Routing Optimization and Wireless Resource Allocation in a Smart Production Network Enabled by Computing Power Network and Digital Twin

Shixun Xu, Fangmin Xu, Chenglin Zhao, Shihui Duan, Huayi Liu

***Abstract*—With virtual-real mapping, predictive maintenance, and simulation optimization capabilities, digital twin (DT) technology can enable smart production, thereby reducing production costs and improving production efficiency. However, intelligent applications provided by DT always require a substantial amount of computing power, such as job-shop scheduling (JSS). Therefore, the efficient computing and network management capabilities of the computing power network (CPN) are needed to coordinate ubiquitously deployed computing infrastructures to support DT intelligent applications. This paper proposes a smart production network that integrates DT and CPN to achieve smart production. During deployment, it is necessary to satisfy the synchronization rates of DT models and the requirements for scheduling computing and network resources. Thus, this paper proposes an improved genetic algorithm inspired by power water filling and a graph neural network (GNN)-based deep Q learning algorithm, combined with the topological transformation of CPN, to realize the joint allocation of channel and power resources and the joint management of computing and network resources. In this way, we can optimize the joint costs of system delay and energy consumption under the constraints of synchronization rates of DT models and computing network scheduling. Finally, the stability and optimization performance of the proposed algorithms are verified by comparing them with common resource allocation and routing algorithms.**

***Index Terms*—Smart production network, digital twin, computing power network, resource allocation, computing power routing, graph neural network, deep reinforcement learning, genetic algorithm.**

I. Introduction

As a cyber-physical technology [1] consisting of physical entities, virtual entities, and interactive bidirectional connections [2], [3], digital twin (DT) technology can construct a virtual twin model for a physical production system by collecting data. It can provide services for every production stage, including monitoring, estimating, predicting, optimizing, and simulating. Therefore, DT can achieve cost optimization and management throughout the entire product lifecycle [1], [2], [3], [4], making it a key enabling technology for intelligent manufacturing.

In addition, the services provided by DT are realized through various industrial intelligent applications, which require substantial computing power. For example, DT can provide job-shop scheduling (JSS) optimization services through a JSS application implemented with a genetic algorithm [5]. Solving a large-scale JSS problem with this application can take hours to a day, necessitating a significant amount of computing power from the DT model. However, traditional cloud computing and multi-access edge computing suffer from poor collaboration of computing resources [6]. As an emerging computer network architecture, the computing power network (CPN) can coordinate the deployment of ubiquitous computing, storage, and network facilities [6], [7], [8]. The CPN can provide computing power support for industrial intelligent applications while significantly improving the utilization rate of computing power and network resources by dynamically loading various industrial intelligent applications on the computing nodes in the CPN and jointly scheduling computing power and network resources in a unified manner [6], [7], [8].

This paper aims to build an industrial Internet architecture empowered by DT and CPN to realize smart production. DT and CPN have different technical requirements when implemented. On the one hand, different application scenarios will have different synchronization rate requirements for DT models [2]. In particular, some applications that require precise operation will have higher synchronization rate requirements, necessitating low-latency data collection and transmission. On the other hand, to provide computing power for industrial intelligent applications in CPN, computing power routing (CPR) is a crucial problem to be solved [6], [8]. Traditional routing problems only consider link states, such as transmission rate, and have specific routing source and destination nodes, while CPR needs to consider both link states and node states, like computing resources, without clear destination nodes. Therefore, achieving CPR requires selecting a proper computing node based on the status of computing and network resources during data routing.

In combination with the above requirements, this paper

This work was supported by the Beijing Natural Science Foundation (L234080) and the 2023 Open Topics of the Key Laboratory of Internet and Industrial Integration Innovation, Ministry of Industry and Information Technology (Research on Key Technologies for Practical Deployment of TSN Networks).

Shixun Xu, Fangmin Xu and Chenglin Zhao are with the School of Information and Communication Engineering (SICE), Beijing University of Posts and Telecommunications (BUPT), Beijing 100876, China (e-mail: XUsx0130@bupt.edu.cn; xufm@bupt.edu.cn; clzhao@bupt.edu.cn).

Shihui Duan is with the Communications Standards Research Institute, China Academy of Telecommunication Research, MIIT, Beijing 100191, China (e-mail: duanshihui@ritt.cn).

Huayi Liu is with the Department of Electronic Technology, Ministry of Industry and Information Technology, Beijing 100804, China (e-mail: liuhuayi@sohu.com).

analyzes the process of data transmission for DT model synchronization as well as the routing and computing process of the JSS task. We construct a combinatorial optimization problem to minimize the joint cost of system delay and energy consumption. The optimization problem is decomposed into two sub-problems: a wireless communication resource allocation problem and a CPR problem. This paper proposes an improved genetic algorithm to solve the wireless communication resource allocation problem and a graph neural network (GNN)-based deep reinforcement learning algorithm to solve the CPR problem.

In summary, the main contributions of this paper are as follows:

1) A smart production network enabled by DTs and the CPN is proposed.
2) The joint costs of latency and energy consumption in the smart production process are analyzed and modeled as a combinatorial optimization problem.
3) A GNN-based deep reinforcement learning algorithm is proposed to achieve the joint scheduling of computing and network resources, and an improved genetic algorithm for the joint allocation of channel and power resources. The stability and optimization performance of the proposed schemes are verified through simulation experiments.

The remainder of the paper is structured as follows. Section II discusses related work on the synchronization time of DTs and the management of computing network resources. Section III introduces the smart production network, and Section IV constructs the optimization problem. Proposed algorithms are detailed in Section V and validated in Section VI. Section VII presents conclusions and future work.

II. Related Work

Most of the existing research on digital twin deployment focuses on the design of a general deployment framework for digital twins [9], [10], the development of digital twin instances in specific fields [11], research on digital twin data security [12], [13], and the design of digital twin modeling methods [14]. However, the discussion on the synchronization requirements of digital twin models has been neglected.

In addition, Nwoke et al. discussed the use of Field Programmable Gate Array (FPGA)-based embedded digital twins to avoid the data transmission delay caused by edge or cloud-based digital twins [15]. However, due to the limited computing resources of terminal devices, it is difficult to provide diversified digital twin intelligent applications to terminal devices. Han et al. proposed a digital twin deployment framework based on edge-cloud collaboration, implementing digital twin models at different levels on the edge and cloud to reduce model deployment delay [16]. However, the improvement in delay performance is primarily due to the reduction of computing delay in model deployment, while the data transmission delay, which accounts for a significant proportion of model deployment delay, has not been optimized.

Zhang et al. used a wireless computing power network to provide computing power support for the digital twin model and adopted a cooperative game method to decide the deployment location of the digital twin model in the computing power network to reduce the data transmission delay of model synchronization [17]. However, the wireless transmission process of data was not analyzed and optimized. When the amount of data is large, merely deciding the deployment location may not meet the model synchronization rate requirements.

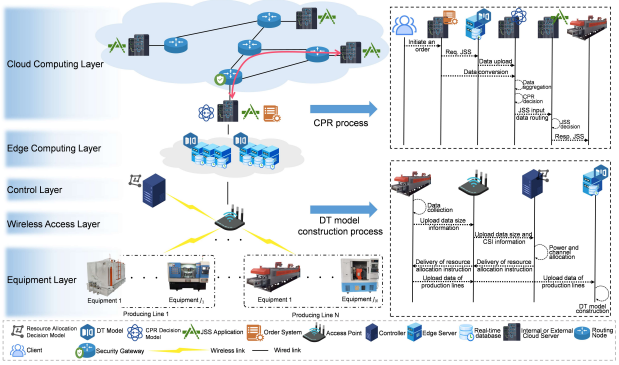

Fig. 1. System architecture and processes of DT construction and CPR.

For the joint scheduling of computing and network resources in a computing power network, most studies have achieved computing power management by offloading computing tasks through computing node matching [18], [19], [20], [21] or achieved network management through routing optimization [22], [23], [24]. However, these studies have not discussed how to achieve both computing power and network management under the new computing network management mode of the computing power network. Xie et al. combined network attributes and computing power node attributes to construct a multi-objective optimization problem and used a non-dominated sorting genetic algorithm II to search for the optimal feasible path while using a soft actor-critic algorithm to adjust the genetic algorithm hyperparameters, thus achieving computing power and network resource management [25]. In contrast, this paper leverages the graph neural network's ability to process graph-structured data to improve the generalization ability of the deep Q-learning algorithm in achieving computing power routing.

III. System Architecture and Operation Process

This section outlines the smart production network (Fig. 1) enabled by DT models and a CPN. Then, we use JSS as an example to detail the operation process of the system. The architecture is structured in layers:

1) Equipment layer: It comprises multiple production lines and collects real-time status information through sensors installed in the equipment of the production lines. Moreover, it uploads the collected data and carries out optimization instructions;
2) Wireless access layer: It consists of wireless access devices, which are connected to the edge computing layer by wire and to the equipment layer and the control layer by wireless. The wireless access devices relay equipment data and optimization instructions;

3) Edge computing layer: This layer houses an edge real-time database and edge computing nodes. The edge real-time database stores the data of production lines, and the edge computing nodes maintain production line DT models to provide intelligent services, such as fault prediction and JSS;
4) Control layer: This layer features a controller with a resource allocation decision model that makes decisions based on the production line data size information, the synchronization requirements of the production line DT models, and the channel state information (CSI) reported by the wireless access devices;
5) Cloud computing layer: The cloud computing layer consists of cloud servers located inside and outside the factory, which we call internal servers and external servers hereafter. All the cloud servers are connected through routing devices to form a cloud CPN. The cloud CPN provides computing power for users by automatically deploying corresponding applications on the cloud servers [6]. Moreover, the internal cloud server is equipped with an order management system and a computing network joint scheduling decision model, making decisions based on the size of scheduled data and the status information of computing network resources.

The following takes a JSS task as an example to illustrate the detailed operation process of the architecture, which consists of the DT model construction process and the CPR process. In this case, each cloud server is equipped with the JSS application to support the computing power requirement of the JSS task.

For the construction process of the production line DT models, sensors installed in each device collect the status information of each production line in real time. The generated data size information is then uploaded together with the CSI to the controller through the wireless access points. Next, the controller makes channel allocation and transmission power control decisions, and the production line equipment uploads the collected data to the edge real-time database according to the decision instructions made by the controller. The edge computing nodes then synchronize the production line DT models with the uploaded production line data. To meet the synchronization requirements of the DT models, the above construction process needs to be executed periodically within a short synchronization cycle.

As for the CPR process, when the order system running on the internal cloud server receives an order, the internal cloud server generates a scheduling request and converts the customer order information and the production line data into the input data for the JSS application. The customer order information includes product supply time, product type (product material, product size, product process, product packaging), product supply quantity, and quality requirements. The input data for the JSS application includes product workpieces, workpiece production processes, production equipment and total quantity of each process, processing time of each workpiece at each process equipment, and product delivery period. After the generation of the scheduling request, the internal cloud server runs the computing network joint scheduling decision model to select the appropriate service node and scheduling path for the JSS application input data. Then it forwards the JSS application input data and the JSS task request to the corresponding service node according to the scheduling decision. After receiving the request, the service node runs the JSS application to schedule production according to the input data and generates control instructions for production line optimization. Finally, the instructions are sent back to the corresponding production lines and executed to improve production efficiency.

In the actual production process, the time interval between the arrival of adjacent orders is relatively long, so before the next order arrives, the computing network joint scheduling decision and the JSS task of the current order have been completed. Therefore, it can be considered that the scheduling request of the current order does not affect the status of computing network resources when the request of the next order is scheduled.

IV. Mathematic model

As mentioned above, the synchronization period of the DT model and the arrival time interval of the order scheduling request have different time scales. It is assumed that the computing network decision period T_{JSS} consists of T synchronization periods t_s of the DT model, i.e., $T_{JSS} = Tt_s$, and the synchronization period is indexed by $\tau = 1, 2, ..., T$. Since the computing network scheduling decisions of adjacent order requests do not affect each other in the actual production process, this section, without loss of generality, analyzes the delay and energy consumption model of the system within a T_{JSS} period. Then, we construct the corresponding optimization problem and decompose it through an analysis of the independence among decision variables and constraints.

A. Delay and energy consumption model during wireless data transmission

This paper assumes there are N production lines at the industrial site, with the production line set represented by $\mathcal{PL} = \{1, ..., n, ..., N\}$. The nth production line consists of J_n devices, and the device set of the nth production line is represented by $\mathcal{D}_n = \{1, ..., j_n, ..., J_n\}$, $n \in \mathcal{PL}$. The production lines are connected to the edge server through one or more wireless access points. During the data synchronization period, data from each device on a production line is aggregated through a high-speed wired method, and then uploaded to the edge server via wireless transmission through the access points to update the DT model. The data size of the jth device in production line n during the τth synchronization period is $d^D_{n,j}(\tau)$, and the real-time status information size of the nth production line collected by sensors during the τth synchronization period is $d^{PL}_n(\tau) = \sum_{j=1}^{J_n} d^D_{n,j}(\tau)$.

Assuming non-overlapping coverage of wireless access nodes, the channel resource allocation under multiple access nodes can be simplified to the scenario of a single node. For simplicity, multiple production lines are assumed to access the edge server through a single wireless access node. Wireless transmission between the production lines and the access point

is conducted using orthogonal frequency division multiplexing (OFDM). The number of orthogonal sub-channels is C, where $C \gg N$, and each sub-channel has a constant bandwidth B. The channel set is denoted as $\mathcal{C} = \{1, ..., c, ..., C\}$. Assuming the channel coherence time exceeds the data synchronization period t_s, the channel power gain remains constant within t_s and changes between adjacent periods. Thus, the channel between the production lines and the wireless access point is quasi-static. The power gain of production line n in sub-channel c during the τth synchronization period is $h_{n,c}(\tau)$.

Suppose the channel allocation decision vector between production line n and the wireless access point generated by the controller in the τth synchronization period is $\boldsymbol{x}_n(\tau) = (x_{n,1}(\tau), ..., x_{n,c}(\tau), ..., x_{n,C}(\tau))$, where $x_{n,c}(\tau) \in \{0, 1\}$. When $x_{n,c}(\tau) = 1$, it means channel c is allocated to production line n, and $x_{n,c}(\tau) = 0$ otherwise. The set of sub-channels allocated to production line n in synchronization period τ is $\mathcal{C}_n(\tau) = \{c \mid x_{n,c}(\tau) = 1, c \in \mathcal{C}\}$. Suppose the transmission power decision vector of production line n on the allocated channel generated by the controller in synchronization period τ is $\boldsymbol{p}_n(\tau) = (P_{n,c}(\tau) \mid c \in \mathcal{C}_n(\tau))$, where $P_{n,c}(\tau)$ represents the transmission power allocated by production line n on channel c in synchronization period τ. Let $\mathcal{X}(\tau) = \{\boldsymbol{x}_n(\tau) \mid n \in \mathcal{PL}\}$ and $\mathcal{P}(\tau) = \{\boldsymbol{p}_n(\tau) \mid n \in \mathcal{PL}\}$ represent the channel allocation and power allocation set within synchronization period τ, respectively. The data transmission rate of the nth production line on the allocated channel c within synchronization period τ is:

$$r_{n,c}(\tau) = B \lg(1 + \frac{P_{n,c}(\tau) h_{n,c}(\tau)}{N_0 B}), c \in \mathcal{C}_n(\tau) \quad (1)$$

where N_0 is the power spectrum density of Gaussian white noise. The total data transmission rate of production line n within the synchronization period τ is:

$$r_n(\tau) = \sum_{c \in \mathcal{C}_n(\tau)} r_{n,c}(\tau) \quad (2)$$

The data upload delay of production line n within the synchronization period τ is:

$$t_n^{access}(\tau) = \frac{d_n^{PL}(\tau)}{r_n(\tau)} \quad (3)$$

The total data upload delay of all production lines within the synchronization period τ is:

$$t_{access}(\tau) = \sum_{n=1}^{N} t_n^{access}(\tau) \quad (4)$$

Since the data size of channel allocation instructions and power control instructions generated by the controller is much smaller than the production line data, the instruction delivery delay can be ignored. The energy consumption generated by wireless data transmission of production line n during synchronization period τ is denoted as:

$$E_n^{access}(\tau) = t_n^{access}(\tau) \sum_{c \in \mathcal{C}_n(\tau)} P_{n,c}(\tau) \quad (5)$$

The total energy consumption generated by wireless data transmission of all production lines within τ is denoted as:

$$E_{access}(\tau) = \sum_{n=1}^{N} E_n^{access}(\tau) \quad (6)$$

The edge server packages the real-time data of the production line and transmits it to the internal cloud server via high-speed wired mode to complete data aggregation. The data aggregation delay can be ignored.

B. Delay and energy consumption model during computing network scheduling

The internal cloud server, the external cloud servers, and the routing devices constitute the cloud CPN. The set of computing power nodes is represented as $\mathcal{SV} = \{1, ..., m, ..., M\}$; the set of routing nodes is represented as $\mathcal{RV} = \{1, ..., i, ..., I\}$; the set of nodes in the CPN is represented as $\mathcal{V} = \{v | v \in \mathcal{SV} \cup \mathcal{RV}\}$. Denote the set of links in the computing network as $\mathcal{ES} = \{\langle v_k, v_{k+1} \rangle | v_i, v_j \in \mathcal{V}\}$, the wired transmission distance of the link $\langle v_k, v_{k+1} \rangle \in \mathcal{ES}$ as $l_{i,j}$, and the wired transmission bandwidth available for JSS applications as $r_{i,j}$. The network topology is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{ES})$. We use set $\mathcal{F}_v \in \mathcal{V}$ to represent the neighbor nodes of node $v \in \mathcal{V}$.

After receiving the customer order, the order system running on the internal cloud server converts the order information into the available input of the JSS application. The converted data together with the corresponding production line data, constitutes the JSS application input d_{JSS}. The computing network joint scheduling decision model installed on the internal cloud server decides the appropriate routing path and service node for the production scheduling task request. The decision path vector is $\boldsymbol{y} = (v_1, ..., v_k, ..., v_K)$, $K \geq 2$, where $v_1, v_k \in \mathcal{SV}$ represent the source node (internal cloud server) and the target node respectively. For node v_k, when $2 \leq k < K$, we have $v_k \in \mathcal{RV} \cap \mathcal{F}_{v_{k-1}} \setminus \{v_{k-2}\} (\{v_0\} = \emptyset)$ to ensure the continuity of decision links and prevent link loops. In addition, when $K = 2$, $v_k = v_1$ indicates that the JSS task is calculated on the internal cloud server. Assuming the routing nodes are high-speed routes and the traffic intensities of the routing nodes are small, the queuing delay of the data at routing nodes can be ignored compared with other delays. Suppose a routing node takes a fixed routing table lookup delay t_{table} to determine the next forwarding. According to the decision vector $\boldsymbol{y}$, the number of routing nodes the data passes through during routing is:

$$q_{rn} = K - 2 \quad (7)$$

Then the forwarding delay t_{re}, transmission delay t_{trans}, and propagation delay t_{prop} of JSS application input data are denoted as:

$$t_{re} = q_{rn} t_{table} \quad (8)$$

$$t_{trans} = \sum_{k=1}^{K-1} \frac{d_n^{JSS}}{r_{k,k+1}} \quad (9)$$

$$t_{prop} = \sum_{k=1}^{K-1} \frac{l_{k,k+1}}{c_{light}} \tag{10}$$

where $c_{light} = 3 \times 10^8 m/s$ is the speed of light.

Assume the service node of the production scheduling task is cloud server m, which serves various applications in parallel, and the remaining computing frequency available for the JSS application is f_m. Denote the computing resource required to process a unit bit as w, then the task processing delay is:

$$t_{compute} = \frac{d_{JSS} w}{f_m} \tag{11}$$

When the production scheduling task is completed, the generated optimization control instructions will be returned to the production line along the original path. Since the amount of instruction data is small compared to the amount of JSS application input data, the delay in issuing the instructions can be ignored. Therefore, the total delay caused by CPR in a computing network decision cycle is:

$$t_{CPR} = t_{re} + t_{trans} + t_{prop} + t_{compute} = t_{net} + t_{compute} \tag{12}$$

According to the service frequency of the selected service node and the computing resources required for the scheduled tasks, the computing energy consumption [18] can be denoted as:

$$E_{compute} = \epsilon_m d_{JSS} w f_m^2 \tag{13}$$

where ϵ_m represents the capacitance constant of cloud server m. In addition to computing energy consumption $E_{compute}$ and wireless communication energy consumption $E_n^{access}(\tau)$, there is also wired communication energy consumption in the system [26], [27]. Assume the power consumption of the link $\langle v_k, v_{k+1} \rangle$ is $g(r_{k,k+1})$, where $g(x)$ is the power consumption function representing the power consumed when the transmission rate is x. As same as in the literature [26], [27], this paper takes $g(x) = \sigma + \mu x^2$, $\sigma, \mu, x \in \mathbb{R}^+$, where σ is the start-up power and μ is a constant representing the power consumption coefficient. Then wired communication energy consumption generated by the transmission of JSS application input data in the CPN is expressed as:

$$E_{net} = \sum_{k=1}^{K-1} \frac{g(r_{k,k+1}) d_{JSS}}{r_{k,k+1}} \tag{14}$$

Therefore, the total energy consumption generated by CPR in a computing network decision cycle is:

$$E_{CPR} = E_{compute} + E_{net} \tag{15}$$

C. Problem formulation

The resource allocation decision model and the computing network joint scheduling decision model are designed to minimize the global delay and energy consumption cost by making decisions on the joint allocation of channel and power resources for data upload of N production lines as well as on the joint scheduling of the routing path and the service node for JSS task data scheduling respectively. Since the dimensions of delay and energy consumption are not uniform and there are numerical differences in magnitude, we cannot directly combine the two into an objective function, so we normalize the delay and energy consumption:

$$t_{access}^{norm} = \frac{\sum_{\tau=1}^{T} t_{access}(\tau)}{NTt_s} \tag{16}$$

$$\begin{aligned} t_{CPR}^{norm} &= \frac{t_{net}}{t_{table} + \max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} \frac{d_{JSS}}{r_{i,j}} + \max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} \frac{l_{i,j}}{c_{light}} + \frac{t_{compute}}{\max\limits_{m \in \mathcal{SV}} \frac{d_{JSS} w}{f_m}}} \\ &= t_{net}^{norm} + t_{compute}^{norm} \end{aligned} \tag{17}$$

$$E_{access}^{norm} = \frac{1}{NT} \sum_{\tau=1}^{T} \sum_{n=1}^{N} \frac{E_n^{access}(\tau)}{P_{n,max} t_s} \tag{18}$$

$$\begin{aligned} E_{CPR}^{norm} &= \frac{E_{net}}{\max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} g(r_{i,j}) \frac{d_{JSS}}{r_{i,j}}} + \frac{E_{compute}}{\max\limits_{m \in \mathcal{SV}} \epsilon_m d_{JSS} w f_m^2} \\ &= E_{net}^{norm} + E_{compute}^{norm} \end{aligned} \tag{19}$$

where $P_{n,max}$ represents the maximum transmission power of the production line n. In summary, the normalized system delay cost and energy consumption cost within a computing network decision cycle can be expressed as:

$$t_{total}^{norm} = t_{access}^{norm} + t_{CPR}^{norm} \tag{20}$$

$$E_{total}^{norm} = E_{access}^{norm} + E_{CPR}^{norm} \tag{21}$$

Therefore the optimization problem can be modeled as:

$$\mathcal{P}_0 : \min_{\mathcal{X}, \mathcal{P}, \boldsymbol{y}} \quad \eta t_{total}^{norm} + (1 - \eta) E_{total}^{norm} \tag{22}$$

$$\begin{aligned} \text{subject to} \quad & C1 : 0 < \eta < 1 \\ & C2 : \sum_{c=1}^{C} x_{n,c}(\tau) \geq 1, \forall n \in \mathcal{PL}, \forall \tau \in \{1, ..., T\} \\ & C3 : \sum_{n=1}^{N} x_{n,c}(\tau) \leq 1, \forall c \in \mathcal{C}, \forall \tau \in \{1, ..., T\} \\ & C4 : 0 \leq \sum_{c \in \mathcal{C}_n(\tau)} P_{n,c}(\tau) \leq P_{n,max}, \\ & \qquad \forall n \in \mathcal{PL}, \forall \tau \in \{1, ..., T\} \\ & C5 : r_n(\tau) \geq \frac{d_n^{PL}(\tau)}{t_s}, \forall n \in \mathcal{PL}, \forall \tau \in \{1, ..., T\} \\ & C6 : v_k \in \mathcal{RV} \cap \mathcal{F}_{v_{k-1}} \setminus \{v_{k-2}\} (\{v_0\} = \emptyset), \\ & \qquad 2 \leq k < K \; or \; v_1 = v_k, K = 2 \\ & C7 : v_1, v_k \in \mathcal{SV} \\ & \qquad \text{and } v_1 \text{ is the internal cloud server} \end{aligned}$$

where $\mathcal{X} = \{\mathcal{X}(\tau) | \tau \in \{1, ..., T\}\}$ is the channel allocation decision set, $\mathcal{P} = \{\mathcal{P}(\tau) | \tau \in \{1, ..., T\}\}$ is the transmission power decision set, and η is the delay cost weighting coefficient. C1 is the weight constraint; C2 and C3 represent the channel allocation constraints, which respectively indicate that in the same upload cycle, each production line can be

assigned to at least one sub-channel, and each sub-channel can be assigned to at most one production line; C4 represents the power constraint of the production line; C5 ensures the data of each production line can be uploaded within a synchronization cycle, which reflects the fairness to a certain extent; C6 ensures the continuity of links and avoids link loops; C7 restricts the source node and target node of the CPR.

D. Problem decomposition

Suppose a finite number of modulation and coding combinations are used in the wireless transmission process, so the transmission power is discretized to several power levels. Then the problem $\mathcal{P}_0$ is a combinatorial optimization problem. To solve this problem, we decompose the optimization problem. It can be seen from the constraint C5 and equations (1)-(6) that there is a coupling relationship between the decision set $\mathcal{X}$ and $\mathcal{P}$. However, regardless of the constraints or optimization objectives, the decision vector $\boldsymbol{y}$ has no coupling terms with the decision set $\mathcal{X}$ and $\mathcal{P}$, so they can be separated from each other. Then the original problem $\mathcal{P}_0$ can be decomposed into sub-problems $\mathcal{P}_1$ and $\mathcal{P}_2$:

$$\begin{aligned} \mathcal{P}_1: \min_{\mathcal{X},\mathcal{P}} \quad & \eta t_{access}^{norm} + (1-\eta) E_{access}^{norm} \\ \text{subject to} \quad & C1 - C5 \end{aligned} \tag{23}$$

$$\begin{aligned} \mathcal{P}_2: \min_{\boldsymbol{y}} \quad & \eta t_{CPR}^{norm} + (1-\eta) E_{CPR}^{norm} \\ \text{subject to} \quad & C1, C6 - C7 \end{aligned} \tag{24}$$

When the optimal solutions of the two sub-problems are obtained respectively, the optimal solution of the original problem is obtained. Furthermore, for sub-problem $\mathcal{P}_1$, since constraint C5 mandates that each production line completes uploading data within every synchronization cycle, the constraints between cycles are independent. This means that the resource allocation decisions made in one synchronization cycle do not influence the constraints or decision-making process of adjacent cycles. Therefore, we face the same form of combinatorial optimization problem in each synchronization cycle, which can be solved independently in a distributed manner. So the method to solve the sub-problem $\mathcal{P}_1$ modeled in a computing network decision cycle is equivalent to the method to solve the combinatorial optimization problem $\mathcal{P}_3$ modeled in a single synchronization cycle:

$$\mathcal{P}_3: \min_{\mathcal{X}(\tau),\mathcal{P}(\tau)} \quad \eta \sum_{n=1}^{N} \frac{t_n^{access}(\tau)}{N t_s} + (1-\eta) \sum_{n=1}^{N} \frac{E_n^{access}(\tau)}{N P_{n,max} t_s} \tag{25}$$

$$\begin{aligned} \text{subject to} \quad & C1, C2': \sum_{c=1}^{C} x_{n,c}(\tau) \geq 1, \forall n \in \mathcal{PL} \\ & C3': \sum_{n=1}^{N} x_{n,c}(\tau) \leq 1, \forall c \in \mathcal{C} \\ & C4': 0 \leq \sum_{c \in \mathcal{C}_n(\tau)} P_{n,c}(\tau) \leq P_{n,max}, \forall n \in \mathcal{PL} \\ & C5': r_n(\tau) \geq \frac{d_n^{PL}(\tau)}{t_s}, \forall n \in \mathcal{PL} \end{aligned}$$

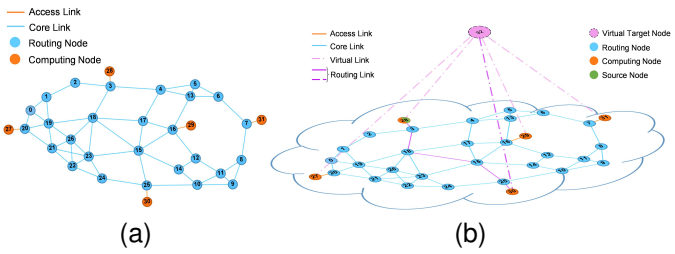

Fig. 2. Topology transformation. (a) Original Noway topology. (b) Transformed topology.

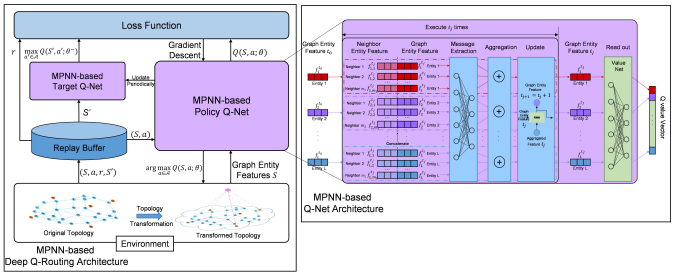

Fig. 3. Architectures of the MPNN-based Q-Net and the MPNN-based Deep Q-Routing.

V. Proposed Algorithms

As mentioned above, the original problem can be solved by independently solving sub-problems $\mathcal{P}_2$ and $\mathcal{P}_3$. This section proposes a message passing neural network (MPNN)-based Deep Q-routing algorithm for sub-problem $\mathcal{P}_2$ to achieve computing network joint scheduling, and an improved genetic algorithm for sub-problem $\mathcal{P}_3$ to achieve joint allocation of channel and power resources. The MPNN-based Deep Q-routing algorithm and the improved genetic algorithm are introduced in detail as follows.

A. MPNN-based Deep Q-routing

Unlike traditional routing problems with specified source and target pairs, the CPR problem in the CPN does not specify a clear target node. Instead, it is necessary to autonomously select a suitable computing power node based on the network and computing power node resource status, and use it as the target node to generate a routing path. This is a multi-target routing problem, and traditional routing algorithms cannot be directly applied to this problem. To achieve the joint selection of a computing power target node and a routing path, as shown in Fig. 2, this paper transforms the original topology. We first add a common virtual target node and add virtual links between the virtual target node and computing power nodes, then convert attributes of computing power nodes into those of corresponding virtual links. Therefore, this paper realizes the transformation of the multi-target routing optimization problem into a single-target routing optimization problem. After the transformation, v_{K+1} in the CPR decision vector $\boldsymbol{y} = (v_1, ..., v_K, v_{K+1})$ is the virtual target node.

In line with the inherent graph structure of the CPN topology, we use the GNN to process the graph structure representation of the transformed CPN. GNN is designed to

specifically process graph structure data [28]. In recent years, it has been widely used in the modeling and optimization of communication networks [29], [30] and has shown good generalization to network configuration and topology [30], [31]. This paper adopts a general definition framework of GNN: MPNN [32] to define the GNN structure, and uses MPNN-based Q-Net to refer to it hereafter.

1) MPNN-based Q-Net Architecture: As shown in Fig. 3, MPNN-based Q-Net mainly consists of four stages: message extraction, aggregation, update, and readout. First, in the message extraction stage, the feature vector $\boldsymbol{f}_l^{t_j}$ of each graph entity (such as a graph node or link) is concatenated with the feature vectors of its neighboring graph entities $\boldsymbol{f}_{l,m}^{t_j}$ for $m = 1, ..., m_l$. Each concatenated feature vector $(\boldsymbol{f}_{l,m}^{t_j}, \boldsymbol{f}_l^{t_j})$ is then processed by a common multilayer perceptron (MLP) to obtain a message vector with the same number of dimensions as the original graph entity feature vector. Secondly, in the aggregation stage, the message vectors of each graph entity are aggregated via element-wise summation to obtain the aggregated graph entity feature vector. This aggregated feature vector incorporates the features of neighboring entities. In the update stage, the aggregated graph entity feature vector and the corresponding input graph entity feature vector are used as inputs to a common recurrent neural network (RNN) to update the input graph entity feature vector. The updated vector $\boldsymbol{f}_l^{t_{j+1}}$ retains the same number of dimensions as the original input vector $\boldsymbol{f}_l^{t_j}$.

These three stages together form an MPNN layer, and multiple MPNN layers constitute an MPNN block. Processing through an MPNN block means that the feature vectors of graph entities are iteratively processed by these layers. As the number of MPNN layers increases, each layer aggregates and encodes information from a wider range of neighboring entities, broadening the field of view of the output graph entity features across the original graph structure.

Upon entering the readout phase, the output graph entity features are fed into a valuation network composed of another MLP. Each graph entity feature vector undergoes valuation to derive a corresponding Q-value scalar. These Q-values are sequentially concatenated to form the final Q-value vector. Unlike the approach in literature [33], where all output graph entity feature vectors are aggregated before valuation, this paper evaluates each graph entity feature vector individually using the valuation network. This method avoids information loss due to aggregation and allows for a more detailed estimation of Q-values.Different designs of neural network structures will lead to variations in the state space, as discussed in the following subsection.

The MPNN-based Q-Net takes the graph entity feature vector set as input. After the MPNN block extracts the graph entity features, the valuation network estimates the Q value of each entity in the graph. In the CPR problem addressed in this article, the graph entities are the links in the CPN. The output Q value of each entity represents the estimated long-term return expectation if the corresponding link is adopted as the next hop.

2) MPNN-based Deep Q-Routing Algorithm: We use the MPNN-based Q-Net as the Q value network in the Deep Q-learning Network (DQN) framework to achieve the optimization goal. This paper designs the DQN agent based on the structural characteristics of the MPNN-based Q-Net. As shown in Fig. 3, in the training phase of the DQN framework, the agent observes the environment to obtain the state $S \in \mathcal{S}$, where $\mathcal{S}$ is the environment state set. The policy Q network is used to estimate the Q values of the actions that can be taken in the current state and select the corresponding action $a \in \mathcal{A}$ according to the exponentially decaying ε-greedy strategy, where $\mathcal{A}$ is the action set. The environment gives a reward r as feedback based on the current state and the selected action and then transfers to the next state $S' \in \mathcal{S}$. (S, a, r, S') constitutes an experience stored in a replay buffer, whose memory size is denoted as MS. The agent and the environment repeat this interaction process continuously. When a certain amount of experience is accumulated, a small batch of experience with size BS is randomly selected from the replay buffer. (S, a) and S' are processed by the policy Q network and the target Q network respectively to obtain the estimates of Q values, $Q(S, a; \boldsymbol{\vartheta})$ and $\max_{a' \in \mathcal{A}} Q(S', a'; \boldsymbol{\vartheta}^-)$. The two together with the reward r constitute the loss function:

$$Loss(\boldsymbol{\vartheta}) = \mathbb{E}\left(\left[r + \gamma \max_{a' \in \mathcal{A}} Q(S', a'; \boldsymbol{\vartheta}^-) - Q(S, a; \boldsymbol{\vartheta})\right]^2\right) \quad (26)$$

where $\gamma \in (0, 1)$ is a discount factor indicating the importance of future rewards, a' represents the selected action in state S', and $\boldsymbol{\vartheta}$ and $\boldsymbol{\vartheta}^-$ represent the parameters of the policy Q network and the target Q network, respectively. The Adam optimizer [34] is then used to update the parameters of the policy Q network. The initial learning rate of the Adam optimizer is α. The parameters of the target Q network are periodically copied from the policy Q network, without gradient descent updates.

In summary, at each step, the policy Q network needs to process the environment state S and output the action a based on the ε-greedy strategy. Then, the agent obtains reward r, and the environment state is transferred. Next, the state set $\mathcal{S}$, action set $\mathcal{A}$, the state transfer process, and the reward function $r(S, a)$ are introduced in turn.

Since the MPNN-based Q-Net is used to process the environment state, the state in the set S needs to have a graph structure. This paper uses the real and virtual links in the transformed CPN as the graph entities. Therefore, the set of link feature vectors and the topological information after transformation are used as the environment state S. Assume the link feature $\boldsymbol{f}_l^{t_0}$ consists of H dimensions. The first three dimensions are the link length, the available bandwidth, and the computing frequency, respectively, representing the configuration of the CPN. The computing frequency dimension of real links as well as the length and available bandwidth of virtual links are set to zero. The value of the computing frequency dimension of a virtual link is the available frequency of the computing power node connected to it. The fourth, fifth, and sixth dimensions represent the routing state, marking respectively the historically selected link, the current link, and the optional neighboring links of the current link. The seventh and eighth dimensions represent the routing task, marking

TABLE I
MAJOR NOTATIONS

Notations	Description
T_{JSS}	The CPR decision period
t_s	The DT synchronization period
T	The number of DT synchronization periods in one CPR decision period
$\mathcal{PL}$, N	The set and number of production lines
$\mathcal{D}_n$, J_n	The set and number of equipment in production line n
$d_n^{PL}(\tau)$, $d_{n,j}^{D}(\tau)$	Data sizes of production line n and equipment j in production line n in the τth t_s
d_{JSS}	The data size of the input of the JSS application
$\mathcal{C}$, C, c	The set, number, and index of wireless channels
$\boldsymbol{x}_n(\tau)$, $\boldsymbol{p}_n(\tau)$	Decision vectors of channel allocation and power allocation of production line n in the τth t_s
$h_{n,c}(\tau)$, $P_{n,c}(\tau)$, $r_{n,c}(\tau)$	The power gain, allocated power, and transmission rate of production line n on channel c,
$r_n(\tau)$, $P_n(\tau)$	The transmission rate and power of production line n in the τth t_s
$t_n^{access}(\tau)$, $E_n^{access}(\tau)$	Data access delay and energy consumption of production line n in the τth t_s
$t_{access}(\tau)$, $E_{access}(\tau)$	The data access delay and energy consumption in the τth t_s
$\mathcal{SV}$, $\mathcal{RV}$, M, I	Sets and numbers of computing nodes and routing nodes
$\mathcal{V}$, $\mathcal{ES}$, $\mathcal{E}_{visited}$	Sets of nodes, links and visited links in the graph $\mathcal{G}$
$l_{i,j}$, $r_{i,j}$	The distance and available transmission rate of the link $\langle v_i, v_j \rangle$
v_1, v_K, v_{K+1}	The internal cloud server, computing power node, and virtual target node in the decision vector $\boldsymbol{y}$ of CPR
t_{re}, t_{trans}, t_{prop}	The delay of looking up routing tables, transmission, and propagation during the CPR period
f_m, ϵ_m	The available computing frequency and capacitance constant of computing power node m
f_{heavy}, w	The available computing frequency of computing power node with heavy load, number of cycles needed by processing one bit
$t_{compute}$, $E_{compute}$	The delay and energy consumption of processing JSS task in the selected computing power node
t_{net}, E_{net}	The delay and energy consumption of transmitting input data of JSS application from the source node to the selected computing power node
t_{CPR}, E_{CPR}	The delay and energy consumption of CPR
σ, μ	The start-up power and power dissipation coefficient of routing links
t_{access}^{norm}, E_{access}^{norm}, $t_{compute}^{norm}$, $E_{compute}^{norm}$, t_{net}^{norm}, E_{net}^{norm}, t_{CPR}^{norm}, E_{CPR}^{norm}	Normalized delay and energy consumption of data access, JSS task processing, JSS input data transmission, and CPR in one T_{JSS}
t_{total}^{norm}, E_{total}^{norm}	Normalized system delay and energy consumption in one T_{JSS}
$\boldsymbol{f}_l^{t_j}$, $\boldsymbol{f}_{l,m}^{t_j}$, m_l, H	Feature vectors of graph entity l and the mth neighbor entity of the graph entity l after being processed t_j times by the MPNN layer, number of neighbor entities of graph entity l, number of dimensions of the feature vector
α, γ, ε	The learning rate, reward decay coefficient, and the initial probability of taking a greedy policy
$t_{k,k+1}^{norm}$, $E_{k,k+1}^{norm}$, $t_{k,k+1}$, $E_{k,k+1}$	The delay and energy consumption of transmitting data through the link $\langle v_k, v_{k+1} \rangle$ or processing JSS task on the node v_{k+1} with and without normalization
$Cost_{k,k+1}$, $Cost_y$, $Cost_{Dijkstra}$	Cost of transmitting data through the link $\langle v_k, v_{k+1} \rangle$ or processing the JSS task on the node v_{k+1}, accumulative cost of CPR taking the policy $\boldsymbol{y}$, accumulative cost of CPR taking the policy derived from Dijkstra algorithm
X, $X_{elitism}$	Sizes of the population P, and the preserved elitism group
P_{level}, N_{bits}, P_{bits}	Number of power levels, encoding bits for channel allocation and power allocation
pr_c, pr_m, pr_i	Adaptive probabilities for crossover and mutation, probability for individual i to be selected

respectively the source node access link and the virtual links. Finally, the ninth to Hth dimensions are set to zero and reserved. Since the agent makes decisions hop by hop, the action set $\mathcal{A}$ includes all forwarding links and virtual links. To ensure the continuity of selected links and avoid link loops, the available actions for the agent only involve neighbor links of the current link but do not include the previous hop. When the routing decision is made, the environment state transfers according to the decision, that is, the historically selected link and current link dimensions of the decision link are set to one; at the same time, for the neighbor links of the decision link (excluding the previous hop), the available neighbor link dimensions are set to one; while the current link dimension of the previous link and the available neighbor link dimensions of its neighbor links are set to zero, thereby completing the state transfer.

The purpose of deep reinforcement learning is to learn an optimal strategy from the experience gained from the continuous interaction between the agent and the environment so that the agent can follow the strategy to obtain the maximum cumulative reward during the task cycle. Therefore, to solve the optimization problem $\mathcal{P}_2$, the reward function needs to be designed according to the optimization goal. Suppose the historical link set of the agent is $\mathcal{E}_{visited}$, the virtual target node is v_{target}, and the normalized delay and energy consumption caused by transmitting data through the current link $\langle v_k, v_{k+1} \rangle$ are $t_{k,k+1}^{norm}$ and $E_{k,k+1}^{norm}$ respectively:

$$t_{k,k+1}^{norm} = \begin{cases} t_{compute}^{norm}, & \text{if } \langle v_k, v_{k+1} \rangle \text{ is a virtual link,} \\ \dfrac{t_{k,k+1}}{t_{table} + \max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} \dfrac{d_{JSS}}{r_{i,j}} + \max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} \dfrac{l_{i,j}}{c_{light}}}, & \text{otherwise.} \end{cases} \tag{27}$$

$$E_{k,k+1}^{norm} = \begin{cases} E_{compute}^{norm}, & \text{if } \langle v_k, v_{k+1} \rangle \text{ is a virtual link,} \\ \dfrac{E_{k,k+1}}{\max\limits_{\langle v_i, v_j \rangle \in \mathcal{ES}} g(r_{i,j}) \dfrac{d_{JSS}}{r_{i,j}}}, & \text{otherwise.} \end{cases} \tag{28}$$

where $t_{k,k+1}$ is composed of routing table lookup delay, transmission delay, and propagation delay, and $E_{k,k+1}$ is the wired communication energy consumption generated by data transmission. Denote the cost of the link $\langle v_k, v_{k+1}\rangle$ as $Cost_{k,k+1} = \eta t^{norm}_{k,k+1} - (1-\eta)E^{norm}_{k,k+1}$. Then the cost $Cost_{\boldsymbol{y}}$ generated by data transmission and computation following a complete CPR decision $\boldsymbol{y}$ equals the objective function of $\mathcal{P}_2$. Similar to the literature [24], the reward function $r(S,a)$ or $r(S,\langle v_k, v_{k+1}\rangle)$ is defined as:

$$
\begin{aligned}
&r(S,\langle v_k, v_{k+1}\rangle)\\
&= \begin{cases}
2, & \text{if } v_{k+1} = v_{target},\\
& Cost_{\boldsymbol{y}} \le Cost_{Dijkstra};\\
-0.5 + \frac{Cost_{Dijkstra}}{Cost_{\boldsymbol{y}}}, & \text{if } v_{k+1} = v_{target},\\
& Cost_{\boldsymbol{y}} \ge Cost_{Dijkstra}\\
1 - Cost_{k,k+1}, & \text{if } v_{k+1} \text{ is closer than } v_k,\\
& \langle v_k, v_{k+1}\rangle \notin \mathcal{E}_{visited};\\
Cost_{k,k+1}, & \text{if } v_{k+1} \text{ is closer than } v_k,\\
& \langle v_k, v_{k+1}\rangle \in \mathcal{E}_{visited};\\
-1 - Cost_{k,k+1}, & \text{if } v_{k+1} \text{ is further than } v_k,\\
& \langle v_k, v_{k+1}\rangle \notin \mathcal{E}_{visited};\\
-2 - Cost_{k,k+1}, & \text{if } v_{k+1} \text{ is further than } v_k,\\
& \langle v_k, v_{k+1}\rangle \in \mathcal{E}_{visited}.
\end{cases}
\end{aligned} \tag{29}
$$

where $Cost_{Dijkstra}$ is the cost of data transmission and computation following a complete CPR decision obtained by the Dijkstra algorithm. The proposed algorithm assesses whether the current decision $\langle v_k, v_{k+1}\rangle$ brings the agent closer to the virtual target node by comparing the routing cost. Specifically, it checks if the cost of routing data from node v_{k+1} to the target node using the decision made by the Dijkstra algorithm is lower than the cost from the current node v_k to the target node. Additionally, to avoid link loops, an additional penalty is imposed on the action that results in a loop.

B. Joint Allocation through Genetic Algorithm

This paper proposes an improved genetic algorithm to jointly allocate channels and power to solve $\mathcal{P}_3$. The genetic algorithm simulates the natural selection process in biological evolution to perform a global search for solutions to an optimization problem. The proposed algorithm consists of seven operations: population initialization, fitness calculation, crossover, mutation, elitism preservation, selection, and power rearrangement inspired by a water-filling solution of a power allocation problem. Next, these seven operations are introduced respectively, and the algorithm process is summarized.

1) Population Initialization: The genetic algorithm searches for solutions based on individuals I_x (also known as chromosomes) and the population P. Each individual constitutes a possible solution to the optimization problem. The population is composed of individuals of a certain size, denoted as $P = (I_1, ..., I_x, ..., I_X)$. The initialization process initializes the individuals in the population randomly. For the joint allocation of channels and power, the chromosome designed in this paper consists of a channel allocation part and a power allocation part. The channel allocation part indicates whether each channel is allocated (0 means the channel is reserved) and to which production line it is allocated; the power allocation part indicates the power level allocated to each channel (0 means no power is allocated on this channel). Both parts use binary coding, and to avoid the Hamming cliff problem, the binary coding is converted into Gray coding. Suppose the number of channels is C, the number of production lines is N, and the number of power levels of the production lines is P_{level}. Then the gene encoding of each channel allocation decision occupies N_{bits} bits ($N_{bits} = \lceil \log_2(N+1) \rceil$, $\lceil\ \rceil$ means rounding up), and the gene encoding of the power allocation decision on each channel occupies P_{bits} bits ($P_{bits} = \lceil \log_2(P_{level}+1) \rceil$). So the total encoding length of the chromosome is $C(N_{bits} + P_{bits})$ bits. Fig. 4 shows the chromosome design when $N_{bits} = 3$ and $P_{bits} = 3$.

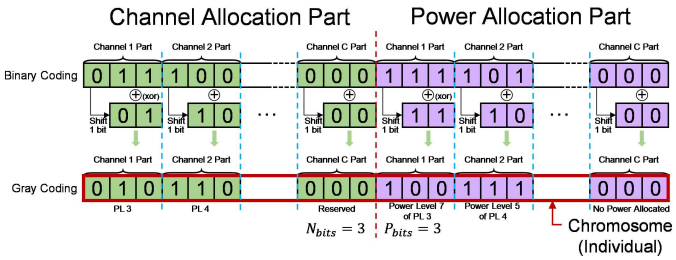

Fig. 4. Design of the chromosome.

2) Fitness Calculation: The search process of the genetic algorithm is guided by the fitness of the individuals in the population. Individuals with high fitness are obtained through continuous search. Since the genetic algorithm is used to solve the optimization problem with constraints, the inverse of the optimization objective cannot be directly used as the fitness function. This paper adopts the dynamic-objective constraint-handling method [35] to design the fitness function, which involves additional evaluation of solutions outside the feasible domain:

$$
\begin{aligned}
&F(X(\tau), \mathcal{P}(\tau))\\
&= \begin{cases}
\left(\eta \sum\limits_{n=1}^{N} \frac{t_n^{access}(\tau)}{N t_s} + \right. & \\
\quad \left. (1-\eta) \sum\limits_{n=1}^{N} \frac{E_n^{access}(\tau)}{N P_{n,max} t_s}\right)^{-1}, & \text{if constraints of}\\
& \mathcal{P}_3 \text{ are satisfied,}\\
-\sum\limits_{n=1}^{N} \frac{1}{P_{n,max}} \left[\sum\limits_{c \in \mathcal{C}_n} P_{n,c}(\tau) P_{n,max}\right]^+ + & \\
\quad \frac{t_s}{d_n^{PL}(\tau)} \left[\frac{d_n^{PL}(\tau)}{t_s} - r_n(\tau)\right]^+, & \text{otherwise.}
\end{cases}
\end{aligned} \tag{30}
$$

where $[x]^+ = x$ when $x > 0$ or $[x]^+ = 0$ otherwise.

3) Crossover: The crossover operation simulates the biological reproduction process, enabling the algorithm to explore solutions widely. This paper adopts the grouped scattered crossover [36]. Firstly, individuals in the population are sorted by fitness from smallest to largest, and the sorted population is denoted as $P' = (I'_1, ..., I'_x, ..., I'_X)$. Secondly, pairs I'_1 and $I'_{X/2+1}$, I'_2 and $I'_{X/2+2}$, and so on up to $\frac{X}{2}$ pairs are selected

for group crossover. Each group decides whether to perform a crossover operation based on the adaptive probability pr_c. pr_c is defined as follows:

$$pr_c = \begin{cases} pr_{c1}, & \text{if } F' < F_{avg} \\ pr_{c1} - \dfrac{(pr_{c1} - pr_{c2})(F' - F_{avg})}{F_{max} - F_{avg}}, & \text{otherwise.} \end{cases} \tag{31}$$

where F_{max} is the maximum fitness value in the population, F_{avg} is the average fitness value of each generation's population, F' is the higher fitness value in each crossover group, $pr_{c1}, pr_{c2} \in (0,1)$ are constants, and $pr_{c1} > pr_{c2}$. If the randomly generated value is less than pr_c, the crossover group implements the crossover operation. As depicted in Fig. 5, the crossover operation first randomly generates a $2C$-dimensional binary crossover vector. Then, the two individuals in the crossover group exchange the gene parts where the corresponding elements of the crossover vector are 1 and retain the gene parts where the corresponding elements are 0, thereby producing two new individuals after crossover.

4) Mutation: Different from crossover, which achieves large-scale search, the mutation operation can achieve local search for solutions. Similar to the crossover operation, each individual mutates according to the adaptive mutation probability pr_m, which is defined as follows:

$$pr_m = \begin{cases} pr_{m1}, & \text{if } F' < F_{avg} \\ pr_{m1} - \dfrac{(pr_{m1} - pr_{m2})(F_{max} - F)}{F_{max} - F_{avg}}, & \text{otherwise.} \end{cases} \tag{32}$$

where F is the fitness value of the individual to be mutated, $pr_{m1}, pr_{m2} \in (0,1)$ are constants with $pr_{m1} > pr_{m2}$, and other symbols are the same as above. When the generated random value is less than pr_m, the individual undergoes scattered bit flip mutation [36]. As shown in Fig. 5, a binary mutation vector with $C(N_{bits}+P_{bits})$ dimensions is randomly generated. Each individual performs bit flips on the gene bits where the corresponding mutation vector elements are 1, while retaining the gene bits where the corresponding mutation vector elements are 0, thereby obtaining a mutated new individual. Additionally, when $2^{N_{bits}} > N+1$ or $2^{P_{bits}} > P_{level}+1$, the mutated gene encoding may become meaningless. For instance, when $N = 5$ and $N_{bits} = 3$, the gene encodings $110 \oplus 11 = 101$ (Gray) and $111 \oplus 11 = 100$ (Gray) are meaningless codes. In such cases, each meaningless code is randomly mapped to a meaningful code.

5) Elitism Preservation: To prevent the loss of elite individuals with high fitness values during the search process due to crossover and mutation operations, an elitism preservation strategy is employed. The individual with the highest fitness in each generation of the population is moved to a reserved population. As the number of reserved individuals grows to the predefined size $X_{elitism}$, the fitness of any newly added elite individual is compared with that of the individual with the lowest fitness value in the reserved population, and the one with the lower fitness is removed. As the iteration count of the genetic algorithm increases, the reserved population is continually updated. Upon completion of the genetic algorithm, the individual with the highest fitness in the reserved population is selected as the final solution to the optimization problem.

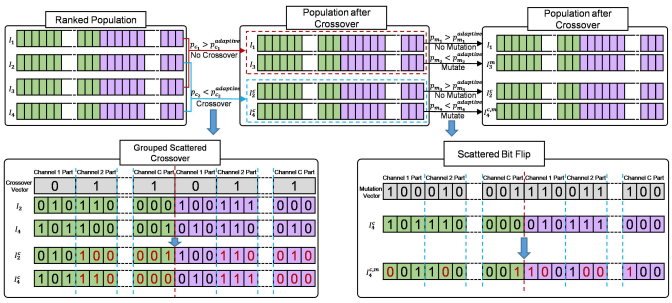

Fig. 5. Grouped scattered crossover and scattered bit flip.

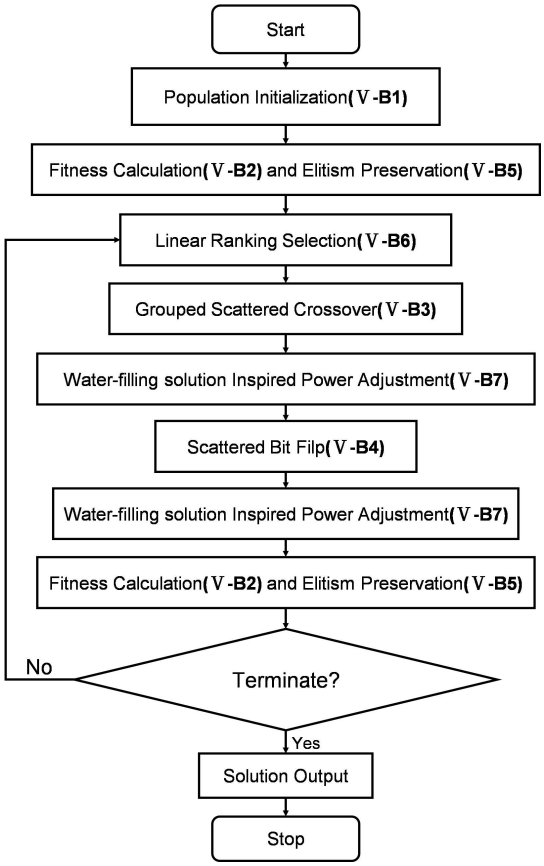

Fig. 6. Flow chart of joint allocation of channels and power through genetic algorithm.

6) Selection: The search process of the genetic algorithm progresses through selection, crossover, and mutation operations. The selection operation ensures that gene patterns with higher fitness continue to proliferate in the population. During the selection operation, the elite reserved population is sorted together with the main population by fitness in ascending order, and a linear ranking selection method is applied. The probability P_i of an individual with a ranking position i being selected into the next-generation population is calculated as

follows:

$$pr_i = pr_{min} + (pr_{max} - pr_{min})\frac{i-1}{X + X_{elitism} - 1}, \quad \forall i \in \{1, ..., X + X_{elitism}\}/\{1, X + X_{elitism}\} \quad (33)$$

where $pr_{min} = pr_1 = \frac{2}{(X+X_{elitism})(X+X_{elitism}+1)}$, and $pr_{max} = pr_{X+X_{elitism}} = \frac{2}{X+X_{elitism}+1}$.

7) Power Adjustment Inspired by the Water-Filling Solution: When searching for solutions through the genetic algorithm, given the channel allocation scheme $\mathcal{C}_n(\tau)$ and the total power value $P_n(\tau)$ for each production line represented by each individual, we can deduce the optimization objective for each production line from Equation (25):

$$\begin{aligned}
&\frac{\eta}{Nt_s}\frac{d_n^{PL}(\tau)}{\sum\limits_{c\in\mathcal{C}_n(\tau)} Blog_2\left(1+\frac{P_{n,c}(\tau)h_{n,c}(\tau)}{N_0B}\right)} + \\
&\qquad \frac{1-\eta}{NP_{n,max}t_s}\frac{P_n(\tau)d_n^{PL}(\tau)}{\sum\limits_{c\in\mathcal{C}_n(\tau)} Blog_2\left(1+\frac{P_{n,c}(\tau)h_{n,c}(\tau)}{N_0B}\right)} \\
&= \left(\frac{\eta d_n^{PL}(\tau)}{Nt_s} + \frac{(1-\eta)P_n(\tau)d_n^{PL}(\tau)}{NP_{n,max}t_s}\right) \\
&\qquad \frac{1}{\sum\limits_{c\in\mathcal{C}_n(\tau)} B\log_2\left(1+\frac{P_{n,c}(\tau)h_{n,c}(\tau)}{N_0B}\right)} \\
&= \frac{1}{\varsigma_n(\tau)r_n(\tau)}
\end{aligned} \quad (34)$$

where $\varsigma_n(\tau) = \left(\frac{\eta d_n^{PL}(\tau)}{Nt_s} + \frac{(1-\eta)P_n(\tau)d_n^{PL}(\tau)}{NP_{n,max}t_s}\right)^{-1}$. Without considering the rate constraint, the corresponding optimization problem can be expressed as:

$$\begin{aligned}
\mathcal{P}_4 : \max_{\boldsymbol{p}_n(\tau)} \quad & \varsigma(\tau)r_n(\tau) \\
\text{subject to} \quad & C6 : \sum_{c\in\mathcal{C}_n(\tau)} P_{n,c}(\tau) = P_n(\tau) \\
& C7 : P_{n,c} \geq 0, \forall c \in \mathcal{C}_n(\tau)
\end{aligned} \quad (35)$$

Then, from the Karush-Kuhn-Tucker(KKT) conditions of $\mathcal{P}_4$, we can deduce the power water-filling solution to $\mathcal{P}_4$:

$$P_{n,c}(\tau) = \left[\frac{1}{\beta} - \frac{1}{H_{n,c}(\tau)}\right]^+ \quad (36)$$

where $\beta = \frac{\lambda \ln 2}{\varsigma_n(\tau)}$ is a constant during each DT model synchronization period. From this water-filling solution, we can derive a conclusion applicable to each production line: To achieve higher transmission rates, channels with greater power gain should be allocated higher power levels. Based on this conclusion, this paper adjusts the power allocation for each production line after the crossover and mutation operations. First, given a solution represented by an individual, the allocated channels and power levels of each production line are sorted from high to low based on their power gains and power levels, respectively. Then, channels and power levels are matched one-to-one in the sorted order. Consequently, each production line assigns higher power levels to channels with greater power gains, guiding the genetic algorithm to search for solutions with higher fitness.

8) Steps of Proposed GA: Based on the previous section, the improved genetic algorithm in this paper proceeds as follows: First, initialize a population of a certain size randomly. Then, calculate the fitness values of individuals in the population using formula (30), and migrate the individual with the highest fitness value to the elite reserved group. Next, perform linear ranking selection on both the elite reserved group and the population. Subsequently, apply grouped scattered crossover to the selected population, and adjust the power allocation strategy following the method described in Section V-B7. Afterward, conduct mutation operations and further adjust the power allocation. Then recalculate the fitness values of individuals in the population, and select the elite individual to join the reserved group. Finally, if the termination condition of the algorithm is met, decode the optimal individual in the reserved group to obtain the optimized solution. Otherwise, repeat the process of selection, crossover, power adjustment, mutation, power adjustment, fitness calculation, and elitism preservation until the algorithm terminates. The algorithm flowchart is depicted in Fig. 6.

VI. Simulation and result analysis

This section conducts simulations to evaluate the performance of the two proposed algorithms for solving optimization problems $\mathcal{P}_2$ and $\mathcal{P}_3$. For the MPNN-based Deep Q-routing algorithm (MPNN-DQN), the MLP-based DQN algorithm (MLP-DQN), and the open shortest path first (OSPF) algorithm are used for comparison. For the genetic algorithm-based channel power joint allocation algorithm (JAGA), two step-by-step optimization algorithms are used for comparison.

Based on the topological transformation, MLP-DQN utilizes MLP as the Q-value network and employs the expansion vector of the CPN adjacency matrix as the environment state [37]. The action space and reward function settings are consistent with the MPNN-DQN algorithm. The OSPF algorithm selects the computing power node with the strongest computing power as the target node and chooses the path with the shortest hops as the routing path.

Regarding the step-by-step optimization algorithms for channel and power allocation, one is similar to the literature [38], which first employs the suboptimal subcarrier allocation algorithm [39] for channel allocation, followed by a meta-heuristic method for power allocation (referred to as UA-GA hereafter). Given equal power allocation, the production line with the highest priority for selecting the best channel is determined based on the urgency degree to transmit data. In this paper, urgency degree is denoted as $urgency = \frac{d_n^{JSS}(\tau)/t_s}{r_n(\tau)}$, where a higher value indicates higher priority for channel selection. After channel allocation, a genetic algorithm is used for power allocation. The genetic algorithm utilizes only the power allocation encoding part of the chromosome designed in this paper, without incorporating the water-filling solution-inspired power adjustment during the search process. The other step-by-step optimization algorithm allocates channels randomly and performs equal power allocation for the allocated channels of each production line (referred to as RA-EPA hereafter).

TABLE II
SIMULATION PARAMETERS

Parameters	Values	Parameters	Values
d_{JSS}	$[0.5, 0.9]$Gbit	η	0.8
$l_{i,j}$	$[10, 100]$m	α	0.012
$r_{i,j}$	$[12.5 : 2.5 : 30]$Mbps	γ	0.9
f_m	$[1.7 : 1 : 4.7]$GHz	ε	0.95
f_{heavy}	0.7GHz	t_J	3
w	1×10^3cycle/bit	H	10
ϵ_m	5×10^{-26}W $\cdot$ s^2/cycle^3	Memory size	12000
t_{table}	100μs	Batch size	32
σ	200W	Total episodes	7000
μ	1×10^{-4}W/Gbps^2	Number of steps to update the target net	14000
N	$[2 : 2 : 8]$	X	300
$P_{n,max}$	2W	$X_{elitism}$	4
P_{level}	10	G	300
$d_n^{PL}(\tau)$	$[0.5, 0.9]$Gbit	pr_{c1}	0.9
C	64	pr_{c2}	0.7
$C \times B$	15MHz	pr_{m1}	0.1
t_s	15s	pr_{m2}	0.003
N_0	-200dBm/Hz		

The experiment in this paper utilizes the Norway topology [40] shown in Fig. 2a as the computing network. The network topology consists of 32 vertices and 56 edges, including 5 computing nodes, 27 routing nodes, 5 access links, and 51 core links. The high-load computing node indexed as 28 serves as the source node. Parameters such as the data size of the JSS application input d_{JSS}, link lengths $l_{i,j}$, link transmission bandwidth $r_{i,j}$, and computing frequencies f_m were uniformly randomly generated within specified ranges, with discretization applied to link transmission bandwidth and computing frequency. The specific ranges for these parameters, along with constant configurations such as the computing frequency f_{heavy} of the high-load computing power node, computing resources w required per bit, energy consumption coefficient ϵ_m of computing power nodes, routing table lookup delay t_{table}, link startup power σ, and link energy consumption coefficient μ, are detailed in Table II.

During the training phase, 12 CPN parameter configurations were randomly generated as the training environment set, and cyclic training was conducted on this set. The training process spanned 7000 episodes, with the environment configuration switching every 100 episodes. Training hyperparameters are outlined in Table II. Fig. 7 illustrates the cumulative rewards obtained by each algorithm across training episodes on the training environment set. It is observed that the cumulative reward curves of MPNN-DQN and MLP-DQN steadily increase with the number of training episodes, tending to converge after approximately 2800 and 3500 episodes, respectively. The average convergence values of these two algorithms on the training environment set surpass the average cumulative rewards achieved by OSPF on the same set.

Subsequently, to compare the generalization ability of the MPNN-DQN and MLP-DQN algorithms across environmental parameter configurations, an independent and identically distributed zero-shot generalization test environment setting [41] was employed. In this setting, the random parameter configurations of the test environment set were independent and identically distributed with those of the training environment set but were not encountered during the training process. During testing, the MPNN-DQN and MLP-DQN agents, along with the OSPF algorithm, were evaluated across 20 different environments. The evaluation focused on the delay-energy joint cost, specific average delay, and average energy consumption of the three algorithms.

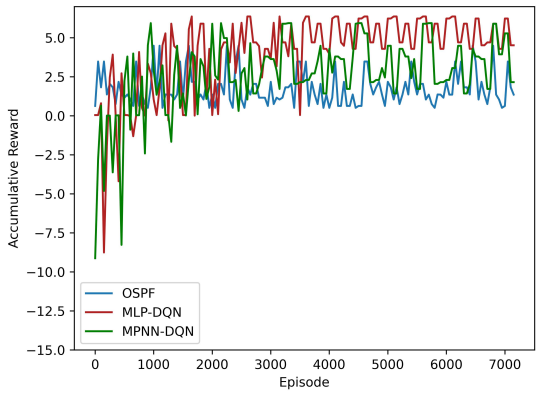

Fig. 7. Accumulative reward on training set.

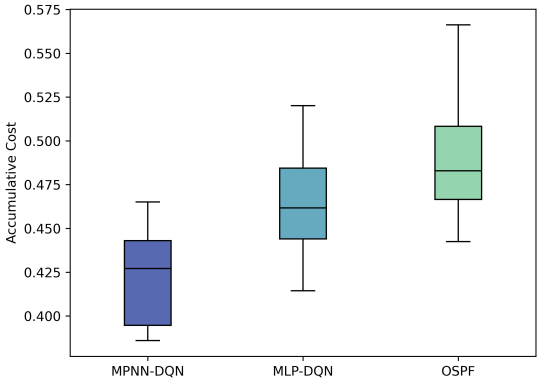

Fig. 8. Accumulative cost on testing set.

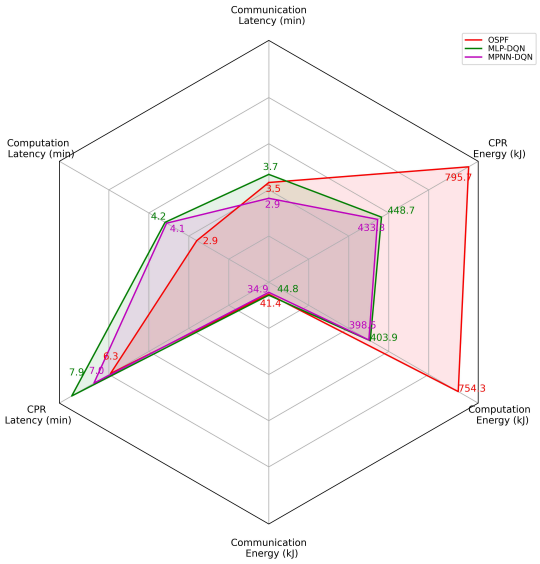

Fig. 9. Average delay and energy consumption on testing set.

The results are depicted in Fig. 8 and Fig. 9. Fig. 8 presents a box plot illustrating the cumulative cost distribution of the three algorithms on the test set. It is observed that OSPF exhibits a wider cumulative cost distribution range compared to MPNN-DQN and MLP-DQN, indicating higher variability in performance. MPNN-DQN shows a narrower distribution range than MLP-DQN, suggesting more consistent performance. Furthermore, the radar chart in Fig. 9 illustrates the average communication delay and energy consumption, average computing delay and energy consumption, and average CPR delay and energy consumption of the three algorithms on the test set. MPNN-DQN outperforms MLP-DQN across all indicators, indicating superior generalization ability across environmental configurations. Although the OSPF algorithm achieves lower average computing delay by selecting nodes with the strongest computing power during routing, this approach significantly increases CPR's computing energy consumption. In contrast, MPNN-DQN achieves a 45.5% reduction in energy consumption compared to OSPF while only incurring an 11.1% increase in delay, effectively balancing delay and energy consumption costs in CPR. In conclusion, combining the findings from Fig. 8 and Fig. 9, MPNN-DQN demonstrates superior generalization ability for environmental parameter configurations compared to MLP-DQN. It also achieves a better balance between delay and energy consumption costs, thereby achieving CPR with lower delay-energy joint costs.

For the wireless resource allocation experiment, the channel between production lines and the wireless access point was simulated using a 6-path independent Rayleigh channel. The power delay envelope decayed exponentially as $e^{-0.5l}$ with the multipath index $l \in \{1, 2, ..., 6\}$. The synchronization period of DT was 15 seconds, and the total available bandwidth was 15 MHz, divided into 64 subcarriers. The power spectral density of Gaussian white noise was set to -200 dBm/Hz, and the maximum transmit power of each production line was 2 W. Each production line had 10 discrete power levels during information transmission. The environmental parameters and hyperparameter settings for the JAGA and UA-GA algorithms are summarized in Table II.

First, to verify the effectiveness of the water-filling solution-inspired power adjustment, the performance of the JAGA algorithm with and without the power adjustment operation was compared for different numbers of production lines (2, 4, 6, and 8). Fig. 10 illustrates the convergence curves of the optimal individual fitness obtained by the search. It is evident that when the power adjustment operation is removed, the JAGA algorithm prematurely converges to a local optimum. The power adjustment operation guides the algorithm to search for better solutions.

Next, the delay-energy joint cost, specific system throughput, and energy consumption of the JAGA, UA-GA, and RA-EPA algorithms were compared across different numbers of production lines. Fig. 11 shows that as the number of production lines increases, the joint cost of each algorithm also increases. This increase occurs because with more production lines, channel resources become scarcer, and the limited maximum power per production line reduces the average

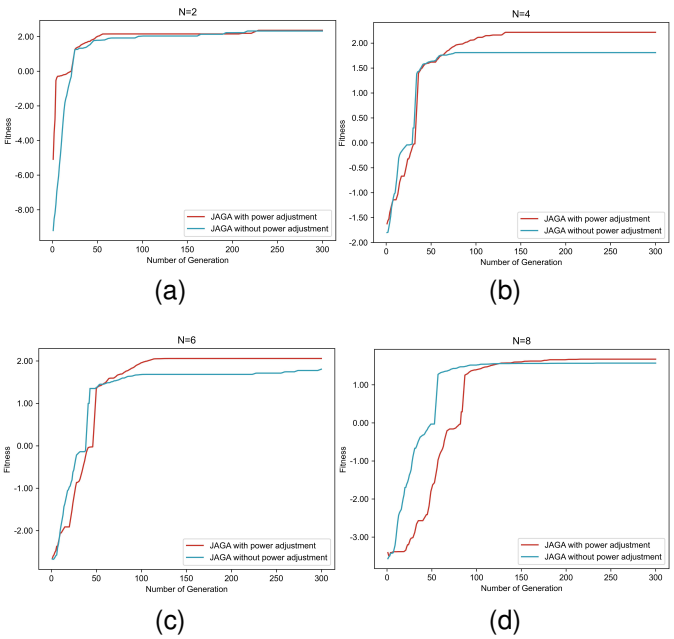

Fig. 10. Comparison of convergence performance of JAGA with and without power adjustment when (a) N=2 (b) N=4 (c) N=6 (d) N=8.

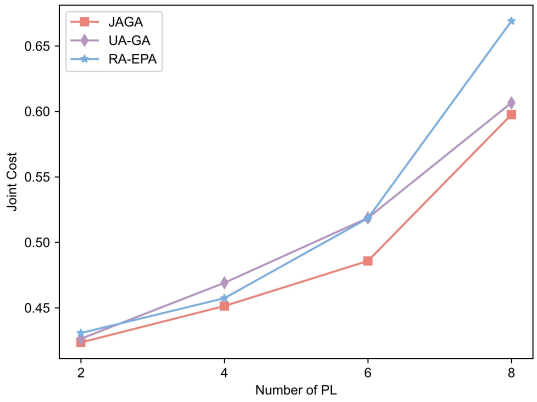

Fig. 11. Comparison of joint cost with different number of production lines.

transmission rate achievable per production line compared to scenarios with abundant spectrum resources. Consequently, the increased transmission delay leads to higher average energy consumption. Additionally, it is observed that the joint cost of the JAGA algorithm consistently outperforms that of the UA-GA and RA-EPA algorithms across all scenarios. However, the performance of the RA-EPA algorithm fluctuates: it achieves a lower joint cost than UA-GA when there are 4 production lines, matches UA-GA with 6 production lines, and exceeds UA-GA with 2 and 8 production lines. This variability in RA-EPA's performance is attributed to the randomness in channel allocation.In summary, based on the results in Fig. 11, the RA-EPA algorithm exhibits unstable performance due to its random channel allocation strategy. Conversely, the JAGA algorithm consistently demonstrates superior performance in minimizing joint cost, indicating its effectiveness in wireless resource allocation tasks.

Fig. 12 presents the system throughput and energy con-

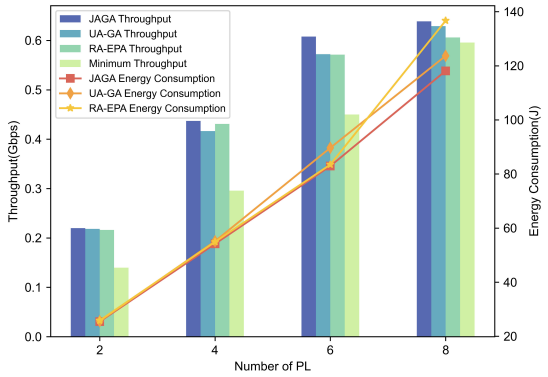

Fig. 12. Throughput and energy consumption vary with different number of production lines.

sumption of each algorithm alongside the minimum system throughput required under DT synchronization constraints. It is evident that as the number of production lines increases, the system throughput achieved by each algorithm approaches the minimum system throughput required for DT synchronization, while the system energy consumption continues to rise. This trend aligns with the analysis explaining why joint costs increase with an increasing number of production lines. Furthermore, the JAGA algorithm achieves higher system throughput than UA-GA and RA-EPA, while consuming less system energy compared to UA-GA and RA-EPA. Conversely, RA-EPA exhibits unstable performance in both system throughput and energy consumption.In summary, we conclude that the JAGA algorithm outperforms UA-GA and RA-EPA algorithms in terms of both system throughput and energy consumption, demonstrating good stability across varying conditions.

VII. Conclusion

This paper first proposes a smart production network empowered by DTs and the CPN. We then study the combinatorial optimization problem of minimizing the joint cost of system delay and energy consumption during a CPR period in the context of the JSS task. The problem is decomposed into the wireless communication resource allocation problem and the CPR problem. JAGA is proposed to solve the resource allocation problem while meeting the synchronization requirement of the production line DT model. Additionally, MPNN-DQN is introduced to achieve CPR with high generalization ability. We evaluate the optimization performance and stability of the proposed algorithms through extensive simulation experiments. The results demonstrate that JAGA achieves better performance than UA-GA and RA-EPA in reducing the delay and energy consumption of data transmission, and exhibits better stability than RA-EPA across different numbers of production lines. Moreover, MPNN-DQN shows superior generalization performance for CPN configurations compared to MLP-DQN and achieves greater energy savings at a lower delay cost compared to OSPF.

Furthermore, since the production process requires deploying multiple industrial intelligent applications and computing nodes in the computing network are often heterogeneous (such as GPU, CPU, TPU, FPGA, etc.), we need a uniform measure of heterogeneous computing power and efficient matching of different computing nodes for computing requests from various applications. Therefore, to support a variety of industrial intelligent applications in the smart production network, we plan to conduct research on scheduling computing network resources under conditions involving multiple users, multiple industrial application services, and heterogeneous computing resources in the future.

References

[1] F. Tao, Q. Qi, L. Wang, and A. Nee, "Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison," *Engineering*, vol. 5, pp. 653–661, 08 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S209580991830612X

[2] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad, H. Venkataraman, R. Trestian, and H. X. Nguyen, "Digital twins: a survey on enabling technologies, challenges, trends and future prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, pp. 1–1, 2022.

[3] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, 07 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520301072

[4] T. L. Speicher and J. F. DeFranco, "Industry 4.0 and digital twins," *Computer*, vol. 56, pp. 84–88, 2023.

[5] F. Yang, T. Feng, F. Xu, H. Jiang, and C. Zhao, "Collaborative clustering parallel reinforcement learning for edge-cloud digital twins manufacturing system," *China Communications*, vol. 19, pp. 138–148, 08 2022.

[6] Y. Sun, B. Lei, L. Juniin, H. Haonan, Z. Xing, P. Jing, and W. Wenbo, "Computing power network: a survey," *China Communications*, pp. 1–37, 01 2024.

[7] X. Tang, C. Cao, Y. Wang, S. Zhang, Y. Liu, M. Li, and T. He, "Computing power network: the architecture of convergence of computing and networking towards 6g requirement," *China Communications*, vol. 18, pp. 175–185, 02 2021.

[8] Q. Zhao, B. Lei, and M. Wei, "Survey of computing power network," *ITU Journal on Future and Evolving Technologies*, vol. 3, 12 2022.

[9] G. N. Schroeder, C. Steinmetz, R. N. Rodrigues, R. V. B. Henriques, A. Rettberg, and C. E. Pereira, "A methodology for digital twin modeling and deployment for industry 4.0," *Proceedings of the IEEE*, vol. 109, p. 556–567, 04 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9247401

[10] M.-H. Hung, Y.-C. Lin, H.-C. Hsiao, C.-C. Chen, K.-C. Lai, Y.-M. Hsieh, H. Tieng, T.-H. Tsai, H.-C. Huang, H.-C. Yang, and F.-T. Cheng, "A novel implementation framework of digital twins for intelligent manufacturing based on container technology and cloud manufacturing services," *IEEE Transactions on Automation Science and Engineering*, vol. 19, pp. 1614–1630, 07 2022.

[11] *Design and Application of Flexible Manufacturing Unit for Chuck Parts Based on Digital Twin*, 2023 3rd International Conference on Digital Society and Intelligent Systems (DSInS). IEEE, 2023.

[12] *Data Security Management Framework for Digital Twins of Industrial Pipeline*, 2021 International Conference on Maintenance and Intelligent Asset Management (ICMIAM). IEEE, 2021.

[13] C. Zhang, G. Zhou, H. Li, and Y. Cao, "Manufacturing blockchain of things for the configuration of a data- and knowledge-driven digital twin manufacturing cell," *IEEE Internet of Things Journal*, vol. 7, pp. 11 884–11 894, 12 2020.

[14] H. Zhang, Q. Qi, and F. Tao, "A multi-scale modeling method for digital twin shop-floor," *Journal of Manufacturing Systems*, vol. 62, pp. 417–428, 01 2022.

[15] *FPGA-Based Digital Twin Implementation for Power Converter System Monitoring*, 2023 IEEE 3rd International Conference on Digital Twins and Parallel Intelligence (DTPI). IEEE, 2023.

[16] *Cloud-Edge Collaborative-Based Digital Twin System for Hardware Limited IIoT Scenario*, 2023 IEEE Smart World Congress (SWC). IEEE, 2023.

[17] Y. Zhang, H. Zhang, Y. Lu, W. Sun, L. Wei, Y. Zhang, and B. Wang, "Adaptive digital twin placement and transfer in wireless computing power network," *IEEE internet of things journal*, pp. 1–1, 01 2024.

[18] H. Liao, Z. Zhou, N. Liu, Y. Zhang, G. Xu, Z. Wang, and S. Mumtaz, "Cloud-edge-device collaborative reliable and communication-efficient digital twin for low-carbon electrical equipment management," *IEEE Transactions on Industrial Informatics*, vol. 19, pp. 1715–1724, 02 2023.
[19] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "Meets: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, p. 4076–4087, 10 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8382251/
[20] *Dynamic Computation Offloading and Resource Allocation for Multi-User Mobile Edge Computing*, GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020.
[21] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, pp. 881–892, 09 2021.
[22] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, pp. 64–69, 05 2019.
[23] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 19, pp. 918–953, 01 2017.
[24] S. S. Bhavanasi, L. Pappone, and F. Esposito, "Dealing with changes: Resilient routing via graph neural networks and multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 20, pp. 2283–2294, 09 2023.
[25] Y. Xie, X. Huang, J. Li, and T. Liu, "Computing power network: Multi-objective optimization-based routing," *Sensors*, vol. 23, pp. 6702–6702, 07 2023.
[26] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, pp. 107–112, 01 2014.
[27] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu, "Greendcn: a general framework for achieving energy efficiency in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 4–15, 04 2013.
[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, pp. 61–80, 01 2009.
[29] Y. Shen, J. Zhang, S. Song, and K. B. Letaief, "Graph neural networks for wireless communications: from theory to practice," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2022. [Online]. Available: https://arxiv.org/pdf/2203.10800.pdf
[30] J. Suarez-Varela, P. Almasan, M. Ferriol-Galmes, K. Rusek, F. Geyer, X. Cheng, X. Shi, S. Xiao, F. Scarselli, A. Cabellos-Aparicio, and P. Barlet-Ros, "Graph neural networks for communication networks: Context, use cases and opportunities," *IEEE Network*, vol. 37, pp. 146–153, 2022.
[31] *Challenging the Generalization Capabilities of Graph Neural Networks for Network Modeling*. Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos, 2019.
[32] *Neural Message Passing for Quantum Chemistry*, International Conference on Machine Learning. PMLR, 2017.
[33] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194, 2022.
[34] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.
[35] H. Lu and W. Chen, "Dynamic-objective particle swarm optimization for constrained optimization problems," *Journal of Combinatorial Optimization*, vol. 12, pp. 409–419, 09 2006.
[36] *Enhanced Genetic Algorithm for Optimized Classification*. 2020 International Conference on Promising Electronic Technologies (ICPET), 2020.
[37] S. Q. Jalil, M. Husain Rehmani, and S. Chalup, "Dqr: Deep q-routing in software defined networks," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 07 2020.
[38] *Fair Adaptive Resource Allocation in NC-OFDM Based Cognitive Radio System*, 2010 International Conference on Wireless Communications & Signal Processing (WCSP). IEEE, 2010.
[39] Z. Shen, "Multiuser resource allocation in multichannel wireless communication systems," *The University of Texas at Austin*, 2006.
[40] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0-survivable network design library," *Networks*, vol. 55, pp. 276–286, 2009.
[41] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, pp. 201–264, 07 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

Shixun Xu received the B.E. degree in electronic information science and technology from Shanxi University, Shanxi, China, in 2022. He is currently pursuing the M.E. degree with the Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Post and Telecommunications, China. His research interests include edge computing, digital twins and deep reinforcement learning.

Fangmin Xu received the M.S. and Ph.D. degrees in communication engineering from the Beijing University of Posts and Telecommunication (BUPT), China, in 2003 and 2008, respectively. He is currently an Associate Professor with the School of Information and Communication Engineering, BUPT, China. From 2008 to 2014, he was with Samsung Electronics, where he actively contributed to 3GPP LTE/LTE-A and IEEE 802.16m. He has authored two books, 20 peer-reviewed international research papers, and 50 standard contributions and the Inventor of 15 is sued or pending patents among which four have been adopted in the specifications of 4G (3GPP LTE/LTE-A and IEEE 802.16m) standards. His research interests include advance technologies in wireless networks, especially the IoT field.

Chenglin Zhao received the bachelor's degree in radio-technology from Tianjin University, in 1986, and the master's degree in circuits and systems and the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications, Beijing, China, in 1993 and 1997, respectively. He serves as a Professor with BUPT. His research interests include emerging technologies of short-range wireless communication, cognitive radios, and industrial internet.

Shihui Duan received the B.S. degree in radio engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1996, the M.A. degree in communication engineering from the Research Institute of Telecommunication Science and Technology, Beijing, China, in 1999, and the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, in 2009. He is currently a Senior Engineer with the Key Laboratory of Internet and Industry Integration, which belongs to the China Academy of Telecommunication Research, Ministry of Industry and Information Technology, China. His current research interests include networked control systems, industrial network architecture, and cyber-physical systems.

Huayi Liu is currently with the Electronic Information Department of the Ministry of Industry and Information Technology, where he engages in policy research and formulation, industry management, and project management. He has co-authored more than ten national industry standards and holds an invention patent for ”A Method and Device for Evaluating the Performance of Smart TVs.” His research interests include digital transformation and the development of testing platforms for emerging technologies.