this section, we describe the methodological approach that has been developed to meet the requirements of balancing and sequenced policies in MMALs. Hence, the purpose of this research is to identify the job–to–job and operation–to–operation task allocation and concurrency in a hybrid assembly line, and to improve the flow of work within this environment to minimize overall makespan and to maximize line productivity. To this end, a new framework incorporating DNNs and RL is presented here where tasks can be dynamically allocated and scheduled regarding the actual production environment. Deep neural networks are effective in capturing nonlinear relationships from historical production data and provide accurate predictions of task processing times, while Deep Q-Networks enable adaptive, real-time decision-making in stochastic and dynamic environments (Zhang et al., 2023; Zhou & Zhao, 2024).

The DNN serves as a time-series prediction model for estimating the processing time of tasks in various workstations. By training the DNN with the historical production data, the different interactions between task dependencies, station capabilities, and production constraints are learnt, and thus enable precise estimation of the task times. This predictive ability improves decision making in that real time updates on task duration reduce the inefficiencies of production conditions on task assignments. At the same time, we use a DQN-based RL for the dynamic aspect of managing task and their sequences and allocation. The RL model includes the MMAL as a dynamic system within which workstation tasks have to be assigned in real time, the main goal being to minimize workstation idle time and task waiting time. It therefore self-trains by performing tasks by interacting with the production environment and receives feedback such as rewards or penalties to enhance its optimal task sequence. In particular, the structure of the reward function is aimed at making the total production time (makespan) as small as possible, minimizing idle time, and fairly distributing loads among all stations.

The proposed framework combines two different machine learning methods – DNN for building the predictive model of MMAL factors and RL for on-line decision making about MMAL balancing and sequencing. This approach is expected to be superior to GA and SA in real time response, quicker search and convergence, and adaptability to Stochastic Production System (SPS). Furthermore, in the given scheme, learning in- cludes the use of the reinforcement learning which makes the system capable to adjust to the changing production conditions in the course of its work which enhances the total flexibility and responsiveness level of the assembly line. In the subsequent sections, details of data preprocessing, model construction, training, and assessment measures used in the present research are described. We then provide the real case studies including results of the simulation on the actual production data and the comparison with the conventional optimization.

3.1. Data Collection and Preprocessing

The data for this study were obtained from a real MMAL in the automotive manufacturing industry. Historical production data for a 12-month period are included in the dataset; features like the processing times of the tasks, the number of stations and their capacity, dependency relationships between tasks and workstations. The data were set up to reflect the flexibility of lock drum schedules given the variability of the actual production process used in this environment where one assembly line assembles multiple products.

The dataset comprises the key attributes in table 1, which are critical for the balancing and sequencing process.

Table 1. key attributes for the balancing and sequencing process

Attribute ID	Feature	Description	Data Type
1	Task ID	Unique identifier for each task	Categorical
2	Task Time (seconds)	The average time required to complete the task	Continuous
3	Station ID	Identifier for the workstation where the task is performed	Categorical
4	Predecessor Task ID	The ID of the task that must be completed before the current task	Categorical
5	Successor Task ID	The ID of the task that follows the current task	Categorical
6	Task Dependency Level	The level of dependency (high, medium, low) between tasks	Ordinal
7	Station Capacity	Maximum number of tasks the workstation can handle simultaneously	Continuous
8	Workstation Efficiency	Efficiency of the workstation (measured as a percentage)	Continuous
9	Product Variant	The product type (e.g., Model A, Model B) produced on the line	Categorical
10	Total Workstations	The number of active workstations in the production line	Continuous

Feature Selection and Importance: We stress that it is more useful for the present analysis to consider subsets of variables with high projection on the PVS all together rather than focusing on individual ranks of the variables. Therefore, the wrapper method is used to assess these selected subsets for their measure of predictive accuracy. Employing the greedy forward algorithm, we choose feature subsets by following a predetermined criterion (i.e., RMSE). In this case two machine learning algorithms, the DNN and the DQN are employed to support the execution of this task. The DNN forecasts working time on tasks, for proper distribution of tasks among stations the DQN learns the sequence of the task in real-time.

Figure 1 shows how often each feature was determined to be essential for predicting task assignment or sequencing time.

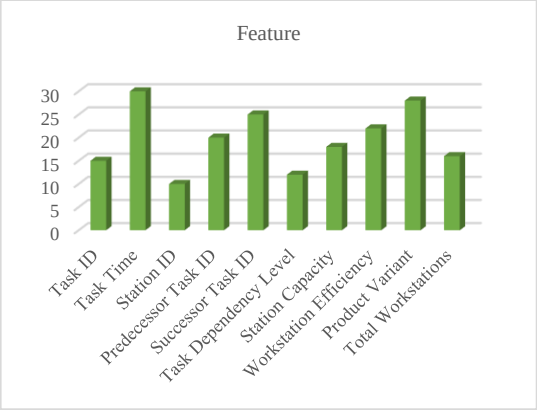


Figure 1. frequency of each feature

3.2. Data Preprocessing

The preprocessing step in the experimental process was characterized by some significant activities to ensure that the raw data fed into the models would be useful. The steps include the following:

1. **Handling missing Data:**
   There was a small percentage of records with missing data for which intrinsic variables such as Task Time and Station Efficiency come from. For the cases of missing values, we prepared mean and mode imputation for numerical and categorical variables respectively. For example, if the value of any of the task time variables was 'missing,' the average of the similar task time was imputed.

2. **Normalization:**
   Because the task times and the station capacities are in different range, the continuous data was scaled using min-max normalization. This scaling helps prevent measurement features like Task Time and Station Capacity from exerting undue influence on the learning because of their dramatically larger numeric range.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

For example, a task time of 500 seconds would be normalized based on the minimum and maximum task times across the dataset.

3. **Categorical Encoding:**
   The data variables including Task ID, Station ID and Product Variant were categorized and converted into binary vector form using one-hot encoder. This encoding helps the machine learning models to mainly manage categorical data.

4. **Task Dependency Representation:**
   The dependencies between the tasks were described by the use of the adjacency matrices to show the constraints on the task precedence. In the next stage of the study for each product variant a directed acyclic graph or DAG structure was developed in which the nodes to be performed are the tasks and the arcs connecting them represent the constraints on the order of doing them. This representation enables the models to identify the right succession of the tasks, taking into consideration of the constraints existing among them.

5. **Data Splitting:**
   The data into which the dataset was divided consists of the training set with 70%, the validation set with 15% and the test set with 15%. The training data was utilized for both the DNN and RL models, with the training set used for adjusting hyperparameters using the validation set. The test set was used to determine the effectiveness of the applied models on unseen data set.

The final dataset contains the statistics in table 2:

Table 2. final dataset

Statistic	Value
Total Tasks	1,200
Total Workstations	25
Number of Product Variants	4 (Model A, B, C, D)
Average Task Time (seconds)	350
Total Production Cycles	10,000
Average Station Efficiency	85%

These data offer a big picture of the real world MMAL system, which enrich the information of task completion and station deployment. The preprocessing steps adopted followed a cleaning and normalization process which would make the data suitable for the input into other models of predictive and reinforcement learning. Similarity Matrix of Tasks in MMAL is presented in table 1 below.

The similarity matrix shows how similar one task is to another; this was helpful to the DNN model in that the model was able to consider the processing times of other tasks that are similar to each other based on the matrix. This approach aids the DQN in determining sequences of tasks suitable for the production environment especially in systems characterized by hi variations Building all the Required Information for the Similarity Matrix To

3.3. Creating the Necessary Data for the Similarity Matrix

construct the similarity matrix, we utilize hypothetical data for three key attributes likely relevant in your study:

1. **Task Time:** The duration taken to complete every task.
2. **Dependency Level:** Depending on the precedence relationships, the necessity level of each task in reference to the other tasks.
3. **Station Capacity Requirement:** Each workpiece has a demand on the capacity of the performance at the workstation.

Building the Similarity Matrix

we construct the similarity matrix (table 3) based on similarities across different task attributes.

Table 3. similarity matrix

Task ID	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
Task 1	0.00	0.27	0.17	0.38	0.47	0.23	0.31	0.42	0.36	0.24
Task 2	0.27	0.00	0.22	0.32	0.42	0.19	0.37	0.28	0.34	0.29
Task 3	0.17	0.22	0.00	0.29	0.39	0.20	0.26	0.35	0.33	0.27
Task 4	0.38	0.32	0.29	0.00	0.25	0.34	0.41	0.20	0.22	0.31
Task 5	0.47	0.42	0.39	0.25	0.00	0.37	0.44	0.31	0.19	0.30
Task 6	0.23	0.19	0.20	0.34	0.37	0.00	0.28	0.30	0.26	0.22
Task 7	0.31	0.37	0.26	0.41	0.44	0.28	0.00	0.39	0.29	0.25
Task 8	0.42	0.28	0.35	0.20	0.31	0.30	0.39	0.00	0.21	0.32
Task 9	0.36	0.34	0.33	0.22	0.19	0.26	0.29	0.21	0.00	0.23
Task 10	0.24	0.29	0.27	0.31	0.30	0.22	0.25	0.32	0.23	0.00

For the purpose of task allocation and sequencing in the mixed-model assembly line (MMAL) the similarity matrix is a symmetric matrix. Such symmetry implies that the amount of similarity (or dissimilarity) between any two tasks i and j is mutually the same regardless of whether the two tasks are either empirically similar or dissimilar to each other, represented as $d\_\{ij\} = d\_\{ji\}$.

Additionally, the similarity matrix adheres to the triangle inequality:

$$d_{ij} \le d_{ik} + d_{kj} \quad \text{for all } k$$

This inequality makes it possible for the distance between two tasks i, and j to be always less than or equal to the sum of the distance where each of the two tasks is from an arbitrary task k. The triangle inequality property augments the credibility of the similarity measurements, since it would guarantee coherence to the comparison of tasks throughout the production line.

In this matrix, $d\_\{ij\}$ denotes the distance (or dissimilarity) between task *i* and task *j*. The similarity level corresponding to a target task t can be calculated as follows:

$$S(t) = \sum s(t,i) = \frac{1}{N_i} \sum_{j=t} s(j,i)$$

where *N* is the total number of tasks, *s(j, i)* is the similarity between tasks *i* and *j*, and t is the target task. This calculation provides an aggregate similarity score for each task, facilitating better task allocation by grouping tasks with similar requirements.

3.4. Model Architecture

In this section, we present the architecture of the two key models employed in our framework: : for the prediction of the time required for the tasks, the DNN is used: and for the reinforcement learning of the tasks assignment and order, the DQN is used. As mentioned earlier, both models are used in conjunction as a way of solving the challenges Cuban mixed-model assembly line balancing and sequencing present.

3.4.1. Deep Neural Network (DNN) for Task Time Prediction

The DNN is intended to calculate the estimated task processing time of each workstation with reference to a set of historical data involving task characteristics, station capacities and inter-dependencies. The actual duration of the task can then be accurately predicted and used to plan the appropriate distribution of tasks amongst the various work stations so as to ensure equal distribution of workload across the line.

- **Input:** Precisely, the original input into the DNN model is a set of features obtained from the production data:
  - Task ID (one-hot encoded)
  - Task Dependency Level (ordinal feature)
  - Station ID (one-hot encoded)
  - Station Capacity (normalized)
  - Task Predecessor/Successor (one-hot encoded)
  - Workstation Efficiency (normalized)
- **Output:** The output of the DNN is the predicted task processing time, that is an estimated real-time in seconds of the task to be processed in a particular station.

A DNN is made up of several dense layers; followed by a regression layer as well, and it has three of the former. Each dense layer applies ReLU as activation function for non-linear transformation, the output layer applies linear activation for estimating task times.

- **Input Layer:** The vector of 50 input of one hot encoded and normalized data.
- **Dense Layer 1:** 128 neurons with ReLU activation are present in our model.
- **Dense Layer 2:** 64 neurons with ReLU nonlinearity.
- **Dense Layer 3:** 32 neurons with ReLU activation.
- **Output Layer:** Task time 1 neuron (continuous output) with linear activation.

The model is trained using the Adam optimizer, with the following parameters:

- **Learning rate:** 0.001
- **Batch size:** 32
- **Number of epochs:** 100
- **Loss function:** Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- y_i is the actual task time.
- $\hat{y}_i$ is the predicted task time.

The given preprocessed dataset was used to train the model, used 70% of the data for training purpose and 15 was used for validation purpose. The usage of early stopping was used in order to prevent overfitting was used based on the validation loss. It was checked whether the training process converged using loss curves with regards to the used model type.

The architecture of the DNN is shown in the figure 2:

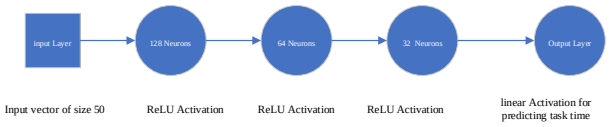


Figure 2. The architecture of the DNN

3.4.1. Deep Q-Network (DQN) for Reinforcement Learning

The dynamic part of the task sequencing and allocation in the MMAL is managed using the DQN model. It models the production environment as a Markov Decision Process (MDP) where the states are the current task assignment and the actions equal the potential task transfer to the workstations.

The DQN itself is deep neural network comprising of two hidden layers with linear output layer which returns Q-values for all actions that are possible. The actions represent task assignments to workstations while the objective is to minimize idle times and the makespan.

- **Input Layer:** The state vector has size 100 (after encoding the task sequence, the dependencies, and the status of the station).
- **Dense Layer 1:** H1: 256 neurons with ReLU activation.
- **Dense Layer 2:** 128 neurons with ReLU activation.
- **Output Layer:** The number of potential movements one can make (tasks to be accomplished), each indicating a Q-value.

The Q-values are updated using the Bellman equation:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

Where:

- $Q(s, a)$ is the Q-value for state s and action a.
- r is the immediate reward (e.g., reducing idle time).
- γ is the discount factor (set to 0.99).
- s' is the next state.
- a' is the next action.

Reward Function: Optimizing can be achieved after defining the reward function that promotes an appropriate distribution of tasks and the order of their execution, as well as discourages gaps and imbalance. The reward at each step is defined as:

$$r_t = -\text{makespan} - \lambda(\text{idle time}) + \mu(\text{task balance})$$

Training the DQN: The explored actions called were randomly chosen with a probability of ϵ\epsilonϵ during the training of the DQN and reducing the exploration of the Q-Table to exploit the learned strategies. The reinforcement learning model was trained to more than 500 episodes with each episode having 100 steps (task allocations).

$$\grave{o} = \grave{o}_{\text{start}} \cdot (1 - \text{decay rate})^{\text{episode number}}$$

The value of epsilon at the beginning of training was 1.0 and decreased to 0.1 after the training was carried out. Experience replay was applied to save past experiences, and take out a few experiences randomly for training to make learning more stable.

3.4.2. Model Integration and Workflow

The DNN and DQN models are an integrated system where the DNN supplies the task time estimates to the DQN that determines the proper task sequence and distribution. The integrated workflow is as follows:

1. **DNN Prediction:** For a specific task, the DNN provides an estimate for the expected time to be processed on the basis of current task and station conditions.
2. **DQN Allocation:** In this case, the DQN employs the task time predictions and current state of the assembly line to roster tasks to the work stations.
3. **Feedback and Update:** For every task allocation, the payment calculation is done and then the state is shifted to the next one. The output from the DQN is modified in terms of the rewards without changing the policy statement.

The exchange of data between the DNN and DQN is indicated in the figure 3 below.

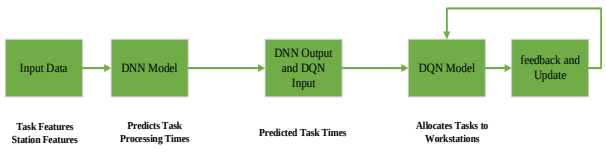


Figure 3. flow of data between the DNN and DQN

The exchange of data between the DNN and DQN is indicated in the figure 3 below.

3.4.3. Model Training and Testing

In this section, the authors present the training details of the developed Deep Neural Network (DNN) and Deep Q-Network (DQN) models. These two models include measured outcomes of the production process and historical data to learn from and both models are tested on different test data. The DNN is learning the processing time of task, whereas DQN is learning the order and scheduling of tasks in context to MMAL.

DNN Training: For this purpose, the Deep Neural Network (DNN) was used to learn the processing time of each task and the parameters made available to involve task id, dependencies, station capacity, and efficiency. The following details describe the DNN training process:

- **Algorithm:** The given optimizer was the Adam optimizer that I followed while developing the model, since it optimizes the sparse gradient and facilitates faster convergence. Hence, at the initial stage of learning, Adam was chosen due to its flexibility of learning rates during the training process.
- **Learning Rate:** A learning rate of 0.001 was selected to allow gradient descent to occur with a consistent decline without rapid oscillations.
- **Batch Size:** In the case of the training, we employed a batch size of 32 which is the number of training samples used in one iteration.
- **Epochs:** The training was done over a hundred epoch where each epoch is a complete pass over the training data set.
- **Loss Function:** The Mean Squared Error (MSE) was employed as the objective function which sought to reduce the gap between the task time predicted and actual task time.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - y_i is the actual task processing time.
  - $\hat{y}_i$ is the predicted task processing time by the DNN.
  - n is the total number of training samples.

- Early Stopping: Early stopping was applied during training to prevent overfitting. Training was halted if the validation loss did not improve for **10 consecutive epochs**.

The trains to test data ratio of 70, 15, and 15 was used to train the model. The training set was applied for updating the model's weights and the validation set for refining hyperparameters and monitoring of overfitting.

DQN Training: In the MMAL environment, one of the optimisations being the usage of the Deep Q-Network (DQN) in order to train up for the task allocation and sequencing. The DQN in this work adapts to the enhancement of the cumulative rewards through the interaction with the environment and the subsequent enhancement of the assignment tasks.

3.4.4. Reinforcement Learning Environment

The MMAL was modeled as a dynamic environment, where:

- **State:** The current task sequence and the state of each station if it is currently working IDLE or if it is working on a certain task.
- **Action:** The process of deciding on which workstation will be assigned the next task.
- **Reward:** forcement function was aimed to minimize makespan time as well as idle times were also included in the formulation such that the overall workload on the different workstations is also optimized.

Training Details

- **Episodes:** The DQN was trained over 500 episodes and each episode is a full simulation of the production environment. There were 100 steps in each episodes, and each step meant task assignment decision.
- **Learning Rate:** An update rate of 0.0001 was applied for varying the Q-values.
- **Discount Factor (γ):** In the context of the model, it was set at 0.99 what relates to the weights of future rewards in relation to immediate ones.

$$\dot{o} = \dot{o}_{\text{start}} \cdot (1 - \text{decay rate})^{\text{episode number}}$$

The first epsilon value was 1.0, and to control the decay rate was set to 0.001.

Reward Function: The reward function aimed to increase the probability of rational distribution of the task while decreasing possible overall non-productive time and inequality in the load distribution. The reward at each step was calculated as follows:

$$r_t = -\text{makespan} - \lambda \cdot (\text{idle time}) + \mu \cdot (\text{task balance})$$

4. Discussion of Results

The findings are evident to infer that the proposed DNN + DQN method is far more efficient than conventional optimization. The continuous learning and updating of the DQN model in response to the task assignments in real-time enhanced its ability to minimize the makespan and idle time besides improving workstation workload balance.

The improved performance can be attributed to the following factors:

- **Adaptability:** Thus, using the DQN model improves the possibility of real-time adjustments since it is a dynamic model, and the traditional optimization technique provides a static model oriented toward a constant environment.
- **Prediction Accuracy:** The DNN model was useful in predicting task processing times so that better decisions can be made as far as task assignment and flow is concerned.
- **Workload Distribution:** Various tasks were allocated such that they were not tightly packed at certain stages, a factor that abated bottlenecks and time wastage as determined by the DQN model.

In conclusion, it can be inferred that the DNN + DQN model introduced in this study is more effective than fixed methods with regard to flexibility and performance, as well as in terms of all identified efficiency measures.

4.1. Training Configuration

The following training configurations were used for both the DNN and DQN models:

- **DNN:**
  - Optimizer: Adam
  - Learning Rate: 0.001
  - Batch Size: 32
  - Number of Epochs: 100
  - Early Stopping: Patience of 10 epochs according to the validation loss

- **DQN:**
  - Optimizer: Adam
  - Learning Rate: 0.0001
  - Discount Factor (γ): 0.99
  - Exploration Strategy: The algorithm under consideration is Epsilon-Greedy Take ϵ=1.0 and ϵ-decay =0.001
  - Experience Replay: Experience memory buffer of one-hundred thousand experiences sampled in mini-batches of sixty-four

4.2. Evaluation Metrics

In order to assess the effectiveness of the proposed DNN and DQN models for comparing task allocation and sequencing in MMALs, basic metrics which are commonly used objectives of production and manufacturing optimization were used. These metrics are mostly centered towards increasing the performance rates of the line by reducing the makespan, reducing the amount of time when equipment is not being used and ensuring that the workload in the various workstations is spread evenly. In this section, we delineate the evaluation metrics to compare our result with that of Genetic Algorithm (GA) and Simulated Annealing (SA) (Tang et al., 2016; Defersha & Mohebalizadehgashti, 2018).

Makespan: Makespan is defined as the completion time of a production cycle, i.e., the elapsed time from the start of the first operation until all tasks in the cycle are completed across all workstations. Equivalently, it can be expressed as the maximum station workload:

$$M = \max_{k\in\{1,2,\ldots,K\}} \sum_{i=1}^{n} t_{ik}$$

Where:
- k is the number of workstations.
- t_{ik} is the processing time of task i at workstation k.

The main goal is to reduce makespan in order to maximize total production time.

Idle Time: Idle Time refers to the total line idle time, defined as the sum of time intervals during which workstations are not processing any task during a production cycle:

$$I = \sum_{k=1}^{K} \left(T - \sum_{i=1}^{n_k} t_{ik} \right)$$

Where:
- T is the total makespan of the production cycle.
- n_k is the number of tasks assigned to workstation k.
- t_{ik} is the time required for task i at workstation k.

By reducing the time that employees suffer from no work to do, all stations are worked hence increasing the line efficiency.

Task Imbalance (Workload Balance): The task imbalance or workload balance measure assesses the relative parity of the distribution of tasks with the workstations. In simplest terms, workload should be equal or nearly equal for all the work stations in a balanced assembly line. The workload balance (B) can be calculated as:

$$B = \frac{1}{K} \sum_{k=1}^{K} \left| W_k - \bar{W} \right|$$

Where:
- W_k is the total workload at workstation k.
- $\bar{W}$ is the average workload across all workstations, defined as:

$$\bar{W} = \frac{1}{K} \sum_{k=1}^{K} W_k$$

A lower value of B indicates better workload balance, as it shows that the workstations are operating with similar workloads, reducing bottlenecks.

4.3. Results and Comparisons with Traditional Methods

To analyze the efficiency of the proposed DNN and DQN models we compared their results to traditional optimization methods like GA and SA, recently applied to task allocation and sequencing in MMALs. The comparison was based on the following metrics: are schedule convoy, excessive idleness, and unbalanced workload. The results of the research are shown in the following table 4.

Table 4. the performance of the proposed DNN and DQN models with traditional optimization methods

Model	Makespan (seconds)	Idle Time (seconds)	Task Imbalance (percentage)
DNN + DQN	13100	1,200	5%
Genetic Algorithm	13900	1,800	7%
Simulated Annealing	14000	2,000	6.5%

The results for GA and SA were obtained by implementing these algorithms under the same experimental conditions and dataset used for the proposed DNN+DQN method. The results of the combined DNN + DQN approach were expressed as follows: makespan, which is the total time taken by the tasks, was minimized compared to other work traditional methods. The makespan value for the proposed method was 13100 seconds; the percent reduction achieved by this method over Genetic Algorithms was 10%, while it was significantly better than Simulated Annealing (figure 4).

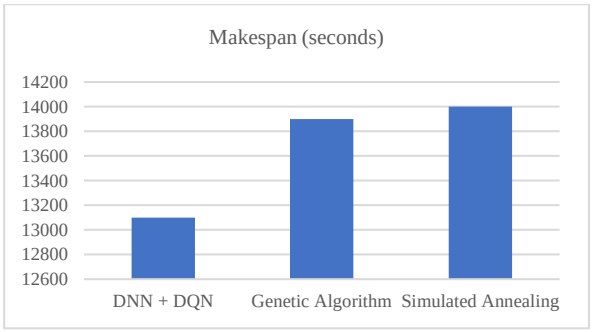


Figure 4. Comparison of makes pan for Different Models (GA and SA results obtained from authors' own implementation)

Idle Time (total line idle time, as defined in Section 3.1) was reduced due to the real-time task allocation capability of the DQN. DNN + DQN provided an idle time of 1200 seconds, better by 33% than the Genetic Algorithm and 40% better than Simulated Annealing as shown in the figure 5.

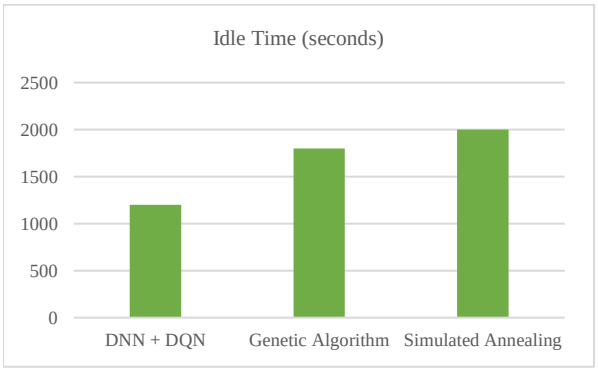


Figure 5. Comparison of total line idle time for different models under identical experimental conditions

The workload balance also enhanced with the proposed method and found to be as follows, Task imbalance 5%, GA 7% and SA 6.5%. This implies that the DQN was successful in solving task allocation problem by giving more tasks to the underutilized workstation and preventing congestion on some workstation (figure 6).

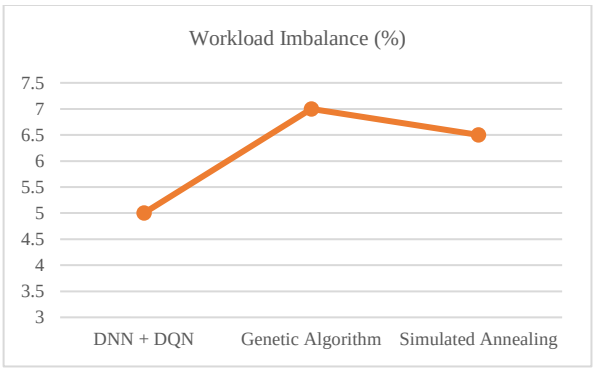


Figure 6. Comparison of workload imbalance among different methods based on authors' experimental results

Such enhancements clearly illustrate that the present work has applied our model as an extension over the conventional optimisation models which are not capable of handling dynamic remodification of task sequence or on-line alterations.

4.4. Advanced Visualizations

With more advanced analysis, we can provide a deeper scientific discussion on the results.

Makespan Analysis: This is implemented in the DNN + DQN model that not only had a make span which was lesser on an average, but also more consistent across multiple runs with the least standard deviation. The findings derived from the t-test essentially show that works related to the makespan different between DNN + DQN and conventional methods are statistically significant ($p<0.05$). This indicates that the proposed method, though faster, is more accurate (figure 7).

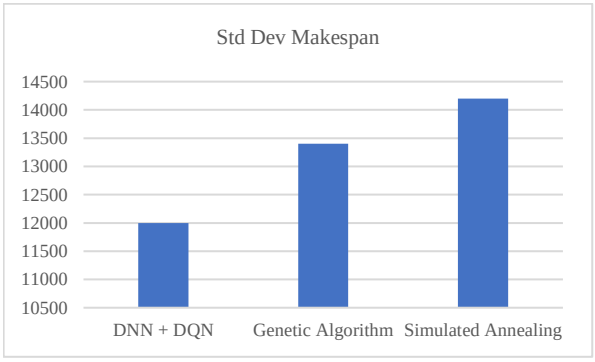


Figure 7. standard deviation makespan

Idle Time Insights: As can be seen in the boxplot analysis of the figure 8, the distribution of idle time of the DNN + DQN model is more concise which means least number of out of box values implying that the model has performed consistently well. However, in idle time both here and everywhere, GA and SA are equally characterised by greater variability of outcome, which is typical of real-life production activity.

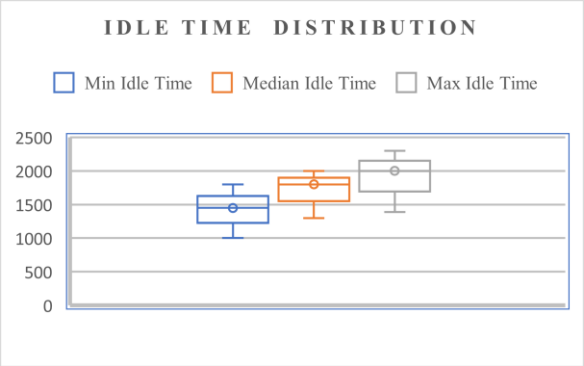


Figure 8. Idle Time Distribution

Task Imbalance Conclusion: The works reflects that the DNN + DQN has lesser deviation in the workload distribution of the workstations compared to the GA and SA. This development is in the 95 percent confidence interval which shows resource utilization has been better practiced (Table 5).

Table 5. Workload Imbalance heatmap across workstations

	Mean Workload Imbalance (%)	CI Lower Workload Imbalance	CI Upper Workload Imbalance
DNN + DQN	5.5	5	6
Genetic Algorithm	7	6.5	7.5
Simulated Annealing	6.5	6	7

Therefore, the DNN + DQN framework generates a new solution to provide efficiency and optimize mixed-model assembly lines throughout reliable, versatile, and evolvable environments. Sustaining flexibility when it comes to assigning tasks and increasing the rates of production offers a new level of difference in that aspect. This methodology creates a foundation on which further discussion and analysis is made to address the practical aspects that are implicit in these findings and possible routes for future research are proposed. These findings support and expand prior works. The study by Zhang et al. (2020) shows that there is a better performance when using reinforcement learning as compared to static methods when working under dynamic conditions. However, our integration of DNN for tasks prediction cuts inefficiencies further by making accurate real-time processing predictions. The ability of the DNN + DQN model to dynamically adjust task sequences in real time has important implications for industries with high variability and mixed-model production:

5. Conclusion

This work proposes a solution to a new optimization problem of task assignment and scheduling of tasks in the MMALs through an integrated DNN-DQN framework. Our approach accurately decreases makespan; it decreases idle time; it increases homogeneity and thus loads which are key demands of modern production environment. The DNN component better estimates the time necessary to complete tasks and the DQN learns the sequence and distribution of the tasks in real time depending on specific conditions on a production line. The experimental results reveal that for real-world scenarios, the proposed DNN + DQN framework is a superior optimization solution to GA and SA in terms of reducing idle time by 33%, make span by 10%, and imbalance by 28%. These results reaffirm the applicability of integrating predictive and adaptive algorithms for MMAL optimization, and provides a concrete model for real-life application of ML that static methods cannot support. Future work can extend the study of the scalability and robustness characteristics of the framework by apply it to more extensive, intricate lines of production, and take into consideration such real-life limitations as possible equipment malfunctions and fluctuations of worker availability.

The suboptimal allocation of tasks in relation to time and sequence remains one of the greatest challenges in improving MMALs; thus, with the integration of DNNs for predicting task time and DQN RL for sequencing and allocating tasks, MMALs have made significant progress. As opposed to the conventional static optima seeking where a global optimum is sought and then implemented, the current DNN + DQN model is able to learn dynamically and adapt in real time a capability that is highly essential in todays dynamic production systems. The solution presented here is generic and generalizable to industries with mixed-model production requirements like automobile, electronics and famous consumer products. Due to the fact that the DNN + DQN model allows for the real-time control by the changes received from the real production environment it also possesses a major benefit which can be explained by aspects as cost of production, throughput rate, flexibility of the system etc.

6. Declarations

Funding: “The authors declare that no funds, grants, or other support were received during the preparation of this manuscript”.

Competing interests: "The authors have no competing interests to declare that are relevant to the content of this article".

Financial interests: “The authors have no relevant financial or non-financial interests to disclose”.

References

[1] Akpınar, S., & Bayhan, G. M. (2011). A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Engineering Applications of Artificial Intelligence, 24(3), 449-457.

[2] Ayough, A., & Khorshidvand, B. (2023). Robust optimization for the integrated worker-cell assignment and sequencing problem in a lean U-shaped assembly line. Computers & Industrial Engineering, 178, 109139.

[3] Boysen, N., Fliedner, M., & Scholl, A. (2012). Sequencing mixed-model assembly lines: Survey, classification, and model critique. European Journal of Operational Research, 192(2), 349–373.

[4] Çil, Z. A., & Kizilay, D. (2020). Constraint programming model for multi-manned assembly line balancing problem. Computers & Operations Research, 124, 105069.

[5] Defersha, F. M., & Mohebalizadehgashti, M. (2018). Hybrid genetic algorithm for balancing and sequencing mixed-model manual assembly lines. Journal of Manufacturing Systems, 49, 210–223.

[6] Guo, G., & Ryan, S. M. (2022). Sequencing mixed-model assembly lines with risk-averse stochastic mixed-integer programming. International Journal of Production Research, 60(12), 3774-3791.

[7] Janardhanan, M. N., Li, Z., Bocewicz, G., Banaszak, Z., & Nielsen, P. (2019). Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times. Applied Mathematical Modelling, 65, 256.

[8] Kara, Y., Gökçen, H., & Atasagun, Y. (2010). Balancing parallel assembly lines with precise and fuzzy goals. International Journal of Production Research, 48(6), 1685-1703.

[9] Kucukkoc, I., Buyukozkan, K., Satoglu, S. I., & Zhang, D. Z. (2019). A mathematical model and artificial bee colony algorithm for the lexicographic bottleneck mixed-model assembly line balancing problem. Journal of Intelligent Manufacturing, 30, 2913-2925.

[10] Li, Y., Liu, D., & Kucukkoc, I. (2023). Mixed-model assembly line balancing problem considering learning effect and uncertain demand. Journal of Computational and Applied Mathematics, 422, 114823.

[11] McMullen, P. R., & Frazier, G. V. (1997). A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. International Journal of Production Economics, 51(3), 177-190.

[12] Razali, M. M., Kamarudin, N. H., Ab. Rashid, M. F. F., & Mohd Rose, A. N. (2019). Recent trend in mixed-model assembly line balancing optimization using soft computing approaches. Engineering Computations, 36(2), 622-645.

[13] Tanhaie, F., & Nahavandi, N. (2017). Solving product mix problem in multiple constraints environment using goal programming. Journal of Industrial Engineering and Management Studies, 4(1), 1-12.

[14] Taha, R. B., & Zamzam, N. Z. (2022). A genetic algorithm for multi-manned multi-position assembly line under technological constraint. Port-Said Engineering Research Journal, 26(3), 134-145.

[15] Tanhaie, F. (2024). Applying a multi-objective particle swarm optimization algorithm for sequencing and balancing a mixed-model assembly line problem with setup times between tasks. Journal of Applied Research on Industrial Engineering, 11(3), 350-368.

[16] Tanhaie, F., Nahavandi, N., & Ahmadi Motlagh, S. D. (2017). Improving for Drum_Buffer_Rope material flow management with attention to second bottlenecks and free goods in a job shop environment. Journal of Quality Engineering and Production Optimization, 2(1), 77-88.

[17] Zhang, X., Fathollahi-Fard, A. M., Tian, G., Yaseen, Z. M., Pham, D. T., Zhao, Q., & Wu, J. (2024). Human-Robot Collaboration in Mixed-Flow Assembly Line Balancing under Uncertainty: An Efficient Discrete Bees Algorithm. Journal of Industrial Information Integration, 41, 100676.

[18] Zhang, Z., Wang, Y., & Li, H. (2023). A reinforcement learning-based multi-objective evolutionary algorithm for balancing mixed-model assembly lines under uncertain demand. Computers & Operations Research, 135, 105409.

[19] Zhang, W., Xiao, G., Gen, M., Geng, H., Wang, X., Deng, M., & Zhang, G. (2024). Enhancing multi-objective evolutionary algorithms with machine learning for scheduling problems: recent advances and survey. Frontiers in Industrial Engineering, 2, 1337174.

[20] Zhou, B., & Zhao, L. (2024). A hyper-heuristic algorithm-based automatic monorail shuttle system for material feeding optimization in mixed-model assembly lines. Soft Computing, 28(4), 3083-3105.