Multi-Agent Deep Reinforcement Learning Based Demand Response for Discrete Manufacturing Systems Energy Management

Renzhi Lu[a], Yi-Chang Li[b,*], Yuting Li[c], Junhui Jiang[c], Yuemin Ding[d,*]

[a]*Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China*
[b]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[c]*Department of Electronic Systems Engineering, Hanyang University, Ansan 15588, Korea*
[d]*Department of Energy and Process Engineering, Norwegian University of Science and Technology, Trondheim 7034, Norway*

Abstract

With advances in smart grid technologies, demand response has played a major role in improving the reliability of grids and reduce the cost for customers. Implementing the demand response scheme for industry is more necessary than for other sectors, because its energy consumption is often considered the largest. This paper proposes a multi-agent deep reinforcement learning based demand response scheme for energy management of discrete manufacturing systems. In this regard, the industrial manufacturing system is initially formulated as a partially-observable Markov game; then, a multi-agent deep deterministic policy gradient algorithm is adopted to obtain the optimal schedule for different machines. A typical lithium-ion battery assembly manufacturing system is used to demonstrate the effectiveness of the proposed scheme. Simulation results show that the presented demand response algorithm can minimize electricity costs and maintain production tasks, as compared to a benchmark without demand response. Moreover, the performance of the multi-agent deep reinforcement learning approach against a mathematical model method is investigated.

Keywords:
Artificial intelligence, deep reinforcement learning, demand response, industrial energy management, discrete manufacturing system.

1. Introduction

With the development of smart grid (SG) technologies, demand response (DR) is playing an increasingly significant role in facilitating economic efficiency [1], enhancing operational flexibility [2], and improving system reliability [3] of the SG. The DR programs offer demand flexibility by motivating end users to adapt their energy consumption profiles in response to time-varying electricity prices or other grid signals [4]. In terms of various electricity customers, industrial facilities consume the largest portion of energy, compared to other end-use sectors [5]. According to the International Energy Outlook from Energy Information Administration [6] in 2017, industry accounted for about 54% of the world's total delivered energy. This proportion is larger in developing countries; for instance, in China it has even reached to 72%. Thus, it is both essential and urgent to realize DR programs for industrial energy management.

However, realizing successful DR schemes in industrial facilities is challenging and complicated because many industrial processes are sequential, interdependent and correlated; modeling DR problems in industrial sectors should capture the physical characteristics of different machines, which increases the complexity of the model [7]. In addition, among different industrial applications, their processes and load profiles vary considerably, making it difficult to design a universal model for industrial DR. Until now, there have been few studies demonstrating the feasibility and benefit of industrial DR. For instance, the study of [8] applied an adaptive multi-objective memetic algorithm for industrial DR to prevent a rise in electricity and labor costs. In [9], the authors proposed a decision model for industrial load management in face of time-changing electricity prices. The work of [10] introduced an active time-based DR model enabling industrial customers to shift their energy consumption by following daily price curves. An optimal industrial load control model was investigated in [11] to minimize the energy cost. The study of [12] proposed a DR scheme for industrial energy management based on the state task network and mixed integer linear programming (MILP). Similarly, an intelligent energy management framework with DR capability for industrial facility was developed in [13], wherein the industrial processes were also modeled by the state task network, then they were optimized via MILP. The study in [14] presented a structure for industrial DR aggregators to provide operational flexibility for the power system, and a robust self-scheduling approach was utilized to optimize the entire production line. The performances of the aforementioned approaches, however, rely directly on the precision of the model employed in each optimization problem. Also, the mathematical formulations of

*Corresponding authors
Email addresses: liyichang2012@hotmail.com (Yi-Chang Li), yuemin.ding1986@gmail.com (Yuemin Ding)

Nomenclature

Abbreviations

SG	smart grid
DR	demand response
MILP	mixed integer linear programming
AI	artificial intelligence
DRL	deep reinforcement learning
RL	reinforcement learning
DL	deep learning
DPG	deep policy gradient
DDPG	deep deterministic policy gradient
MDP	Markov decision process
POMG	partially-observable Markov game
MADDPG	multi-agent deep deterministic policy gradient
SM	smart meter
GW	gateway
FEMC	factory energy management center
UPL	utility power line
FPL	facility power line
WAN	wide area network
LAN	local area network
DQN	deep Q-network
SF	side frames
BC	battery cells
CP	cooling plates
IF	intermediate frames
CF	compression foams
ReLU	rectified linear unit

Variables and Parameters

$M_{i,j}$	machine representation
i	serial production-line branch index
j	machine index in the ith branch
$B_{i,j}$	buffer representation
$Z^h_{i,j}$	the decision variable of machine $M_{i,j}$
h	hour index
$e^h_{i,j}$	energy consumption of machine $M_{i,j}$ in hour h
$e^{op}_{i,j}$	energy demand of machine $M_{i,j}$ in operation mode
$e^{idle}_{i,j}$	energy demand of machine $M_{i,j}$ in idle mode
E^h	total energy consumption of all machines in hour h
$E_{\max}$	a threshold of total energy consumption
$P^h_{i,j}$	generated quantity of machine $M_{i,j}$
$C^h_{i,j}$	consumed quantity of machine $M_{i,j}{}'$
$p^h_{i,j}$	production rate of machine $M_{i,j}$
$c^h_{i,j}$	consumption rate of machine $M_{i,j}{}'$
$B^{\min}_{i,j}$	lower bound of buffer capacity
$B^{\max}_{i,j}$	upper bound of buffer capacity
v	unit value of output good
g	output good
c	unit cost of input material
m	input material
π^h	electricity price at hour h
S	a set of states
O	a set of observations
A	a set of actions
R	real-valued reward function
$\mu_{i,j}$	policy of an agent
$r_{i,j}$	immediate reward of an agent
$R_{i,j}$	cumulative rewards of an agent
γ	a discount factor
$o_{i,j}$	observation of an agent
$a_{i,j}$	action of an agent
s	system state
a	system action
r	system reward
$Q_{i,j}$	individual Q-network
$\phi_{i,j}$	the parameter of Q-network
$\theta_{i,j}$	the parameter of policy network
$\zeta_{i,j}$	loss function of Q-network
$Q^{\mu}_{i,j}$	critic network under policy $\mu_{i,j}$
$Q^{\mu}_{i,j}{}'$	target critic network under policy $\mu_{i,j}$
D	experience replay buffer
$\phi_{i,j}{}'$	target parameter of Q-network
$\theta_{i,j}{}'$	target parameter of policy network
χ	a noise process

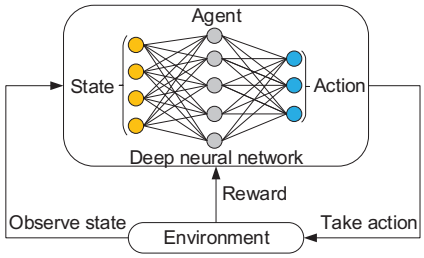

Figure 1: Deep reinforcement learning (DRL) diagram.

model-based methods are usually complicated [15], developing and maintaining an accurate model is a significant task, where the cost of doing so might outweigh its financial benefits. To overcome these issues, there is an impending need for a solution that avoids complex modeling while maintaining production tasks and minimizing energy costs.

Over the past few years, with the rapid evolution of artificial intelligence (AI), deep reinforcement learning (DRL) has become a focus due to its success in addressing challenging sequential decision-making problems [16]. DRL combines the decision making of reinforcement learning (RL) and the information perception of deep learning (DL), as shown in Fig. 1 [17]. RL is a type of machine learning algorithm concerned with how an agent chooses the best behaviors in a stochastic environment, so as to maximize the cumulative rewards [18]. DL can be integrated with RL for representing states and approximating functions. Within the agent and environment interaction of RL, DL can maintain the internal policy of the agent, which determines the next action based on the current state of the environment. Owing to the unique features of being model free and no need for a priori domain knowledge paradigms [19]; DRL can use raw state representations directly and train policies with effective and efficient approaches for high-dimensional feature extraction and non-linear generalization, to ensure the optimal control of complex systems [20].

Some research has been published on employing DRL for solving DR problems in SG energy management. The work described in [21] demonstrated the benefits of using DRL to perform on-line optimization scheduling for building energy management systems; its learning procedure was explored by deep Q-learning and deep policy gradient (DPG). The study of [22] proposed an energy management strategy for a plug-in hybrid electric bus based on deep deterministic policy gradient (DDPG), which is an actor-critic, model-free DRL algorithm that can assign the optimal energy split of the bus over continuous spaces. In [23], a DR algorithm was presented to determine a charging policy for electric vehicles, considering the stochastic of user behaviors and utility prices. The scheduling problem was formulated as a Markov decision process (MDP) with an unknown transition probability; afterwards, DRL was used to obtain the optimal strategy. Similarly, the work in [24] applied a DRL algorithm for local energy trading to promote the action of customers joining a localized energy ecosystem, wherein the decision-making process was also built by an MDP with continuous variables; then, this decision-making process of local market participation was solved by deep Q-learning without analytical calculations or prior knowledge of the market model. In [25], the authors investigated DR management for an energy internet, where the practical energy management problem was formulated as a constrained optimal control scheme; and DRL algorithm was applied to obtain the desired control solution. The authors of [26] verified how a deep neural network can be integrated with fitted Q-iteration in a realistic DR setting for residential load control subject to partial observability. The work of [27] developed an energy management scheme on economical operation for a microgrid, in which the approximate dynamic programming and deep recurrent neural network learning were employed to derive the optimal scheduling policy considering uncertainties in, and various power flow constraints on, electricity loads, renewable resources and real-time prices. In [28], a distributed operation strategy was proposed to manage the operation of a battery energy storage system in microgrid via double deep Q-learning method, which is capable of handling uncertainties in the system with both grid-connected and islanded modes. Although there have been several successful examples illustrating the effectiveness of DRL in energy management systems, they did not take into account the inner physical characteristics of industrial facilities, and thus few can be directly utilized in industrial settings, for two reasons. First, the existing literature features relatively simple scenarios in which individual items (e.g., residential load or electric vehicle) are operated independently, whereas industrial units are highly correlated and, inherently function together; different machines in production lines must follow particular operational sequences. Second, most studies only account for electricity costs; however, while reducing energy consumption is desirable, considering the overall expenses of industrial facilities, normal production cannot be compromised to gain more revenue based on this variable alone.

Considering the aforementioned issues, this work proposes a multi-agent DRL based DR scheme for energy management of industrial facilities, to minimize electricity cost and maintain production task. There is a trend for applying multi-agent DRL to complex cooperative learning scenarios: research has shown its promise for a variety of problems, including optimizing energy sharing, allocating task resources, and controlling traffic lights and so on [29]. Multi-agent DRL algorithms are practical for solving multi-objective scheduling problems, as they are characterized by the feature of finding high-quality solutions in a reasonable time without the building of complex models, especially in the cooperative setting of industrial manufacturing processes that are sequential, interdependent and correlated. Specifically, we first formulate the industrial DR problem as a partially-observable Markov game (POMG). After that, multi-agent deep deterministic policy gradient (MADDPG) algorithm is adopted to obtain the optimal schedule for different machines. Finally, the proposed DR scheme is verified in the case of a general discrete lithium-ion battery assembly manufacturing system. To the best of our knowledge, this is the first paper to address the industrial DR problem via multi-agent DRL. The main contributions of this work are shown below:

(1) Propose an AI-based DR scheme for industrial facility

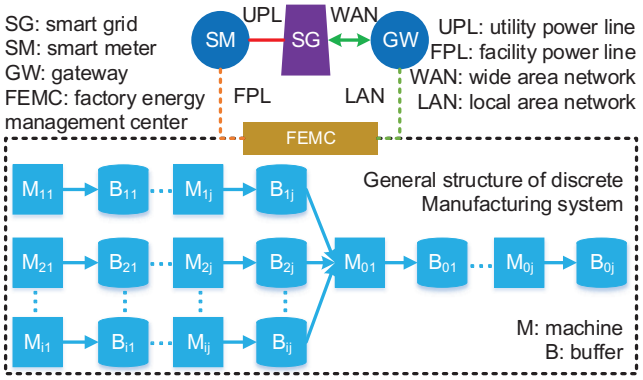

Figure 2: Industrial facility energy management system.

energy management, wherein an entire discrete manufacturing process is considered, to verify the effectiveness of the proposed DR scheme.

(2) The industrial DR problem is formulated as a POMG, and MADDPG algorithm is adopted to learn the optimal policy without requiring any system model information.

(3) The performance of the presented approach by multi-agent DRL against a mathematical model method is investigated, indicating that multi-agent DRL is a promising solution for complex industrial DR problems.

(4) Two different cases with and without AI-based DR are compared, showing that the proposed DR scheme can significantly reduce the electricity cost.

The remainder of this paper is organized as follows. Section 2 describes the problem formulation of the discrete manufacturing system. Section 3 introduces the multi-agent DRL methodology to solve the DR problem. Section 4 provides the case study and numerical results. Section 5 concludes the paper and gives an outlook on future research.

2. Problem Formulation

Conforming to the prototype of standard [30], Fig. 2 shows an energy management system for industrial facilities, including the smart grid (SG), smart meter (SM), gateway (GW), factory energy management center (FEMC), utility power line (UPL), facility power line (FPL), wide area network (WAN), and local area network (LAN). The SG and FEMC belong to the electricity supply and demand sides, respectively. Among them, the SM and GW are used as the interface to deliver energy between the UPL and FPL, and exchange information (i.e., electricity price) between the WAN and LAN. On the demand side, the LAN transmits message and the FPL distributes electricity, between different machines. The FEMC serves as the system core, to determine a working schedule for each industrial load based on the pre-installed energy management algorithm, according to the target set by a production planner and the hourly electricity price received from the GW. In this work, the proposed AI-based DR algorithm is embedded in the FEMC.

For the industrial facility, we consider a general discrete manufacturing assembly system [31], as exhibited in Fig. 2. In which, $M_{i,j}$ represents a machine, i indicates the ith serial production-line branch, and j denotes jth machine in the ith branch. $B_{i,j}$ represents the buffer used to store the products produced by machine $M_{i,j}$.

2.1. System Electricity Consumption

Generally, in the discrete manufacturing system, each machine works under "impulse mode" (i.e., operation or idle) considering the highest machine efficiency [32]. Operation means that the machine is fully operated, while idle means that the machine enters a low-power mode. Let $Z^h_{i,j}$ indicates the decision variable of machine $M_{i,j}$, i.e., $Z^h_{i,j} = 1$ if $M_{i,j}$ is operating, and $Z^h_{i,j} = 0$ if $M_{i,j}$ is idle. During hour h, each machine can only choose one working state. Thus, the energy consumption of machine $M_{i,j}$ in hour h is:

$$e^h_{i,j} = e^{op}_{i,j} \cdot Z^h_{i,j} + e^{idle}_{i,j} \cdot \left(1 - Z^h_{i,j}\right) \tag{1}$$

where $e^{op}_{i,j}$ and $e^{idle}_{i,j}$ represent the energy demands of machine $M_{i,j}$ in its operation and idle working states, respectively.

Thus, the total energy consumption of all machines during hour h is:

$$E^h = \sum_{i \in I,\ j \in J} e^h_{i,j} \tag{2}$$

$$E^h \le E_{\max} \tag{3}$$

Eq. 3 indicates that the total energy consumption during hour h should be under a threshold $E_{\max}$, which is determined by the limit of the transmission lines in the SG [33].

2.2. System Production Buffer

Between two consecutive machines, there is a buffer $B_{i,j}$ used to provide an opportunity for consociation in different pieces of equipment. The production storage of buffer $B_{i,j}$ at hour h is equal to its storage at hour $h-1$, plus the total quantity $P^h_{i,j}$ that machine $M_{i,j}$ generated and minus the total quantity $C^h_{i,j}$ that the following machine $M_{i,j}{}'$ consumed with hour h, as follows:

$$B^h_{i,j} = B^{h-1}_{i,j} + P^h_{i,j} - C^h_{i,j} \tag{4}$$

$$P^h_{i,j} = Z^h_{i,j} \cdot p^h_{i,j} \tag{5}$$

$$C^h_{i,j} = Z^h_{i,j}{}' \cdot c^h_{i,j} \tag{6}$$

$$B^{\min}_{i,j} \le B^h_{i,j} \le B^{\max}_{i,j} \tag{7}$$

where $p^h_{i,j}$ and $c^h_{i,j}$ denote the production and consumption rates of machine $M_{i,j}$ and $M_{i,j}{}'$ with operating states $Z^h_{i,j}$ and $Z^h_{i,j}{}'$, respectively. Eq. 7 indicates that the production storage of buffer $B_{i,j}$ should maintain a minimum amount of material flow, while not exceeding the maximum capacity [9].

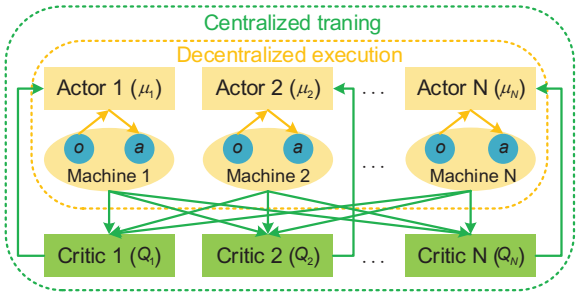

Figure 3: Multi-agent DRL with centralized training and decentralized execution.

2.3. System Objective Function

Microeconomic theory suggests that consumers will increase their demand up to the point at which the marginal benefit they derive from doing so is equal to the expenditure they have to pay [34]. For example, in the manufacturing system, a planner might not produce products if the cost required to produce them makes their sale unprofitable. In other words, the consumer will determine whether to produce, as well as when and how to produce, in such a way to maximize its profits. Thus, the objective function is as follows:

$$\max \left\{ \sum_{h=1}^{H} \left[\sum_{v \in V,\, g \in G} v \cdot g - \sum_{c \in C,\, m \in M} c \cdot m - E^h \cdot \pi^h \right] \right\} \tag{8}$$

where v is the unit value of output good g, c is the unit cost of input material m, and π^h is the electricity price at hour h.

3. Multi-Agent Deep Reinforcement Learning Methodology

In this section, we consider a partially-observable Markov game (POMG), an extension to a partially-observable MDP, to formulate the interactions among the multiple agents (each machine has an agent) of the industrial discrete manufacturing environment. After that, we develop a multi-agent deep deterministic policy gradient (MADDPG) algorithm of centralized training with decentralized execution as shown in Fig. 3, to solve the POMG. This algorithm is a variation on actor-critic policy gradient method, where the critic is augmented with extra information about the policies of other agents, while the actor only has access of local information (i.e., its own observation) to learn the optimal policy.

3.1. Partially-Observable Markov Game (POMG)

The multi-agent POMG is defined as a 4-tuple (S, O, A, R), where S represents a set of states for the entire system, describing the possible configurations of all agents; O indicates a set of observations, and each agent acquires private and limited information from the state through its own observation; A denotes a set of actions that agents can select to take; and R is the real-valued reward function. To choose actions, each agent has a policy $\mu_{i,j} : o_{i,j} \rightarrow a_{i,j}$, that maps the local observation to the action. When all the agents execute actions $\overrightarrow{\mathbf{a_{i,j}}}$, each agent gets its own immediate reward $r_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}\right)$ from the environment, and the global state s evolves to the next state s' according to the state transition $\Gamma : s \times \overrightarrow{\mathbf{a_{i,j}}} \rightarrow s'$. The agent (i, j) aims to maximize its cumulative rewards $R_{i,j} = \sum_{h=1}^{H} \gamma \cdot r_{i,j}$, where γ is a discount factor, and H is the time horizon. Note that, the state transition probability is not given in the definitions, for the reason of the POMG problem is solved by a model-free multi-agent DRL algorithm, which do not necessarily acquire the knowledge of state transition probabilities [21, 26]. Instead, the agents learn the optimal decision policy from the transition tuples obtained by interacting with the environment [23]. In the following, the detailed information of each element is given.

3.1.1. System State Formulation

The state of the FEMC includes the internal facility state and the external information state. In the facility, each agent has its own observation $o_{i,j} = e^h_{i,j} \times B^h_{i,j}$, containing the machine energy consumption state and buffer storage state. The external state covers the time-related components, i.e., the current hour of the day and electricity price received from the SG, which are relevant to the dynamic of the system. Thus, the system state s is spanned by hour h, electricity price π^h, and the observation vector $\overrightarrow{\mathbf{o_{i,j}}}$ of all agents:

$$s = h \times \pi^h \times \overrightarrow{\mathbf{o_{i,j}}} \tag{9}$$

3.1.2. System Action Formulation

The FEMC schedules the energy consumption of all machines via the binary control action $Z^h_{i,j} \in \{0, 1\}$ defined in Section 2. Therefore, the system action a is composed by the vector $\overrightarrow{\mathbf{Z^h_{i,j}}}$ of control actions determined by all agents:

$$a = \overrightarrow{\mathbf{Z^h_{i,j}}} \tag{10}$$

3.1.3. System Reward Formulation

In terms of the objective function defined in Section 2, the reward r of the system should be the value of output goods, minus the total cost of input materials and energy consumption:

$$r = \sum_{v \in V,\, g \in G} v \cdot g - \sum_{c \in C,\, m \in M} c \cdot m - E^h \cdot \pi^h \tag{11}$$

In the multi-agent cooperative setting of this work, each agent acts according to its own policy $\mu_{i,j}$ and receives a shared reward $r_{i,j} = r$ [35]. Hence, the problem of POMG is to find a policy that maximizes the expected shared return for all agents, which can be solved as a joint maximization MADDPG algorithm, as described in the next subsection.

3.2. Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Traditional DRL approaches such as deep Q-network (DQN) or deep policy gradient (DPG) are poorly suited to complex multi-agent industrial environments. One issue is that the policy of each agent changes continually as training progresses, and the industrial environment becomes non-stationary

from the perspective of any individual agent; this prevents the straightforward use of past experience replay, which is crucial for a DQN to learn stability. In addition, the DPG method typically suffers from extremely high variance with more agents [36]. The details of DQN and DDPG are given in the Appendix A and Appendix B.

Recently, MADDPG [35], an extension of DDPG to multi-agent systems, has been proposed to solve POMG. Different from optimizing a single policy network in DDPG, MADDPG inherits the decentralized actor and centralized critic framework, so that each agent maintains an individual policy network: the *actor*, $a_{i,j} = \mu_{i,j}\left(o_{i,j}|\theta_{i,j}\right)$, mapping the local observation $o_{i,j}$ to the action $a_{i,j}$ on behalf of maximizing the expected return, which is approximated as an individual Q-network $Q_{i,j}$, the *critic*. MADDPG allows each agent to receive its own personal reward signal $r_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}\right)$. Under this circumstance, the main idea of MADDPG is to learn each agent's critic network with local reward and derive the decentralized actor network using the centralized critic network.

The parameter $\phi_{i,j}$ of the centralized Q-network for each agent (i, j) is optimized as minimizing the loss $\zeta_{i,j}$:

$$\zeta_{i,j}\left(\phi_{i,j}\right) = \mathrm{E}_{s,\overrightarrow{\mathbf{a_{i,j}}},\overrightarrow{\mathbf{r_{i,j}}},s'\sim D}\left[Q^{\mu}_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}|\phi_{i,j}\right) - y_{i,j}\right]^2 \tag{12}$$

$$y_{i,j} = r_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}\right) + \gamma Q^{\mu}_{i,j}{}'\left(s', \overrightarrow{\mathbf{a_{i,j}}}'|\phi_{i,j}{}'\right)|_{a_{i,j}'=\mu_{i,j}'\left(o_{i,j}'|\theta_{i,j}'\right)} \tag{13}$$

where $Q^{\mu}_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}|\phi_{i,j}\right)$ is a centralized action-value function that takes as input the actions and state information of all agents, and outputs the Q-value for agent (i, j). $y_{i,j}$ is the target value computed by critic target network $Q^{\mu}_{i,j}{}'\left(s', \overrightarrow{\mathbf{a_{i,j}}}'|\phi_{i,j}{}'\right)$ with a slowly updating parameter $\phi_{i,j}{}'$. Additionally, the experience replay buffer D contains $\left(s, \overrightarrow{\mathbf{a_{i,j}}}, \overrightarrow{\mathbf{r_{i,j}}}, s'\right)$. It is worth noting that, in state s, agent (i, j) evaluates which action will be performed by utilizing inferred policy networks for other agents [37]. In brief, the individual critic network catches the effects of other agents' joint actions, influencing the future accumulated rewards of agent (i, j).

The parameter $\theta_{i,j}$ of policy network for agent (i, j) is optimized using a gradient ascent algorithm with its gradient computed as follows:

$$\begin{aligned}\nabla_{\theta_{i,j}}\psi\left(\theta_{i,j}\right) = \\ \mathrm{E}_{s\sim D}\left[\nabla_{\theta_{i,j}}\mu_{i,j}\left(o_{i,j}|\theta_{i,j}\right)\nabla_{a_{i,j}}Q^{\mu}_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}|\phi_{i,j}\right)\right]\end{aligned} \tag{14}$$

By iteratively updating the parameters of the actor and critic networks, the algorithm eventually produces the optimal policy network $a^*_{i,j} = \mu^*_{i,j}\left(o_{i,j}|\theta_{i,j}\right)$ [35]. It should be noticed that, the centralized Q-function is only utilized during learning, while execution is decentralized, during which each policy only requires a local observation to induce an action.

3.3. The Detailed Algorithm

The detailed algorithm is shown in Table 1. Specifically, from lines 1 to 3, the algorithm first randomly initializes the critic and actor network with weights $\phi_{i,j}$ and $\theta_{i,j}$ for each agent (i, j), and initializes each agent's target network parameters $\phi_{i,j}{}'$ and $\theta_{i,j}{}'$, as well as the replay buffer D, so that the accumulated experience is preserved for later learning.

Afterwards, the algorithm starts running with episodic iteration. At the outset of each episode, the system sets a noise process χ randomly for action exploration, and the agents begin to observe the environment initial state s (lines 5 to 6). Then, the algorithm goes in for experience accumulation from lines 7 to 11. Particularly, as a step counter h increments, each agent chooses action $a_{i,j}$ based on the current policy and the exploration noise. The primary challenge of learning in action selection is exploration. In this work, an exploration policy is constructed though appending noise sampled from a noise process χ to the actor policy $\mu_{i,j}\left(o_{i,j}|\theta_{i,j}\right)$. An Ornstein-Uhlenbeck process [38] is used to generate temporally correlated exploration for efficiency in physical control problems with inertia. The Ornstein-Uhlenbeck process models the velocity of a massive Brownian particle under the influence of friction, which results in temporally correlated values centered around 0. After executing all the selected actions, each agent gains an immediate reward $r_{i,j}$ according to Eq. 11 and the system evolves to the next state s'. In this procedure, every pair of samples $\left(s, \overrightarrow{\mathbf{a_{i,j}}}, \overrightarrow{\mathbf{r_{i,j}}}, s'\right)$ is saved in the experience buffer D.

Finally, the algorithm enters into the learning phase from lines 12 to 17. In detail, for each agent (i, j), it samples a minibatch of random K samples $\left(s^k, \overrightarrow{\mathbf{a^k_{i,j}}}, \overrightarrow{\mathbf{r^k_{i,j}}}, s^{k\prime}\right)$ from experience buffer D, and sets the target value using Eq. 13. After that, every agent updates the critic by minimizing the loss using Eq. 12, and the actor using the sampled policy gradient by Eq. 14. Since the critic being updated in Eq. 12 is also used in calculating the target value in Eq. 13, the critic update is apt to diverge. Thus, copies of the critic and actor networks are created, to calculate the target values. The weights of these target networks are then updated by having them slowly track the learning networks in line 18: $\phi_{i,j}{}' \leftarrow \tau\phi_{i,j} + (1-\tau)\,\phi_{i,j}{}'$ and $\theta_{i,j}{}' \leftarrow \tau\theta_{i,j} + (1-\tau)\,\theta_{i,j}{}'$ with $\tau < 1$. This stands for the target values are constrained to change slightly, greatly improving the learning stability. At last, the algorithm goes to the next episodic iteration and the learning process begins afresh, until the cumulative reward reaches its maximum value.

4. Case Study and Numerical Results

In this section, a case study is carried out to demonstrate the effectiveness of the proposed multi-agent DRL algorithm for energy management in a discrete manufacturing system.

4.1. Case Study

We consider a lithium-ion battery assembly system, which is a typical example of a discrete manufacturing system. Generally, a battery module has a layered structure as shown in Fig. 4, comprising side frames (SF), battery cells (BC), cooling plates (CP), intermediate frames (IF) and compression foams (CF) [39]. Its manufacture consists of four processes: assembly, saturating, formation and grading [40]. First, the components are put together into a battery module (assembly); next,

Table 1: Multi-Agent Deep Deterministic Policy Gradient (MADDPG) Algorithm

Algorithm: MADDPG for discrete manufacturing system energy management

1. Randomly initialize critic network $Q_{i,j}\left(s, \overrightarrow{\mathbf{a_{i,j}}}|\phi_{i,j}\right)$ and actor network $\mu_{i,j}\left(o_{i,j}|\theta_{i,j}\right)$ with weights $\phi_{i,j}$ and $\theta_{i,j}$ for each agent (i, j)
2. Initialize each agent target network parameters: $\phi_{i,j}{}' \leftarrow \phi_{i,j}$, $\theta_{i,j}{}' \leftarrow \theta_{i,j}$
3. Initialize replay buffer D
4. **For** episode =1 to N **do**
5. Initialize a random process χ for action exploration
6. Observe initial state s
7. **For** h = 1 to H **do**
8. For each agent (i, j), select action $a_{i,j} = \mu_{i,j}\left(o_{i,j}|\theta_{i,j}\right) + \chi$ according to the current policy and exploration noise
9. Execute actions $\overrightarrow{\mathbf{a_{i,j}}}$, observe rewards $\overrightarrow{\mathbf{r_{i,j}}}$ and next state s'
10. Store transition $\left(s, \overrightarrow{\mathbf{a_{i,j}}}, \overrightarrow{\mathbf{r_{i,j}}}, s'\right)$ in replay buffer D
11. $s \leftarrow s'$
12. **For** agent (i, j), $i \in I$, $j \in J$ **do**
13. Sample a minibatch of random K samples $\left(s^k, \overrightarrow{\mathbf{a_{i,j}^k}}, \overrightarrow{\mathbf{r_{i,j}^k}}, s^{k\prime}\right)$
14. Set $y_{i,j}^k = r_{i,j}^k + \gamma Q_{i,j}^{\mu}{}'\left(s^{k\prime}, \overrightarrow{\mathbf{a_{i,j}^k}}{}'|\phi_{i,j}{}'\right)|_{a_{i,j}^k{}'=\mu_{i,j}^k{}'\left(o_{i,j}^k{}'|\theta_{i,j}{}'\right)}$
15. Update critic by minimizing the loss:
$$\zeta_{i,j}\left(\phi_{i,j}\right) = \frac{1}{K}\sum_k\left[Q_{i,j}^{\mu}\left(s^k, \overrightarrow{\mathbf{a_{i,j}^k}}|\phi_{i,j}\right) - y_{i,j}^k\right]^2$$
16. Update actor using the sampled policy gradient:
$$\nabla_{\theta_{i,j}}\psi\left(\theta_{i,j}\right) = \frac{1}{K}\sum_k \nabla_{\theta_{i,j}}\mu_{i,j}\left(o_{i,j}^k|\theta_{i,j}\right)\nabla_{a_{i,j}}Q_{i,j}^{\mu}\left(s^k, \overrightarrow{\mathbf{a_{i,j}^k}}|\phi_{i,j}\right)$$
17. **End for**
18. Update the target network parameters for each agent (i, j):
$$\phi_{i,j}{}' \leftarrow \tau\phi_{i,j} + (1-\tau)\,\phi_{i,j}{}'$$
$$\theta_{i,j}{}' \leftarrow \tau\theta_{i,j} + (1-\tau)\,\theta_{i,j}{}'$$
19. **End for**
20. **End for**

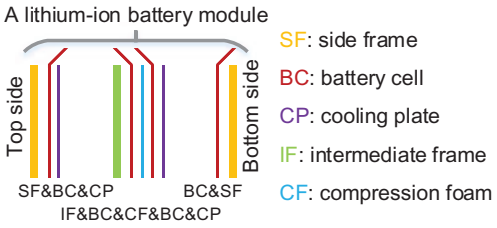

Figure 4: Structure of a lithium-ion battery module.

the module is endowed with the sufficient electrolyte (saturating); and the module is then transformed into an useable form by charging and discharging within a specific time (formation); finally, according to resistance and capacitance measurements, the battery module is rated on the basis of performance (grading). For more information, readers can refer to Appendix C [39, 40].

Fig. 5 illustrates the detailed battery module assembly process, and these processes are divided into ten tasks, where each is assigned to the appropriate working machine. The power consumption rate of each machine is listed in Table 2, together with its production rate and buffer capacity. All parameters are taken from [39, 41]. Table 3 lists the costs of input materials and the value of output good, derived from [40, 41]. The maximum power $E_{\max}$ drawn from the SG is set to 500 kW, and the hourly electricity prices on September 5, 2018, used in the simulation are obtained from [42].

4.2. Numerical Results

To start learning, we use Adam optimizer [37] for learning the actor and critic network parameters with the learning rate of 0.0001 and 0.001, respectively. Each network has two hidden layers with 64 neurons per layer, and rectified linear unit (ReLU) is used as the active function between all hidden layers. The Adam optimizer is an extension to stochastic gradient descent that has recently seen broader adoption for machine learning applications, and it proved itself as an efficient and effective optimization method in many success works. In regard

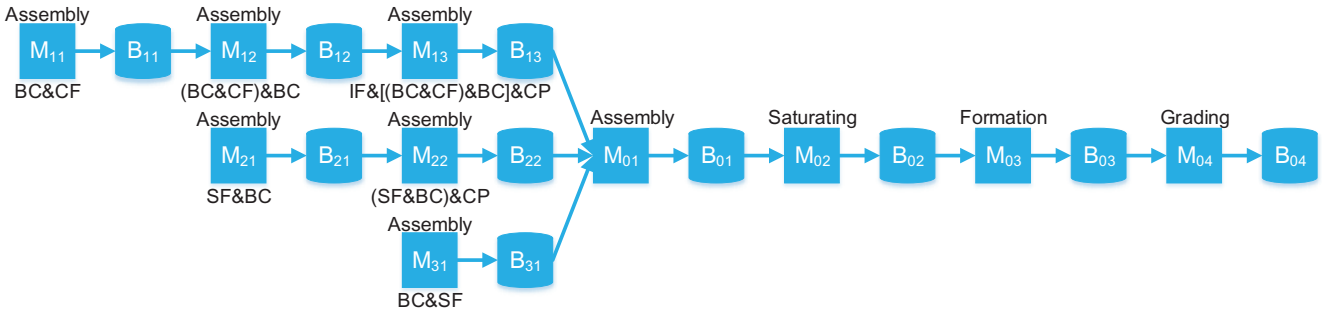

Figure 5: A lithium-ion battery module assembly process.

Table 2: Task Information

Task	Task description	Machine	Working state	Production rate (Unit/h)	Power consumption rate (kW/h)	Buffer capacity (Unit)
1	Assembling BC&CF	M_{11}	operation idle	35 0	22.8 2.28	80
2	Assembling (BC&CF)&BC	M_{12}	operation idle	32 0	20.8 2.08	80
3	Assembling IF&[(BC&CF)&BC]&CP	M_{13}	operation idle	40 0	25.6 2.56	100
4	Assembling SF&BC	M_{21}	operation idle	30 0	26.2 2.62	80
5	Assembling (SF&BC)&CP	M_{22}	operation idle	26 0	24.6 2.46	80
6	Assembling BC&SF	M_{31}	operation idle	30 0	26.2 2.62	80
7	Assembling all	M_{01}	operation idle	25 0	12.4 1.24	60
8	Saturating	M_{02}	operation idle	24 0	10.2 1.02	60
9	Formating	M_{03}	operation idle	30 0	13.6 1.36	80
10	Grading	M_{04}	operation idle	28 0	9.5 0.95	500

Table 3: Input Cost and Output Value

Item	Unit price (¢)
Input material - side frame (SF)	35
Input material - battery cell (BC)	20
Input material - cooling plate (CP)	15
Input material - intermediate frame (IF)	25
Input material - compression foam (CF)	16
Output good - lithium-ion battery module	416

of its benefits as compared to other optimizers, the readers can refer to [43]. For the critic Q-network, an L2 regularization [35] with weight decay of 0.01 is used to reduce overfitting, and the discount factor γ is set to 0.95 [44]. The final output layer of the actor policy network is a tanh layer, to bound the actions. The size of the replay buffer is 10^6 and we update the network parameters after every 100 samples added to the replay buffer. We train with a minibatch sizes of 1024, and the maximum number of iteration episodes is set to 5×10^4. For the exploration noise process, we use temporally correlated noise (realized by an Ornstein-Uhlenbeck process [38] with θ=0.15 and σ=0.2) as this is effective for exploring physical environments. The soft target update τ is assigned to 0.001.

Upon executing the simulation, the system converges to the optimal value, as shown in Fig. 6. It is clear that during the first iterations, the agents have limited knowledge on how to

Table 4: Operating Points of All Machines During Each Time Interval

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
M_{11}	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
M_{12}	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0
M_{13}	0	0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	0	0	0	1	0	0	0	0
M_{21}	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
M_{22}	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	0
M_{31}	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0
M_{01}	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0
M_{02}	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0
M_{03}	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	0
M_{04}	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1

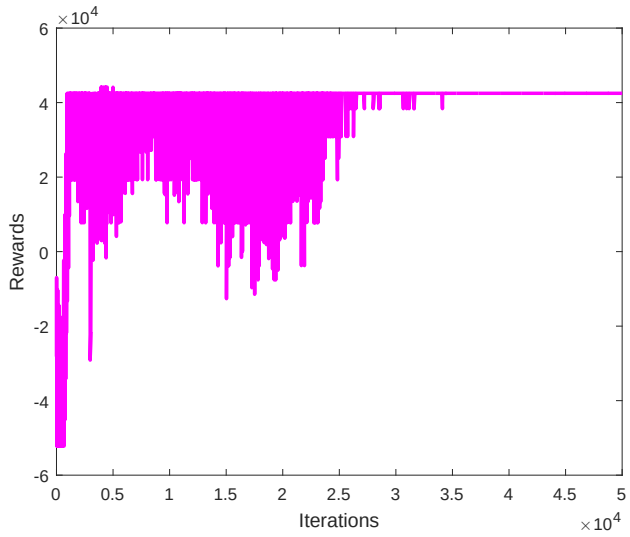

Figure 6: Cumulative rewards during the learning process.

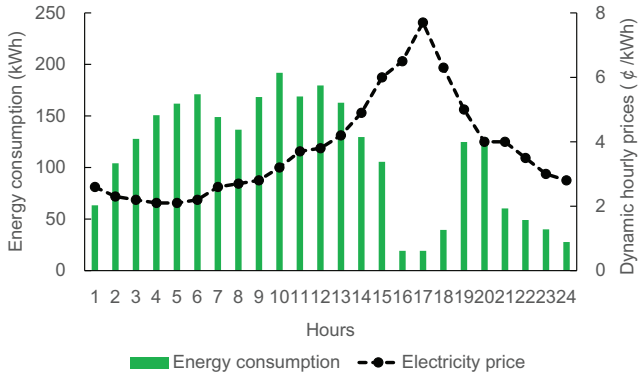

Figure 7: Aggregated energy consumption of all machines with DR.

select actions to yield high rewards, and are randomly explore action space based on the actor policy and exploration noise described in Section 3.3. However, as iteration passed, the agents gain experience by learning from episodic iterations through trial and error methodology within the multi-agent DRL algorithm. Gradually, the system converges its maximum profit at close to 3×10^4 iterations. Once the cumulative rewards become steady, the optimal policy is identified, in which the corresponding optimal control action of each agent is determined. Table 4 exhibits the operating points of all machines during each time interval, where "1" and "0" denote operation and idle, respectively. To illustrate the optimal results more clearly, the aggregated energy consumption of all machines under the proposed DR scheme is plotted in Fig. 7. We can see that the machines consume more electricity when the price is low, and then reduce their demand when it is high, such that energy consumption at peak times is avoided. More precisely, the industrial loads consume more energy during time slots 1-15 and 19-24, and consume less during 16-18. Particularly, the energy consumption of all machines is reduced to their minimum in time slots 16 and 17, since the price reaches its highest value. This not only relieves the stress on, and improves the reliability of, the SG but also reduces the electricity cost for industrial consumers.

In order to further emphasize the capability of the proposed DR scheme, Fig. 8 shows the energy consumption of the entire manufacturing process during each stage without DR, wherein the electricity price fluctuations are ignored (using fixed flat prices that equal to the average of the dynamic prices). Obviously, the system has no intention to shift or reduce its energy demand; all machines simply operated to accomplish production, sequentially. Fig. 9 compares the total electricity costs under the two cases, where the energy cost with the DR scheme (blue) is 9.8% less than that when no DR (yellow) is applied, which serves as the core motivation for industrial facilities to participate in the DR program.

To assess the performance of the proposed multi-agent DRL methodology in DR algorithm, a benchmark without learning is investigated, wherein the optimization problem is formulated under a MILP framework and solved by a commercial Gurobi solver [45]. This benchmark could be identified as an ideal strategy, since it is developed according to the ideal assumption that it has full information about the system parameters and utilizes the accurate model to maximize the objective function

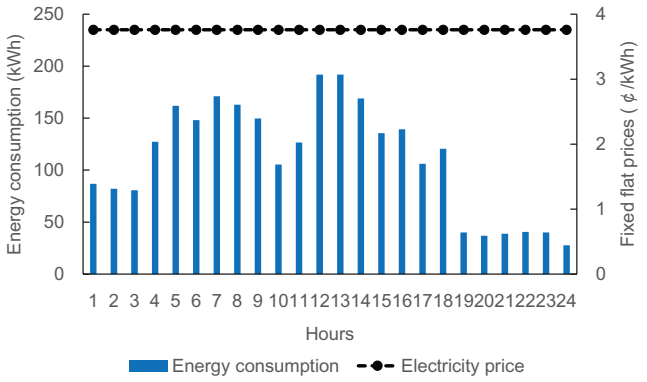

Figure 8: Aggregated energy consumption of all machines without DR.

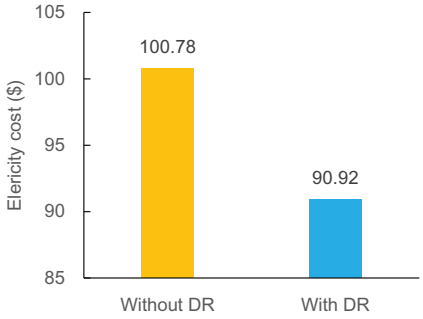

Figure 9: Electricity costs without and with DR.

defined in Eq. 8, leading to an ideal mathematically optimal result. By contrast, the multi-agent DRL strategy enabling taking advantage of learning ability to select different actions to maximize the reward defined in Section 3. Fig. 10 illustrates the total revenue procured via these two methods. It can be seen that, due to the assumption that a perfect model is available and given as a priori, the benchmark optimization achieves the best performance with maximum profit. However, this perfect model eliminates all of the uncertainties concerning industrial facility, which is not realistic in practice. For the multi-agent DRL approach, it does not work well at the initial stage, since it is engaging in its learning process by trial and error; whereas, as more experience is gained via running more iterations, the multi-agent DRL starts to autonomously adapt to the facility characteristics and adjust its policy as described in Section 3. At last, the multi-agent DRL algorithm converges toward the optimal value calculated by the benchmark. Considering that it is model-free and no need for prior domain knowledge about the features of volatile energy management situations, it is reasonable to suggest multi-agent DRL as a promising solution for complex industrial DR problems.

To evaluate the generality and flexibility of the proposed learning algorithm for energy management, we also conduct the simulation from a single day to three different days, wherein the electricity prices are obtained from ComEd [42] on the date from September 2 to September 4, 2018. Figs. 11 and 12 show the convergence of the cumulative rewards during learning process and the corresponding optimal aggregated energy

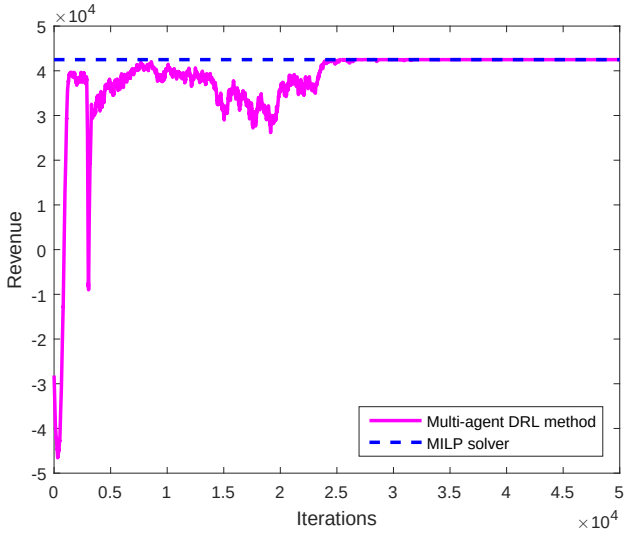

Figure 10: Total revenue procured by multi-agent DRL and Gurobi solver.

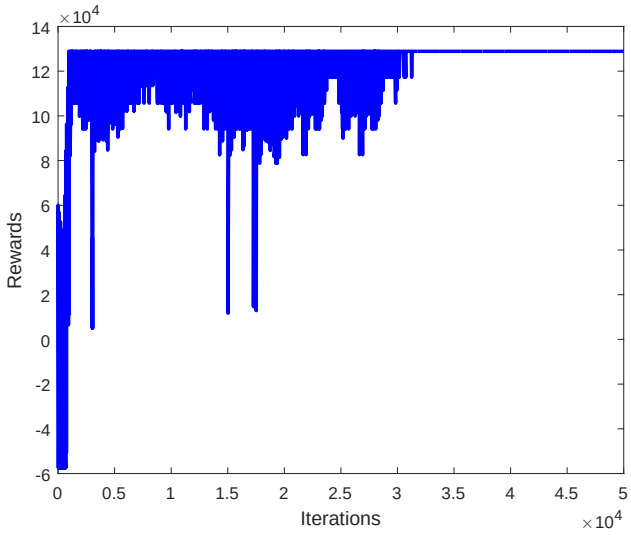

Figure 11: Convergence of the cumulative rewards from September 2 to September 4, 2018.

consumption of all machines under the DR case within these three days, respectively. As shown in Fig. 12, similar trend of energy consumption profiles with the previous single day are repeated on each of the three days that verifies the entire electricity demand of all machines is scheduled to off-peak slots to ensure the bill savings; which further enhances the simulation analysis before, indicating that this proposed DR scheme with DRL methodology can handle the industrial energy management well.

Finally, Table 5 gives the computational statistics for the case study. All the simulations are conducted on a laboratory computer with 64-bit Windows 7 OS, a 3.3 GHz 4-core i5-6600 CPU, 16 GB of RAM and an Nvidia GTX 1080 GPU. All the learning related codes are running on GPU, such as the procedure of updating actor and critic network parameters, and the

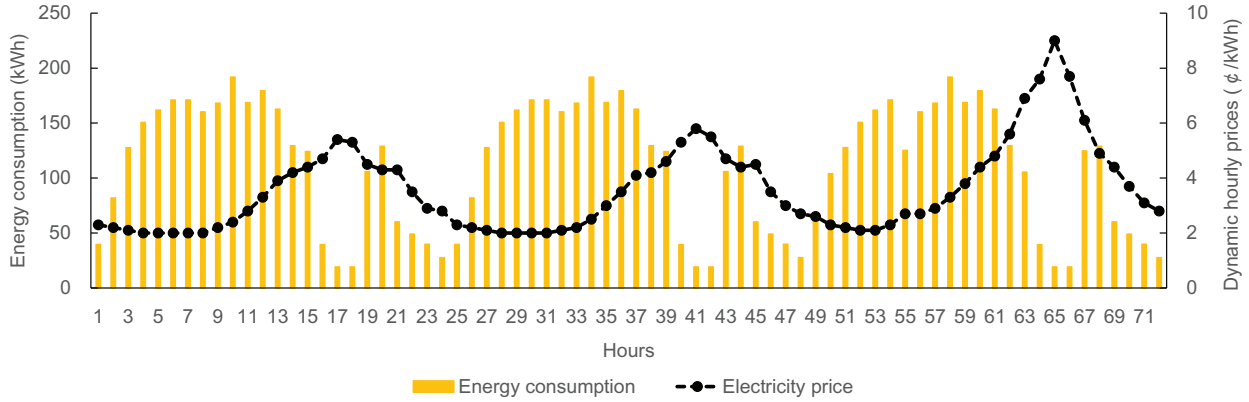

Figure 12: Aggregated energy consumption of all machines from September 2 to September 4, 2018.

action selection process. We also grab a bunch of transitions from the experience buffer and use the GPU to optimize the learning objective with Adam optimizer. The powerful GPU allows for the networks to update with gradients that have higher efficiency, speeding up the learning process [46]. The computation times for obtaining the optimal results with the multi-agent DRL and the Gurobi solver for a single day and multi-agent DRL for three continuous days are, on average about 2 min, 100s and 7 min, respectively. Such a time can fully meet the time requirement to deploy the proposed DR scheme for industrial facility energy management.

5. Conclusions and Future Work

This paper proposes a multi-agent DRL based DR scheme for discrete manufacturing systems energy management, aimed at minimizing the electricity cost and improving the grid stability. In particular, the industrial manufacturing system is initially formulated as a POMG; after that, a MADDPG algorithm is employed to obtain the optimal energy consumption for each machine. Through a case study of a lithium-ion battery assembly process, this multi-agent DPL algorithm is proved effective at managing the energy consumption without knowing the system dynamics. Compared to the case of no DR being employed, this proposed DR scheme is able to reduce the total electricity cost by 9.8%, which serves as the core motivation for the industrial consumer to participate in the DR program. Moreover, the performance of the presented approach with multi-agent DRL against the mathematical method by Gurobi solver is investigated, indicating that multi-agent DRL is a promising solution for complex industrial DR problems.

For future work, one valuable direction would be to execute the proposed DR scheme for different real-world industrial facilities to test its performance. Another desirable direction would be to standardize the communication networks of DR scheme interfaces for practical industrial applications.

Appendix A. Deep Q-network (DQN)

DQN [47], a DRL approach to solve MDP, aims to find an optimal policy υ that maximizes the expected return. DQN approximates the expected return using a deep neural network as $Q(s,a|\phi) \approx \mathrm{E}[R|s^t = s, a^t = a]$. The parameter ϕ of $Q(s,a|\phi)$ is optimized by minimizing the loss ζ defined as:

$$\zeta(\phi) = \mathrm{E}_{s,a,r,s'\sim D}[Q(s,a|\phi) - y]^2 \tag{A.1}$$

$$y = r + \gamma \max Q'(s', a'|\phi') \tag{A.2}$$

where ϕ' is the target network parameter, which is periodically updated with the most recent ϕ. D is experience replay buffer, in which (s, a, r, s') is stored for each step of each episode. Both the experience replay buffer and the target network are intended to stabilize learning [47].

Once the optimal network parameter ϕ^* is obtained, the optimal policy can be expressed as $\upsilon^*(s) = \arg\max Q(s,a|\phi^*)$.

Appendix B. Deep Deterministic Policy Gradient (DDPG)

The policy gradient method is another common choice for DRL tasks [48]. The main idea is to adjust directly the parameter θ of the policy to maximize the objective $\psi(\theta) = \mathrm{E}_{s\sim\rho^\mu, a\sim\mu_\theta}[R|s^t = s, a^t = a]$ in the direction of the performance gradient $\nabla_\theta\psi(\theta)$. DDPG [48], as an actor-critic algorithm, is aimed at deriving the deterministic policy network directly, referred to as an *actor*, $a_t = \mu(s_t|\theta)$ that maximizes the expected return, defined as:

$$\psi(\theta) = \mathrm{E}_{s\sim\rho^\mu, a\sim\mu_\theta}[R] \approx \mathrm{E}_{s\sim\rho^\mu, a\sim\mu_\theta}[Q^\mu(s,a|\phi)] \tag{B.1}$$

The Q-network $Q^\mu(s,a|\phi)$ in Eq. B.1, referred to as a *critic*, is optimized like Eq. A.1 in DQN. The parameter θ of the policy network is optimized using a gradient ascent algorithm, computed as follows:

$$\nabla_\theta\psi(\theta) = \mathrm{E}_{s\sim D}\left[\nabla_\theta\mu_\theta(s|\theta)\nabla_a Q^\mu(s,a|\phi)|_{a=\mu(s|\theta)}\right] \tag{B.2}$$

The gradient is computed using the chain rule, as the expected value of the product can be decomposed into the gradient of the policy network in relation to its parameters, and the gradient of the Q-network with respect to actions. By iteratively updating parameters for the actor and critic network, the algorithm deduces the optimal policy network $a_t^* = \mu(s_t|\theta^*)$.

Table 5: Computational Statistics for the Case Study

Approach	Hardware	Software	Computation time
Multi-agent DRL (one day)	Windows 7 OS 64-bit, 16GB RAM,	Python programming, PyCharm IDE	2 min
Gurobi solver (one day)	4-core i5-6600 CPU 3.30GHz,	C++ programming, Visual Studio IDE	100 s
Multi-agent DRL (three days)	Nvidia GTX 1080 GPU	Python programming, PyCharm IDE	7 min

Appendix C. Lithium-ion Battery Assembly System

A lithium-ion battery module usually has a hierarchical structure consisting of battery cells (BC) and ancillary members, such as side frames (SF), cooling plates (CP), intermediate frames (IF) and compression foams (CF), as shown in Fig. 4 [39]. These components are assembled or stacked together in a certain pattern, for example, IF&[(BC&CF)&BC]&CP. The details of a lithium-ion battery assembly process is illustrated in Fig. 5 [40]. And these processes can be decomposed into ten tasks, where different tasks are independently performed at the different branches, i.e., preassembling different components into subassemblies at different branches. Then, the different subassemblies are fed to an assembly station which processes the finished modules.

Table 2 lists the tasks information for subassemblies (from Task 1 to Task 6), and the final assembly together with another three processes (from Task 7 to Task 10). Specifically, the BC and auxiliary components (SF, CP, IF, and CF) are crated onto the assembly line, and are then assembled into a module by appropriate stacking machines via a series of operations. After that, each module is filled with the electrolyte, and is then clamped or sealed with end plates (Task 8). Once the module assembly and saturating are complete, the module should be put through at least one precisely controlled charge and discharge cycle to activate the working materials, transforming them into their useable form, named the formation process (Task 9). Finally, the battery modules are inspected and graded according to their properties (type, size), performances (good, damaged) and different electrochemical characteristics (capacity, voltage) by optical, electrical, ultrasonic, or mechanical sorting devices (Task 10). Upon the completion of these operations, the modules can be welded together into a battery pack.

Acknowledgements

This work was supported in part by the Fundamental Research Funds for the Central Universities under HUST Grant 2020kfyXJJS084; and in part by the National Natural Science Foundation of China under Grants 61802184 and 61702369.

References

[1] K. Wohlfarth, M. Klobasa, R. Gutknecht, Demand response in the service sector–theoretical, technical and practical potentials, Applied Energy 258 (2020) 114089.

[2] J. Wang, H. Zhong, Z. Ma, Q. Xia, C. Kang, Review and prospect of integrated demand response in the multi-energy system, Applied Energy 202 (2017) 772–782.

[3] M. Á. Lynch, S. Nolan, M. T. Devine, M. O'Malley, The impacts of demand response participation in capacity markets, Applied Energy 250 (2019) 444–451.

[4] A. A. Desta, H. Badis, L. George, Demand response scheduling in industrial asynchronous production lines constrained by available power and production rate, Applied Energy 230 (2018) 1414–1424.

[5] A. Abdulaal, R. Moghaddass, S. Asfour, Two-stage discrete-continuous multi-objective load optimization: An industrial consumer utility approach to demand response, Applied Energy 206 (2017) 206–221.

[6] E. I. Administration, International Energy Outlook, https://www.eia.gov/outlooks/ieo/pdf/industrial.pdf, [Online; accessed Septermber-2019] (2019).

[7] G. May, B. Stahl, M. Taisch, Energy management in manufacturing: Toward eco-factories of the future–a focus group study, Applied Energy 164 (2016) 628–638.

[8] X. Gong, Y. Liu, N. Lohse, T. De Pessemier, L. Martens, W. Joseph, Energy-and labor-aware production scheduling for industrial demand response using adaptive multiobjective memetic algorithm, IEEE Transactions on Industrial Informatics 15 (2) (2018) 942–953.

[9] M. Yu, R. Lu, S. H. Hong, A real-time decision model for industrial load management in a smart grid, Applied Energy 183 (2016) 1488–1497.

[10] F. Y. Xu, L. L. Lai, Novel active time-based demand response for industrial consumers in smart grid, IEEE Transactions on Industrial Informatics 11 (6) (2015) 1564–1573.

[11] A. Gholian, H. Mohsenian-Rad, Y. Hua, Optimal industrial load control in smart grid, IEEE Transactions on Smart Grid 7 (5) (2015) 2305–2316.

[12] Y. M. Ding, S. H. Hong, X. H. Li, A demand response energy management scheme for industrial facilities in smart grid, IEEE Transactions on Industrial Informatics 10 (4) (2014) 2257–2269.

[13] J. Wang, Y. Shi, Y. Zhou, Intelligent demand response for industrial energy management considering thermostatically controlled loads and evs, IEEE Transactions on Industrial Informatics 15 (6) (2018) 3432–3442.

[14] H. Golmohamadi, R. Keypour, B. Bak-Jensen, J. R. Pillai, M. H. Khooban, Robust self-scheduling of operational processes for industrial demand response aggregators, IEEE Transactions on Industrial Electronics 67 (2) (2019) 1387–1395.

[15] D. Ernst, M. Glavic, F. Capitanescu, L. Wehenkel, Reinforcement learning versus model predictive control: a comparison on a power system problem, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (2) (2008) 517–529.

[16] Y. Li, H. He, A. Khajepour, H. Wang, J. Peng, Energy management for a power-split hybrid electric bus via deep reinforcement learning with terrain information, Applied Energy 255 (2019) 113762.

[17] P. Kou, D. Liang, C. Wang, Z. Wu, L. Gao, Safe deep reinforcement learning-based constrained optimal control scheme for active distribution networks, Applied Energy 264 (2020) 114772.

[18] R. Lu, S. H. Hong, Incentive-based demand response for smart grid with reinforcement learning and deep neural network, Applied Energy 236 (2019) 937–949.

[19] J. R. Vázquez-Canteli, Z. Nagy, Reinforcement learning for demand response: A review of algorithms and modeling techniques, Applied Energy 235 (2019) 1072–1089.

[20] F. Pallonetto, M. De Rosa, F. Milano, D. P. Finn, Demand response algorithms for smart-grid ready residential buildings using machine learning models, Applied Energy 239 (2019) 1265–1282.

[21] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, J. G. Slootweg, On-line building energy optimization using deep reinforcement learning, IEEE Transactions on Smart Grid 10 (4) (2019) 3698–3708.

[22] Y. Wu, H. Tan, J. Peng, H. Zhang, H. He, Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus, Applied Energy 247 (2019) 454–466.

[23] Z. Wan, H. Li, H. He, D. Prokhorov, Model-free real-time ev charging scheduling based on deep reinforcement learning, IEEE Transactions on

Smart Grid 10 (5) (2018) 5246–5257.
[24] T. Chen, W. Su, Local energy trading behavior modeling with deep reinforcement learning, IEEE Access 6 (2018) 62806–62814.
[25] H. Hua, Y. Qin, C. Hao, J. Cao, Optimal energy management strategies for energy internet via deep reinforcement learning approach, Applied Energy 239 (2019) 598–609.
[26] B. J. Claessens, P. Vrancx, F. Ruelens, Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control, IEEE Transactions on Smart Grid 9 (4) (2016) 3259–3269.
[27] P. Zeng, H. Li, H. He, S. Li, Dynamic energy management of a microgrid using approximate dynamic programming and deep recurrent neural network learning, IEEE Transactions on Smart Grid 10 (4) (2019) 4435–4445.
[28] V.-H. Bui, A. Hussain, H.-M. Kim, Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties, IEEE Transactions on Smart Grid 11 (1) (2019) 457–469.
[29] T. T. Nguyen, N. D. Nguyen, S. Nahavandi, Deep reinforcement learning for multi-agent systems: a review of challenges, solutions and applications, arXiv preprint arXiv:1812.11794 (2018).
[30] IEC-TS-62872, Industrial-process measurement, control and automation system interface between industrial facilities and the smart grid, Standard, International Electrotechnical Commission (IEC) (December 2015).
[31] C. Ocampo-Martinez, et al., Energy efficiency in discrete-manufacturing systems: Insights, trends, and control strategies, Journal of Manufacturing Systems 52 (2019) 131–145.
[32] IEC-TR-62837, Energy efficiency through automation systems, Standard, International Electrotechnical Commission (IEC) (Septermber 2013).
[33] R. Lu, S. H. Hong, X. Zhang, A dynamic pricing demand response algorithm for smart grid: reinforcement learning approach, Applied Energy 220 (2018) 220–230.
[34] D. S. Kirschen, Demand-side view of electricity markets, IEEE Transactions on Power Systems 18 (2) (2003) 520–527.
[35] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems, 2017, pp. 6379–6390.
[36] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, S. Russell, Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4213–4220.
[37] H. Ryu, H. Shin, J. Park, Multi-agent actor-critic with generative cooperative policy network, arXiv preprint arXiv:1810.09206 (2018).
[38] Wikipedia, Ornstein–Uhlenbeck process, https://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process, [Online; accessed June-2019] (2019).
[39] S. Li, Designing productive assembly system configurations based on hierarchical subassembly decomposition with application to automotive battery packs, Doctor of Philosophy, Mechanical Engineering in University of Michigan, USA (2012).
[40] H. Heimes, A. Kampker, C. Lienemann, M. Locke, C. Offermanns, Lithium-ion battery cell production process, VDMA Battery Production, 2019.
[41] Y.-C. Li, S. H. Hong, Real-time demand bidding for energy management in discrete manufacturing facilities, IEEE Transactions on Industrial Electronics 64 (1) (2016) 739–749.
[42] ComEd, ComEd's Hourly Pricing Program, https://hourlypricing.comed.com/live-prices/, [Online; accessed June-2019] (2019).
[43] R. Sun, Optimization for deep learning: theory and algorithms, arXiv preprint arXiv:1912.08957 (2019).
[44] R. Lu, S. H. Hong, M. Yu, Demand response for home energy management using reinforcement learning and artificial neural network, IEEE Transactions on Smart Grid 10 (6) (2019) 6629–6639.
[45] Gurobi, Gurobi - The Fastest Solver, https://www.gurobi.com/products/gurobi-optimizer/, [Online; accessed June-2019] (2019).
[46] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, J. Kautz, Reinforcement learning through asynchronous advantage actor-critic on a gpu, arXiv preprint arXiv:1611.06256 (2016).
[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[48] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, 2014, pp. 387–395.