Online Reinforcement Learning for Designing Automotive Hybrid Assembly Sequence: A Task Clustering-Guided Approach

Anass Elhoud[1,2,*], Benoit Piranda[1], Raphael De Matos[2], and Julien Bourgeois[1]

[1] University of Franche-Comte, FEMTO-ST Institute, CNRS, Montbeliard, France
[2] Forvia Clean Mobility, R&D Center, Bavans, France
anass.elhoud@forvia.com

Abstract. This paper proposes a novel approach that integrates hierarchical clustering (HC) into reinforcement learning algorithms to address the simultaneous resolution of hybrid assembly line balancing (ALB-1) and assembly sequence planning (ASP). The proposed approach attempts to capture implicit constraints, derived from accumulated experiences and industry-specific knowledge, enhancing the adaptability of solutions. The inclusion of the clustering algorithm enhances the decision-making process of the reinforcement learning agent through the introduction of a problem-specific similarity reward. To evaluate the effectiveness of the approach, three adapted methods are implemented and tested: QL-HC, SARSA-HC, and SARSA without HC. The experimental results demonstrate the superior performance of our novel approach compared to traditional techniques, with SARSA-HC exhibiting particularly impressive results with 82 % of similarity to the expert's sequence. This approach offers increased flexibility, adaptability, and the capacity to incorporate expert knowledge and lessons learned from prior assembly line design experiences and proves to be a valuable tool for enhancing efficiency and reducing time-to-market in manufacturing settings.

Keywords: Reinforcement Learning · Hierarchical Clustering · Assembly Sequence Design · Assembly Line Balancing · One Piece Flow Optimization

1 Introduction

1.1 Background & Motivation

In the ever-changing landscape of the automotive industry, efficiency, precision, and adaptability are crucial. As the demands for innovation and quality continue to rise, automotive manufacturers face a big challenge: figuring out how to make machines and human operators work together smoothly on the assembly line. The fusion of advanced robotic technologies with skilled humans has given birth to the concept of the hybrid assembly line. In response to these remarkable technological advancements, the industrial manufacturing sector, with

a particular focus on assembly lines, has seen a significant transformation with the integration of artificial intelligence (AI) and cutting-edge technologies. The key motivation behind this integration is to address and effectively manage the growing complexity of problems encountered [21]. AI, with its remarkable capacity for adaptation, learning, and data-driven decision-making, holds the promise of substantially elevating efficiency, quality, and overall productivity and ultimately contributing to the competitiveness and sustainability of manufacturing businesses in a fast-evolving global market [6, 7].
On the other hand, designing hybrid assembly lines has now become increasingly challenging and demands the collaboration of multiple experts due to the involvement of various types of operation times, including machine time, manual time, and technical time. This blending of different behaviors in a hybrid system introduces a level of complexity that conceals time variables, making them difficult to quantify.

1.2 Challenges in Assembly Process Design

The current approach to designing assembly processes is time-consuming and repetitive, especially when applied to each new product with the same assembly process. Organizing industrial design workshops becomes expensive as complexity increases, and navigating organizational challenges becomes more difficult, compounded by the need to ensure the availability of all expert participants. While these workshops are crucial for validating manufacturing processes, there is room for optimization. Moreover, deeply rooted human biases toward familiar patterns can restrict the exploration of potentially superior solutions. While standardized rules are essential, blindly adhering to them risks stagnation. Hence, a more efficient and innovative approach is necessary, balancing the comfort of established rules with the potential of novel ideas. Only through this approach can we genuinely optimize assembly processes and unleash their full potential.

This paper introduces and evaluates a novel, fast, and semi-automated method for designing assembly sequences. Our approach prioritizes optimizing resource utilization, and production costs while adhering to standard rules, and maintaining a continuous integration of lessons learned from past projects.

2 Related Works

In recent years, the integration of AI solutions in the industrial sector has seen substantial growth. These AI-driven systems have demonstrated their significant utility in tackling complex challenges within the industrial landscape. AI's role in industrial manufacturing has evolved rapidly, proving to be a powerful tool in enhancing process control [18], optimizing decision-making, and hence improving the overall efficiency of the industry.

Over the past decade, researchers have invested substantial effort in developing techniques to address assembly line balancing and assembly sequence planning challenges. Traditional methods, such as linear programming [4], nonlinear

programming [23], and dynamic programming [24], were designed to tackle simpler assembly line scenarios characterized by deterministic tasks and relatively straightforward processes and well defined and stable constraints. However, these conventional methods often fall short when dealing with hybrid and complex assembly lines. They struggle to identify optimal solutions, get trapped in local optima, exhibit a risk of divergence, and become computationally intensive, making them impractical for real-world applications.

To overcome computational limitations, researchers have turned their attention to metaheuristic algorithms [17]. Metaheuristics, a category of optimization techniques, operate without explicit gradients. Swarm intelligence algorithms [5], evolutionary algorithms [11], and physics-based or human-based metaheuristics [22] have been explored in various studies to tackle assembly line problems. These algorithms employ a set of search agents that explore alternative solutions through a series of operations, aiming to improve solutions iteratively. Despite their effectiveness in obtaining sub-optimal solutions within a reasonable amount of time, metaheuristics exhibit stochastic behavior, which is why they cannot guarantee the discovery of the global optimal solution, and each optimization run operates independently without incorporating learning from past experiences — an essential aspect in assembly line design. Notably, these methods lack flexibility and demand explicit expression of constraints and rules [1].

Numerous studies have explored the idea of framing Reinforcement Learning (RL) as a global sequence modeling problem to predict a sequence of actions that result in a sequence of high rewards [10]. This concept has found application in a variety of sequencing problems within the IT and industrial domains [16]. However, only a limited number of research works have addressed the simultaneous resolution of assembly line balancing and assembly sequence planning problems using RL techniques. Furthermore, among those that have, their focus has predominantly been on relatively simple assembly lines, either staffed exclusively by manual operators or entirely automated with robots [13].

RL has the potential to be a valuable tool in this context [2], as it can be used to identify optimal policies for sequencing assembly tasks in a manner that minimizes production costs and adheres to various operational constraints. What's particularly intriguing is that reinforcement learning enables the incorporation of rules and insights gained from previous experiences into the design policy, allowing for adaptability and refinement over time. Unfortunately, this RL's potential in real-life hybrid assembly lines has received limited exploration, leaving room for future research and innovation.

3 Problem Statement

The design of the hybrid assembly process consists of defining the assembly sequence and scenario, the number of machines and the human resources while ensuring that the proposed solution is both feasible and capable of meeting the specified Target Cycle Time (TCT). This problem is a general version of two combined problems known as type-1 assembly line balancing (ALB-1) and

assembly sequence planning (ASP), both considered NP-hard.
The proposed assembly process should respect predefined constraints which can be organized into three categories:

- Precedence Constraints: The assembly process is subject to a network of precedence constraints. These constraints dictate the global order in which tasks must be executed, ensuring that critical dependencies are respected.
- Restrictions: In certain cases, restrictions are imposed due to ergonomic, quality considerations or process-related factors.
- Hypotheses: Depending on the specific characteristics of each company, foundational assumptions underlie the proposed assembly process. In our case, these assumptions are articulated as follows:
  - All assembly process durations are treated as deterministic and constant.
  - Each assembly operation is allocated to a single, dedicated workstation.
  - Each workstation is operated by a unique operator.

We introduce the concept of **implicit constraints** — a set of unspoken guidelines — that emerge from accumulated experiences, industry-specific knowledge, and the insights of experts. These constraints are challenging to formalize but crucial in problem-solving, drawing from cumulative expertise in a domain. Unlike codifiable explicit constraints, implicit constraints resist standardization and align with specific problem nuances. Insights from post-project reviews across industries inform implicit constraints for future projects. These constraints are not static rules but adapt to dynamic environments, as seen in experts adjusting designs based on unforeseen circumstances.
Dynamically modeling these implicit constraints represents a pivotal advancement in addressing assembly process design challenges. It is crucial to highlight a key differentiation in our problem statement from previous research efforts. We are addressing the concurrent optimization of type-1 assembly line balancing and assembly sequence planning (ASP) simultaneously, employing novel approaches to formalize implicit constraints continuously.

4 Mathematical Model

Let N be the number of components to be assembled. Each task i has a processing time or duration t_i and a set of constraints. We define the assembly sequence by the vector X_i and its corresponding assembly assignment by the matrix Y_{ij}.

$$x_i = (\text{Position of task } i \text{ in the sequence}),$$

$$y_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned to workstation } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{pj} = \begin{cases} 1, & \text{if operator } p \text{ is assigned to workstation } j, \\ 0, & \text{otherwise.} \end{cases}$$

where $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$ and $p = 1, 2, \ldots, W$.

The first objective function f_1, known as the McGovern–Gupta balance function, seeks to achieve a dual objective: it aims to reduce the number of machines while also striving to maintain balanced idle times at each workstation. We recall that TCT represents the Target Cycle Time.

$$f_1 = \sum_{j=1}^{M} (TCT - CT_j) = \sum_{j=1}^{M} \left(TCT - \sum_{i=1}^{N} (t_i \times y_{ij}) \right) \tag{1}$$

Similar to the optimization for machines, the second objective function aims to reduce both the number of operators and their idle times.

$$f_2 = \sum_{p=1}^{W} (TCT - CT_p) = \sum_{p=1}^{W} \left(TCT - \sum_{j=1}^{M} \left(t_j^{man} \times z_{pj} \right) \right) \tag{2}$$

Remark 1. Manual operating time t_j^{man} is variable and dependent on the task sequence and assignments. This variability arises because manual operations can be adjusted or reduced based on the specifics of the assembly task. For instance, an operator at a particular workstation might handle and pick only 2 components instead of 3, as is the case at another workstation. This adaptability in manual operating time is considered in the optimization process.

We define the mathematical model of the combined ALB-1 and ASP:

$$\min_{(x_i, y_{ij}, z_{pj})} \quad w_1 f_1 + w_2 f_2 \tag{3a}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_i = \frac{N(N+1)}{2}, \tag{3b}$$

$$\sum_{i=1}^{N} y_{ij} = 1, \quad \forall j = 1, 2, \ldots, M, \tag{3c}$$

$$\sum_{p=1}^{W} z_{pj} = 1, \quad \forall j = 1, 2, \ldots, M, \tag{3d}$$

$$\sum_{j=1}^{M} z_{pj} \geq 1, \quad \forall p = 1, 2, \ldots, W, \tag{3e}$$

$$x_i < x_k, \quad \forall (i,k) \in P \tag{3f}$$

$$y_{ij} \neq y_{pj} \quad \forall (i,p) \in F \tag{3g}$$

$$\Phi(x_i) = 0 \quad \forall i \in \{1, 2, \ldots, n\} \tag{3h}$$

$$x_i \in \mathbb{N} \quad \forall i \in \{1, 2, \ldots, N\} \tag{3i}$$

$$y_{ij} \in \{0, 1\} \quad \forall j \in \{1, 2, \ldots, M\} \tag{3j}$$

Constraint 3b ensures that each task is assigned a unique position in the sequence, Equation 3c verifies that every task is assigned to a unique workstation, Equation 3d expresses the requirement that each workstation has one

unique operator, Equation 3e asserts that an operator can work on at least one workstation. Equation 3f refers to the precedence constraint. Constraint 3g, also known as the restrictions, ensures that incompatible tasks are not assigned to the same workstation. Equation 3h encodes the implicit constraints that are process-related and not explicitly defined.

5 Background on Methods

5.1 Hierarchical Clustering

Hierarchical clustering is an unsupervised learning method involving recursive partitioning or grouping n data points into clusters. Unlike traditional clustering methods that require a predefined number of clusters, hierarchical clustering offers flexibility, allowing data to be organized into a varying number of clusters, ranging from 2 to n clusters [12]. This attribute is particularly useful when we aim to construct a hierarchy of clusters and seek to visualize data at multiple levels of detail. The hierarchical nature of this approach, often portrayed through dendrograms, enables the exploration of complex data structures and can reveal insights at both macro and micro levels. The greater the height difference, the more dissimilar the concerned data points. In the rest of this paper, we will employ the term *Hierarchical clustering* in a global context, referring specifically to the *Agglomerative method*, as it is our chosen method.

5.2 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning inspired by behaviorist psychology. It entails an agent learning to make decisions through trial and error in an environment. The agent follows a policy (π), executing actions that influence the environment's state and receives feedback in the form of rewards, guiding its learning process. This reward tells the agent whether the action taken was good or bad, helping it to distinguish desirable behaviors from undesirable ones [15]. Reinforcement learning traces back to optimizing solutions for sequential decision problems, often modeled as Markov Decision Processes (MDPs) in the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ denotes transition probability, and $\mathcal{R}$ is the reward function. The agent's primary objective is to maximize the accumulation of rewards over time by adapting its decision-making policy based on received rewards. A key concept involves a discount factor, typically denoted as γ, which accounts for the immediate impact of an action and the diminishing influence of past actions over time. The discounted accumulated future reward, known as expected return (G_t), is expressed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{4}$$

In this equation, R_{t+k} is the reward at time step $t+k$. The summation is often finite, considering a terminal state T due to finite environmental horizons.
Let π^* be the optimal policy that an agent wants to learn. It has been proven in [3] that, if the optimal policy exists, it should satisfy Equation 5 for any pair (s, a). This equation is known as Bellman Optimality Equation

$$q_{\pi^*}(s,a) = \mathbb{E}\left[R_{t+k+1} + \gamma \max_{a'} q_{\pi^*}(s', a')\right] \tag{5}$$

This equation tells us that under the optimal policy π^*, the expected total reward for a state-action pair (s, a) is equal to the immediate reward for taking action a in the state s plus the maximum expected cumulative reward that can be obtained by making optimal choices in subsequent states. This formulation opened up the door to building algorithms for iterative learning of the optimal policy. For the rest of the paper, we will focus on two methods of RL: Q-Learning Algorithm and State–Action–Reward–State–Action (SARSA) Algorithm.

5.3 Q-Learning

Q-Learning is one of the most common model-free, value-based, off-policy methods in reinforcement learning. Its primary objective is to iteratively learn the optimal Q-value function by leveraging the Bellman Optimality Equation (5). In this approach, the algorithm maintains a table containing Q-values for various state-action pairs, and these values are updated at each time step according to the update formula expressed in Equation (6).

$$q_{t+1}(s,a) = q_t(s,a) + \alpha\left(R + \gamma \max_{a'}\left(q_t(s', a')\right) - q(s,a)\right) \tag{6}$$

In the update formula, $q_{t+1}(s, a)$ and $q_t(s, a)$ correspond to the new and previous Q-values associated with the state-action pair (s, a). The variable R signifies the reward obtained by taking action a in state s. The algorithm adapts these Q-values through trial and error, using a learning rate α to balance the influence of new information with prior knowledge. The term $\gamma \max_{a'} q_t(s', a')$ reflects the maximum expected future return, considering the best possible action a' in the resulting state s'. This way, Q-Learning iteratively refines its Q-value estimates to approximate the optimal action-value function progressively.

5.4 SARSA

SARSA is another algorithm for reinforcement learning (RL), its name stands for State-Action-Reward-State-Action. This algorithm is a model-free, value-based, and on-policy method that differs from Q-Learning on this on-policy side, and exactly in how it updates the Q-values. SARSA also updates these values iteratively based on the Bellman Optimality Equation but taking the current policy instead of another independent policy, following this equation:

$$q_{t+1}(s,a) = q_t(s,a) + \alpha\left(R + \gamma q_t(s', a') - q(s,a)\right) \tag{7}$$

SARSA uses the policy it aims to learn to update its Q-table. In contrast, Q-Learning employs distinct policies: one for selecting the next action and another for updating its Q-table. Consequently, Q-Learning directly converges towards the optimal policy, while SARSA learns a near-optimal policy when actively exploring alternative options. Thanks to this, SARSA exhibits a cautious approach, inching towards convergence while avoiding dangerous sub-optimal paths.
Advancements in RL have witnessed the emergence of deep learning-based algorithms such as Deep Q-Network (DQN) [14] and Proximal Policy Optimization (PPO) [20]. While these methods have shown significant effectiveness in addressing various complex issues by leveraging advanced neural network architectures and optimization techniques, we argue that employing such sophisticated RL methods may not be necessary in our particular scenario. The problem structure at hand could be effectively tackled using simpler approaches.

6 Methodology

We reframe the assembly sequence as a Markov Decision Process (MDP). In this model, the actions space is represented as lists of assembly tasks, while the state space consists of lists of assembly tasks extended by the required number of resources up to instant t. When we choose a task as an action, it transitions the agent to the next state in the sequence. This simplifies the modeling and learning process, making it more tractable. Additionally, it allows for a more efficient representation of the system, as the state encapsulates the essential information needed to make decisions at any given moment. The assembly process is constructed step by step throughout the episode, with the episode concluding when all assembly tasks are executed, marking it as the terminal state.

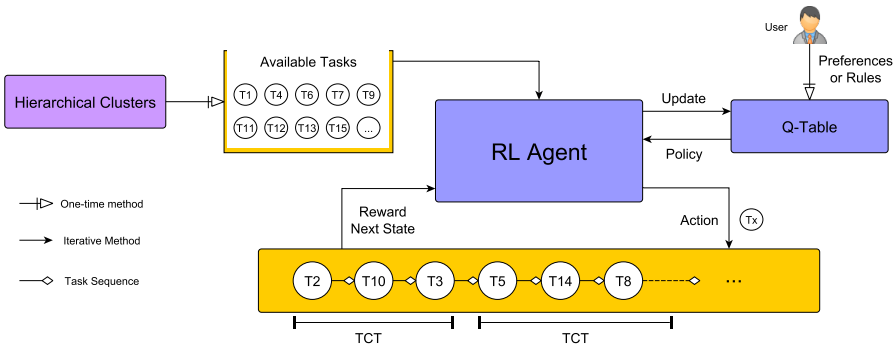

Fig. 1. Our Clustering-Guided RL Approach

The primary challenge in this modeling approach lies in determining the reward function. The final objective function, as outlined in Equation (3a), is

not immediately computable after each action is taken. Rather, it can only be assessed at the end of the cycle, once all tasks have been completed.

Proposition 1. *Value-based methods like Q-Learning and SARSA are more convenient for solving the assembly line balancing and assembly sequence planning due to their ability to estimate the value of states, facilitate exploration and exploitation trade-offs, and leverage historical data into the model by focusing on value estimation rather than policy derivation.*

Our approach comprises two fundamental components. The initial component, the clustering block, processes input data associated with various tasks and their attributes. It employs a hierarchical algorithm to classify tasks based on multiple factors, such as the components to be assembled, their complexity, the involved technology, physical characteristics, task precedence, and constraints. The clustering block has two main functionalities: it organizes tasks into groups based on similarity, which can help in identifying common patterns or characteristics among tasks, and it helps the RL algorithm to generalize better across different manufacturing technologies. The clustering distance primarily considers the attributes of assembly tasks, with non-numerical attributes being one-hot encoded. After the clustering phase, we implement the second vital component, which incorporates the reinforcement learning algorithm. It undergoes a learning phase guided by the outcomes of the hierarchical clustering. Instead of relying only on precedence constraints and restrictions to guide the learning algorithm, we introduce a novel element r_{sim}: a similarity reward derived from the hierarchical clustering of tasks. This reward offers additional insights to RL agents, influencing their exploration and exploitation processes. Moreover, expert-specific rules can be easily integrated by initializing the Q-table with specific action-state values based on the expert's preferences. In times of production ramp-down or ramp-up, the method can rapidly determine the ideal quantity of machines to maintain operational and pinpoint any excess machines that could be utilized on alternative assembly lines. Additionally, lessons learned can also be incorporated as an attribute, influencing the clustering distance and therefore the agent's reward. We define the reward as a function of a local immediate reward denoted as $R_{local}(a, s)$ and a global delayed reward labeled as R_{global}.

$$R(a, s) = R_{local}(a, s) + \mathbb{1}_D \times R_{global}$$

$$R_{local}(a, s) = \begin{cases} +X & \text{if precedences of } \boldsymbol{a} \text{ are respected.} \\ -X & \text{if precedences of } \boldsymbol{a} \text{ are NOT respected.} \\ +r_{sim} & \text{if } \boldsymbol{a} \text{ is in the same cluster of } \boldsymbol{s}. \\ -r_{sim} & \text{if } \boldsymbol{a} \text{ is NOT in the same cluster of } \boldsymbol{s}. \end{cases}$$

$$R_{global} = -w_1 f_1 - w_2 f_2$$

r_{sim} denotes the similarity results derived from the hierarchical clustering. We summarize the pseudo-code of our proposed approach in the Algorithm 1.

In summary, hierarchical clustering acts as a recommendation system for the RL algorithm by continuously organizing tasks into a hierarchical structure. This

Algorithm 1 Hierarchical Clustering-Guided Reinforcement Learning

Input: RL Method, Number of episodes, α, γ
Output: Best Sequence

```
Initialize Q-table with user preferences or zeros.
Initialize clustering parameters.
Cluster tasks based on all their attributes (HC).
Compute similarity rewards of all actions/states based on HC.
for each episode do
    Initialize state s
    while s is not terminal do
        Choose action a and observe new state s' based on the chosen RL Method.
        Update Q-value with R_local(a, s) based on the chosen RL Method.
        Update state: s ← s'
        Update and re-cluster available actions (HC).
    end while
    Update All Q-values with R_global based on the chosen RL Method.
end for
Get best policy π* based on Q-values.
return Best sequence based on π*.
```

guides reward allocation based on task similarities, relationships, and lessons learned, enhancing the algorithm's learning and decision-making abilities.

7 Results and Discussions

We conducted three distinct experiments for each RL algorithm, namely QL-HC, SARSA-HC, and SARSA. In the third experiment, SARSA was executed without task clustering to serve as a benchmark for evaluating the potential benefits of task clustering before RL agent training.

Table 1. Parameters of implemented methods.

	QL-HC	SARSA-HC	SARSA
Clustering Method	Hierarchical	Hierarchical	None
Number of Episodes	100K	100K	100K
Max Iterations	100	100	100
Learning Rate α	10^{-3}	10^{-3}	10^{-3}
Discount Factor γ	0.5	0.5	0.5
Training Strategy	ϵ-greedy	ϵ-greedy	ϵ-greedy
Learning Method	Off-Policy	On-Policy	On-Policy

All experiments were conducted on a Windows computer equipped with an Intel(R) Core(TM) CPU i9 running at 3.50 GHz and 128 GB of RAM. Each

method was subjected to 50 independent runs, each consisting of 100,000 episodes with a distance threshold of 0.5 for the hierarchical clustering. The tested use case involves a real automotive assembly process.
First, we present the average best objective values generated by these methods, which were averaged over the 50 independent runs. We recall that the objective functions are designed to be minimized, representing the number of workstations and operators required and their balancing ratio as expressed in Equation 3a.

Table 2. Average Performance Indicators for each method in SI.

	QL-HC	SARSA-HC	SARSA	Expert
Objective f_1	33.8	**29.2**	34.1	35.0
Objective f_2	10.0	**8.8**	10.3	15.0
Sequence Errors	**0**	**0**	4	**0**
Execution Time	10	10	**8**	480
Global Cost Ratio	0.26	**0.22**	0.65	0.45

Table 2 illustrates that, on average, all three methods generate sequences with superior objective values compared to those produced by the expert. Notably, SARSA-HC emerges as the method that proposes more promising sequences without encountering any constraint violations. In contrast, SARSA, when executed without clustering, exhibits a higher incidence of sequence errors on average, underscoring the importance of incorporating the clustering phase into the methodology. We note that the estimated cost ratio is normalized to facilitate a more meaningful and confidential comparison.

The learning curves depicted in Figure 2 provide a comprehensive insight into the convergence and stability of three distinct methods during the learning phase. To capture the global behavior of these methods, we computed the average reward over a window of 100 episodes. SARSA-HC stands out as the most consistently stable method among its counterparts, primarily due to its distinctive policy update strategy. In contrast, SARSA without clustering stabilizes at a lower reward level compared to the other two methods due to its limited knowledge of explicit constraints with no incorporation of clustering information. This observed fluctuation has been quantified, revealing a remarkable 93 % consistency for SARSA-HC across all 50 runs. In comparison, QL-HC exhibits an 86 % consistency, and the conventional SARSA method lags slightly behind with an 83 % consistency.

In the evaluation of sequence similarity, diverse scoring metrics are used to quantify the resemblance between two sequences. Notably, the Longest Common Subsequence score (LCS) [19] is employed for sequence matching, measuring the length of the longest common subsequence between two sequences. Kendall's Tau Correlation (KTC) [9] focuses on ranking comparison, capturing pairwise disagreements in rankings. Finally, Dynamic Time Warping (DTW) [8] facilitates the assessment of sequence similarity in time series data by allowing non-linear

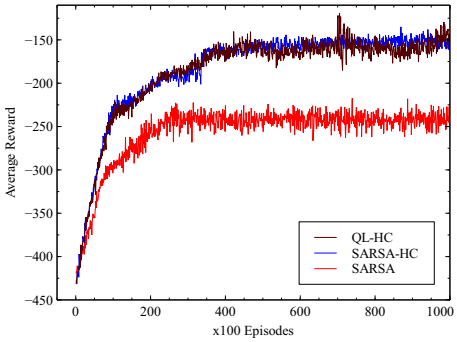

Fig. 2. Learning Curve of the three methods.

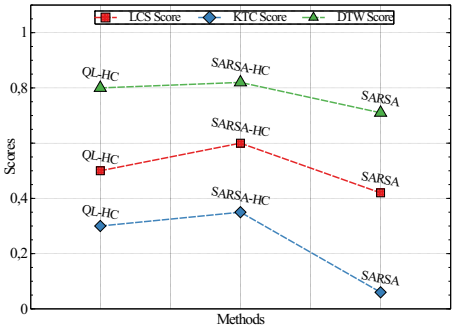

Fig. 3. Comparison of similarity scores among different methods.

alignment of elements. Each of these scoring methods serves a unique purpose in capturing different aspects of sequence similarity. KTC stands out as a particularly severe scoring metric due to its diligent consideration of sequence rankings.

The findings underscore the superiority of SARSA-HC over other methods, highlighting the effectiveness of incorporating hierarchical clustering in adjusting the feasibility of generated solutions. Sequences produced by methods with clustering exhibit a higher likelihood of acceptance by experts, particularly in terms of lessons learned adherence, as indicated by the three scores in Figure 3. The statistical analysis reveals significant results at a 95 % confidence level for SARSA-HC and QL-HC with respective p-values of 6×10^{-4} and 2×10^{-2}. However, there is insufficient statistical evidence to support the significance of results for SARSA without clustering.

8 Conclusion

This paper addressed the complex challenges of hybrid assembly process design by simultaneously considering assembly line balancing (ALB) and assembly sequence planning (ASP) problems. The proposed methodology introduced a rapid, expert-guided, and adaptable tool for the initial design of assembly lines. It integrated explicit constraints, such as precedence constraints and restrictions, with a novel exploration of implicit constraints derived from accumulated experiences and industry-specific knowledge. Additionally, during periods of reduced or increased production rates (ramp-down or ramp-up), the approach could swiftly determine the optimal number of machines to be kept operational and identify the surplus machines available for utilization on other assembly lines.
The mathematical model presented in this work combined ALB and ASP into a unified optimization problem, aiming to find a feasible high-rewarding sequence while minimizing the number of machines and operators while balancing their workloads. The introduction of implicit constraints represented a significant ad-

vancement in capturing real-world complexities that traditional explicit constraints may overlook.

We introduced a reinforcement learning (RL) approach guided by hierarchical clustering, providing a versatile and expert-friendly tool for assembly line sequencing. Three RL methods, namely QL-HC, SARSA-HC, and SARSA, were implemented and evaluated. The results indicated that the SARSA-HC method outperformed the other two in terms of convergence stability and sequence quality. In addition, the analysis of sequence similarity using different metrics (LCS, KTC, DTW) revealed that the RL methods consistently produced sequences that aligned well with expert-generated sequences and standard rules. The statistical significance of the results underscored the reliability and effectiveness of the proposed RL approach with hierarchical clustering.

This work contributes not only to the advancement of hybrid assembly process design methodologies but also to the broader field of optimization and RL applications in complex manufacturing scenarios. The combination of explicit and implicit constraints, along with the integration of clustering-guided RL, opens avenues for further research and application in diverse manufacturing environments. In future work, the proposed methodology can be extended to consider dynamic aspects of assembly processes, allowing adaptation to changing production demands and unforeseen disruptions. Additionally, the integration of real-time data and feedback loops can enhance the responsiveness and adaptability of the assembly line sequencing approach.

References

1. A. Alorf. A survey of recently developed metaheuristics and their comparative analysis. *Engineering Applications of Artificial Intelligence*, 117:105622, 2023.
2. L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.
3. Z. Ding, Y. Huang, H. Yuan, and H. Dong. Introduction to reinforcement learning. *Deep reinforcement learning: fundamentals, research and applications*, 47, 2020.
4. M. Eghtesadifard, M. Khalifeh, and M. Khorram. A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. *Computers & Industrial Engineering*, 139:106182, 2020.
5. A. El Houd, B. Piranda, R. De Matos, and J. Bourgeois. Swarm intelligence-based framework for accelerated and optimized assembly line design in the automotive industry. *Journal of Intelligent Manufacturing*, pages 1–15, 2023.
6. C. El Mazgualdi, T. Masrour, N. Barka, and I. El Hassani. A learning-based decision tool towards smart energy optimization in the manufacturing process. *Systems*, 10(5):180, 2022.
7. Y. Guo, W. Zhang, Q. Qin, K. Chen, and Y. Wei. Intelligent manufacturing management system based on data mining in artificial intelligence energy-saving resources. *Soft Computing*, 27(7):4061–4076, 2023.
8. Q. He, F. Borgonovi, and J. Suárez-Álvarez. Clustering sequential navigation patterns in multiple-source reading tasks with dynamic time warping method. *Journal of Computer Assisted Learning*, 39(3):719–736, 2023.

9. S. Jadhav and S. Ma. Kendall's tau for functional data analysis. *arXiv preprint arXiv:1912.03725*, 2019.
10. M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.
11. H.-Y. Kang and A. H. Lee. An evolutionary genetic algorithm for a multi-objective two-sided assembly line balancing problem: a case study of automotive manufacturing operations. *Quality Technology & Quantitative Management*, 20:66–88, 2023.
12. B. Karthikeyan, D. J. George, G. Manikandan, and T. Thomas. A comparative study on k-means clustering and agglomerative hierarchical clustering. *International Journal of Emerging Trends in Engineering Research*, 8(5), 2020.
13. Y. Lv, Y. Tan, R. Zhong, P. Zhang, J. Wang, and J. Zhang. Deep reinforcement learning-based balancing and sequencing approach for mixed model assembly lines. *IET Collaborative Intelligent Manufacturing*, 4(3):181–193, 2022.
14. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
15. S. Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.
16. Y. Ping, Y. Liu, L. Zhang, L. Wang, and X. Xu. Sequence generation for multi-task scheduling in cloud manufacturing with deep reinforcement learning. *Journal of manufacturing systems*, 67:315–337, 2023.
17. A. N. Ramli and M. F. F. Ab Rashid. A review of assembly line balancing optimisation with energy consideration using meta-heuristic algorithms. *Proceedings of the Institution of Mechanical Engineers*, 236(5):475–485, 2022.
18. B. Rolf, I. Jackson, M. Müller, S. Lang, T. Reggelin, and D. Ivanov. A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research*, 61(20):7151–7179, 2023.
19. A. Rubinstein and Z. Song. Reducing approximate longest common subsequence to approximate edit distance. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1591–1600. SIAM, 2020.
20. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
21. A. Simeth and P. Plapper. Artificial intelligence based robotic automation of manual assembly tasks for intelligent manufacturing. In *Smart, Sustainable Manufacturing in an Ever-Changing World: Proceedings of International Conference on Competitive Manufacturing (COMA'22)*, pages 137–148. Springer, 2023.
22. K. Wang, X. Li, L. Gao, P. Li, and S. M. Gupta. A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing*, 107:107404, 2021.
23. T. Yin, Z. Zhang, Y. Zhang, T. Wu, and W. Liang. Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations. *Robotics and Computer-Integrated Manufacturing*, 73:102251, 2022.
24. Y. Zhang, Z. Zhang, Y. Zeng, and T. Wu. Constraint programming for multi-line parallel partial disassembly line balancing problem with optional common stations. *Applied Mathematical Modelling*, 2023.