

Article

Task Allocation in Human–Machine Manufacturing Systems Using Deep Reinforcement Learning

Taejong Joo ¹, Hyunyoung Jun ²  and Dongmin Shin ^{2,*} 

¹ Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208, USA; tjoo@u.northwestern.edu

² Industrial and Management Engineering Department, Hanyang University, Ansan 15588, Korea; hy1911@hanyang.ac.kr

* Correspondence: dmshin@hanyang.ac.kr

Abstract: Catering for human operators is a critical aspect in the sustainability of a manufacturing sector. This paper presents a task allocation problem in human–machine manufacturing systems. A key aspect of this problem is to carefully consider the characteristics of human operators having different task preferences and capabilities. However, the characteristics of human operators are usually implicit, which makes the mathematical formulation of the problem difficult. In addition, variability in manufacturing systems such as job completion and machine breakdowns are prevalent. To address these challenges, this paper proposes a deep reinforcement learning-based approach to accommodate the unobservable characteristics of human operators and the stochastic nature of manufacturing systems. Historical data accumulated in the process of job assignment are exploited to allocate tasks to either humans or machines. We demonstrate that the proposed model accommodates task competence and fatigue levels of individual human operators into job assignments, thereby improving scheduling-related performance measures compared to classical dispatching rules.

Keywords: deep learning; dynamic task allocation; human factors; intelligent manufacturing systems; manufacturing scheduling; reinforcement learning



Citation: Joo, T.; Jun, H.; Shin, D.

Task Allocation in Human–Machine Manufacturing Systems Using Deep Reinforcement Learning.

Sustainability **2022**, *14*, 2245. <https://doi.org/10.3390/su14042245>

Academic Editor: Paolo Renna

Received: 11 January 2022

Accepted: 9 February 2022

Published: 16 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Although automation has been adopted for repetitive physical tasks in manufacturing systems, human operators are indispensable components, even in advanced manufacturing systems. For instance, in automotive door assembly tasks, an operator holds and guides the door using an assistive material handling apparatus or equipment to attach it to the main body of the vehicle. These types of hybrid systems can take advantages of both humans and machines. For physical manufacturing activities, humans offer the inimitable senso-motoric ability for complex tasks, and machines can achieve higher speed, constant precision, and force [1,2].

This paper presents a task allocation problem in human–machine manufacturing systems. It aims to find desirable job allocation to manufacturing resources in such a way that tasks are assigned to human operators with consideration of their competence and preference while guaranteeing reasonably good operational performance measures such as makespan, flowtime, and/or tardiness.

Over the decades, optimization-based techniques have long been a mainstream for addressing scheduling problems. In doing so, mathematical objective functions and constraints are first defined, and then the optimization algorithm is employed to find solutions of the formulated problem. This optimization-based approach has assumed availability of explicit formulations concerning objective functions and constraints of the target systems.

Allocating manufacturing tasks to human operators requires careful considerations for their characteristics and properties. While machines usually perform specific tasks such as machining process or assembly, the human operators can perform various tasks depending

on their specialization and experiences [2]. This results in more room for alternatives in assigning manufacturing tasks to human operators than to dedicated machines. More importantly, human performances tend to significantly deteriorate when fatigue exceeds thresholds. Therefore, task allocation involving human operators should carefully consider human competence levels influenced by working duration and types of tasks that can cause fatigue [3–5].

Another important characteristic of manufacturing systems is their variabilities, which lead to complex changes in functional states of jobs and resources. Due to the variabilities induced by the stochastic nature, the event occurrences within the system such as arrivals and completions of tasks cannot easily be determined [6]. Therefore, rigid scheduling constructed before the occurrence of such events may not be efficient and may not even be feasible [7], which incurs the need for constructing dynamic scheduling.

It is a challenging problem to allocate tasks dynamically with consideration of inherent variabilities in human operator performance and manufacturing systems. Although a relationship between fatigue and task performance needs to be modeled, there are few universally agreed models to describe this relationship. Instead, the previous records of task allocation can be useful in inferring latent characteristics of human. For instance, the times taken to complete various types of manufacturing can be used to infer operator fatigue threshold.

This paper addresses task allocation by means of reinforcement learning (RL) and then exploits a neural network to derive the desirable assignment. Specifically, the agent in RL contains a recurrent neural network, which can capture hidden characteristics of human operators and the variabilities of a manufacturing system implicitly contained in previous sequences of observations. This sequential modeling is performed on the semantically meaningful representation space provided by a feedforward neural network. Based on the sequential representation, the agent evaluates performance and improves task allocation. Herein, the question to be investigate in this work is to whether tasks can be allocated to machines and humans while achieving balanced workload to humans as well as assuring manufacturing operational performance. Numerical experiments are conducted with simulated data, and it is shown that the agent successfully captures various hidden characteristics of human operators, which results in significantly increased scheduling performance.

This paper is organized as follows: Section 2 reviews the related work and provides the motivation of the task allocation with consideration of human operators and the stochastic nature of manufacturing systems; we also present related works on data-driven scheduling. In Section 3, we formulate task allocation as a deep RL problem. Section 4 verifies the effectiveness of the proposed approach and interpretations of the results, which is followed by a conclusion in Section 5.

2. Related Work

The basic forms of scheduling and task allocation deal with machines and jobs, given job arrivals and processing times of tasks. Various approaches have been proposed and have contributed to resolve task allocation problems. However, there remains an important aspect to be considered from a practical and sustainable perspective of manufacturing systems. For a more realistic task allocation, it is necessary to consider a critical component, such as a human operator, which should be considered with the variabilities in manufacturing systems at the same time.

Agnetis et al. [2] consider scheduling problems wherein jobs require the collaboration between a human operator and a machine, which can be called the scheduling with human operators. In their work, human operators are characterized as agents having their own unique skill sets, which become constraints to job allocations. Along this approach, several works have proposed the algorithms for obtaining a better solution to task allocation among humans and machines [8,9]. However, they treat human operators as flexible general-purpose “machines” performing multiple types of tasks. In this regard, it is necessary to

consider inherent characteristics of human operators such as performance deterioration due to fatigue accumulation.

Task allocation problems have widely been examined to accommodate the stochastic nature of manufacturing systems. For example, the availability of resources is affected by machine breakdown [10]. The stochastic job processing time makes it difficult to determine when jobs and resources will be available [11]. Since the states of the system are changing over time and are even unpredictable, rigid job assignments would be inefficient in practice.

While a variety of techniques such as genetic algorithm [9,11,12], evolutionary algorithm [10], and memetic algorithm [8] are suggested, data-driven approaches supported by real-time data in manufacturing systems have gained popularity in recent years. Among them, neural networks have received much attention because of their capability of learning representations of high-dimensional data [13].

Neural network (NN)-based supervised learning approaches aim to learn input–output relationships between input data and corresponding targets. Input data could include information about individual jobs such as processing times and due dates, and/or operational information such as the number of jobs and rate of tardy jobs. Based on accumulated records of such data and corresponding task allocation results, it is aimed to predict a desirable priority of jobs waiting in the buffer [14,15]. Summary statistics of simulations such as inter-arrival times of products and batch sizes of raw materials are also used to predict total profit [16]. However, these supervised learning approaches may have limited practical strength because their performances are critically influenced by the quality of historical data concerning task allocations.

Another line of research, NN-based RL approaches, aims to maximize the expected cumulative rewards obtained by the execution of actions [17]. The effectiveness of the RL with NNs has been demonstrated in various applications domain including playing Atari game [18], playing Go [19], and terrain-adaptive dynamic locomotion skills [20].

In the scheduling field, multi-agent learning algorithms with NNS have been proposed for solving the job shop scheduling problems (JSP). Gabel and Riedmiller presented a novel view on the JSP by modeling them as multi-agent RL problems [21]. They additionally investigated interactions among agents, each of which was regarded as a machine [22]. Inspired by such successes, this paper uses the RL approach to solve the JSP in human–machine manufacturing systems, aiming to capture unobservable characteristics of human operators and the variabilities in a manufacturing system.

This work presents a task allocation in human–machine collaborative manufacturing using deep learning (DL)-based RL. Task allocation problems are extensively investigated in the field of air traffic control [23], manufacturing systems [24], and human–robot interactions [25]. Moreover, RL has shown significant benefits in scheduling [26] and task allocation in manufacturing systems [27]. Complexity of tasks in allocating tasks is also considered in [28]. Meanwhile, task allocation between human and machine in consideration of task complexity, human competence levels, and fatigue accumulation has not been investigated. Table 1 shows the research gaps among related works.

Table 1. Contribution table for related work.

Approach	Mfg. Tasks	Task Complexity	Mfg. Performance	Human Operator	Issues			
					Competence Level	Fatigue	DL	RL
Hoc (2000), Mencia (2015)	✓		✓	✓				
Digiesi et al. (2009), Hu and Chen (2017)	✓			✓		✓		
Park et al. (2020), Gabel and Riedmiller (2008, 2012), Oliff et al. (2020)	✓		✓	✓			✓	✓
Malik and Bilberg (2019), Krüger et al. (2009)	✓	✓		✓	✓			✓
Agnetis et al. (2014), Mencia (2016)	✓		✓	✓	✓			
Ferjani (2017)	✓			✓	✓	✓		
This work	✓	✓	✓	✓	✓	✓	✓	✓

3. Proposed Framework

3.1. Task Allocation in Human–Machine Manufacturing Systems

We consider a dynamic scheduling problem in human–machine manufacturing systems as shown in Figure 1. A job consists of several tasks, each of which is subject to specific task precedence relations. Then, solving the JSP corresponds to find the optimal assignment of a task in a job into a manufacturing resource such as machines, robots, and material handlers, which minimizes the flowtime or tardiness. In addition, it considers variabilities in job arrivals, processing times, and unexpected machine breakdowns. Therefore, possible job assignments are dynamically changing depending on the availability of resources and task precedence requirements.

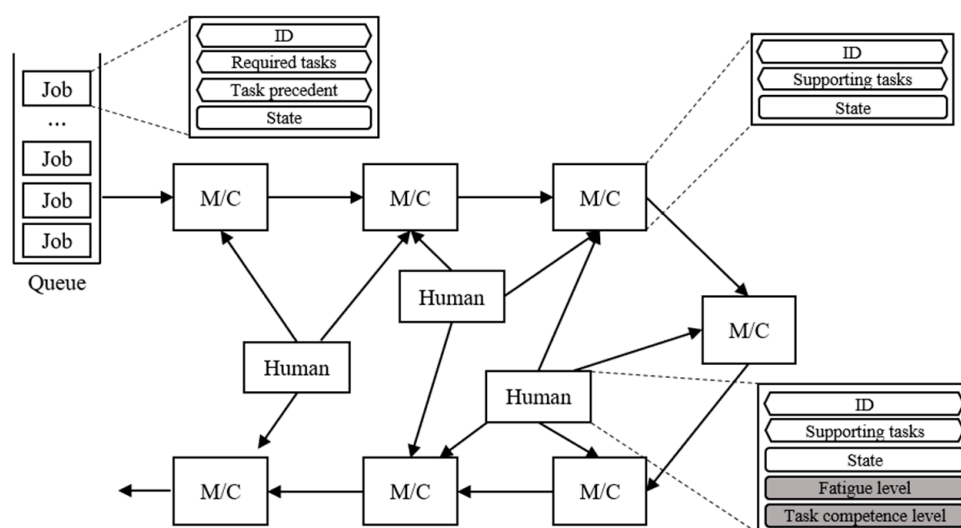


Figure 1. Schematic diagram of a task allocation problem in human–machine manufacturing systems.

In addition to constraints of resource and job availability, special care is taken in assigning jobs to accommodate characteristics of human operators. Human operators can perform a wide range of tasks with different competence levels. This characteristic can give room for flexibility in task allocation. Conversely, their individual competence levels vary depending on task types. Therefore, appropriate tasks need to be allocated to human operators depending on their competence levels to attain desirable scheduling performances.

In scheduling problems, the operational states of manufacturing resources can usually be recognized such as in the form of working, idle, and breakdown. However, human operator states are subtle compared to machines. Although a human is in a working state, it is subject to dynamically changing and unobservable factors such as fatigue and situation awareness. In this work, we consider fatigue accumulation—one of the major unobservable factors affecting the task performances of human operators [3,4,25,29,30]. There is no universally agreed model for describing the relationship between fatigue and task performance. However, the task processing time is known to increase as fatigue accumulates due to the number of tasks or the duration of working time [3,4].

In this work, observable factors refer to the conditions that can be formulated by mathematical representations, such as whether a machine or a human is working or idle and whether task precedence relations are satisfied or violated [31]. Conversely, unobservable conditions may not be recognized explicitly and measured clearly; for instance, it is difficult to determine the competence levels of human operators for various task types or for how much fatigue they are suffering from in manufacturing tasks.

Nevertheless, capturing the unobservable conditions is necessary to achieve the desired performance of the task allocation. For example, assigning a task to a human operator with heavy workload can result in delayed task completion by her or him due to perfor-

mance deterioration. Meanwhile, job allocation regardless of human operator competence would not fully exploit the capability of them.

In this regard, this paper aims to allocate tasks to either machines or human operators while catering for fatigue and competence of human operators, which are unobservable factors affecting scheduling performance.

3.2. Reinforcement Learning Framework for Task Allocation

The proposed framework is based on deep RL, wherein a goal is formulated as the RL problem, and then deep learning techniques are used to improve the goal. RL deals with the behavior of an agent that interacts with an environment, and it aims to maximize the desirability of the agent's actions by means of repeating the observation–action–reward loops [17]. In this subsection, we formulate the task allocation into an RL problem and discuss how to evaluate the desirability of the agent's job assignments.

Consider a human–machine manufacturing system consisting of the n_M machines, n_H human operators, n_T classes of tasks, and n_J types of jobs. Formally, let $\mathcal{M} = \{m_1, \dots, m_{n_M}\}$ and $\mathcal{H} = \{h_1, \dots, h_{n_H}\}$ denote sets of machines and human operators, respectively. $\mathcal{T} = \{t_1, \dots, t_{n_T}\}$ denotes a set of task classes and $\mathcal{J} = \{j_1, \dots, j_{n_J}\}$ denotes a set of job classes. Herein, each job $j_k \in \mathcal{J}$ is a partially ordered set of tasks such that $j_k = (\{t_{k_i}\}, \leq_k)$ where $\{t_{k_i}\} \subseteq \mathcal{T}$ is a set of subtasks contained in j_k and $\leq_k$ is a partial ordering that dictates task precedent relationships.

We also let $Q = \{q_1, \dots, q_{n_Q}\}$ be a set of storages, or queue, with n_Q being a maximum number of jobs subject to task allocation where each $q_i \in \mathcal{J}$ contains a job. In this paper, it is assumed that every task requires both a human operator and a machine, and each machine is dedicated to a specific task class, i.e., a machine m_i is dedicated to a task t_i and $n_M = n_T$.

To formulate the task allocation into the RL problem, we characterize the scheduling into the partially observable Markov decision process (POMDP), which can be defined as a septuple:

$$\langle \mathcal{S}, \mathcal{X}, \mathcal{A}, r, \mathbf{p}^{(I)}, \mathbf{P}^{(T)}, \mathbf{P}^{(O)} \rangle \quad (1)$$

where $\mathcal{S}$ is a (partially observable) state space; $\mathcal{X} \subseteq \mathcal{S}$ is an observation space; $\mathcal{A}$ is an action space; $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a reward function. We denote random variables of a state, an observation, an action, and a reward at time t as s_t , x_t , a_t , and r_t , respectively. $\mathbf{p}^{(I)} = \{p_i^{(I)}\}$ is an initial probability vector where $p_i^{(I)} = p(s_0 = i)$; $\mathbf{P}^{(T)} = \{p_{ijk}^{(T)}\}$ is a transition probability tensor where $p_{ijk}^{(T)} = p(s_{t+1} = k | s_t = i, a_t = j)$ for all t ; $\mathbf{P}^{(O)} = \{p_{ij}^{(O)}\}$ is an observation probability matrix where $p_{ij}^{(O)} = p(x_t = j | s_t = i)$ for all t .

Considering the scheduler as an actor in the RL framework, we define $\mathcal{X}$ and $\mathcal{A}$ and then introduce a reward function. The reward function is a surrogate function of scheduling goals including minimizing flowtime and balancing workload among human operators. Note that we treat the remaining components $\mathcal{S} \setminus \mathcal{X}, \mathbf{p}^{(I)}, \mathbf{P}^{(T)}, \mathbf{P}^{(O)}$ as unknowns which are handled in implicit manner.

Let $\mathcal{X} = \mathcal{X}^{(H)} \times \mathcal{X}^{(M)} \times \mathcal{X}^{(Q)}$ where $\mathcal{X}^{(H)}$, $\mathcal{X}^{(M)}$, and $\mathcal{X}^{(Q)}$ denote observation spaces of human operators, machines, and system queues, respectively. $\mathcal{X}^{(H)}$ is composed of the Cartesian product of identical copies of n_H sets $\mathcal{X}^{(h_i)} = \{\text{idle}\} \cup \mathcal{T}$ indicating h_i is at a state of idle or working on task $t \in \mathcal{T}$. Similarly, $\mathcal{X}^{(M)} = (\mathcal{X}^{(m_1)} \times \dots \times \mathcal{X}^{(m_{n_M})})$ with $\mathcal{X}^{(m_i)} = \{\text{working}, \text{idle}, \text{breakdown}\}$ and $\mathcal{X}^{(Q)} = (\mathcal{X}^{(q_1)} \times \dots \times \mathcal{X}^{(q_{n_Q})})$ with $\mathcal{X}^{(q_i)} = (\emptyset \cup \mathcal{J}) \times (\{\text{idle}\} \cup \mathcal{H}) \times (\{\text{idle}\} \cup \mathcal{T})$.

For example, $(j, h, t) \in \mathcal{X}^{(q_i)}$ represents a situation that a human operator h processes a task t in job j with a machine for the task. Meanwhile, $(j, \text{idle}, \text{idle}) \in \mathcal{X}^{(Q)}$ means tasks in job j have not yet been allocated to both a human and a machine and $(\emptyset, \text{idle}, \text{idle}) \in \mathcal{X}^{(Q)}$ represents that there is no job subject to task allocation.

Action $a = (t, q, h) \in \mathcal{A}$ where $\mathcal{A} = \mathcal{T} \times \mathcal{Q} \times \mathcal{H}$, is considered an assignment of task t of a job in system queue q to human operator h .

The reward function is defined as:

$$r_L = r(\mathbf{x}_1, \mathbf{a}_1, \mathbf{x}_2, \dots, \mathbf{x}_L) = \begin{cases} b^{(u)} + b_L^{(u)} n_L^{(q)} & \text{if } L > t_{\text{terminate}} \\ b^{(c)} + b_L^{(c)} (t_{\text{terminate}} - L) & \text{otherwise} \end{cases} \quad (2)$$

where $\mathbf{x}_i$ is an observation at time step i and $n_L^{(q)}$ is the number of remaining jobs in a queue at time L . $b^{(u)}$ and $b^{(c)}$ are penalty and bonus rewards for the desired situation, respectively. For example, $b_L^{(u)}$ is a penalty for an increased work-in-process, and $b_L^{(c)}$ is the bonus reward for timely completion. It is important to note that it may be impossible to assign a reward for each pair of $(s, a) \in (\mathcal{S}, \mathcal{A})$ due to the unobservability of the state s .

Let a policy of an agent π be a mapping from an observation to an action; then, the expected reward of π is defined as $\mathbb{E}_{\tau_1^\pi} [\sum_{i=1}^{\infty} \gamma^{i-1} r(s_i, a_i)]$ where τ_1^π is a probability distribution of a future trajectory starting from the time step 1 under a policy π and $\gamma \in [0, 1]$ is a discount factor. With the Bellman equation, we define the Q-value function under the policy π , which is the desirability of action $\mathbf{a}_t$ at state $\mathbf{s}_t$ at time step t :

$$Q^*(s_t, a_t) = \max_{\pi} \mathbb{E}_{\tau_1^\pi} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t = s_t, a_t = a_t \right] \\ = \max_{\pi} \mathbb{E}_{s_{t+1}} \left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s_t, a_t = a_t, \pi \right] \quad (3)$$

As noted above, Q-value function is subject to unavailable information: the current state information $\mathbf{s}_t$ due to human operators states and the state-transition probability $P^{(T)}$. One naïve solution to this problem can be to define the Q-function of a sequence of observations $(\mathbf{x}_1, \dots, \mathbf{x}_t)$ instead of the state $\mathbf{s}_t$ and approximating the transition probability with an empirical distribution based on an accumulated history of state-transitions.

However, considering that the number of possible trajectories $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t)$ grows exponentially with the time step, the probability of observing exactly the same trajectories from repetitive episodes would be extremely low, especially for large t . Therefore, this approach would require prohibitively many repetitions of scheduling episodes for reasonably approximating the transition probability.

Instead, we exploit NNs to handle high-dimensional data for approximating Q-value function directly from the accumulated scheduling history data, which is covered in the next sub-section.

3.3. Deep Learning for Optimizing the Scheduling Objective

This subsection presents the architecture of an NN used as a Q-value approximator a strategy for learning parameters of the NN based on the accumulated history of scheduling assignments.

3.3.1. Architecture Design

An NN agent receives an observation from a system $\mathbf{x}_i = (\mathbf{x}_i^{(h)}, \mathbf{x}_i^{(m)}, \mathbf{x}_i^{(q)})$ where $\mathbf{x}_i^{(*)} = \{x_i^{(*)k}\}_{k \in \{1, \dots, n_*\}}$ with $x_i^{(*)j} \in \mathcal{X}^{(*)j}$ for $\star \in \{h, m, q\}$ at each time step i . Based on the observation, the agent produces the approximated Q-value, which can be used to task allocation. Figure 2 illustrates the architecture of an agent, which consists of three main computational building blocks: (1) pre-processing observations, which reduces input dimensionality; (2) modeling sequential observations in the context of the previously accumulated data, which allows for one to implicitly capture unobservable and unknown conditions; (3) evaluating the desirability of task allocations.

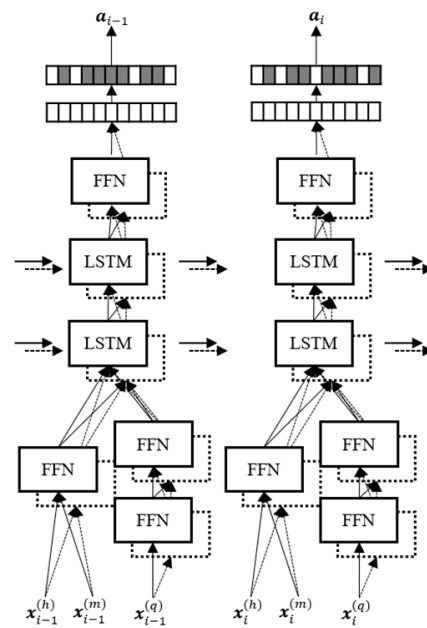


Figure 2. Architecture of the deep reinforcement learning agent for job allocation. FFN block and LSTM block represent feedforward layer and recurrent layer, respectively.

Preprocessing observations: One appealing aspect of using NNs for pre-processing an observation x_i is its ability to learn meaningful representations of observations. This enables the semantically similar observations to have a closer distance in the learned feature space compared to their distance in the raw observation space.

In this work, an observation space $\mathcal{X}$ of a human–machine manufacturing system contains several different observations but semantically similar in terms of task allocations. For instance, the observations $x \in \mathcal{X}$ wherein tasks in the different order are similar if they cause the same changes in resource states, humans or machines because they are not yet to be allocated. NNs can make these different observations have similar semantics in terms of task allocation to have a close distance with each other in the representation space.

The representation space can be constructed by means of multiple hierarchical levels of abstractions and multiple parallel hidden units [13]. Specifically, each layer contains several units each of which computes non-linear mapping of a weighted sum of incoming inputs given by the previous layer, and these computations are performed recursively:

$$y_i^{(l)} = f(W^{(l)T} y_i^{(l-1)} + b^{(l)}) \quad (4)$$

where $y_i^{(l)} \in \mathbb{R}^{d_l}$ represents the input at time step i in l -th layer (typically $d_n \geq d_m$ for $n < m$), $y_i^{(0)}$ corresponds to the input x_i , $(W^{(l)}, b^{(l)})$ are learnable parameters, called weights and biases, respectively, and $f(\cdot)$ is a non-linear activation function. In this paper, we use ReLU non-linearity [32], which is defined as $f(x) = \max\{x, 0\}$.

While, in theory, we can construct the NN agent by applying Equation (4) to the observation $x_i = (x_i^{(h)}, x_i^{(m)}, x_i^{(q)})$, this procedure comes with prohibitive computational complexity and an enormous number of parameters. In the case of direct application of Equation (4) to x_i , we preprocess them by one-hot encoding representation denoted by $\tilde{x}$. Then, one-hot encoding of all possible observations in the observation space has a dimension of $\mathcal{O}(n_H^2 n_M^3 n_Q n_J)$; thus, the matrix multiplication $W^{(1)T} y_i^{(0)}$ is computationally prohibitive.

To resolve this problem, we process human and machine observations, $\mathbf{x}_i^{(h)}$ and $\mathbf{x}_i^{(m)}$, and system queue observation $\mathbf{x}_i^{(q)}$ separately and then concatenate resulting representations and re-process the concatenated representation such that:

$$\hat{\mathbf{x}}_i^{(r)} = f_r^{(f)}(\tilde{\mathbf{x}}_i^{(h)} \oplus \tilde{\mathbf{x}}_i^{(m)}) \quad (5)$$

$$\hat{\mathbf{x}}_i^{(q)} = f_q^{(f)}(\tilde{\mathbf{x}}_i^{(q)}) \quad (6)$$

$$\hat{\mathbf{x}}_i = f_x^{(f)}(\hat{\mathbf{x}}_i^{(r)} \oplus \hat{\mathbf{x}}_i^{(q)}) \quad (7)$$

where $\oplus$ denotes concatenation operator. $f_{i \in \{r, q, x\}}^{(f)}(\cdot)$ is a function of a stack of feedforward layers.

Since the output of each layer has smaller dimension than its input, the above formulation allows for one to encode the observation $\mathbf{x}_i$ at most $\max\{\mathcal{O}(n_H^2 + n_M^3), \mathcal{O}(n_Q n_I)\}$. Besides, since the intermediate representations of human and machine observations $\hat{\mathbf{x}}_i^{(r)}$ and system queue observation $\hat{\mathbf{x}}_i^{(q)}$ are gathered and re-processed using Equation (7), their aggregated semantics can still be captured.

Modeling sequential observations: We aim to build a task allocation agent that can infer unobservable conditions such as fatigue thresholds of human operators and stochastic dynamics of the manufacturing system from a sequence of observations. To achieve this, we utilize recurrent NNs (RNNs) that maintain the sufficient statistics of previous observations and then process the current state information of the system ($\hat{\mathbf{x}}_i$) in the context of the history:

$$\mathbf{h}_i = f^{(r)}(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_i) = f^{(r)}(\mathbf{h}_{i-1}, \hat{\mathbf{x}}_i) \quad (8)$$

where $f^{(r)}(\cdot)$ is a function of a stack of RNNs. Here, we can see that $\mathbf{h}_{i-1}$ plays a role of sufficient statistics of all previous observations ($\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_{i-1}$). This enables representation $\mathbf{h}_i$ to contain the unobservable information implicitly contained in a sequence of observations, such as fatigue accumulation due to intensive workload.

From a theoretical perspective, the RNNs can capture all information contained in a long-term sequence through the learning process. However, training RNNs to store the long-term memory is extremely challenging due to the vanishing and exploding gradient problems [33]. Therefore, we employ long short-term memory (LSTM) [34–36] among variants of RNNs, which mitigates the vanishing and exploding gradient problems.

The basic intuitions behinds LSTM is to maintain an explicit state vector called a memory cell and update the state vectors through several gates that control the amount of storing new information and forgetting existing information. Based on these intuitions, the representation $\mathbf{h}_i$ is computed by the following series of computations:

$$\mathbf{z}_i = \tanh(\mathbf{W}^{(z)} \hat{\mathbf{x}}_i + \mathbf{R}^{(z)} \mathbf{h}_{i-1}) \quad (9)$$

$$\begin{pmatrix} \mathbf{i}_i \\ \mathbf{f}_i \end{pmatrix} = \sigma \left(\begin{pmatrix} \mathbf{W}^{(i)} \\ \mathbf{W}^{(f)} \end{pmatrix} \hat{\mathbf{x}}_i + \begin{pmatrix} \mathbf{R}^{(i)} \\ \mathbf{R}^{(f)} \end{pmatrix} \mathbf{h}_{i-1} + \begin{pmatrix} \mathbf{p}^{(i)} \\ \mathbf{p}^{(f)} \end{pmatrix} \odot \mathbf{c}_{i-1} \right) \quad (10)$$

$$\mathbf{c}_i = \mathbf{z}_i \odot \mathbf{i}_i + \mathbf{c}_{i-1} \odot \mathbf{f}_i \quad (11)$$

$$\mathbf{o}_i = \sigma(\mathbf{W}^{(o)} \hat{\mathbf{x}}_i + \mathbf{R}^{(o)} \mathbf{h}_{i-1} + \mathbf{p}^{(o)} \odot \mathbf{c}_i) \quad (12)$$

$$\mathbf{h}_i = \tanh(\mathbf{c}_i) \odot \mathbf{o}_i \quad (13)$$

where $\odot$ denotes element-wise multiplication. $\sigma(\cdot)$ is a sigmoid function defined as $\sigma(x) = 1/(1 + \exp\{-x\})$. For $\star \in \{z, i, f, o\}$, $\mathbf{W}^{(\star)}$, $\mathbf{R}^{(\star)}$, and $\mathbf{p}^{(\star)}$ are weights. Here, we omit the bias terms in Equations (9), (10) and (12) for brevity. $\mathbf{i}_i$ is called an input gate corresponding to probabilities of receiving new information in each cell, $\mathbf{f}_i$ is called a forget

gate corresponding to probabilities of forgetting information in each cell, and o_i is called an output gate corresponding to probabilities of outputting information in each cell. z_i is block input and h_i is block output.

Task allocation: Finally, the agent allocates tasks based on estimated Q-values for each action. Specifically, another stack of feedforward layers $f_v^{(f)}(\cdot)$ evaluates the desirability of an action in the context of the summarized information of observation sequence h_i :

$$v_i = f_v^{(f)}(h_i) \quad (14)$$

where $v_{ij} = Q((x_1, \dots, x_i), a_j)$ and a_j is j -th action.

The approximated Q-value v_i may not conform to the constraints of precedent requirement, especially in the early stages of training. Although it can be learned by placing extreme penalties on actions that violate the constraints, this would be extremely inefficient since the executable actions take only a small proportion of the entire action space. Therefore, we introduce a function $s: \mathcal{X} \rightarrow 2^{|\mathcal{A}|}$ that represents executable actions at observation $x \in \mathcal{X}$ satisfying constraints. This significantly reduces the search space of actions as well as automatically guarantees that task allocation conforms to explicit conditions. As a result, the agent can provide the Q-values for only executable actions by filtering non-executable ones out.

3.3.2. Learning Objective

Optimization of the Q-value approximation enables the NN agent to learn from the experiences. In this work, the NN agent performs real-time task allocations through observing x_i , executes a_i based on Q-values and receives new observation x_{i+1} for each time step i . In this way, after one episode is completed, the agent obtains a reward r_L , and the history of the episode e with run length L_e , while $D^{(e)} = (x_1^{(e)}, a_1^{(e)}, x_2^{(e)}, \dots, x_{L_e}^{(e)}, a_{L_e}^{(e)}, r_{L_e}^{(e)})$ being stored. Based on accumulated records of different episodes stored in a replay buffer $\mathcal{E}$, parameters of the agent are updated to make approximated Q-values close to true Q-values defined in Equation (3). To this end, a loss function is used to quantify the difference between its Q-value approximation and the target Q-value.

Designing a loss function: In this paper, we define a loss function based on Mnih et al. [18]. An idea behind this loss function is to introduce a target NN that computes the target Q-value and then define a minimization objective as a distance between the target Q-value and Q-value computed by an NN agent. Specifically, the target NN is obtained by copying the neural network at the initialization stage. Then, at each iteration of the optimization process, the parameters of the target network $\Theta^{(t)}$ are updated such that they slowly track parameters of the NN agent Θ through the following rule:

$$\theta^{(t)} \leftarrow \delta \theta + (1 - \delta) \theta^{(t)} \quad (15)$$

where $\theta^{(t)} \in \Theta^{(t)}$ and $\delta \in [0, 1]$. With the target NN, the target Q-value for the history of the episode $e \in \mathcal{E}$ up to time step $i + 1$ is obtained based on Equation (3):

$$\hat{Q}^T(x_{1:i+1}^{(e)}, r_{L_e}^{(e)}) = r_{L_e}^{(e)} 1_{\{L_e\}}(i) + \gamma \max_{a' \in \mathcal{A}} Q^T(x_{1:i+1}^{(e)}, a') \quad (16)$$

where $1_{\{L_e\}}(i)$ is an indicator function that takes one if $i = L_e$ and zero otherwise, and $Q^T(x_{1:i+1}^{(e)}, a')$ is a Q-value for a' computed by the target neural network for inputs $x_{1:i+1}^{(e)}$.

This target value incurs our minimization objective for optimizing parameters of the neural network agent Θ :

$$\mathcal{L}(\Theta) = -\mathbb{E}_{D_{1:i+1}^{(e)}} \left[\left(\hat{Q}^T(x_{1:i+1}^{(e)}, r_{L_e}^{(e)}) - Q(x_{1:i}^{(e)}, a_i^{(e)}) \right)^2 \right] \quad (17)$$

where $D_{1:i+1}^{(e)} = (\mathbf{x}_{1:i+1}^{(e)}, \mathbf{a}_{1:i}^{(e)}, r_{L_e}^{(e)})$, $D_{1:i+1}^{(e)}$ is sampled from a probability distribution $p^{\mathcal{E}}$ that samples both episode e and time step i from the replay buffer $\mathcal{E}$, and $Q(\mathbf{x}_{1:i}^{(e)}, \mathbf{a}_i^{(e)})$ is an approximated Q-value computed by the neural network agent for job assignment $\mathbf{a}_i^{(e)}$.

Finally, the replay buffer requires actions with both high and low rewards for better exploration. Therefore, we use the Boltzmann exploration strategy during training:

$$p(\mathbf{a}_i = \mathbf{a}_j) = \frac{\exp\{\mathbf{v}_i / \tau\}_j}{\sum_{k \in \mathcal{S}(\mathbf{x}_i)} \{\exp\{\mathbf{v}_i / \tau\}\}_k} \quad (18)$$

where τ is an exploration noise. Note that a small τ encourages exploitation of the current approximation of Q-values $\mathbf{v}_i$ by biasing the probability to the peak distribution; conversely, a large τ encourages exploration by biasing the probability to the uniform distribution.

Optimization of parameters: Parameters Θ are adjusted to optimize the loss function based on gradient-based optimization techniques. To compute the gradient, backpropagation [35] recursively applies the chain rule. For instance, gradients of $\mathcal{L}(\Theta)$ with respect to activation at layer l , $\mathbf{a}^{(l)} = \mathbf{W}^{(l)T} \mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}$, are computed by:

$$\begin{aligned} \nabla_{\mathbf{y}^{(l)}} \mathcal{L}(\Theta) &= \left(\frac{\partial \mathbf{y}^{(l+1)}}{\partial \mathbf{y}^{(l)}} \right)^T \nabla_{\mathbf{y}^{(l+1)}} \mathcal{L}(\Theta) \\ &= \mathbf{W}^{(l+1)} \text{diag}(\mathbf{f}'(\mathbf{a}^{(l+1)})) \nabla_{\mathbf{y}^{(l+1)}} \mathcal{L}(\Theta) \end{aligned} \quad (19)$$

$$\begin{aligned} \nabla_{\mathbf{a}^{(l)}} \mathcal{L}(\Theta) &= \left(\frac{\partial \mathbf{y}^{(l)}}{\partial \mathbf{a}^{(l)}} \right)^T \nabla_{\mathbf{y}^{(l)}} \mathcal{L}(\Theta) \\ &= \text{diag}(\mathbf{f}'(\mathbf{a}^{(l+1)})) \nabla_{\mathbf{y}^{(l)}} \mathcal{L}(\Theta) \end{aligned} \quad (20)$$

which gives the gradient with respect to parameters: $\nabla_{\mathbf{W}^{(l)}} \mathcal{L}(\Theta) = \mathbf{y}^{(l-1)} (\nabla_{\mathbf{a}^{(l)}} \mathcal{L}(\Theta))^T$ and $\nabla_{\mathbf{b}^{(l)}} \mathcal{L}(\Theta) = \nabla_{\mathbf{a}^{(l)}} \mathcal{L}(\Theta)$.

Among various gradient-based optimization techniques, this paper employs the Adam optimizer that provides fast convergence with desirable optima for a wide range of problems in practice [37]. Specifically, for each $\theta \in \Theta$ with gradient $\mathbf{g}$, the Adam optimizer maintains momentum vectors $\mathbf{m}$ and $\mathbf{v}$, which were initialized by zeros:

$$\mathbf{m} \leftarrow \beta_1 \mathbf{m} + (1 - \beta_1) \mathbf{g} \quad (21)$$

$$\mathbf{v} \leftarrow \beta_2 \mathbf{v} + (1 - \beta_2) \mathbf{g} \odot \mathbf{g} \quad (22)$$

where $\beta_1, \beta_2 \in [0, 1)$ are decay rates. Then, parameter θ is updated as:

$$\theta \leftarrow \theta - \alpha \frac{\hat{\mathbf{m}}}{\epsilon + \sqrt{\hat{\mathbf{v}}}} \quad (23)$$

where $\hat{\mathbf{m}} = \mathbf{m} / (1 - \beta_1^t)$ and $\hat{\mathbf{v}} = \mathbf{v} / (1 - \beta_2^t)$ are bias-corrected momentums, α is step size, and ϵ is a small constant used for numerical stability.

4. Experiments and Results

This section illustrates a task allocation for the proposed methodology. We first describe a human-machine manufacturing system and then apply the proposed framework to the system. Experiment results with interpretations are also presented.

4.1. Target Environment

4.1.1. Static Characteristics

The target system consists of five human operators and ten machines to process five types of jobs. Each job has its own precedence requirement for ten classes of tasks as shown

in Table 2. For instance, Job j_1 can be completed by performing task t_1 followed by t_6 , after which both t_8 and t_{10} are performed in any order. Tasks are different in terms of complexity and operator competence levels as shown in Tables 3 and 4. Task complexity takes a normalized value ranging from 0 to 1. Complexity value of 0 indicates that a task requires little skill to perform, while the value of 1 indicates the most complicated task. Task complexity can be significantly related to competence levels of human operators. For task t_2 of somewhat less complexity, most of the five operators have similar competence levels. Conversely, only operator h_1 has a high competence level for task t_{10} , which has high complexity. The complexity of the task and competence levels need to be assessed in consideration of a target manufacturing system. Table 4 shows the competence level of the operator for the corresponding task class, ranging from 0 to 1. Herein, c_{ij}^O and c_j^T denote operator i competence level for task t_j and task complexity of t_j , respectively.

Table 2. Task precedence relations.

Job	j_1	j_2	j_3	j_4	j_5
Task precedents	$\{1\}, \{6\}, \{8, 10\}$	$\{1\}, \{5, 6\}, \{8\}$	$\{2, 4\}, \{7\}, \{9\}$	$\{3, 4\}, \{7\}, \{10\}$	$\{2, 3\}, \{5\}, \{9\}$

Table 3. Task complexity for each task class.

Task Class	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Complexity	0.23	0.17	0.39	0.37	0.33	0.73	0.54	0.37	0.77	0.81

Table 4. Competence levels of human operators for tasks.

Operator	Task									
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
h_1	0.51	0.85	0.89	0.99	0.49	0.13	0.02	0.00	0.02	0.89
h_2	0.93	0.57	0.93	0.93	0.04	0.47	0.74	0.07	0.87	0.00
h_3	0.98	0.90	0.88	0.99	0.54	0.00	0.08	0.95	0.00	0.04
h_4	0.96	0.91	0.89	0.97	0.80	0.76	0.65	0.02	0.00	0.31
h_5	0.92	0.59	0.96	0.89	0.87	0.81	0.87	0.00	0.29	0.38

4.1.2. Dynamic Characteristics

In addition to human operators, the manufacturing system presented in this experiment has variabilities in batch arrivals of jobs and task processing times as well as machine breakdowns. Specifically, the inter-arrival time of batches is assumed to follow the exponential distribution with $\mu = 1500$ s, and the number of jobs in each batch follows the binomial distribution with $n = 10$, $p = 0.4$. Furthermore, the time between breakdowns is assumed to follow the exponential distribution with $\mu = 20,000$ s, and repair time following the uniform distribution between 2000 and 6000 s.

Standard time taken to complete a task by an operator with zero fatigue level follows the exponential distribution with $\mu = \left(300 \left(1 - c_{ij}^O\right) c_j^T\right)$. This suggests that task processing time takes longer, as complexity of a task increases or competence level of an operator in performing the task decreases. Parameters of the probability distributions characterizing the stochastic property of the system are synthesized from [38].

Fatigue level $\psi^{(i)}(t)$ of human operator i at time t after performing task j during δ is assumed to increase as follows [3,29,39]:

$$\begin{aligned} & \psi^{(i)}(t + \delta) \\ &= \psi^{(i)}(t) + \alpha \left(1 - \psi^{(i)}(t)\right) \left(1 - \exp\left\{-\frac{c_j^T \delta}{3600}\right\}\right) \end{aligned} \quad (24)$$

where α is a control coefficient and 3600 in the denominator is a time scale (hour) constant. Figure 3 shows an example of fatigue accumulation depending on different utilizations when $c_j^T = 0.5$.

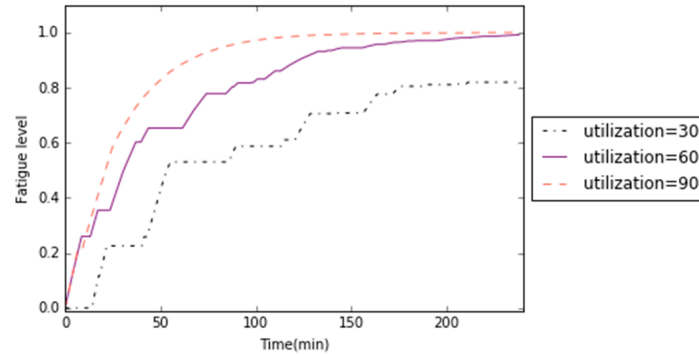


Figure 3. Example of fatigue accumulation during four hours of work with different utilizations.

We note that the fatigue accumulations can vary depending on both the task complexity and time taken to perform a task. For example, assigning a task to an operator with less experiences can cause more fatigue than assigning it to a skillful operator. This is partly because task-performing time tends to increase as operator competency level for the task decreases, resulting in more fatigue.

To simulate performance decay incurred by fatigue accumulation, an additional time factor is added to the task time. The additional time to perform task j by operator i with fatigue level of $\psi^{(i)}(t)$ is given by:

$$T' = \left(\frac{1 - c_{ij}^O}{\ln 2} \right) \left(\ln(1 + \psi^{(i)}(t)) \right) \quad (25)$$

where $(1 - c_{ij}^O) / \ln 2$ controls the upper bound of the additional task time, which indicates that skillful operators are less vulnerable to performance deterioration by fatigue accumulation. In this regard, it is desirable to allocate tasks to human operators such that the effect of fatigue accumulation on their performances can be minimized.

4.2. Deep Reinforcement Learning Framework Configuration

In this experiment, an observation space $\mathcal{X}$ and an action space $\mathcal{A}$ are determined as:

$$\mathcal{X}^{(h_i)} = \{\text{idle}, t_1, t_2, \dots, t_{10}\}$$

$$\mathcal{X}^{(m_i)} = \{\text{idle}, \text{working}, \text{breakdown}\}$$

$$\mathcal{X}^{(q_i)} = \{\emptyset, j_1, j_2, \dots, j_5\} \times \{\text{idle}, h_1, h_2, \dots, h_5\} \times \{\text{idle}, t_1, t_2, \dots, t_{10}\}$$

$$\mathcal{A} = \{j_1, j_2, \dots, j_5\} \times \{h_1, h_2, \dots, h_5\} \times \{t_1, t_2, \dots, t_{10}\}$$

A reward function r is given by Equation (2), and constants for computing reward are set to $t_{\text{terminate}} = 120$, $b^{(u)} = -10$, $b^{(c)} = 150$, $b_L^{(u)} = -1/40$, $b_L^{(c)} = 1/60$. Here, we design the reward function such that base bonus $b^{(c)}$ and penalty $b^{(u)}$ dominate overall scale of rewards to promote the agent to learn how to complete the task allocation episode first. We also define a discount factor as $\gamma = 0.99$, following the commonly used value in practice.

As an agent, each of the two feedforward layers processes waiting jobs in queue and observations of humans and machines, respectively. Then, another feedforward layer is exploited to further process concatenated observations processed by the earlier layer. After that, two LSTM layers update a cell state c_t and an output y_t , and a feedforward layer computes Q -value for each action.

Parameters of the agent are updated every five episodes. In addition, we used a replay buffer size of 1000, a minibatch size of 64, hyperparameters ($\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) for Adam optimizer. $\delta = 10^{-3}$ is used for updating parameters of the target network as in Equation (15). For regularizing the network, parameter weight decay with parameter 10^{-4} is used. For exploration, $\tau = 0.5$ is used in Equation (18).

4.3. Experimental Results and Interpretation

Figure 4 shows the cumulative reward for each episode during training of 20,000 episodes. It is observed that the reward value becomes higher as trainings are repeated, which implies that the agent allocates tasks while avoiding penalty and achieving rewards by completing given jobs on time.

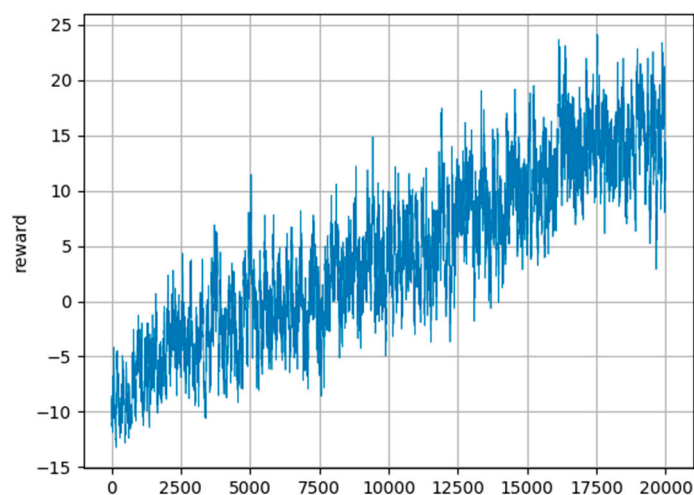


Figure 4. Cumulative rewards during training step.

The principal goal of the method proposed in this work is to cater to human operators. Therefore, we first investigate fatigue accumulations of human operators. Before training of the agent, it is observed that fatigue accumulations of human operators abruptly increase after performing of a certain job as shown in Figure 5 (left). For instance, operator h_1 and h_5 suffer from sharp increases in fatigue accumulation at time 30 and 100 min, respectively. Conversely, a trained agent is shown to alleviate such steep increases in fatigue of each human operator as shown in Figure 5 (right).

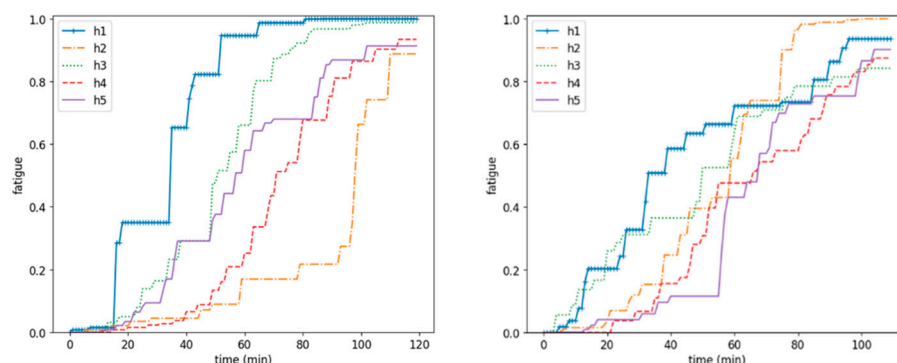


Figure 5. Fatigue accumulation: before training (left) and after training (right).

Furthermore, the variations of fatigue accumulation rates among human operators are shown to decrease. In other words, they tend to be distributed evenly through human operators, which means that the agent successfully considers operator fatigue levels in allocating tasks to operators.

We then examined task allocation from the viewpoint of human operator task competence levels, which corresponds to how tasks are allocated to human operators whose competence level is higher for the allocated tasks. Figure 6 shows human competence levels for ten classes of tasks (left) and the distribution of allocated tasks to operators (right). The left diagram illustrates that operators have a similarly high level of competence for task t_4 , while only operator h_1 has high competence for task t_{10} . It seems to be reasonable for a task to be allocated to an operator with a higher competence level for it. This corresponds to a distribution of task allocation in the right diagram where operator h_1 has performed task t_{10} the most.

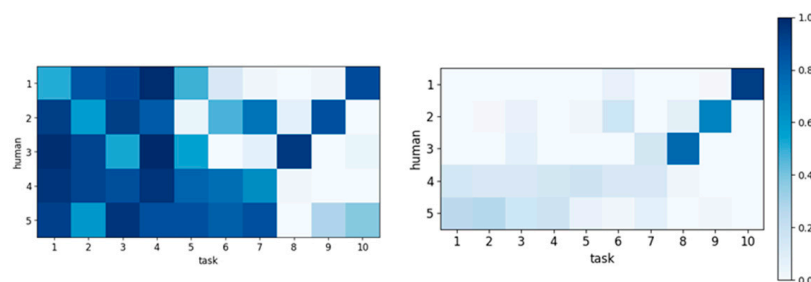


Figure 6. Human competence levels and task (left) and task allocation ratio (right).

Consider task t_4 for which operator h_1 has a high competence level among all the operators. Based on the argument above, it should have been allocated to operator h_1 . However, it is evenly allocated to operators. This is because other operators also have similar high competence levels for the task. This is also observed in task allocations for operators h_2 and h_3 . These operators are assigned to perform tasks for which they show exceptionally higher competence levels compared to other operators.

We compared the task allocation results with those by two extreme cases of FIFO and SPT rules. The FIFO (first-in, first-out) rule assigns tasks to any human operator who is at the idle state regardless of the competence levels of operators. On the contrary, the SPT (shortest processing time) rule assigns tasks to a human who has the highest competence for the tasks.

As shown in the left part of Figure 7, tasks performed by humans are evenly distributed among operators under the FIFO (first-in, first-out) rule. This is obvious because tasks are allocated to humans in such a way that operators are considered to be mere manufacturing resources available at the time of job arrival without considering their unobservable states in terms of fatigue accumulation and competence level.

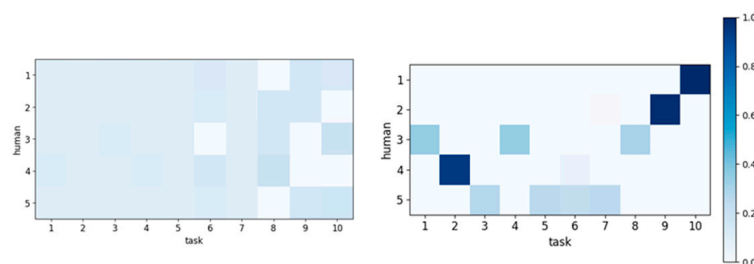


Figure 7. Task allocation ratios: FIFO (left) and SPT (right).

Under the SPT rule, tasks are allocated to humans with a high competence level for tasks in order to shorten the processing time. For instance, in the right part of Figure 7, it is observed that task t_1 is mostly assigned to human operator h_3 , who has the highest competence level for t_1 . Similarly, human operator h_4 , with a relatively high competence level for task t_2 is assigned t_2 much more than the other operators. With this allocation, it could be conjectured that operators with a high competence level for specific tasks can suffer from rapid accumulation of fatigue due to repetitive performance of tasks requiring a high competence level as in Equation (24).

In this regard, it is suggested that dispatching rules to allocate tasks to manufacturing resources without considering human states has much room for improvement in terms of unbalanced workloads across the operators. The proposed method, however, allocates tasks to humans while considering their competence level and balanced fatigue among them. Tasks requiring a high competence level are allocated to corresponding human operators, while other operators can also be assigned to perform tasks when fatigue levels are to be distributed among all the operators.

The proposed method mostly aims to cater to human operators. However, it is also desirable to ensure reasonably good scheduling performance. To examine the scheduling performance, we compared the mean flowtimes under the proposed method and those of two dispatching rules—shortest processing time (SPT) and first-in first-out (FIFO).

As shown in Figure 8, the proposed approach yields a shorter mean flowtime with less variation. Specifically, a mean and standard deviation of mean flowtime for 1500 episodes under the proposed method, SPT rule, and FIFO rule are $(\mu = 25.97, \sigma = 16.44)$, $(\mu = 39.98, \sigma = 29.19)$, and $(\mu = 39.84, \sigma = 24.47)$, respectively. This shows an encouraging outcome in terms of performance of flowtime and could be attributed to workload balance among operators and assigning tasks to operators with an appropriate competence level for them.

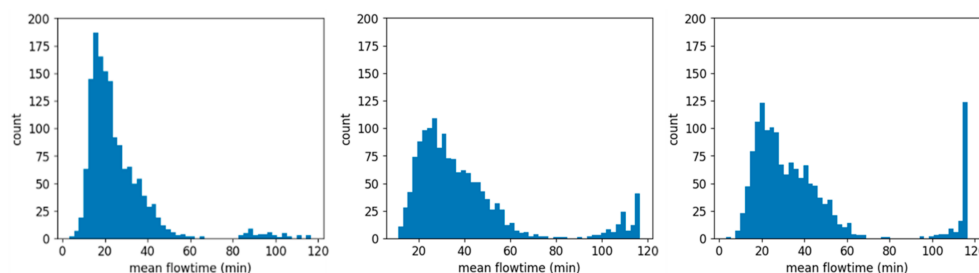


Figure 8. Mean flowtimes: proposed method (left), FIFO rule (middle), and SPT rule (right) for 1500 episodes.

From the managerial point of view, it is desirable for operators to be treated fairly in terms of workload. Unbalanced workload among operators could be considered a factor causing bottlenecks in line with balancing problems. In this sense, the results suggest that tasks are allocated in a way that fatigue accumulation of operators are balanced with small variations. This is achieved by allocating tasks to operators with high competence levels for corresponding tasks. It can be meaningful to the management in evaluating operators since assessment of operator job performance needs to be carried out in accordance with the tasks that are allocated to appropriate operators.

Catering to human operators does not mean degradation of manufacturing performances from the management viewpoint. The method proposed in this work also achieves relatively good performance in terms of mean flowtime and variation. Compared to conventional dispatching rules, short flowtime with small variation is achieved. This can mean that care for humans improves managerial performances rather than compromises them. This is an important aspect from managerial and sustainability perspectives.

5. Conclusions

Although automation has become more and more pervasive in manufacturing systems, human operators are still essential for many manufacturing activities. In this regard, it is desirable to take the characteristics of human operators into account in task allocation. While humans can perform a variety of subtle tasks in manufacturing systems, which increases flexibility of resource allocation, one of the several challenges to overcome is to consider unobservable characteristics of human operators. Several studies in the literature investigate human flexibility in task allocation, and the unobservable characteristics of human operators have not received much attention.

In this regard, we present a framework for task allocation in human–machine manufacturing systems and develop an RL-based method. By training an agent in an RL framework with task allocation data in an iterative manner, unobservable states of human operators are assessed and improved upon using recurrent layers. The agent allocates tasks according to an expected cumulative discounted reward in each task allocation episode. The agent considers fatigue accumulation and task competence level of human operators, while achieving faster mean flowtime compared to classical dispatching rules.

In this work, deep learning (DL) serves as a framework, while reinforcement learning (RL) is employed for updating the performance of task allocation in the form of a reward function. RL captured the system behavior in terms of human fatigue and operational aspects of a manufacturing system such as precedence requirement and the number of waiting jobs for allocation. As task allocation episodes are iterated in a framework of DL, RL continuously improves the task allocation performance.

RL captured the system behavior in terms of human fatigue and operational aspects of manufacturing systems such as precedence requirement and the number of waiting jobs for allocation. As task allocation episodes are iterated in a framework of DL, RL continuously improves the task allocation performance.

As a future direction of this work, the proposed approach should be extended into a human-in-the-loop environment, wherein human supervisors are involved as well as operators. In addition, considering the learning effects of operators in the middle of performing assigned tasks could possibly be a promising topic for future work. This direction of extension corresponds to the current trends in manufacturing where cognitive aspects of humans are becoming more important in accordance with advances in automation.

Human–machine interfaces can be observed, and implemented cases are also available in practice. They usually take the form of a computer or mechanical apparatus for assisting operators in performing tasks. However, the task allocation between human and machine remains at an early stage of implementation. This is partly because the outcome and the benefits of these kind of approaches cannot easily be assessed quantitatively. In this sense, the further effort toward boosting industrial and academia interest in human operators in task allocation needs to be taken.

Finally, we note that deep RL-based task allocation may have limited practical values in the systems in case of frequent changes in operators and machines. This is because such modifications require retraining of the existing agent. However, in the cases of minor modifications where the changes are relatively small, incremental learning or transfer learning could be exploited. These techniques allow the NN agent to adapt to the new systems by “recycling” what the system already learned before rather than being trained from scratch. Transfer learning can be one of the promising techniques in applying results gained from physical task allocation to allocation of cognitive or supervisory tasks. The complexity of physical tasks and competence levels of operators could take a “similar” form as supervisory tasks or missions. We leave the employment of incremental learning or transfer learning in the task allocation of human–machine manufacturing systems as an important future direction of research.

Author Contributions: Conceptualization, D.S. and T.J.; methodology, T.J.; software, T.J.; validation, D.S. and T.J.; formal analysis, H.J.; data curation, T.J.; writing—original draft preparation, T.J.; writing—review and editing, D.S.; visualization, T.J. and H.J.; supervision, D.S.; project administration, D.S.; funding acquisition, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was supported by the Korea Institute for Advancement of Technology (KIAT), grant funded by the Korean Government (MOTIE) (P0008691), HRD Program for Industrial Innovation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krüger, J.; Lien, T.K.; Verl, A. Cooperation of human and machines in assembly lines. *CIRP Ann.* **2009**, *58*, 628–646. [\[CrossRef\]](#)
2. Agnetis, A.; Murgia, G.; Sbrilli, S. A job shop scheduling problem with human operators in handicraft production. *Int. J. Prod. Res.* **2014**, *52*, 3820–3831. [\[CrossRef\]](#)
3. Digiesi, S.; Kock, A.A.A.; Mummolo, G.; Rooda, J.E. The effect of dynamic worker behavior on flow line performance. *Int. J. Prod. Econ.* **2009**, *120*, 368–377. [\[CrossRef\]](#)
4. Lodree, E.J.; Geiger, C.D.; Jiang, X. Taxonomy for integrating scheduling theory and human factors: Review and research opportunities. *Int. J. Ind. Erg.* **2009**, *39*, 39–51. [\[CrossRef\]](#)
5. Sedighi Maman, Z.; Alamdar Yazdi, M.A.; Cavuoto, L.A.; Megahed, F.M. A data-driven approach to modeling physical fatigue in the workplace using wearable sensors. *Appl. Ergon.* **2017**, *65*, 515–529. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Trentesaux, D.; Moray, N.; Tahon, C. Integration of the human operator into responsive discrete production management systems. *Eur. J. Oper. Res.* **1998**, *109*, 342–361. [\[CrossRef\]](#)
7. Zandieh, M.; Adibi, M.A. Dynamic job shop scheduling using variable neighbourhood search. *Int. J. Prod. Res.* **2010**, *48*, 2449–2458. [\[CrossRef\]](#)
8. Mencía, R.; Sierra, M.R.; Mencía, C.; Varela, R. Memetic algorithms for the job shop scheduling problem with operators. *Appl. Soft Comput.* **2015**, *34*, 94–105. [\[CrossRef\]](#)
9. Mencía, R.; Sierra, M.R.; Mencía, C.; Varela, R. Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators. *Integr. Comput.-Aided Eng.* **2016**, *23*, 269–285. [\[CrossRef\]](#)
10. Xiong, J.; Xing, L.; Chen, Y. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *Int. J. Prod. Econ.* **2013**, *141*, 112–126. [\[CrossRef\]](#)
11. Yoshitomi, Y.; Yamaguchi, R. A genetic algorithm and the Monte Carlo method for stochastic job-shop scheduling. *Int. Trans. Oper. Res.* **2003**, *10*, 577–596. [\[CrossRef\]](#)
12. Chen, J.C.; Wu, C.-C.; Chen, C.-W.; Chen, K.-H. Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Syst. Appl.* **2012**, *39*, 10016–10021. [\[CrossRef\]](#)
13. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [\[CrossRef\]](#)
14. Eguchi, T.; Oba, F.; Hirai, T. A neural network approach to dynamic job shop scheduling. In *Proceedings of the Global Production Management: IFIP WG5.7 International Conference on Advances in Production Management Systems, Berlin, Germany, 6–10 September 1999*; Mertins, K., Krause, O., Schallack, B., Eds.; Springer US: Boston, MA, USA, 1999; pp. 152–159.
15. Weckman, G.R.; Ganduri, C.V.; Koonce, D.A. A neural network job-shop scheduler. *J. Intell. Manuf.* **2008**, *19*, 191–201. [\[CrossRef\]](#)
16. Golmohammadi, D. A neural network decision-making model for job-shop scheduling. *Int. J. Prod. Res.* **2013**, *51*, 5142–5157. [\[CrossRef\]](#)
17. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
18. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [\[CrossRef\]](#)
19. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [\[CrossRef\]](#)
20. Peng, X.B.; Berseth, G.; van de Panne, M. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.* **2016**, *35*, 1–12. [\[CrossRef\]](#)
21. Gabel, T.; Riedmiller, M. Adaptive reactive job-shop scheduling with reinforcement learning agents. *Int. J. Inf. Technol. Intell. Comput.* **2008**, *24*, 30.
22. Gabel, T.; Riedmiller, M. Distributed policy search reinforcement learning for job-shop scheduling tasks. *Int. J. Prod. Res.* **2012**, *50*, 41–61. [\[CrossRef\]](#)
23. Hoc, J.-M. From human-machine interaction to human-machine cooperation. *Ergonomics* **2000**, *43*, 833–843. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Hu, B.; Chen, J. Optimal task allocation for human-machine collaborative manufacturing systems. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1933–1940. [\[CrossRef\]](#)
25. Tsarouchi, P.; Makris, S.; Chrysosolouris, G. Human-robot interaction review and challenges on task planning and programming. *Int. J. Comput. Integr. Manuf.* **2016**, *29*, 916–931. [\[CrossRef\]](#)
26. Park, I.; Huh, J.; Kim, J.; Park, J. A reinforcement learning approach to robust scheduling of semiconductor manufacturing Facilities. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1420–1431. [\[CrossRef\]](#)
27. Oliff, H.; Mumar, M.; Williams, M.; Ryan, M. Reinforcement learning for facilitating human-robot-interaction in manufacturing. *J. Manuf. Syst.* **2020**, *56*, 326–340. [\[CrossRef\]](#)
28. Malik, A.; Bilberg, A. Complexity-based task allocation in human-robot collaborative assembly, Industrial Robot. *Int. J. Robot. Res. Appl.* **2019**, *46*, 471–480.
29. Grandjean, E. Fatigue in industry. *Occup. Environ. Med.* **1979**, *36*, 175–186. [\[CrossRef\]](#)
30. Ferjani, A.; Ammar, A.; Pierreval, H.; Elkosantini, S. A simulation-optimization based heuristic for the online assignment of multi-skilled workers subjected to fatigue in manufacturing systems. *Comput. Ind. Eng.* **2017**, *112*, 663–674. [\[CrossRef\]](#)
31. Manne, A.S. On the Job-Shop Scheduling Problem. *Oper. Res.* **1960**, *8*, 219–223. [\[CrossRef\]](#)

32. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; Gordon, G., Dunson, D., Dudik, M., Eds.; PMLR: 2011. Available online: <https://proceedings.mlr.press/v15/glorot11a.html> (accessed on 10 January 2022).
33. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [[CrossRef](#)]
34. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
35. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2222–2232. [[CrossRef](#)]
36. Rumelhart, D.E.; Williams, R.J.; Hinton, G.E. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [[CrossRef](#)]
37. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
38. Shen, X.-N.; Yao, X. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Inf. Sci.* **2015**, *298*, 198–224. [[CrossRef](#)]
39. Konz, S. Work/rest: Part II—The scientific basis (knowledge base) for the guide. *Int. J. Ind. Ergon.* **1998**, *22*, 73–79. [[CrossRef](#)]