INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

IJIE

Vol. 17 No. 1 (2025) 17-30
https://publisher.uthm.edu.my/ojs/index.php/ijie

Adaptive Production Capacity Planning Under Variable Electricity Cost Using Deep Reinforcement Learning

Thachanon Danket[1], Sansiri Tanachutiwat[2]*, Vichai Rungreunganun[1]

[1] *Department of Industrial Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, THAILAND*
[2] *The Sirindhorn International TGGS, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, THAILAND*

*Corresponding Author: sansiri.t@tggs.kmutnb.ac.th
DOI: https://doi.org/10.30880/ijie.2025.17.01.002

Article Info

Received: 1 July 2024
Accepted: 27 October 2024
Available online: 30 April 2025

Keywords

Deep reinforcement learning, production capacity planning, variable electricity cost, asynchronous advantage actor-critic, proximal policy optimization

Abstract

Reinforcement learning is gaining traction for its ability to solve complex tasks that are intractable or impossible for other machine learning techniques. This paper proposes a novel approximation technique for production capacity and inventory planning using deep reinforcement learning (DRL). To address practical implementation challenges, we incorporate demand uncertainty and time-of-use electricity price-driven demand response patterns (PDDR) into the model. We compare the performance of two DRL techniques, A3C and PPO, in learning to optimize production planning over time to minimize total cost. The Discrete-Time MILP with new changeover constraint equations was formulated to take the model's optimal solution as an upper benchmark. Our results show that the PPO outperforms the A3C and expert heuristics with an optimality gap of 4.03% compared to MILP, and its simulation time is 2,502 times faster than that of MILP. Furthermore, our findings suggest that PPO is more robust regarding demand fluctuations than A3C due to its objective clipping mechanism stabilizing policy updates. This makes our PPO-based production planning model a promising candidate for real-world applications where demand fluctuations are common.

1. Introduction

Production capacity planning (PCP) is a critical decision-making process of production planning problems that determines each product's production volume and production period to be consistent with the demand for products or services. In detailed PCP, production tasks are allocated to available machines at any period within a given time frame to produce, store, and deliver products in response to customer orders at different periods [1, 2]. To maximize the total planning profit, these decisions must be precisely arranged in advance concerning many factors, such as the amount of inventory, daily delivery product demands, number of changeover times, electricity cost, etc. Electricity costs are a significant factor, accounting for 18 percent of total energy costs, and Energy cost is 60% of the total production cost [3]. Electricity is transmitted and charged in industrial estates as PDDR (Peak Demand-Driven Rate). Different PDDR patterns will affect the characteristics of the PCP of industrial plants [4, 5].

Production planning problems deal with multiple variables and constraints that must be considered when finding the optimal solutions. As problem complexity increases, the number of variables and constraints also increases, making it exponentially more challenging to find an optimal solution. As a result, production planning problems are considered NP-hard. Traditional optimization methods are still used to create mathematical models for solving production planning problems [6-12]. However, as problems become more complex, many research

studies have used metaheuristics to reduce solution time and find solutions that are close to the optimal plan [13-17].

Machine learning (ML) is a rapidly growing field with the potential to revolutionize many industries, including production planning [18]. In production planning, traditional machine learning, like supervised and unsupervised methods, struggles with dynamic environments and complex relationships. While supervised learning needs pre-labeled data, often unavailable in real-time production, unsupervised learning approaches cannot improve the quality of the results without a clear goal. Conversely, reinforcement learning (RL) can learn through trial and error to adapt to unpredictable demands and non-linear costs. This flexibility allows it to minimize costs across periods, even among changing environments and competing objectives. It can learn to adapt to compatibility with variable changes in the production planning environment. In recent years, there have been several research studies that have applied reinforcement learning (RL) to solve production planning problems [1, 19-24]. Some studies have compared RL to meta-heuristic methods and found that RL can provide better quality answers [23, 25].

Recent research highlights the potential of Reinforcement Learning (RL) for optimizing Production Planning and Control (PPC). Panzer [26] demonstrate Deep RL's (DRL) effectiveness in production tasks, surpassing traditional adaptability and uncertainty handling methods. However, safety, reliability, and real-world validation remain hurdles for widespread adoption. Esteso [2] emphasizes the RL's promise for PPC, particularly in dynamic environments. While RL outperforms traditional methods in handling uncertainties, limitations include adaptation to significant changes and a lack of research on tactical/strategic decision-making in PPC. To address these, Esteso proposes comparative studies between RL methods and the exploration of learning techniques for improved environmental adaptation.

Reinforcement learning encompasses a diverse range of algorithms, each with its unique mechanism developed by researchers. Prominent among these are A2C, A3C, TRPO and PPO. Extensive research has been conducted to compare the performance of these RL algorithms against each other and against metaheuristic approaches in addressing production scheduling and related optimization problems. Windmuller [27] compared PPO, ARA-DiRL, and a specific heuristic, concluding that PPO outperformed the others in production scheduling, achieving lower production costs. Ruan [28] compared A3C with traditional methods, demonstrating A3C's superior performance in terms of efficiency and speed for solving aircraft routing problems. Nahhas [29] compared A3C and PPO in hybrid flow shop (HFS) scheduling, finding A3C more effective. Henderson [30] compared TRPO, PPO, DDPG, and ACKTR, concluding that PPO and TRPO exhibited the highest stability. Schulman [31] conducted a comprehensive comparison of PPO against A2C, ACER, and TRPO, establishing PPO's superior performance across a wide range of continuous control environments. Based on the comparative studies, PPO emerges as the most effective algorithm overall among the four considered. However, certain studies indicate that A3C can outperform PPO in specific scenarios. Consequently, this research employs both PPO and A3C for comparison in the production planning domain.

This paper investigates the effectiveness of Deep Reinforcement Learning (DRL) models in tackling Production capacity planning (PCP) with PDDR constraints, a prevalent challenge in manufacturing optimization. Leveraging the production process of a cement factory as a real-world case study, we train and test two prominent RL algorithms, A3C and PPO, within a custom-designed environment mirroring PCP specifics. Their performance will be compared against a human-crafted PCP heuristic and a Mixed Integer Linear Programming (MILP) model, serving as the optimal solution benchmark. Notably, PDDR's dynamic electricity costs lead to intermittent production with discrete batch sizes, requiring the plant to halt production to avoid periods of high tariffs. We propose a novel memory term for the MILP model's changeover constraint, which enables the accurate exploration of optimal production schedules under fluctuating electricity costs. This research promises to uncover the potential of DRL in navigating PCP problems, paving the way for DRL to be further developed to meet the challenges of increasingly complex and diverse PCP systems in the future. The contribution of this paper has three folds:

1. We pioneered the use of deep reinforcement learning in our new environment design, which is specialized for discrete-time medium-term batch production capacity and inventory planning. Crucially, we integrate price-driven demand response for electricity costs, mirroring real-world production complexities.
2. Our research has practical implications. We have developed a novel discrete-time changeover constraint equations with memory terms that enables the MILP model to handle multi-period production stoppages, thereby avoiding high electricity charges. This practical solution can be applied in real-world scenarios.
3. We find that PPO is a promising approach for production capacity and inventory planning. It achieves superior performance to human experts in terms of time and profits. While PPO produces a production plan with a slightly lower profit than MILP, it is significantly faster.

Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) 17-30

2. Problem Statement

In this article, we model a case study of the cement production process. The process begins with the raw materials used. The mixture formula is based on the ratio of gypsum, limestone, and other ingredients. We assume that raw materials are always available and continuously supplied to two parallel cement grinding machines: M1 and M2. Each machine can produce three types of products. In this factory, employees use a spreadsheet program to implement a computational planning method. The method uses an experiential heuristic approach to determine which machines, M1 and M2, should be used to produce each of the three products, P1, P2, and P3, in each period. We elicited the employees' knowledge of the heuristic algorithm in Section 4. However, this manual planning approach is inefficient and inflexible in situations where product demand data fluctuates, making it difficult to optimize costs and resulting in frequent re-planning. The formulated DRL models are a promising solution to the problems mentioned above. They can be used as a prototype model for industrial plants to improve production planning efficiency by leveraging the characteristics of the same production system.

3. Formulation of MILP

To provide an upper benchmark for DRL models, we formulate a novel MILP model to account for time-varying electricity prices in a discrete-time multi-parallel machine production setting with PDDR electricity prices. This model is consistent with the special electricity price conditions we added to the model. The constraints and objective function were derived from a combination of the research of Castro, Dogan, and Zhao[6, 7, 32].

3.1 Inventory Balance Constraint

The quantity of finished goods r remaining at the end of period t, denoted by $R_{r,t}$, can be computed by Eq. (1).

$$R_{r,t} = R_{r,t-1} + \sum_{m=1}^{M} P_{m,r,t} - d_{r,t} - L_{r,t} \quad \forall r \in R, t \in T \tag{1}$$

Where; $R_{r,t-1}$ is the number of inventories carried forward from the previous period, $P_{m,r,t}$ is the number of finished goods received by machine production m, $d_{r,t}$ is the product demand to be delivered to the customer during period t, and $L_{r,t}$ is the shortage quantity to be delivered to the customer at that time.

3.2 Production Capacity Constraints

Let $P_{r,m,t}$ denote the capacity of the machine m to produce product r for each period t. Each period t has a different T_t hour duration between the on-peak and the off-peak periods. As production capacity constraints, Eq. (2) specifies the limitation on $P_{r,m,t}$ as follows.

$$P_{r,m,t} = T_t p_{r,m} N_{r,m,t} \quad \forall r \in R, m \in M, t \in T \tag{2}$$

$$s.t. \ \Sigma_{r \in R} N_{r,m,t} \leq 1 \tag{3}$$

Where, $p_{r,m}$ is the production rate of product r (tons per hour) of the machine m, $N_{r,m,t}$ is a binary variable indicating whether the machine m produces the product r or not at period t. Eq. (3) determines the constraints for each machine to produce no more than one type of product per period.

3.3 Level of Inventory Constraints

Eq. (4) specifies the product minimum and maximum inventory level in its storage tank.

$$R_r^{\min} \leq R_{r,t} \leq R_r^{\max} \tag{4}$$

where R_r^{max} is the cement storage tank capacity, while R_r^{min} is the minimum possible inventory level, which is zero since lost sales are allowed.

3.4 Switch On-Off Constraints

For each machine m, the decision variable for turning the machine on or off, denoted by $SW_{r,m,t}$, can be computed by Eq. (5 and 6).

$$SW_{r,m,t} \geq \sum_{r \in R} N_{r,m,t} \quad \text{at period } t = 1 \tag{5}$$

$$SW_{r,m,t} \geq \sum_{r \in R} N_{r,m,t} - \sum_{r \in R} N_{r,m,t-1} \quad \text{otherwise} \tag{6}$$

3.5 Production Changeover Constraints

Modifications to production processes or replacement of manufactured products often involve costly activities. We modify the backtracking mechanism with the memory to trace the history of production.

$$C^{(k)}_{r',m,t+k} \geq N_{r,m,t} + N_{r',m,t+k} - \mu_m(t+1, t+k) - 1 \quad \text{if } 1 \leq t \leq k \tag{7}$$

$$C^{(k)}_{r',m,t} \geq N_{r,m,t-k} + N_{r',m,t} - \mu_m(t-k+1, t) - 1 \quad \text{if } t > k \tag{8}$$

Given a window size K, where $2 \leq k \leq K$, for each possible k-step memory. Incorporating the changeover memory, we can define the changeover variable from any product r to product r' at the period $t + k$ for all $2 \leq k \leq K$, where $r \neq r'$. In our scenario, we assume that the changeover period will last for at most three periods, setting $K = 3$.

3.6 Objective Equation

The objective function seeks to maximize the total profit, considering the production capacity and inventory planning environment E. By letting $G = [N_{r,m,t} | \forall r \in R, m \in M, t \in T]$ be a production planning, we define the total profit as the gap between the total revenue and the incurred total cost, as shown in Eq. (9).

$$\Omega(G|E) = \left(\sum_{r \in R} \sum_{t=1}^{T} p_r (d_{r,t} - L_{r,t})\right) - \Psi(G|E) \tag{9}$$

Where, p_r is the price of product r [USD/ton], $d_{r,t}$ is the demand of product r at period t [ton], $L_{r,t}$ is the lost sale of product r due to stockout [ton], given in Eq. (1). $\Psi(G|E)$ is the cost function for a production planning [USD]. The total cost for a production planning G with respect to the environment E is further defined as follows.

$$\begin{aligned}\Psi(G|E) = &\sum_{r \in R} \sum_{m=1}^{M} \sum_{t=1}^{T} (cu_r P_{r,m,t} + ce_t pw_m P_{r,m,t} + f_m N_{r,m,t} + sc_m SW_{m,t}) + \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{r \in R} co_{r'} \widehat{C}_{r',m,t} \\ &+ \sum_{r \in R} \sum_{t=1}^{T} h_r R_{r,t} + \sum_{r \in R} \sum_{t=1}^{T} lsc_r L_{r,t}\end{aligned} \tag{10}$$

Where, cu_r is the unit cost of product r [USD/ton], ce_t is the electricity cost at during time interval t [USD/kWh], pw_m is the amount of electricity used by machines m to produce product [kWh/ton], f_m is the fixed cost incurred when there is a production on machine m [USD], sc_m is the costs incurred from re-setup the machine m for production after it was stopped from the previous period [USD], $co_{r'}$ is the cost incurred by changing production from producing any product to producing r' [USD], h_r is the holding cost of product r per one period [USD/ton], and lsc_r is the penalty cost per ton for shortage of product r [USD/ton].

Note that each parameter used in the cost calculation is derived from actual data collected from the cement production process in Section 2. To find the optimal planning that maximizes the total profit concerning environment E, the objective equation can be written as Eq. (11) subject to the constraints in Eqs. (1, *2*, 3, 4, 5, 6, 7 and 8)

$$G^* = \underset{G}{argmax}\, \Omega(G|E) \tag{11}$$

4. Human-inspired PCP Heuristic Algorithm

In this section, we describe the formulation of a heuristics approach called Expert Heuristic, which is based on the simulation of expert experience. The approach begins with human planners performing monthly tactical planning based on customer product forecasts. The planners use a spreadsheet program to calculate the inventory balance. The quantity to be produced is then filled in based on the re-ordering threshold, which is used to maintain the inventory balance. Each day, planners follow a two-step procedure for each product r and period t.

1. They check the demand forecast $d_{r,t}$ for each product.

2. They then check the inventory level $R_{r,t-1}$ in Eq. (1) for each product. If the inventory level is getting close to or lower than the minimum stock level $R_{r,\min}$ in Eq. (4), then they order each machine m to produce $P_{r,m,t}$ items of that product using Eq. (*2*).

Based on the electricity tariff in the TOU PDDR pattern, the numerical value of the production volume is divided into two periods (shifts): off-peak (10 PM to 9 AM when the electricity cost is low) and on-peak (9:00 AM to 10:00 PM when the electricity cost is high). The planners will first focus on filling out the production plan during the low electricity price period, except Saturday and Sunday, which has a low electricity rate for 24 hours. The production volume in each period is fixed in batch values, depending on the number of hours of the production shift, the machine number, and the production rate of the product of each machine.

5. Reinforcement Learning Models

5.1 Reward Functions

The reward function provides the RL agent with a signal that guides its behavior toward achieving the desired goal [33]. We begin by defining the reward function of our reinforcement learning (RL) model as the total profit function:

$$Q = \Omega(G|E) \tag{12}$$

where Ω is the total profit, and G is a production capacity planning, which is a result of the action trajectory generated from the policy learned from the environment E. This equation contains the total revenue and the total cost, similar to the objective function of the MILP model.

To improve the learning ability of the models, we add extra penalties for inventory shortage and overstock into Eq. (12). However, this can lead to negative rewards, which can hinder training. To address this challenge, we introduce an upper bound U to ensure that the reinforcement of agents' learning is both positive and negative [34]. Eq. (12) will become (13), We use Eq. (13) for agent learning, while Eq. (12) tests the model by comparing it with MILP

$$Q = U - \Psi(G|E) \tag{13}$$

5.2 Asynchronous Advantage Actor-Critic (A3C)

A3C is a technique for running multiple learning agents. In this paper, we used the A3C algorithm, which we adapted the steps and code of from Zai's version [35]. This RL algorithm consists of two mechanisms: actor, which controls the selection of actions of the agent (policy-based method), and critic, which measures whether an agent-selected action yields a good or bad reward (value-based method). A3C employs multiple actor-critic agents, each interacting with its copy of the environment. The actors receive state observations and select actions, while the critics evaluate the value of each state and the advantage of each action. These evaluations update the policy and value functions, guiding the agents towards actions that maximize long-term rewards. The equations elucidating the operation of the actor and critic components in the A3C model are presented below:

Actor: The actor modifies and controls the action selection method with a policy function $\pi_\theta(s, a)$, which is controlled by a set of parameters Θ_π. The policy function can be learned by a deep neural network. Generally, a neural network learns the policy function using the backpropagation algorithm on the loss function J (i.e., prediction error). For reinforcement learning, the gradient of the loss function is defined below.

$$\frac{\partial J_\pi}{\partial \theta_\pi} = \mathrm{E}_{\mathrm{T}\sim\pi_\theta}\left[\sum_{t=1}^{T} \frac{\partial}{\partial \theta} log\pi_\theta(a_t|s_t)\, A(s_t, a_t)\right] \tag{14}$$

The advantage of choosing action at a_t state s_t, denoted by $A(s_t, a_t)$, is defined as follows.

$$A(s_t, a_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{15}$$

The term r_{t+1} is the expected future reward q of taking all possible actions from state s_{t+1}. $0 \le \gamma < 1$ is a discount rate. $V(s_t)$, used as the baseline, can be calculated as Eq. (16). It is the expected discounted reward value that the agent should receive from being in the state s_t and operating under a specific action policy π.

$$V(s_t) = \sum_{t=1}^{T} \gamma^{t-1} r_t \tag{16}$$

Critic: Critic's neural network learns from the critic loss value, calculated from the mean squared error between the discounted rewards obtained by running the agent in the environment and the predicted state values from the trajectory of states processed from the neural network. The critic loss can be calculated as follows.

$$J_v(\theta_v) = \frac{1}{T}\sum_{t=1}^{T}(V(s_t,\theta_v) - V_t^{target})^2 \tag{17}$$

Here, the loss is computed by measuring the difference between $V(s_t, \theta_v)$, the expected reward at state s_t according to the critic, and V_t^{target}, the discounted sum of future rewards of the trajectory from period t:

$$V_t^{\text{target}}(\theta) = \sum_{i=t}^{T}\gamma^{i-1}r(s_i, a_i|\theta) \tag{18}$$

where $r(s_i, a_i|\theta)$ is an expected reward of each period i ranging from t to T given the current parameters θ.

5.3 Proximal Policy Optimization (PPO)

The PPO algorithm was proposed by [31]. It is one of the crucial developments in the policy-based methods that have led to a significant improvement over algorithms such as A2C [36]. In this paper, we used the PPO algorithm, in which we adapted the steps and code of the program from Lapan's version [37]. PPO operates by iteratively updating the policy based on the experiences collected during interactions with the environment. It employs a surrogate objective function that minimizes the policy divergence between the old and new policies, ensuring stable updates and preventing significant changes that could lead to instability. The objective of PPO can be represented by the following equation.

$$J(\theta'|\theta) = \mathrm{E}_{s_t\sim p, a_t\sim\pi_\theta}[I(\theta'|s_t, a_t, \theta)] \tag{19}$$

where I is the improvement ratio of the new parameters θ' given the state s_t, the action a_t, and the previous parameters θ:

$$I(\theta'|s_t, a_t, \theta) = \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}A(s_t, a_t|\pi_\theta) \tag{20}$$

where $A(s_t, a_t|\pi_\theta)$ is the advantage function of the policy π_θ computed by Eq. (15). To limit the size of gradient updates, The PPO method added a term to the equation for clipping the objective. Therefore, the objective function (20) is converted to Eq. (21):

$$J_{\text{CLIP}}(\theta'|\theta) = \mathrm{E}_{s_t\sim p, a_t\sim\pi_\theta}[\min[I(\theta'|s_t, a_t, \theta), g(\varepsilon, A(s_t, a_t|\pi_\theta))]] \tag{21}$$

where g is the clipping function defined below:

$$g(\varepsilon, A(s_t, a_t|\pi_\theta)) = \begin{cases} (1+\varepsilon)A(s_t, a_t|\pi_\theta) & \text{if } A(s_t, a_t|\pi_\theta) \geq 0 \quad (22) \\ (1-\varepsilon)A(s_t, a_t|\pi_\theta) & \text{if } A(s_t, a_t|\pi_\theta) < 0 \quad (23) \end{cases}$$

The objective of the old and the new policy ratio is limited to the range $1-\varepsilon$ to $1+\varepsilon$. Therefore, adjusting ε will affect the learning performance.

6. Deep Reinforcement Learning Formulation

This research involved adapting prototype A3C and PPO models from Alex and Lapan, respectively [35, 37], to address the unique characteristics of the production scheduling problem. The modifications encompassed various aspects of the code, including the integration of the custom environment developed on GitHub with the Python platform. This involved adjustments to agent components and hyperparameters, data generation for model training and experience storage, and the core model code structure for training and processing through the neural network. Moreover, we have designed novel action and state spaces tailored to the production capacity planning problem. These modifications ensured compatibility with the many variables associated with discrete-time production scheduling decisions and the extensive data generated from the interaction between the model's agents and the production environment.

6.1 Definition of Environment, States, and Actions

Environment refers to the formulated system in which the agent can interact. The environment sets rules and constraints on what an agent can and cannot do. The agent will learn to choose the most rewarding action from the environment [36, 38]. Here, A production capacity and inventory planning environment is formulated as the production system, as discussed in Section 2. The entire set of actions the agent will, therefore, select in each state will be 16 discrete actions, as shown in Figure 1.

In constructing a Deep Reinforcement Learning model, two pivotal steps, encoding and decoding, play crucial roles in the bidirectional translation of information between the agent and its environment. Figure 2 shows the interaction pattern between the environment and agent learning. The learning process of DRL involves encoding and decoding, which is translating information between the agent and our PCP environment. The encoding process in both PPO and A3C models transforms raw observation data into a suitable format for the model's neural network. This process involves state representation, where meaningful features are extracted from parameters, and variable values arise from the interaction between the agent and the environment so that agents can prioritize parameters that significantly impact their learning, gain a comprehensive understanding of the current state of the environment, and facilitate faster and more efficient learning. From the extensive set of state observation information parameters, we carefully selected 27 key observational parameters that are essential for learning effective action selection in production capacity planning for each scenario within each state. All parameters were normalized to ensure that all selected features had a similar impact on the model's learning process.

The decoding process entails transforming the policy network's output into actions that the agent can execute within the environment during each state. The policy network generates a probability distribution over potential actions, indicating the likelihood of each action being chosen. Based on this distribution, an action is selected using a sampling strategy. The chosen action is then sent to the environment, causing the agent to perform the corresponding action in the simulated PCP environment.

In the learning process, the agent receives 27 observational state parameters. Given product r ∈ [1, 3], machine m ∈ [1, 2], and the future period k ∈ [1, 2], these variables can be divided into four groups:

- Inventory level of each product: I_r and I_r^k
- Incoming shipment demand for each planning period: d_r and d^k_r
- Machine production information: $N_{m,r}$
- Electricity rates during the planning period: E_{on}

These data are then sent to the agent with the neural network mechanism in the RL model to create and develop a policy that is used to choose one of 16 actions. Each action is to produce one of three product types from the first or second machine. As the action is made, it results in state transition, and the reward is computed in Eq. (1). The reward and observation of transitioned state s_{t+1} obtained from the environment will be sent back to the agent for processing and updating the policy to choose an action that gives better rewards.

In training agent process, the product demand is set to change as random uniform values within the specified intervals in each delivery period. Each episode of the agent's training contains 30 steps or states, representing 30 production planning periods. In each state, the quantity of products produced comes from the action chosen by the agent and is used to calculate the state parameters such as inventory balance and incurred costs. These parameters are calculated similarly to sets of equations in the MILP model.

6.2 Agents Training

The goal of training is to model the agent so that it experiences a dataset of policy values in each state. This dataset will lead to a planning solution that achieves an effective total cost of planning. During the learning process, the agent will explore different actions until the action distribution converges to produce the set of potential actions.

The hyperparameters play a crucial role in determining the learning performance of the agent. The discount factor and learning rate are two of the most important hyperparameters that affect learning [39]. In our experiments, we used a trial-and-error method to adjust the hyperparameters using Hubb's [22] default values. We observed the convergence characteristics of the reward curves until we found the appropriate hyperparameter values for the A3C and PPO models, which are shown in Table 1.

The architecture of a neural network, particularly its depth, is a critical factor that profoundly influences its learning capacity. We designed our policy network (actor) as a multilayer perceptron (MLP) with six hidden layers (820, 820, 656, 524, 420, and 328 nodes). The output layer has 16 nodes or units reflecting the log probability for each action. The value network (critic) has four layers (820, 820, 820, and 656 nodes), and the output layer has one node representing one scalar value specifying the corresponding value of a state. Since the actor and critic networks have more than three layers, we can identify our model as a deep reinforcement model (DRL). [40].

Training PPO: In the PPO model training, we set the learning rate to 5×10^{-6} to allow the agent to learn gradually. The discount factor γ was set to 0.95 to simulate a long enough trajectory of states. The clipping value 0.1 helps the PPO's agent appropriately limit the gradient update size. The training process involved 900 training

time steps, which is equivalent to 300,000 episodes. One training time step consisted of 100 batches, each containing 100 states or 3.33 episodes.

Training A3C: A3C's semi-multi-agent learning makes the number of agents significantly affect the learning performance. For our research, we used six agents, the maximum number the CPU processing system on our computation resource can support. We set the learning rate of the A3C agent to 3×10^{-4}, which allows it to learn faster.

$$\mathbf{a}^{(1)}=\begin{bmatrix}1&0&0\\1&0&0\end{bmatrix}\ \mathbf{a}^{(2)}=\begin{bmatrix}0&1&0\\1&0&0\end{bmatrix}\ \mathbf{a}^{(3)}=\begin{bmatrix}0&0&1\\1&0&0\end{bmatrix}\ \mathbf{a}^{(4)}=\begin{bmatrix}0&0&0\\1&0&0\end{bmatrix}$$
$$\mathbf{a}^{(5)}=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\ \mathbf{a}^{(6)}=\begin{bmatrix}0&1&0\\0&1&0\end{bmatrix}\ \mathbf{a}^{(7)}=\begin{bmatrix}0&0&1\\0&1&0\end{bmatrix}\ \mathbf{a}^{(8)}=\begin{bmatrix}0&0&0\\0&1&0\end{bmatrix}$$
$$\mathbf{a}^{(9)}=\begin{bmatrix}1&0&0\\0&0&1\end{bmatrix}\ \mathbf{a}^{(10)}=\begin{bmatrix}0&1&0\\0&0&1\end{bmatrix}\ \mathbf{a}^{(11)}=\begin{bmatrix}0&0&1\\0&0&1\end{bmatrix}\ \mathbf{a}^{(12)}=\begin{bmatrix}0&0&0\\0&0&1\end{bmatrix}$$
$$\mathbf{a}^{(13)}=\begin{bmatrix}1&0&0\\0&0&0\end{bmatrix}\ \mathbf{a}^{(14)}=\begin{bmatrix}0&1&0\\0&0&0\end{bmatrix}\ \mathbf{a}^{(15)}=\begin{bmatrix}0&0&1\\0&0&0\end{bmatrix}\ \mathbf{a}^{(16)}=\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}$$

Fig. 1 *List of actions. For an action matrix a, each element $a_{m,r}$ represents a flag for machine m to produce the product type r*

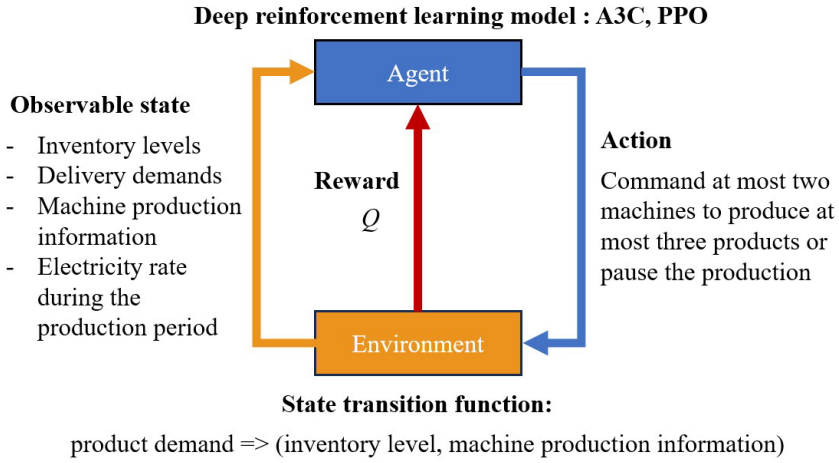


Fig. 2 *Interaction between the learning agents and the environment*

7. Experiment Results

7.1 Training Convergence

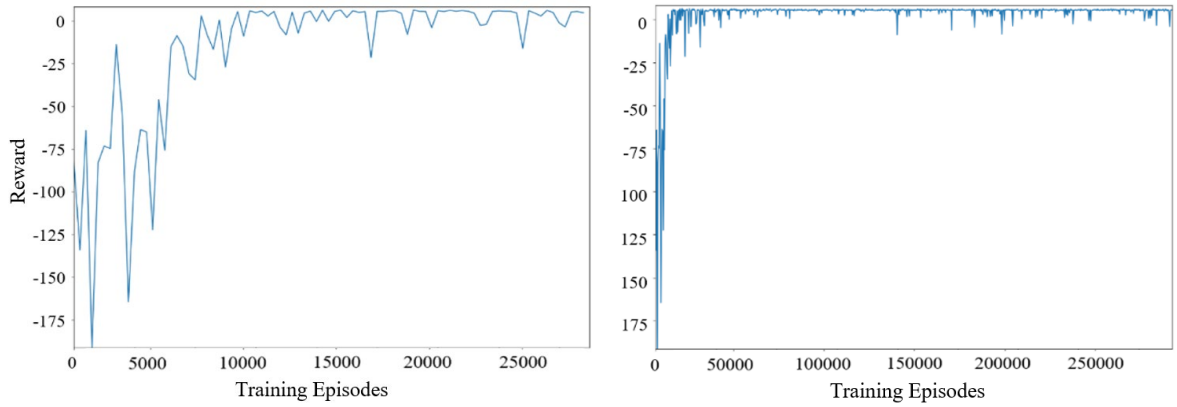
All MILP and DRL models training and testing experiments were performed on a 2.30 GHz Intel i7-11800H CPU with 16 hyperthreads equipped with an NVIDIA RTX-3050 Ti GPU card with 2,560 cores at 1485 MHz and 4GB of VRAM. The MILP model was optimized to the optimal gap of 1% using Pulp 1.6.10 [41]. The proposed DRL models were implemented with PyTorch 1.7.0 [42] on Python 3.10 [43].

Table 1 *Hyperparameter settings*

Types	Hyperparameters	Values	Types	Hyperparameters	Values
Common	Dimension of input vectors	27	PPO	Learning rate	$5\text{x}10^{-6}$
	Actor's hidden layers	6		Number of epochs	30
	Critic's hidden layers	4		Early stopping (time steps)	900
	Units in the first hidden layer	820		Batch size	100
	Entropy regularization	$1\text{x}10^{-4}$		Number of trajectories	2,000
	Dropout probability	0.2		Clipping value	0.1
	Discount factor	0.95	A3C	Number of workers	6
	Upper bound (U)	2,100		Learning rate	$3\text{ x}10^{-4}$

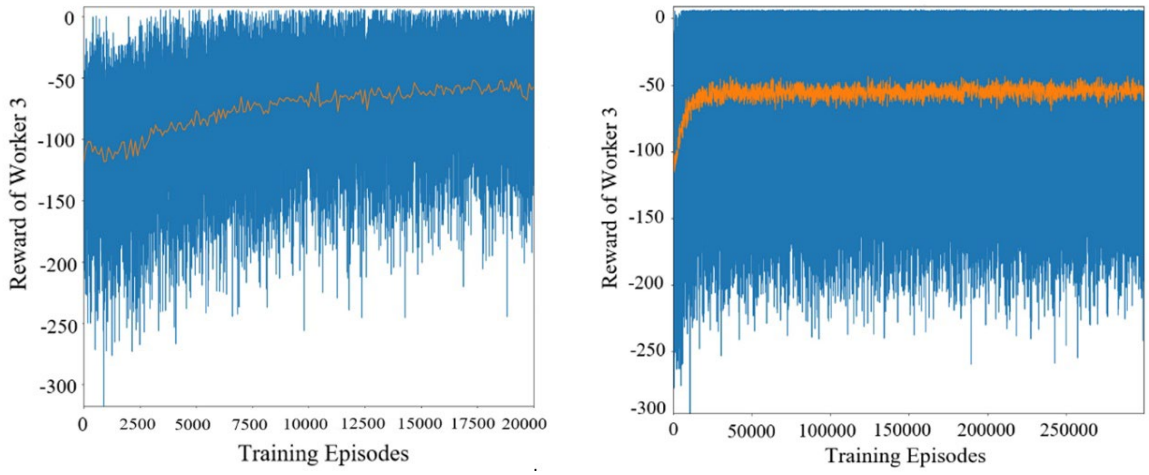
Int. Journal of Integrated Engineering Vol. 17 No. 1 (2025) 17-30

Using the hyperparameters settings in Table 1. Both models are trained by having the agent perform incremental learning in each training round until it yields the highest convergence stable reward. Figure 3 and Figure 4 compare the convergences of the reward of the PPO and A3C models, respectively. In Figure 3a, the PPO model converges at 90 training time steps and yields consistent rewards hereafter. To validate the stability of our model, we trained it for a total of 12 hours, or 900-time steps (300,000 episodes). As shown in Figure 3b, the reward quickly stabilizes at a value of 6.63 million USD, indicating that the model has reached a state of optimal performance. On the other hand, in Figure 4, the A3C model's reward never stabilizes and has a variance ranging from -235.72 to 6.77, with a mean value of -57.85 million USD. The primary objective of plotting the reward curves for both DRL models was to evaluate the training quality by examining convergence speed and reward stability. The PPO model yields a more stable reward graph because it uses a surrogate clipping objective function mechanism. This mechanism enables the PPO to make more aggressive policy updates while still ensuring learning stability. As a result, the PPO agent is better able to accommodate input data with high variation, such as customer demand.



(a) **(b)**

Fig. 3 *Reward convergence of PPO trained for the first 30k (a); and 300k episodes (b)*



(a) **(b)**

Fig. 4 *Reward convergence of A3C trained for the first 20k (a); and 300k episodes (b). The orange line denotes its moving average reward values.*

7.2 Testing Models Efficiency

To evaluate the performance of our four models, we simulated ten demand datasets. The expert heuristic and MILP models were used as baselines, and the two DRL models were compared against them. The MILP model was used as an upper benchmark to compare the reward results.

Demand datasets for testing the models: To simulate the uncertainty of customer demand, we prepared ten datasets with different demand patterns. Each dataset was generated by randomly sampling from the minimum and maximum demand values for each product, which were derived from historical data.

Optimality gap: The reward values obtained for each dataset between the four types of models are shown in Table 2. "%Gap" denotes the corresponding optimality gap with respect to the MILP reward value. The smaller the optimality gap is, the better. The MILP model can find the best answer for production planning from all ten demand data sets, representing an average total production profit of 6.28 million USD. The mean reward values obtained by the PPO and A3C models were 6.02 and 5.11 million USD, respectively, representing an optimality gap of 4.03% and 18.62%. The PPO model offers a promising alternative with a relatively small optimality gap

Computing time: Table 3 compares the efficiency of the four models in terms of the time taken to find the solution. The PPO model is the clear winner, with an average prediction time of 69.06 seconds per round. This is significantly faster than the MILP model, which takes an average of 178,000 seconds per round (2,502 times slower). However, it is important to note that the PPO model requires 43,740 seconds of training before it can be used to find a solution.

A comparison of Table 2 and Table 3 reveals that the PPO model outperforms the A3C and expert heuristic models in terms of both reward and efficiency. PPO's reward results are slightly lower than MILP, but PPO achieves this with a significantly faster prediction time. However, it is important to note that both DRL models require significant training time in the appropriate environment. before they can be used in planning.

Avoidance of on-peak production: Figure 5 shows examples of the production planning and inventory level graph for the demand dataset 1. The green column represents off-peak periods, and the yellow represents on-peak periods.

In Figure 5a, the production planning of the MILP model illustrates the optimizer plan to reduce electricity costs by ordering production during all off-peak periods, while production will be omitted during several on-peak periods. MILP also seeks to reduce changeover costs by allowing the machine to produce the same product repeatedly. In Figure 5c, the PPO model agent can plan production well without any periods of shortage or excessive storage. Imitating the MILP model, the agent was able to plan the production to avoid the cost of electricity quite well in the first machine. However, the agent's ability to avoid electricity costs is not as good as the MILP model because it missed ordering production in some off-peak periods. On the other hand, Figure 5d shows that A3C gives less efficient results. Both machines performed unnecessary production during several on-peak periods, resulting in high production electricity and inventory costs.

Table 2 *Reward values obtained from MILP, Heuristics, PPO, and A3C models*

Data sets	MILP	Heuristics		PPO		A3C	
	Reward	Reward	%Gap	Reward	%Gap	Reward	%Gap
1	6.35	5.82	8.35	5.99	***5.67**	5.19	18.27
2	6.24	5.92	***5.13**	5.91	5.29	4.75	23.88
3	6.07	5.68	6.43	5.92	***2.47**	5.09	16.14
4	6.24	5.93	4.97	6.02	***3.53**	5.76	7.69
5	6.56	6.19	5.64	6.33	***3.51**	5.4	17.68
6	6.44	5.95	7.61	6.15	***4.50**	5.24	18.63
7	6.39	5.77	9.7	6.11	***4.38**	5.33	16.59
8	6.12	5.71	6.7	5.91	***3.43**	4.61	24.67
9	6.33	5.71	9.79	6.05	***4.42**	4.88	22.91
10	6.03	5.49	8.96	5.85	***2.99**	4.83	19.90
Average	6.28	5.82	7.33	6.02	***4.03**	5.11	18.62

* *The asterisks mark the winner in each aspect compared to MILP.*

Table 3 *Model performance in terms of training time and prediction time*

	MILP	Heuristics	PPO	A3C
Training time (s)	-	-	***43,740**	50,400
Prediction time (s)	172,800	194.38	***69.06**	135.9

Avoidance of stockouts and overstock: Shown in parallel on the right side of Figure 5 is the inventory levels graph corresponding to the production planning obtained from each model, the PPO model resulting in a well-balanced inventory level. On the other hand, the A3C model yields an unnecessary inventory increase at the end of the production plan.

The primary reason for PPO's superior performance in production planning compared to A3C lies in its Surrogate Clipping Objective Function mechanism. This mechanism effectively constrains the magnitude of error

values, preventing policy parameter updates from incurring excessively large gradient values. This mechanism enables agents to effectively update policies and formulate production plans that respond to fluctuations in product demand more accurately and stably than A3C.

Despite its demonstrated effectiveness in solving various research problems, A3C may still have certain limitations that could hinder its performance in production planning. One of the primary limitations is that A3C sometimes suffers from learning instability, leading to inconsistent agent behavior and suboptimal outcomes. For instance, A3C may make production decisions that result in overstocking or stockouts during specific periods of the production plan.

7.3 Sensitivity Analysis

To evaluate the sensitivity of the PPO and A3C models to changes in demand data set characteristics, we reformulated a new product demand dataset, 11 to 20, formatted as a monthly seasonality pattern with different seasonal indices [44]. Table 4 shows the results of a sensitivity analysis of the PPO and A3C models, compared to MILP and Expert Heuristic. We gradually increased the seasonality index to observe the effects on optimality gaps. The table shows that when the maximum index was in the low range (1.29 to 1.40), the PPO model had a relatively moderate optimality gap, ranging from 14.42 to 27.61. However, when the seasonal index increased to the high range (1.43 to 1.60), the optimality gap value increased significantly, ranging from 94.02 to 177.15. This suggests that the heuristic, which is manual planning, can better cope with very high variance. The average optimality gap of PPO over the entire test sets is 91.02. In contrast, the A3C model had the worst optimality gap of all the models, with a range of 76.79% to 276.09% and an average of 164.04%. These results suggest that both DRL models are highly sensitive to changes in demand data patterns. Therefore, if demand data patterns change, the models should be retrained to ensure accurate production planning.

8. Conclusions

Reinforcement learning (RL) is a powerful machine learning technique that is becoming increasingly important in various fields, including production planning. This paper tackles discrete-time production capacity planning under fluctuating demand and dynamic electricity tariffs through a novel deep reinforcement learning (DRL) approach. We design and evaluate two DRL models (Proximal Policy Optimization and Asynchronous Advantage Actor-Critic) alongside a heuristic, benchmarked against optimal solutions from Discrete-Time Mixed-Integer Linear Programming (MILP) models, also developed in this study. The MILP model is further enhanced with novel changeover constraints, mitigating production halts during high-cost electricity periods. To train and test the DRL models, we formulated a production capacity planning environment with massive, simulated product demand data sets exhibiting random variation patterns. The PPO model was found to be more efficient than the A3C model and expert heuristic planning in balancing stock levels, avoiding stock-outs, and reducing production during high-cost electricity periods. This demonstrates the potential of PPO to achieve better average inventory and profit in a simulated environment. The PPO model yields slightly less reward than the MILP model but is 2,502 times faster to solve. This trade-off makes the PPO model a more practical choice for production planning in real-world applications, where re-planning is often required due to fluctuating system parameters. However, sensitivity analysis revealed that PPO and A3C models struggle with high seasonality due to unseen demand patterns. Our future work aims to enhance DRL-based production planning by developing a multi-agent PPO model to handle diverse demand patterns through individual agent learning and collaboration. Additionally, we will integrate multi-objective reinforcement learning to facilitate efficient trade-offs between conflicting objectives, like inventory holding and production changeover costs. These advancements promise to significantly improve the adaptability and effectiveness of DRL in real-world production settings.

Table 4 *Sensitivity analysis result of PPO, A3C, and expert heuristics*

Data sets	Max Season Index	MILP	Heuristics			PPO			A3C		
		Profit	Profit	%Gap	%ASO	Profit	%Gap	%ASO	Profit	%Gap	%ASO
11	1.29	7.28	5.67	*22.12	3.33	5.27	27.61	4.44	1.69	76.79	12.64
12	1.32	7.29	5.63	*22.77	4.44	6.22	*14.68	4.44	0.18	97.53	14.94
13	1.36	7.28	4.34	*40.38	4.44	6.23	*14.42	6.67	0.35	95.19	8.05
14	1.4	7.28	4.4	*39.56	5.56	5.55	*23.76	6.67	-2.65	136.4	10.34
15	1.43	7.21	4.3	40.36	3.33	0.06	99.17	12.22	-1.61	122.33	6.9
16	1.47	7.19	1.4	80.53	6.67	0.43	94.02	8.89	-3.2	144.51	10.34
17	1.5	7.16	-2.03	128.35	11.11	-1.94	127.09	8.89	-6.26	187.43	13.79
18	1.53	7.17	-1.08	115.06	8.89	-4.46	162.2	13.33	-10.52	246.72	5.75

19	1.57	7.22	-4.83	166.9	11.11	-5.57	177.15	13.33	-11.69	261.91	13.79
20	1.6	7.11	-6.27	188.33	10	-5.31	174.68	13.33	-12.52	276.09	13.79
Average		7.22	1.15	84.04	6.89	0.65	91.02	9.22	-4.62	164.04	11.03

**The asterisks denote the best optimality gap compared to MILP profit.*
%ASTO stands for average stock-out. Note that there is no stock-out for the MILP model (%ASTO = 0).

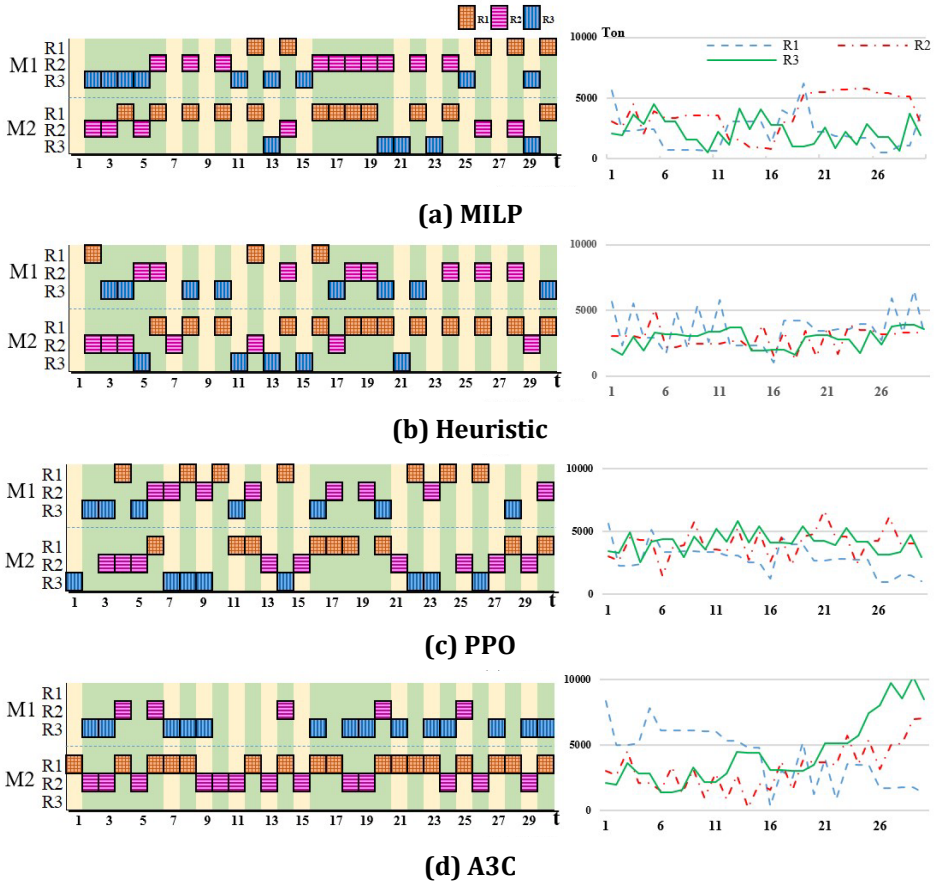


Fig. 5 *Example production scheduling results from four models and corresponding inventory levels*

Acknowledgements

The authors are grateful to King Mongkut's University of Technology North Bangkok for providing the research equipment used in this study. We would also like to extend our thanks to Siam University for their generous doctoral scholarship. Special thanks go to Prachaya Boonkuan for his invaluable guidance and support in troubleshooting the programming bugs encountered during this research.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception, experiment data collection and draft writing****: Thachanon Danket;* ***design research methodology for reinforcement learning****: Sansiri Tanachutiwat;* ***design research methodology for linear programming and production planning****: Vichai Rungreunganun. All authors reviewed results and approved the final version of the manuscript.*

References

[1] F. Kulmer, M. Wolf, and C. Ramsauer, "Medium-term Capacity Management through Reinforcement Learning – Literature review and concept for an industrial pilot-application," *Procedia CIRP,* vol. 107, pp. 1065-1070, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.procir.2022.05.109.

[2] A. Esteso, D. Peidro, J. Mula, and M. Díaz-Madroñero, "Reinforcement learning applied to production planning and control," *International Journal of Production Research,* vol. 61, no. 16, pp. 5772-5789, 2023/08/18 2023, doi: 10.1080/00207543.2022.2104180.
[3] J. A. Swanepoel, E. H. Mathews, J. Vosloo, and L. Liebenberg, "Integrated energy optimisation for the cement industry: A case study perspective," *Energy Conversion and Management,* vol. 78, pp. 765–775, 2014, doi: 10.1016/j.enconman.2013.11.033.
[4] X. Yan, Y. Ozturk, Z. Hu, and Y. Song, "A review on price-driven residential demand response," *Renewable and Sustainable Energy Reviews,* vol. 96, pp. 411–419, 2018, doi: 10.1016/j.rser.2018.08.003.
[5] *V. D. Cosmo*, *S. Lyons*, and *A. Nol*an, "Estimating the Impact of Time-of-Use Pricing on Irish Electricity Demand," *The Energy Journal : IAEE,* vol. 35, 2014, doi: 10.1016/j.eneco.2022.106023.
[6] P. M. Castro, I. Harjunkoski, and I. E. Grossmann, "New Continuous-Time Scheduling Formulation for Continuous Plants under Variable Electricity Cost," *Industrial & Engineering Chemistry Research,* vol. 48, pp. 6701–6714, 2009, doi: https://doi.org/10.1021/ie900073k. ACS Publications.
[7] M. Erdirik-Dogan and I. E. Grossmann, "Simultaneous planning and scheduling of single-stage multi-product continuous plants with parallel lines," *Computers and Chemical Engineering,* vol. 32, pp. 2664–2683, 2008, doi: 10.1016/j.compchemeng.2007.07.010.
[8] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick, "Discrete Time Formulation for the Integration of Scheduling and Dynamic Optimization," *Industrial and Engineering Chemistry Research,* vol. 54, no. 16, pp. 4303–4315, 2015, doi: 10.1021/ie502960p.
[9] A. Obermeier, C. Windmeier, E. Esche, and J.-U. Repke, "A discrete-time scheduling model for power-intensive processes taking fatigue of equipment into consideration," *Chemical Engineering Science,* 2018, doi: 10.1016/j.ces.2018.10.036.
[10] C. A. Floudas and X. Lin, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review," *Computers and Chemical Engineering,* vol. 28, pp. 2109–2129, 2004, doi: 10.1016/j.compchemeng.2004.05.002.
[11] I. Bach, G. Bocewicz, and Z. Banaszak, "Constraint Programming Approach to Multi-product Scheduling," vol. 3, ed: Applied Computer Science, 2007.
[12] M. Larios-gómez, P. M. Quintero-flores, M. Anzures-garcía, and M. Camacho-hernandez, "Application of real-time fan scheduling in exploration-exploitation to optimize minimum function objectives," *Applied Computer Science,* no. *19*(2), pp. 43-54, 2023, doi: https://doi.org/10.35784/acs-2023-13.
[13] H. Mokhtari, I. N. K. Abadi, and S. H. Zegordi, "Production capacity planning and scheduling in a no-wait environment with controllable processing times: An integrated modeling approach," *Expert Systems with Applications,* vol. 38, no. 10, pp. 12630-12642, 2011/09/15/ 2011, doi: https://doi.org/10.1016/j.eswa.2011.04.051.
[14] Y. Ma, F. Qiao, and J. Lu, "Learning-based dynamic scheduling of semiconductor manufacturing system.," presented at the IEEE International Conference on Automation Science and Engineering (CASE 2016), 2016. [Online]. Available: https://doi.org/10.1109/COASE.2016.7743572.
[15] H. Seidgar, M. Zandieh, and I. Mahdavi, "Bi-objective optimization for integrating production and preventive maintenance scheduling in two-stage assembly flow shop problem," *Journal of Industrial and Production Engineering,* 2016, doi: 10.1080/21681015.2016.1173599.
[16] X. Yao, N. Almatooq, R. G. Askin, and G. Gruber, "Capacity planning and production scheduling integration: improving operational efficiency via detailed modelling," *International Journal of Production Research,* vol. 60, no. 24, pp. 7239-7261, 2022/12/17 2022, doi: 10.1080/00207543.2022.2028031.
[17] L. Lin and M. Gen, "Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications," *International Journal of Production Research,* 2018, doi: 10.1080/00207543.2018.1437288.
[18] S. L. Takeda-Berger, E. M. Frazzon, E. Broda, and a. M. Freitag, "Machine Learning in Production Scheduling: An Overview of the Academic Literature," in *Dynamics in Logistics Proceedings of the 7th International Conference LDIC 2020*, Bremen, Germany, 2020: Springer Nature Switzerland AG, p. 409, doi: 10.1080/00207543.2024.2381145. [Online]. Available: https://doi.org/10.1080/00207543.2024.2381145
[19] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt, and S. Wrobel, "A review of machine learning for the optimization of production processes," *The International Journal of Advanced Manufacturing Technology,* vol. 104, no. 5, pp. 1889-1902, 2019/10/01 2019, doi: 10.1007/s00170-019-03988-5.
[20] D. Kalita. "An Overview and Applications of Artificial Neural Networks." Data Science Blogathon. https://www.analyticsvidhya.com (accessed.
[21] Max Mowbray, Dongda Zhang, and E. A. D. R. Chanona, "Distributional Reinforcement Learning for Scheduling of Chemical Production Processes," *arXiv:2203.00636,* 2022, doi: 10.48550/arXiv.2203.00636.
[22] C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "A Deep Reinforcement Learning Approach for Chemical Production Scheduling," *An International Journal of Computers Applications in Chemical Engineering,* 2020, doi: 10.1016/j.compchemeng.2020.106982.

[23] H. Rummukainen and J. K. Nurminen, "Practical Reinforcement Learning Experiences in Lot Scheduling Application," presented at the International Federation of Automatic Control (IFAC) Conference 2019, 2019. [Online]. Available: https://doi.org/10.1016/j.ifacol.2019.11.397.
[24] T. Quah, D. Machalek, and K. M. Powell, "Comparing Reinforcement Learning Methods for Real-Time Optimization of a Chemical Process," *MDPI,* vol. 8, p. 1497, 2020, doi: doi:10.3390/pr8111497.
[25] S. Windmuller, "Smart capacity planning: A machine learning approach," Master of Science, Department of Information Systems, Production and Logistics Management, University of Innsbruck, Innsbruck , Austria, 2020, https://www.scribd.com/document/775594713/Smart-capacity-planning.
[26] M. Panzer, B. Bender, and N. Gronau, "Deep Reinforcement Learning In Production Planning And Control: A Systematic Literature Review," presented at the 2nd Conference on Production Systems and Logistics (CPSL 2021), 2021, https://api.semanticscholar.org/CorpusID:238929996.
[27] S. Windmuller, "Smart capacity planning: A machine learning approach," Master of Science, Department of Information Systems, Production and Logistics Management, University of Innsbruck, Innsbruck , Austria, 2020.
[28] J. H. Ruan, Z. X. Wang, F. T. S. Chan, S. Patnaik, and M. K. Tiwari, "A reinforcement learning-based algorithm for the aircraft maintenance routing problem," *Expert Systems with Applications,* p. 148, 2020, doi: 10.1016/j.eswa.2020.114399.
[29] A. Nahhas, A. Kharitonov, and K. Turowski, "Deep Reinforcement Learning Techniques For Solving Hybrid Flow Shop Scheduling Problems: Proximal Policy Optimization (PPO) and Asynchronous Advantage Actor-Critic (A3C)," 2022. [Online]. Available: http://hdl.handle.net/10125/79538.
[30] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18),* vol. 392, pp. 3207-3214, 2018, doi: 10.48550/arXiv.1709.06560.
[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347,* 2017, doi: 10.48550/arXiv.1707.06347.
[32] S. Zhao, I. E. Grossmann, and L. Tang, "Integrated scheduling of rolling sector in steel production with consideration of energy consumption under time-of-use electricity prices," *Computers and Chemical Engineering,* vol. 111, pp. 55–65, 2018, doi: 10.1016/j.compchemeng.2017.12.018.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press, 2018.
[34] N. Milošević, "Negative Deep Learning," Doctor of Philosophy, Faculty of Sciences, Department of Mathematics and Informatics, University of Novi Sad, Serbia, 2021.
[35] B. B. Alexander Zai, *Deep Reinforcement Learning in Action*. Simon and Schuster, 2020.
[36] E. Bilgin, *Mastering Reinforcement Learning with Python*. Packt Publishing Ltd., 2020.
[37] M. Lapan, *Deep Reinforcement Learning Hands-On - Second Edition*. Packt, 2020, p. 606.
[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Illustrated edition ed. The MIT Press, 2016.
[39] M. Andrychowicz *et al.*, "What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study," *arXiv:2006.05990v1,* 2020.
[40] A. B. W. Putra, R. Malani, B. Suprapty, and A. F. O. Gaffar, "A Deep Auto Encoder Semi Convolution Neural Network for Yearly Rainfall Prediction," in *2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 22-23 July 2020 2020, pp. 205-210, doi: 10.1109/ISITIA49792.2020.9163775.
[41] S. Mitchell, M. OSullivan, and I. Dunning, "PuLP: a linear programming toolkit for python," *The University of Auckland, Auckland, New Zealand,* vol. 65, 2011, https://api.semanticscholar.org/CorpusID:14277904.
[42] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems,* vol. 32, 2019, doi: https://doi.org/10.48550/arXiv.1912.01703.
[43] G. V. Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009, https://dl.acm.org/doi/book/10.5555/1593511.
[44] R. F. Jacobs, W. L. Berry, C. Whybark, and T. E. Vollmann, *Manufacturing Planning and Control for Supply Chain Management: The CPIM Reference*, 2nd Edition ed. New York: McGraw-Hill Education (in en), 2018.