Journal of Network and Computer Applications 217 (2023) 103697

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Collective reinforcement learning based resource allocation for digital twin service in 6G networks

Zhongwei Huang [a,e], Dagang Li [a,b,*], Jun Cai [c], Hua Lu [d]

[a] Macau University of Science and Technology, Macao, 999078, China
[b] Zhuhai-M.U.S.T. Science and Technology Research Institute, Zhuhai, 519031, China
[c] Guangdong Polytechnic Normal University, Guangzhou, 510663, China
[d] Guangdong Communication & Network Institute, Guangzhou, 510663, China
[e] Qianxing Intelligence (Zhuhai) Technology Co., Ltd., Zhuhai, 519000, China

ARTICLE INFO

Keywords:
Digital twin
Resource allocation
Internet of Things
Collective reinforcement learning

ABSTRACT

The 6th generation (6G) mobile communications technology will realize the interconnection of humans, machines, things as well as virtual space. The development of digital twins (DTs) and 6G has accelerated the Internet of Things (IoT) in an unprecedented way. The combination of DTs and edge intelligence (EI) enables powerful digital space synchronized with the real world constructed in the intelligent edge, bringing real-time, and adaptive services delivery of IoT. However, the dynamic features and heterogeneous resources in 6G-enabled IoT make the resource allocation for computation-intensive and delay-sensitive DTs services more challenging. In this paper, we first define the DTs implementation process as a DT service function chain (DTSFC) and address the resource allocation problem of DTs-empowered networks in form of dynamic DTSFCs orchestration. We further propose a novel collective reinforcement learning (CRL) method which is inspired by human collaboration, to realize the effective resource allocation of DTSFCs. Numerical results verify that the proposed CRL algorithm improves the learning efficiency and generalization ability compared with the benchmarks.

1. Introduction

Recent advances in digital twin (DT) and 6th generation (6G) mobile communications technology paved the way for the Internet of things (IoT). Digital twins (DTs) enable near-instant connectivity between the physical and virtual worlds, simulate the running rules of physical objects, and help them be practiced in the real world efficiently and cost-effectively (Tao et al., 2019). In particular, DTs are applicable to these scenarios with "three highs" (high risk, high cost, high pollution) and "four difficulties" (difficult to see, difficult to run, difficult to enter, difficult to reproduce). To support the instant response, the DTs synchronized with the real world are constructed in the network edge by real-time perceiving and modeling of the physical entities, connecting all the elements of the IoTs including users, machines, vehicles, and their connected networks, etc. As shown in Fig. 1, a range of IoT use cases (Wu et al., 2021), such as manufacturing, smart transportation, and smart cities benefit from DTs to customize diversified services for users.

The early DT model contained three dimensions, namely physical equipment, virtual model, and connections between them. Tao et al. (2018) extended the three-dimensional model to five dimensions, i.e., $DT = \{Pe, Ve, Ss, Dd, Cn\}$, where Pe represents a physical entity, Ve represents a virtual equipment, Ss represents the DT service, Dd represents DT data, and Cn represents the connection between various parts. In addition, Ss consists of {Function, Input, Output, Quality, State}, Ss can be scheduled to meet the demands of the PE and VE (Tao et al., 2018). As shown in Fig. 1, the data collected from the physical world will be processed in DT to realize the network modeling, forecasting, and optimization, and the optimized policy in DT will be returned back to the physical network for execution. This modeling and optimizing processes in DT can be regarded as the DT services, and the corresponding functions as well as their sequences are constructed in DT service to realize the specific optimization. For example, the DT service of the manufacturing line maybe includes monitoring and assessment functions, while the intelligent transportation DT requires monitoring, optimization, and control functions. We summarized the commonly used functions in DT service and listed in Table 1 (Glatt et al., 2021).

In recent years, network function virtualization (NFV) technology has emerged as a promising technology to provide flexible service

* Corresponding author at: Macau University of Science and Technology, Macao, 999078, China.
E-mail addresses: 2009853via30001@student.must.edu.mo (Z. Huang), dagang.li@ieee.org (D. Li), caijun@gpnu.edu.cn (J. Cai), luhua@gdcni.cn (H. Lu).

https://doi.org/10.1016/j.jnca.2023.103697
Received 22 February 2023; Received in revised form 16 May 2023; Accepted 28 June 2023
Available online 3 July 2023

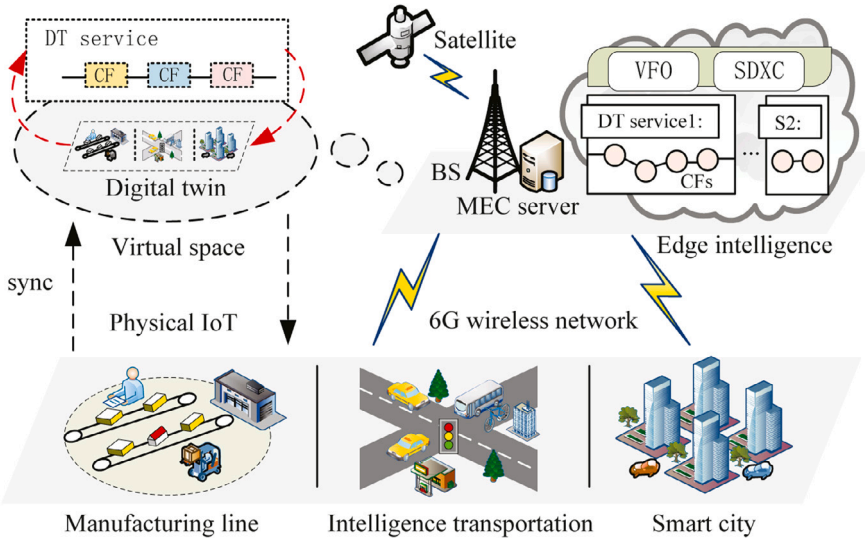


Fig. 1. The architecture of digital twin empowered IoT.

Table 1
The commonly used DT functions, applied scenario, and benefits are partially summarized from the literature (Glatt et al., 2021), and we expanded the table based on the real-world case.

DT functions	Applied scenarios	Benefits
Simulation function	• virtual testing • process planning • potential risks avoidance	Reduce the number of physical experiments, product design cycles, testing costs, and risks and mistakes
Monitoring function	• operation monitoring • fault diagnosis • status monitoring • security monitoring	Identify defects, locate faults, visualize information, and ensure life safety
Assessment function	• status assessment • performance assessment	Realize advance forecast, and guide decision
Prediction function	• failure prediction • life prediction • quality prediction • behavior prediction • performance prediction	Reduce downtime, avoid catastrophic failure, and improve product quality
Optimization function	• design optimization • configuration optimization • performance optimization • energy optimization • process optimization • structure optimization	Improve product development, provide system efficiency, save resources, reduce energy consumption, improve user experience
Controlling function	• operation control • remote control • collaborative control	Improve operational accuracy, adapt to environmental changes, improve flexibility, and real-time corresponding control

provision and efficient resource management for the 5G and its beyond networks. Based on NFV, IoT users define a series of virtual network functions (VNFs) of a service function chain (SFC) to flexibly handle massive heterogeneous IoT services (Fu et al., 2019). After decades of development, SFC is not limited to traditional network services. Dong et al. (2022) introduce the SFC embedding idea of NFV from 5G into the Internet of Vehicles (IoV) by decomposing the road condition recognition application of the IoV into a series of functions (i.e., picture split, object detection, object recognition, road condition perception), and embedding these functions into idle vehicles parked near the roadside. Similarly, the computing functions of DT service perform specific functions and consume resources at the instantiated node, and each service still has the critical latency requirement to guarantee the real-time interaction between the physical and virtual worlds. We define this novel DTs implementation service as *Digital Twin Service Function Chains* (DTSFCs). The implementation of DTs requires a real-time mapping between the physical entities and virtual spaces, its services are delay-sensitive and still have strict delay constraints.

However, due to the limited computing power and battery capacity of IoT equipments (ITEs), the computation-intensive and latency-sensitive services need to be offloaded to well-resourced servers. Therefore, edge intelligence (EI) (Zhou et al., 2019) is considered a promising alternative framework to provide a prompt response for IoT applications by deploying auxiliary computing and intelligence at the network edge, compared to cloud computing. By mapping the IoT systems to the digital twins constructed in EI, digital twin enabled-EI (DTEI) improves the efficiency of machine learning (ML) algorithms in EI and reduces the impact of unreliable communication between IoT devices and edge servers. Additionally, the DTEI directly performs the state analysis and operation optimization instead of dedicated devices in IoT, which reduces the deployment cost of IoT system. Despite this, DTEI still faces challenges in realizing efficient resources allocation when providing DTSFCs in complex and dynamic IoT scenarios:

- Different from the traditional SFCs, the functions in DTSFCs are not just networking functions (e.g., IDS, VPN, NAT, etc Cai et al.,

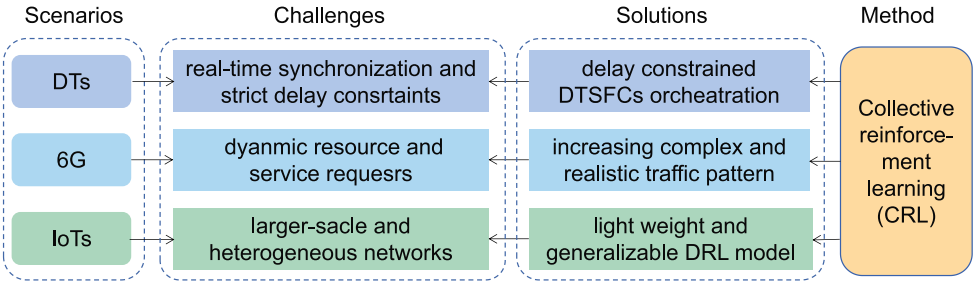


Fig. 2. The research framework of this paper.

2020) but computing functions (CFs) that process the transmitted data and return the computing results to the subsequent function, which makes the DT services more computation-intensive and latency-sensitive;
- The 6G combines multiple access communication technologies such as base stations, satellites, unmanned aerial vehicles, etc, as shown in Fig. 1. Therefore, efficient resource allocation methods for DTSFCs with heterogeneous and dynamic 6G-enables IoT are urgently needed;
- Recently, DRL algorithms have been increasingly used for resource allocation in massive IoT. However, deep learning based methods reliant on abundant samples for acquiring accurate and highly generalizable models, the collection of extensive samples proves time-consuming, labor-intensive, and sometimes unavailable in extreme or hazardous environments.

In this paper, we propose corresponding solutions to address these challenges brought by the new scenarios, as shown in Fig. 2. We propose a novel collective reinforcement learning (CRL) method to realize effective resource allocation of DTSFC in DTEI-empowered IoT. CRL is inspired by human collaboration that jointly combines the agent's own exploration and strategies from other agents. To our best knowledge, this paper is the first study to address the resource allocation problem of DTEI empowered IoT in form of dynamic DTSFCs orchestration. In short, our contributions list as follows:

- By carefully analyzing the phases of DTs implementation, we define this novel DTs service as the DTSFC. We further consider the unique properties of IoT services into account, i.e., the specified delay constraints, as well as the dynamic and heterogeneous characteristics of IoT services.
- We propose a novel CRL-based method to realize the effective resource allocation of DTSFCs, CRL goes beyond traditional DRL by not only focusing on policy exploration but also leveraging the intelligence of other agents to enhance adaptive performance and improve generalization ability. By incorporating the collective knowledge of agents, CRL enables more efficient and versatile learning processes;
- We use TensorFlow to realize the proposed CRL approach and conduct extensive simulations by considering various delay constraints and dynamic traffic patterns of DTSFCs to verify our proposed method.

The rest of the paper organizes as follows: we summarize the state-of-the-art related works in Section 2. In Section 3, we describe the framework of DTEI-enabled IoT and define the problem of delay-constrained DTSFCs resource allocation. Our proposed CRL-based DTSFCs orchestration solution is presented in Section 4, and we conduct the performance verification in Section 5. We further discuss the future potentials and current disadvantages of the proposed method in Section 6, and summarize the paper in Section 7.

2. Related work

We summarize the related works of resource allocation in DT-enabled IoT and DRL-based dynamic service orchestration, and compare the contributions, advantages, and disadvantages in detail, as listed in Table 2.

2.1. Resource allocation in DT empowered IoTs

To make full use of the scattered and limited resources in IoT, Dai et al. (2020) proposed a digital twin network (DTW) to realize the instant mapping of the IoT system and digital space and investigated the stochastic computation offloading problem in DTW using asynchronous actor–critic (AAC). Sun et al. (2020) also consider the architecture of digital twin (DT) empowered IoT, where DTs capture the characteristics of industrial devices to assist federated learning (FL). To reduce the estimation deviations of DTs from the actual value of the device state, a trusted-based aggregation scheme is proposed in federated learning which is based on deep Q-Network (DQN) algorithm. FL is recognized as an effective technology that enables collaborative computing in resource-decentralized IoTs, and further improves data privacy and system reliability. Lu et al. (2020) introduced an FL and blockchain-empowered framework for digital twin 6G wireless networks (DTWN). They address the joint edge association and communication resource allocation problem of DTWN that aims to minimize the system latency and develop a multi-agent RL method to find the optimal solution. The energy consumption of performing DT implementation cannot be ignored in the resource-constrained IoT. Therefore, Zhang et al. (2021) proposed an energy-efficient FL framework for DT-empowered IoT. To reduce the system energy consumption, they jointly address the DTs training server selection and resource allocation problem of the proposed framework and use a DQN-based algorithm to solve it.

These researches paved the way for DTs-empowered IoT, and have studied multiple forms of resource allocation problems, e.g., computation offloading (Dai et al., 2020), edge association (Lu et al., 2020), and real-time IoT traffic routing (Abujassar et al., 2021), etc. However, how to best allocate resources at the edge to support the DTs implementation in IoT, i.e. in the form of dynamic DTSFC orchestration, is still in the blank stage and needs further research.

2.2. DRL for dynamic SFC orchestration in IoT

Recently, deep reinforcement learning (DRL) methods have been recognized as promising methods for dynamic resource management. Fu et al. (2019) proposed a deep Q-learning (DQL) based scheme to achieve dynamic SFC embedding in complex IoT scenarios that aim to minimize service delay. Chen et al. (2021) investigated the distributed orchestration of SFCs in IoT and introduced double DQN-based methods to deploy VNFs and embed SFCs for cost-minimizing. However, the DQL and DQN-based approaches could only obtain the optimal strategy in the discrete action space. Guo et al. (2019) introduced the consortium blockchain and A3C-based scheme to construct a trust SFC orchestration framework, aims to minimize orchestration overhead, they focus

Table 2
Summary of relevant literatures.

Research	Contributions	Method	Pros	Cons
Dai et al. (2020)	Address the stochastic computation offloading problem in DTW	Lyapunov optimization technique; AAC	Jointly optimize offloading decision, transmission power, bandwidth, and computation resource	Did not compare to other continuous DRL method, but only to DQN
Sun et al. (2020)	Propose a trust-based aggregation in DT-driven IIoT	Lyapunov optimization technique; DQN	Considering the deviation of the DT from the true value in the trust-weighted aggregation strategy	DQN-based approach requires a large number of samples for training
Lu et al. (2020)	Propose FL and BC-empowered framework for digital twin 6G wireless networks	MARL	Jointly considering edge association and communication resource allocation to minimize the time cost	The proposed framework does not fully reflect the characteristics of 6G network
Zhang et al. (2021)	Propose energy-efficient FL framework for DT-empowered IoT	double-DQN; dueling-DQN	Address the energy consumption of performing FL and maintaining the virtual object in the digital space	The characteristics of the industrial IoT are not fully considered
Fu et al. (2019)	Decompose the complex VNFs into smaller virtual network function components to make more effective decisions	DQL	Consider the dynamic natures of network conditions and IoT-traffic, which is more realistic in practical IoT	DQL could only obtain the optimal strategy in the discrete action space
Chen et al. (2021)	Formulate the joint optimization problem to maximize total utility and decompose it into SFC selection and dynamic SFC orchestration	Resource-aware matching algorithm; DQN	Taking the timeliness, available resources and obtained reward into account	Two-stage optimization may introduce additional coordination problems; DQN could only obtain the optimal strategy in the discrete action space
Guo et al. (2019)	Integrates blockchain into the distributed SFC orchestration model to realize trusted resource sharing; propose a time-slotted model to support dynamic service migration	Blockchain; A3C	Both congestion control and dynamic SFC migration are taken into account.	Did not take the dynamic nature of IoT services into account; A3C-based algorithm may suffer from local optimum
Schneider et al. (2021)	Propose a novel DRL approach that learns how to best coordinate services and is geared towards realistic assumptions	DDPG	The proposed model-free approach adapts to various objectives and traffic patterns	They only verified the method on small networks, and the performance of the method on large networks is unknown
Zhu et al. (2022)	Propose a hybrid SFC deployment framework, which adopts the centralized training and distributed execution paradigm	Game theory; MARL	Combines the advantages of the distributed solution and centralized solution	MARL needs to pay attention to the joint action of all other agents, the dimension of the joint space will increase as the number of agents increases
Li et al. (2022a)	Propose a blockchain-enabled cloud–edge collaborative resource allocation and task offloading scheme for the IoT system through 6G networks	DQN; CRL	Joint optimizing the offloading decision, block interval, and transmission power	They did not consider the throughput of the blockchain system, and impact of the consensus latency with the increasing BSs
Tang et al. (2022)	Proposes a distributed edge intelligence sharing scheme between different edge nodes	CDRL	Consider the local learning at each edge node and collective learning between different edge nodes	The AI model and intelligence sharing are carried out based on some common assumptions, e.g., possible AI tasks, the quality and energy cost requirement, etc.

on the trust issue of IoT and the proposed A3C-based methods may suffer from local optimum due to the agent parallel training. Schneider et al. (2021) proposed a DDPG-based service coordination method called DeepCoord, which jointly controls service scaling and placement as well as flow scheduling. However, traditional single-agent DRL methods are not applicable to large-scale IoT scenarios. It takes a long time for agents to converge from a larger amount of data. Recently, the distributed multi-agents RL (MARL) methods are increasingly used for resource allocation in larger-scale IoT scenarios, Zhu et al. (2022) propose a hybrid SFC deployment framework, which adopts the centralized training and distributed execution paradigm, they formulate the problem as a multiuser competition game model and solved it by MARL. Nevertheless, MARL needs to pay attention to the joint state and action of all other agents, the dimension of the joint space will increase as the number of agents increases. Therefore, lightweight and efficient multi-agent collaboration is necessary. Li et al. (2022a) investigated 6G and blockchain technologies to improve network performance and assure the authenticity of data sharing for MEC-enabled IoT. Meanwhile, a unique intelligent optimization method called collective RL is developed to realize intelligent resource allocation, meet distributed training results sharing, and minimize excessive system resource consumption. Although DRL considerably enhances decision-making capacity and learning speed in dynamic IoT, huge repetitive model training is ubiquitous due to customers' inescapable requirements for the same sorts of data and training results. Furthermore, a smaller volume of data samples will result in model over-fitting. Tang et al. (2022) proposed a distributed edge intelligence sharing scheme to address these issues, which allows distributed edge nodes to economically improve learning performance by sharing their learned intelligence, where a novel collective deep reinforcement learning (CDRL) algorithm is designed to obtain the optimal intelligence sharing policy, which consists of local DRL learning at each edge node and collective learning between different edge nodes.

In addition, current SFC orchestration methods did not take the properties of the IoT services into account, while in this paper, we consider the unique characteristic of IoT service in three aspects: (1)

Table 3
Symbols used in this paper.

Symbol	Description
$DT_i(t)$	DTSFC generated by device i at time slot t
N	Total amount of device in IoT, $i \in N$
s_i	Service chain of DTSFC DT_i, $s, s_i \in S$
F	A set of VFs chained in s_i, $f \in F$
L	A set of virtual links between VFs, $l \in L$
C_f^i	The computation requirement of VF f in DTSFC DT_i
ω_i	Transmitted data size of DTSFC DT_i
λ_i	Data rate of DTSFC DT_i
Θ_i	Maximum delay tolerance of DTSFC DT_i
G	The substrate edge network
V	The set of physical nodes, $v \in V$
E	The set of physical links, $e \in E$
b_{sl}^e	The bandwidth requirement for l of s mapped on e
r_{sf}^v	The resource requirement of VF f of s deployed on v
cap_{cpu}^v	The computational capacity of node v
b_{sl}^e	The bandwidth requirement of link l mapped on e
cap_{bw}^e	The bandwidth capacity of physical link e
x_{sf}^v	Indicates whether VF $f \in s$ is deployed on node v
y_{sl}^e	Indicates whether virtual link $l \in s$ is mapped to link e

Specific delay constraints of IoT service are studied than just delay minimization; (2) We consider four increasingly complex and realistic traffic arrival patterns in the system model to simulate the dynamic nature of industrial IoT service; (3) We further study the generalization ability of the model by knowledge transfer. In the large-scale and heterogeneous IoT, services are often from different domains with great distinction, the generalization ability of the trained model is extremely important since a more general agent can significantly improve the efficiency and reduce training costs.

3. System model and problem formulation

In this section, we present the system description, dynamic resource allocation of DTSFC, and the mathematical model of DTSFC orchestration.

3.1. System description

The DTSFC requires real-time service delivery to ensure orderly production and collaboration between equipment in IoT. This paper intends to design a DTEI-driven DTSFC orchestration architecture and combine the global awareness of software-defined everything controller (SDXC) and flexible orchestration of virtual function orchestrator (VFO) to realize efficient resource allocation of DTSFCs. As shown in Fig. 1, this architecture mainly includes three layers:

Physical IoT layer: The physical IoT layer composes various entities of the IoT, such as sensors, vehicles, and other equipment. These devices continuously perceive the running status of the physical system and generate massive data. These data will be utilized to analyze the status and optimize resource allocation to improve system efficiency.

Edge intelligence layer: the MEC server equipped with AI capacity is deployed in this layer and connected with the base station by wire link to support the DTs implementation. Heterogeneous data generated from various ITEs are transmitted to the edge server through the 6G wireless network. Both communication and computation resources are limited in the network edge, which should be efficiently allocated to customize the dynamic DTSFCs.

Cloud platform layer: In case of the QoS or resource requirements of the service cannot be guaranteed in the EI layer, the edge will collaborate with the cloud to provide the services, but the delay constraint may be sacrificed. In this paper, we mainly focus on the resource allocation of the edge layer, so the cloud layer is not displayed.

3.2. Dynamic resources allocation of DTSFCs

The early stage of DT implementation in IoT requires mass computation to model, simulate, and render the DTs. After the DTs were constructed, the DTs model will be updated in real-time according to the data collected by ITEs of the physical IoT, as well as mapping the decisions made in DTs to the physical entities. These kinds of services have strict delay constraints and account for most of the DTSFCs. Therefore, this paper focuses on dealing with this kind of DTSFCs.

Assume that the DTSFC $DT_i(t)$ generated by device i at time slot t is represented by a tuple $DT_i(t) = \{s_i(t), C_f^i(t), \omega_i(t), \lambda_i, \Theta_i\}$, $i \in N$, where $s_i = \{f_1, f_2, \ldots, f_m\}, s_i \in S$, represents the service chain which chained by a serial of components f_m that implement corresponding functions of DTSFC, $m \le |F|$, $|F|$ is the total number of chained components. $C_f^i(t)$ represents the computation requirement to process the VF f of DTSFC DT_i, $\omega_i(t)$ represents the transmitted data size of DTSFC, the transmitted data of DTSFC including collected data returns from the physical world to DTs for fusion training, and AI models trained in DTs which need to be mapped to physical devices. λ_i is the data rate of DTSFC, Θ_i represents the maximum delay tolerance of the delay-sensitive DTSFC, the service will be dropped if service delay exceeds the max tolerance. Lets $G = \langle V, E \rangle$ represents the substrate edge network, which consists of multiple nodes v and links e. Each node $v \in V$ has a computation capacity cap_{cpu}^v, we mainly consider the CPU processing frequency due to the computing function in DTSFCs. Each link $e \in E$ has a maximum data rate cap_{bw}^e and interconnects two nodes bidirectionally.

Benefit from the flexible orchestrate capabilities of the VFO and the global awareness and control capabilities of the SDXC. The process of DTSFC resource allocation is divided into two stages: virtual functions (VFs) embedding and traffic scheduling between them (Liu et al., 2020; Cai et al., 2021). VFO first embeds VFs to the substrate node with sufficient resources to process, and then SDXC schedules the traffic flow through the VFs in a pre-defined order to customize the corresponding services. To this end, we define two binary variables x_{sf}^v and y_{sl}^e to represent the orchestration decisions. $x_{sf}^v = \{0, 1\}$ is equal to 1 if a VF f from s_i is mapped to the physical node v, otherwise, it equals 0 (i.e., VFs embedding). Likewise, $y_{sl}^e = \{0, 1\}$ is equal to 1 if virtual link $l \in L$ from s_i is mapped on physical link $e \in E$, otherwise $y_{sl}^e = 0$ (i.e., flow scheduling). Considering the above-mentioned scenario, we present an optimization model for DTSFC resource allocation problem which aims to minimize service delay, while ensuring the service QoS and meeting the resources constraints. The symbols used in the paper are summarized in Table 3.

3.3. Minimizing DTSFCs delay with delay constraint

In this subsection, we first define the end-to-end delay Θ_s of DTSFC, then propose the mathematical formulations of minimizing the D_{end}^s, meanwhile, ensuring the delay tolerance Θ_s of DTSFC.

The end-to-end delay D_{end}^s of DTSFC consists of three parts: VF processing delay, data transmission delay, and link propagation delay. The VF processing delay D_p^s is calculated by the ratio of the total number of CPU cycles required for task processing and the processing power (i.e., CPU frequency) of the instantiated node:

$$D_p^s = \sum_{f=1}^{|F|} \frac{C_f^i}{r_{sf}^v} x_{sf}^v, \forall i \in N, \forall v \in V, \forall s \in S, \tag{1}$$

where x_{sf}^v indicates whether the VF f of service s is embedded to physical node v to process, C_f^i is the computation requirements of VF f in DTSFC DT_i, and r_{sf}^v is the computing power that node v allocated to process the VF f of s. $|F|$ represents the total number of VFs. The total process delay is the sum of all VFs.

The data transmission delay D_t^s is determined by the data size divided by available bandwidth of links:

$$D_t^s = \sum_{l=1}^{|L|} \frac{\omega_i}{b_{sl}^e} y_{sl}^e, \forall i \in N, \forall e \in E, \forall s \in S, \tag{2}$$

where y^e_{sl} indicates whether the virtual link l between *(j-1)*th VF and jth VF is mapped to the physical link e, ω_i represents the transmitted data size of DTSFC, i.e., the size of return data from physical devices or AI models trained in DTs, b^e_{sl} is the required bandwidth that physical link e allocated to transmit the flow of DTSFC.

According to Fu et al. (2019), the link propagation delay D^s_g is determined by $D^s_g = \frac{len}{c}$, where len indicates the DTSFC length and c is the propagation speed of signals determined by the physical link medium. Therefore, the end-to-end delay of DTSFC can be represented as:

$$D^s_{end} = D^s_p + D^s_t + D^s_g, \forall s \in S. \tag{3}$$

The optimization objective expressed as (4), minimizes the overall service end-to-end delay:

$$obj: \quad min \sum_{s=1}^{|S|} D^s_{end}, \forall s \in S. \tag{4}$$

$$s.t. \quad 0 \le D^s_{end} \le \Theta_s, \forall s \in S. \tag{5}$$

$$\sum_{v \in V} \sum_{f \in F} (r^v_{sf} \cdot x^v_{sf}) < cap^v_{cpu}, \forall s \in S. \tag{6}$$

$$\sum_{e \in E} \sum_{l \in L} (b^e_{sl} \cdot y^e_l) < cap^e_{bw}, \forall s \in S. \tag{7}$$

$$\sum_{v \in V} \sum_{f \in F} x^v_{sf} = \mid F \mid, \forall s \in S. \tag{8}$$

$$\sum_{v \in V} x^v_{sf} = 1, \forall s \in S. \tag{9}$$

$$\sum_{e \in E} \sum_{l \in L} y^e_{sl} = \mid L \mid, \forall s \in S. \tag{10}$$

$$\sum_{e_{uv} \in E, l_{pq} \in L} y^{e_{uv}}_{l_{pq}} - \sum_{e_{vu} \in E, l_{qp} \in L} y^{e_{vu}}_{l_{qp}} = 0, \forall p, q \in F, \forall u, v \in V. \tag{11}$$

The constraints of this optimization problem are described in (5)–(11): Firstly, (5) guarantees each service must be processed within the delay tolerance Θ_s consider in IoT to ensure reliable DT services implementation. The computation requirements of VFs mapped on the physical nodes cannot exceed the node resources capacity, therefore, (6) guarantees the resource utilization constraint, where r^v_{sf} represents the allocated computing resource to VF f of s at node v, cap^v_{cpu} is the processing capacity of physical node v. Likewise, constraint (7) avoids link's bandwidth over-utilization, where b^e_{sl} indicates the bandwidth requirement of link l of s mapped on e, and cap^v_{bw} is the bandwidth capacity of physical link e. All VFs of s must be instantiated on physical nodes and chained together to provide corresponding services, which is defined in (8). Nevertheless, each VF of s can be deployed on only one physical node, which is guaranteed by (9). Similarly, the virtual links between each pair of VFs must be embedded in physical links, as shown in constraint (10). The incoming flow must be equal to the outgoing flow for all nodes, as defined in (11), where l_{pq} and e_{uv} indicate a virtual link and a physical link whose incoming and outcoming VF (or nodes) are p and q (or u and v).

4. CRL based DTSFCs resource allocation approach

In this section, we first formulate the problem as a Markov decision process (MDP) and propose a CRL-based solution to solve this problem.

4.1. MDP-based problem formulation

In this paper, we assume that IoT service requests arrive in continuous-time steps and vary over time. The proposed DTSFCs orchestration model involves IoT service requests arriving, being embedded or rejected, and departing after being processed. The substrate network scenarios, as well as traffics of IoT, change dynamically over time, and the movement of the IoT terminal results in the random arrival of service requests. In a dynamic environment where the underlying networks are changing randomly, an intuitionistic model can be achieved using Markov stochastic processes (Liu et al., 2020). The DTSFC orchestration problem also has Markov property since the network state of the current time slot depends on the previous service embedded in the underlying network. To this end, we can formulate the DTSFCs orchestration problem of IoT as a MDP and address it by RL method. The essential elements of RL-based tasks represent by a three-tuple $\mathcal{M} = \langle S, \mathcal{A}, \mathcal{R} \rangle$, which are defined as follows:

State space: $S(t) = \langle G(t), DT_i(t), \mid i \in N \rangle$ denotes the system state at time t, where $G(t) = \langle V, E \rangle$ represents the current network state including the node and link resources capacity, $DT_i(t) = \{s_i(t), C^i_f(t), \omega_i(t), \lambda_i, \Theta_i\}$, represents the DTSFC of device i required at time t.

Action space: $\mathcal{A}(t) = \left\langle p_{v,s,f,v'} \mid v, v' \in V, s \in S, f \in F \right\rangle$ denotes the action set taken at time t, where $p_{v,s,f,v'} \in [0,1]$ is the probability for scheduling a flow arriving at node v, requesting VF f of service s to be processed at node v'. We only define the flow scheduling strategy and leave the VFs embedding as hidden actions (Liu et al., 2020), which means if traffic needs to be processed in node v, the corresponding VF must be deployed on node v first.

Reward function: the DRL method aims to find an optimal policy to maximize the long-term cumulative reward $\mathcal{R}(t) = \sum_{t=1}^{T} r(t), t \in T$ while following the policy, where $r(t)$ it the instant reward of the current time. We use two delay criteria from Schneider et al. (2021) to obtain the instant reward r_i. (1) Hard-deadline: the smaller the service delay, the higher the instant reward is ($r \in [-1, 1]$), the service will be discarded if the service delay exceeds the hard-deadline limit, and set to punish reward r = -1. (2) Soft-deadline-exp: Contains two delay limits d^{soft}_s and d^{hard}_s, and $d^{soft}_s < d^{hard}_s$. Delay reward is maximal (r = 1) if the service delay is smaller than the soft deadline d^{soft}_s and then gradually diminishes with an increasing delay up to hard deadline d^{hard}_s ($r \in [0, 1]$), if the delay is large than d^{hard}_s, this flow will be dropped.

4.2. Problem solution with CRL

In this paper, we propose a novel collective reinforcement learning method (CRL) that combines agents' exploration and extension of other agents to achieve the optimal resource allocation of DTSFC. CRL is inspired by human collaboration (Yu, 2021), imagine that when humans are looking for a restaurant, we either exploring a new restaurant, but we may face the risk of being disappointed or consulting other people's opinions in advance without actually being there. Similar to human learning and cooperative behavior, CRL learns its strategy meanwhile integrating the knowledge from other agents, thereby improving the learning efficiency, using fewer samples to learn more general strategies, and improving its generalization ability. The architecture of collective RL is shown in Fig. 3.

It can be concluded from the structure that CRL is fundamentally different from multi-agent reinforcement learning (MARL): (1) in MARL, the agent changes the environment through actions, thereby affecting other agents, while CRL affects other agents through direct knowledge transfer between agents, in short, MARL indirectly affects other agents, while CRL directly affects other agents; (2) the MARL agent cannot make decisions independently (except independent Q learning), because the agent's policy network needs to know the observations of all other agents, and the entire state space is the sum of the partial observations of all agents, $S = [o^1, o^2, \ldots, o^m]$, CRL can make independent decisions and pass its decision-making knowledge to other agents, that is, MARL collaborated on sharing observations, and CRL collaborated on collaborative knowledge; (3) agent in MARL needs to pay attention to the joint state and action of all other agents. However, the dimension of the joint space will increase as the number of agents increases, so it is difficult to apply to large-scale scenarios, and CRL solves this problem.

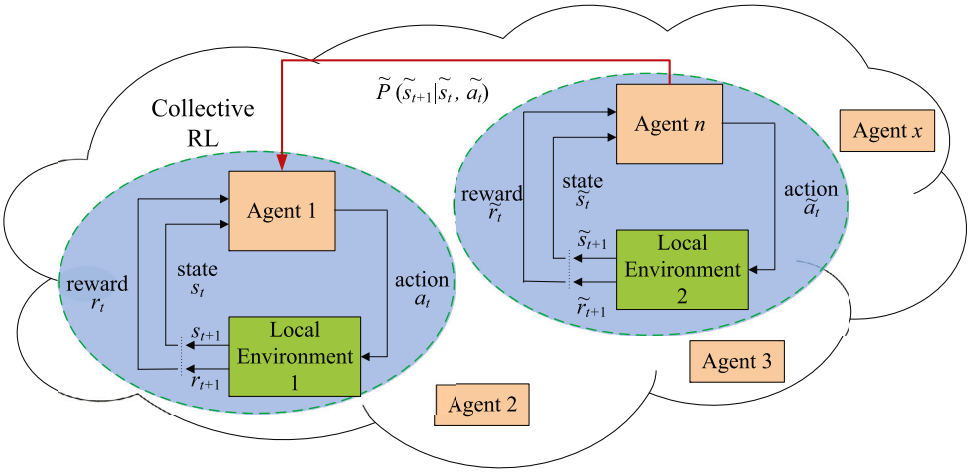


Fig. 3. The architecture of collective reinforcement learning.

4.2.1. From DRL to CRL

The traditional DRL method aims to find an optimal policy $\pi(a|s)$ to maximize the cumulative reward while following that policy.

$$\max_{\pi} R(s_t, s_t) = r_t + \gamma[\mathbb{E}_{s_{t+1}, a_{t+1}} Q(s_{t+1}, a_{t+1})], \tag{12}$$

where γ is the discount factor of the future returns.

Different from the sole objective DRL algorithms, an entropy $H(\pi(a \mid s))$ is added to the value function $Q(s_t, a_t)$ of the soft actor–critic (SAC) (Haarnoja et al., 2018) algorithm to ensure the agent can explore continually. The coefficients ϵ_{ep} balance the exploitation-exploration trade-off during the exploration process. The agent tries to maximize the system reward through local learning in an unknown environment through its own experience. Therefore, the quality of exploration is crucial in the learning process of the SAC algorithm. Exploring leads to new unknown successful behaviors, choosing strategies that the system does not know about, and possibly learning more. Therefore, the SAC algorithm learns the policy by performing an approximate exploration of the following value function:

$$Q(s_t, a_t) = \max_{\pi} R(s_t, s_t) + \underbrace{\epsilon_{ep} H(\pi(a|s))}_{Exploration}. \tag{13}$$

Although exploration incentives encourage agents to take "competitive" actions, agents still struggle to adapt to new environments without pre-training. For example, in the larger-scale and multiple domains IoT scenarios, each domain has its own networking operator, running a different protocol, domains are often reluctant to share data with third parties due to privacy issues, and the finite training samples of the single domain further limit the adaptability of the training model. Therefore, similar to the way humans ask for other people's opinions when making decisions, the extension $E(\pi(a \mid s))$ is added in novel CRL to enable agents to actively cooperate with other agents, i.e., combine the knowledge of other agents. Set ϵ_{et} the extension trade-off factor. The value function of CRL $Q^c(s_t, a_t)$ can be expressed as:

$$Q^c(s_t, a_t) = \max_{\pi} R(s_t, a_t) + \underbrace{\epsilon_{ep} H(\pi(a|s))}_{exploration} + \underbrace{\epsilon_{et} E(\pi(a|s))}_{extension}, \tag{14}$$

where the exploration incentive $H(\pi(a|s))$ can be calculated by the KL-divergence of the actual model P and the learning model P_{θ_t} distribution, in the same way, the extensions $E(\pi(a \mid s))$ can be calculated by the KL-divergence of the actual model P and other agent's model $\widetilde{P}$ distribution (Yu, 2021).

Based on the above definition, the value function of CRL can be rewritten as:

$$\begin{aligned} Q^c(s_t, a_t) &= \max_{\pi} R(s_t, s_t) \\ &\underbrace{-\epsilon_{ep}[log P_{\theta_t}(s_{t+1}|s_t, a_t) - log P_{\theta_{t-l}}(s_{t+1}|s_t, a_t)]}_{exploration} \\ &\underbrace{-\epsilon_{et}[log P_{\theta_t}(s_{t+1}|s_t, a_t) - log \widetilde{P}_{\theta_{t-l+1}}(\widetilde{s}_{t+1}|\widetilde{s}_t, \widetilde{a}_t)]}_{extension}. \end{aligned} \tag{15}$$

Algorithm 1 CRL-based DTSFCs orchestration algorithm

Require: $\epsilon_{ep}, \epsilon_{et}, \gamma$;
Ensure: $p_{v,s,f,v'}, x^v_{sf}, y^e_{sl}$;
1: Initialize network parameters: ϕ, θ, ϑ;
2: **for** each episode **do**
3: **for** each environment step **do**
4: choose action $a \xleftarrow{\pi(\cdot|s,\phi)}$ current state s ;
5: observe $(s', r) \xleftarrow{execute}$ action a;
6: replay buffer $D \xleftarrow{update} (s, a, r, s')$;
7: **end for**
8: **for** each gradient step do **do**
9: (s, a, r, s') $\xleftarrow{sample}$ from D for training:
10: calculate $Q^c(s_t, a_t) \leftarrow$ according to novel $Q^c(s_t, a_t)$ (15) ;
11: $\theta \leftarrow$ update the parameters of value network Q by loss function $L(\theta)$;
12: $\vartheta \leftarrow \theta + \tau\theta$, soft update the parameters of target value network $\overline{Q}$;
13: $\phi \leftarrow$ update the parameters of policy network π by loss function $L(\phi)$;
14: output the optimal strategy: $p_{v,s,f,v'}, x^v_{sf}, y^e_{sl}$;
15: **end for**
16: **end for**

4.2.2. Work flow of CRL-based resource allocation algorithm

The CRL algorithm inherits the actor–critic architecture, therefore, it incorporates four key components: (1) an actor–critic architecture with a separate policy deep neural network (DNN) (named as an actor) and value function DNNs (named as a critic); (2) an off-policy formulation which enables reuse of the past experience stored in the replay memory for sample efficiency, and (3) entropy maximization to ensure stability and exploration of unknown policies. (4) an extension

Table 4
Simulation settings.

Network topology	11–110 computing nodes
Node computation capacity	[1, 5] GHz
CPU cycles required by VFs	[3, 5]M cycles
Link bandwidth	[0.5, 1] Gbps
Input data size of DTSFC	[1, 10] Mb
CFs number of DTSFC	[3, 6] VFs
Traffic requests	4 increasingly complex and realistic traffic patterns

to enable agents to actively cooperate with other agents and share intelligence, thus improving the generalization ability.

The learning process of CRL is shown in Algorithm 1. First, the actor utilizes a parameterized DNN to approximate policy $\pi_\phi(\cdot|s)$, selects and executes an action a at current environment state s (lines 4–5). The system goes to the next state s', and obtains an instant reward r. After that, the experience tuple $\langle s, a, r, s' \rangle$ is stored in replay buffer D (line 6). To mitigate the temporal correlations between different samples, the critic randomly samples the experience $\langle s, a, r, s' \rangle$ from the replay buffer to train the value network Q and target value network $\overline{Q}$ (ling 9).

$$L(\theta) = \mathbb{E}[\frac{1}{2}(Q^c_\theta(s,a) - \overline{Q}^c(s,a))^2]. \tag{16}$$

Then, the loss function $L(\theta)$ of the mean squared error (MSE) between value function Q^c and target $\overline{Q^c}$, will be back-propagated to update the parameters θ of value network Q using stochastic gradient descent method (lines 10–11), the target value network $\overline{Q}$ conduct a soft update form Q (line 12), while policy network parameter ϕ is updated by gradient descent of KL-divergence $L(\phi)$ (line 13). Finally, the CRL agent outputs the optimal strategy (line 14).

$$L(\phi) = \mathbb{E}[D_{KL}(\pi_\phi(\cdot \mid s) \parallel \pi'(\cdot|s))]. \tag{17}$$

4.2.3. Algorithm complexity analysis

Note that the improvement of the algorithm in this study is to reshape the reward on the value function, that is, to incorporate the extensive knowledge of other agents, and that the algorithm's training procedure is the same as that of the SAC algorithm. As a result, the algorithm proposed in this study requires no additional training.

The computational complexity mainly comes from DNN training and parameter update (Sun et al., 2021). As a result, the computational complexity of a DNN with back-propagation and gradient descent is $O(ghKb)$, where g and h represent the number of layers and neurons in each layer, respectively; K represents the number of episodes; and b represents the mini-batch size. The proposed CRL algorithm composes of two DNNs: actor and critic, since the actor and critic have the same network structure, therefore, the computational complexity of the proposed algorithm is $O(2ghKb)$.

5. Simulations results and discussions

This section describes the simulation setup and discusses the results to verify the pros and cons of our proposed method compared with the baseline algorithms.

5.1. Simulations setup

5.1.1. Simulation scenarios

We conduct extensive simulations on real-world network topology selected from the Internet Topology zoo (Knight et al., 2011), the topology scale is selected from 11 nodes to 110 nodes, and assign heterogeneous node computation capacities cap^v_{cpu} between [1, 5] GHz, i.e., CPU frequency. The number of CPU cycles required by VFs is set to [3, 5] M cycles. We set link transmission bandwidth cap^e_{bw} between [0.5, 1] Gbps.

We randomly generate the DTSFC with [3, 6] VFs, each VF has the computing requirement of [3, 5]M cycles, and the size of input data transferred by DTSFC during the real-time sync phase is set to [1, 10] Mb. We assume all service flows to have a unit data rate but consider four increasingly complex and realistic traffic patterns to simulate the dynamic DTSFCs requesting scenarios of IoT: (1) Det-arrival, the request flows arrive at each ingress in fixed intervals of 10 time-steps. (2) Poisson arrival, request flows arrive with a mean inter-arrival interval of 10 time-steps. (3) MMPP-arrival, a more realistic traffic arrival pattern that follows a Markov modulated Poisson process (MMPP) (Fischer and Meier-Hellstern, 1993). (4) Real-world traffic traces, which were recorded from the Abilene network (Orlowski et al., 2010). The specific simulation settings are shown in Table 4.

5.1.2. Baseline DRL algorithms

At present, numerous research has proved that DRL-based methods are substantially superior to the traditional heuristic and exact algorithms (Cai et al., 2021; Li et al., 2022b; Liu et al., 2020). Therefore, this paper only compares advanced DRL-based methods as follows:

Deep deterministic policy gradient (DDPG): the policy gradient-based DRL orchestration algorithm of SFCs, we keep the hyper-parameter settings of the original paper (Schneider et al., 2021). The authors open source code on GitHub.[1] We reuse this code as one of the comparison algorithms.

Soft actor–critic (SAC): the exploration maximum based-DRL algorithm, and configures the following hyper-parameters: (1) learning rate = 0.01, (2) discount factor γ = 0.99. (3) batch size | b | = 64 and buffer size | B | = 10000. Recently, the SAC algorithm has been applied to resource management problems, e.g., optimizing the energy management in a hybrid electric bus (Wu et al., 2020).

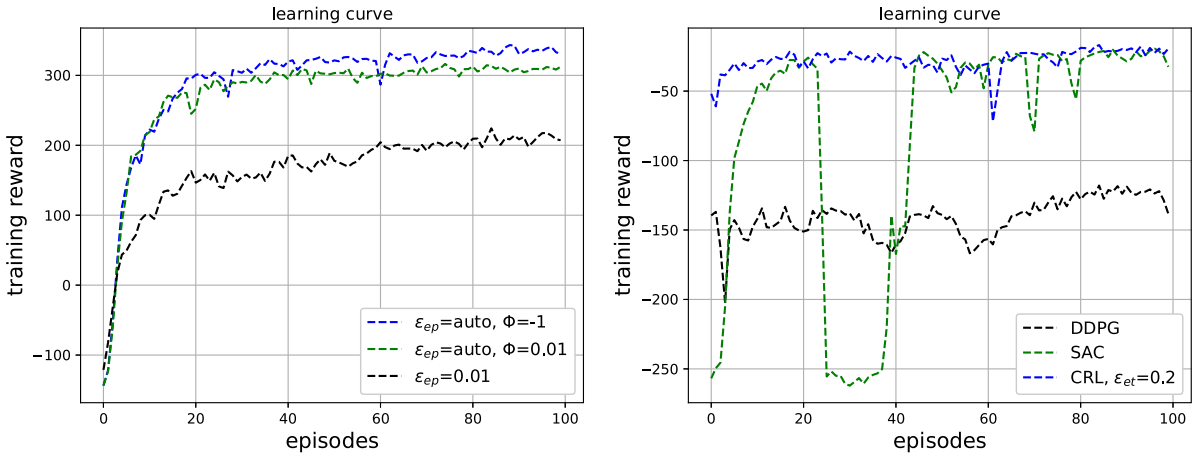
The simulation using Python 3.6 and TensorFlow 1.14 with CUDA 10.0 and implemented in PyCharm using remote Ubuntu 16.04 server equipped with Intel(R) Core(TM) i7-7820X CPU @ 3.60 GHz, GeForce RTX 2080 Ti GPU. The implementation follows the common OpenAI Gym interface, which can be easily extended to other DRL algorithms. We train the agents of three algorithms in 100–500 episodes, each containing 2000 time steps. The trained agent is automatically used for DTSFCs orchestration, we output the average results of all DTSFCs at the end.

5.2. Learning curves comparison

We first verify the learning curves of the DRL algorithms under different parameter settings to obtain the best performance. The exploration coefficient ϵ_{ep} controls the trade-off between exploration and exploitation both in SAC and CRL. Haarnoja et al. (2018) also proposed an automatic-learning ϵ_{ep} mechanism by giving a target entropy Φ. The learning curves of fixed (i.e., ϵ_{ep} = 0.01) and automatic learning ϵ_{ep} are first compared. As shown in Fig. 4(a), DRL (i.e., SAC and CRL) with auto-learning ϵ_{ep} and target entropy Φ at −1 can get rapid convergence and max accumulate reward, where Φ is calculated by the dimensions of action space, i.e., $\Phi = -dim(\mathcal{A})$ (Haarnoja et al., 2018).

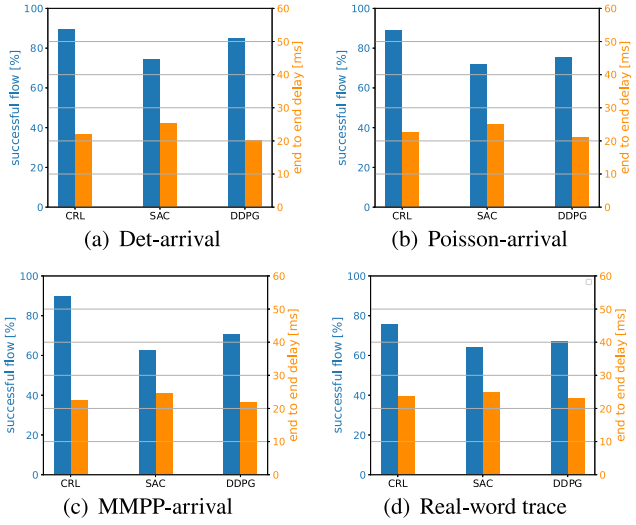
Then we verify the learning efficiency of three algorithms under the same environment setting. We first verify the extension coefficients ϵ_{et} of CRL range from 0.1 to 0.5 and find that ϵ_{et} = 0.2 leads to the best learning performance in our simulation. The learning curves show that both CRL and SAC have a significant superiority over the DDPG algorithm, indicating the entropy maximize-based algorithm is extensively explored to learn the optimal strategy more quickly. However, the SAC may plunge into a poor situation because of random exploration, but it can quickly jump out of the trap and learn a better strategy than before. Nevertheless, the learning efficiency of CRL is better than that of SAC, it obtains a higher reward from the beginning and is even more stable than the original SAC algorithm due to collective learning.

[1] https://github.com/RealVNF/DeepCoord



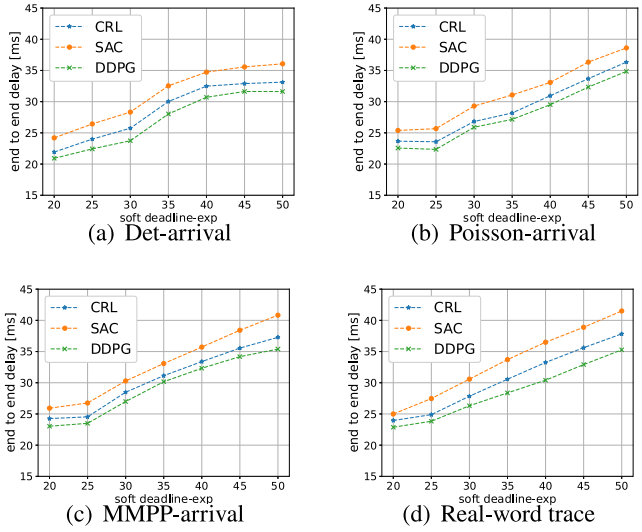
(a) Learning curve of optimal ϵ_{ep} (b) Learning curve of optimal ϵ_{et}

Fig. 4. Learning curve of optimal parameter setting.



(a) Det-arrival (b) Poisson-arrival (c) MMPP-arrival (d) Real-word trace

Fig. 5. Deployment performance of hard-deadline 25.



(a) Det-arrival (b) Poisson-arrival (c) MMPP-arrival (d) Real-word trace

Fig. 6. End-to-end delay of soft-deadline-exp 20–50.

5.3. Performance with different delay constraints

5.3.1. Hard deadline constraint

We employ two delay metrics to verify the algorithm's performance, as discussed in Section 4.1. We first verified the algorithms' performance by dealing with hard-deadline constraints under increasingly complex traffic patterns, concerning the service success rate, and end-to-end delay.

Figs. 5(a)–(d) show the performance of each algorithm in multiple traffic patterns with hard-deadline at 25 ms. The blue and orange histograms denote the success flow rate and the service end-to-end delay, respectively. All three algorithms can adapt to the hard delay constraint well, (i.e., the service delay of less than 25 ms), because services that exceed this constraint of the hard deadline will be discarded, so the successful flow percentage is the critical measure of the algorithm performance. As shown in Fig. 5, the successful flow of CRL is much higher than that of SAC (about 10%–20%) and slightly better than DDPG (about 5%–10%), which can maintain the highest success flow rate about 70%–85% encountered with different traffic patterns.

5.3.2. Exponential soft deadline constraints

In the subsequent simulations, we verify the algorithms with the exponential soft deadline constraint i.e., soft-deadline-exp to evaluate their ability to adapt to more complicated delay situations. The deadline is set from 20 ms to 50 ms.

Figs. 6(a)–(d) show the service delay at different soft-deadline-exp limits encountered with multiple traffic patterns. In general, as the delay limits increase, the average service delay also increases accordingly to allow more services in the network. While the delay limit at 20 ms is too strict for any algorithm to meet, therefore the delay limits are exceeded among all three algorithms when the delay limits are relaxed, DDPG and CRL can quickly adapt at 25 ms limit. Although the service delays of CRL are slightly higher than that of the DDPG's (about

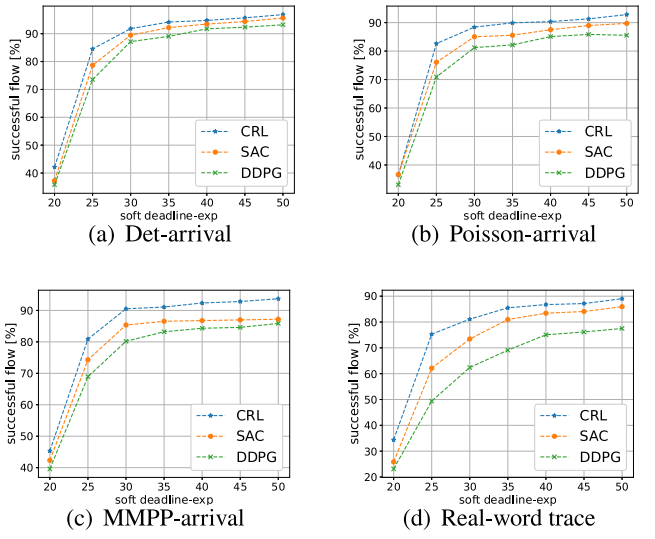


Fig. 7. Successful flow of soft-deadline-exp 20–50.

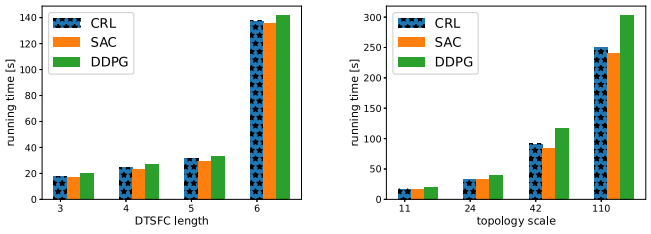


Fig. 8. Running time comparison.

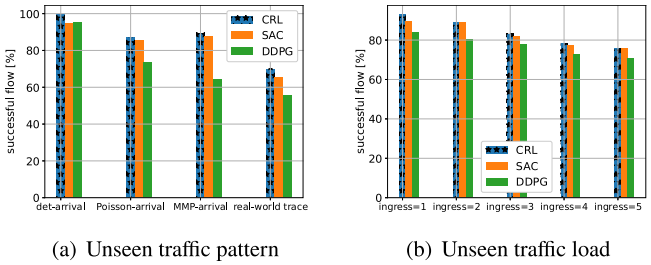


Fig. 9. Self-adapted to unseen traffic scenarios.

2-3 ms), the delay is still within the delay limits as DDPG, so the quality of service is not degraded.

Figs. 7(a)–(d) show the success flow rate of each algorithm under different traffic patterns. In each sub-graph, the successful flow rate of each algorithm increases with the soft-deadline-exp, as more services can meet the constraints. When the traffic gradually becomes more complicated from det-arrival, rand-arrival, mmpp-arrival to real-world traffic trace, the successful flow rate drops slightly for CRL and SAC, while the traffic pattern changes has obvious impacts on DDPG, and CRL always maintains the highest successful flow rate. The above results indicate that strengthening the exploration in value function can indeed improve the adaptability of DRL algorithms. Considering that CRL still meets the delay limit while at the same time achieving a higher success flow rate, as shown in Fig. 7, it surely learns a better strategy.

5.4. Running time comparison

The running time is an important indicator of algorithm's scalability and computational cost, we verify the algorithm's running time with the increasing DTSFCS lengths and topology scales. The running time of one episode DTSFCs orchestration is compared, and the DTSFCs arrive at an average interval of 10-time steps. As shown in Fig. 8, SAC has the shortest running time in two situations (about 2-10s shorter than CRL and 5-70s shorter than DDPG), followed by CRL, while DDPG has the longest running time, because DDPG aims at minimizing the service delay, an extensive search is required to find the short path, therefore, the increasing network scale has the significant impact on DDPG. Although CRL combines intelligence extension to learn better strategies, its complexity is far less than searching the short path.

The learning curve presented in Fig. 4 demonstrates that the agent converges well after 100 episodes of training. Remarkably, this convergence is achieved within a small computational cost, typically ranging from 30 min to a few hours, according to one episode running time shown in Fig. 8. This efficiency is particularly notable when compared to the computational requirements of other large-scale models, e.g., DeepMind's AlphaGo Zero was trained over almost 5 million games. During the practical application phase, the trained agent can be directly employed for service orchestration without further training and only requires fine-tuning when applied to different scenarios. This approach allows for efficient and effective utilization of the trained model in real-world applications.

5.5. Generalization ability comparison

Training a new model is time-consuming and resource-intensive, so it is crucial that the model has good generalization ability, especially in the dynamic IoT environment with changing traffic patterns and network loads.

5.5.1. Generalizes to unseen pattern

We first investigate the performance of all algorithms to unseen traffic patterns. We first train the agents on the simplest det-arrival traffic and test them directly with other complex and realistic traffic, the results are shown in Fig. 9(a). All algorithms were affected by the mismatch between the training and test pattern, but DDPG was affected the most (40% decline at worst). While CRL and SAC can maintain a relatively higher service successful flow rate (30% decline at worst).

5.5.2. Generalizes to changing load

We also study the adaptability of three algorithms to changing traffic load, which is indicated by the traffic ingress node number. We trained the agents with ingress = 3 and tested them with ingress = 1, 2, 4, and 5 in real-world traffic. As shown in Fig. 9(b), the results show a similar trend as in traffic patterns, but less significant (15% decline at worst).

In short, all these algorithms obtained their good performance from their capability to learn well from the training data, therefore unseen patterns and traffic loads will affect their performance. But our proposed collective intelligence-based CRL algorithm still maintains a clear advantage over the SAC and DDPG algorithms indicating that CRL can indeed learn a more generalized strategy than a deterministic strategy.

6. Discussion

In the future 6G era, as we progress towards intelligence networking, the evolution of networks will witness a significant shift from mere information exchange to intelligent decision-making and intelligence exchange (Yu, 2021). This transformation necessitates the deployment

of numerous agents or AI models, coexisting within the network, each catering to specific objectives or serving distinct user groups. Deep learning models, reliant on abundant samples and labels, are crucial for acquiring accurate and highly generalizable models. Similarly, the DRL approach entails an experience collection phase where training samples are generated internally. Nonetheless, the collection of an extensive dataset proves time-consuming, labor-intensive, and sometimes impossible in extreme or hazardous environments. To address this challenge, the CRL-based learning method facilitates agents in acquiring comprehensive models with robust generalization capabilities through knowledge collaboration. By leveraging the collective knowledge of multiple agents, CRL not only saves time and reduces redundant training efforts but also tackles the issue of learning appropriate models even in the absence of specific samples.

However, it is important to acknowledge that the current CRL-based methodology is still in its exploratory stage, and certain challenges persist. For example, ensuring the secure transfer of knowledge between agents and verifying the positive impact of transferred knowledge on performance are critical aspects that require further research and resolution.

7. Conclusion and future work

In this paper, we have investigated the dynamic resource allocation mechanism for the digital twin service within the context of 6G IoT networks. Our study begins by introducing a novel architecture called DTEI, which effectively enables efficient and cost-effective service delivery for IoT applications. Building upon this architecture, we proceed to model real-time DT implementation in IoT as dynamic DTSFCs. To address the unique characteristics of IoT services, we propose a novel approach known as Collective Reinforcement Learning (CRL) to tackle the orchestration problem associated with DTSFCs. Through extensive simulations, we demonstrate that our proposed CRL method outperforms existing state-of-the-art DRL algorithms, exhibits good convergence, effectively adapts to service delay constraints, and significantly improves both throughput and generalization capabilities. Furthermore, CRL leverages the knowledge cooperation among agents, also exhibits promising potential in terms of reducing training time, minimizing redundant model training, and enabling the acquisition of a comprehensive model even in scenarios where sample availability is limited. We are currently working on integrating privacy protection mechanisms and reputation systems into multi-agent collaboration scenarios. These endeavors aim to ensure secure and trustworthy collaboration among agents while preserving privacy and maintaining a reliable reputation system.

CRediT authorship contribution statement

Zhongwei Huang: Conceptualization, Methodology, Investigation, Software, Formal analysis, Writing – original draft, Writing – review & editing. **Dagang Li:** Conceptualization, Methodology, Investigation, Writing – review & editing, Funding acquisition. **Jun Cai:** Supervision, Resources, Funding acquisition, Writing – review & editing. **Hua Lu:** Supervision, Funding acquisition, Writing – review & editing, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Acknowledgments

This work was supported by the National Key R&D Program of China No. 2019YFB1804400, and Macau University of Science and Technology Faculty Research Grants No. FRG-21-031-IINGI.

References

Abujassar, R.S., Yaseen, H., Al-Adwan, A.S., 2021. A highly effective route for real-time traffic using an IoT smart algorithm for tele-surgery using 5G networks. J. Sens. Actuator Netw. 10 (2), 30.

Cai, J., Huang, Z., Liao, L., Luo, J., Liu, W.X., 2021. APPM: adaptive parallel processing mechanism for service function chains. IEEE Trans. Netw. Serv. Manag. 18 (2), 1540–1555.

Cai, J., Huang, Z., Luo, J., Liu, Y., Zhao, H., Liao, L., 2020. Composing and deploying parallelized service function chains. J. Netw. Comput. Appl. 163, 102637.

Chen, H., Wang, S., Li, G., Nie, L., Wang, X., Ning, Z., 2021. Distributed orchestration of service function chains for edge intelligence in the industrial internet of things. IEEE Trans. Ind. Inform. (Early Access).

Dai, Y., Zhang, K., Maharjan, S., Zhang, Y., 2020. Deep reinforcement learning for stochastic computation offloading in digital twin networks. IEEE Trans. Ind. Inform. 17 (7), 4968–4977.

Dong, L., Gao, H., Wu, W., Gong, Q., Dechasa, N.C., Liu, Y., 2022. Dependence-aware edge intelligent function offloading for 6G-based IoV. IEEE Trans. Intell. Transp. Syst.

Fischer, W., Meier-Hellstern, K., 1993. The Markov-modulated Poisson process (MMPP) cookbook. Perform. Eval. 18 (2), 149–171.

Fu, X., Yu, F.R., Wang, J., Qi, Q., Liao, J., 2019. Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach. IEEE Trans. Wireless Commun. 19 (1), 507–519.

Glatt, M., Sinnwell, C., Yi, L., Donohoe, S., Ravani, B., Aurich, J.C., 2021. Modeling and implementation of a digital twin of material flows based on physics simulation. J. Manuf. Syst. 58, 231–245.

Guo, S., Dai, Y., Xu, S., Qiu, X., Qi, F., 2019. Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT. IEEE Internet Things J. 7 (7), 6010–6022.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Knight, S., Nguyen, H.X., Falkner, N., Bowden, R., Roughan, M., 2011. The internet topology zoo. IEEE J. Sel. Areas Commun. 29 (9), 1765–1775.

Li, M., Pei, P., Yu, F.R., Si, P., Li, Y., Sun, E., Zhang, Y., 2022a. Cloud–edge collaborative resource allocation for blockchain-enabled Internet of Things: A collective reinforcement learning approach. IEEE Internet Things J. 9 (22), 23115–23129.

Li, G., Zhou, H., Feng, B., Zhang, Y., Yu, S., 2022b. Efficient provision of service function chains in overlay networks using reinforcement learning. IEEE Trans. Cloud Comput. 10 (1), 383–395.

Liu, Y., Lu, H., Li, X., Zhang, Y., Xi, L., Zhao, D., 2020. Dynamic service function chain orchestration for NFV/MEC-enabled IoT networks: A deep reinforcement learning approach. IEEE Internet Things J. 8 (9), 7450–7465.

Lu, Y., Huang, X., Zhang, K., Maharjan, S., Zhang, Y., 2020. Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks. IEEE Trans. Ind. Inform. 17 (7), 5098–5107.

Orlowski, S., Wessäly, R., Pióro, M., Tomaszewski, A., 2010. SNDlib 1.0—Survivable network design library. Networks 55 (3), 276–286.

Schneider, S., Khalili, R., Manzoor, A., Qarawlus, H., Schellenberg, R., Karl, H., Hecker, A., 2021. Self-learning multi-objective service coordination using deep reinforcement learning. IEEE Trans. Netw. Serv. Manag. 18 (3), 3829–3842.

Sun, W., Lei, S., Wang, L., Liu, Z., Zhang, Y., 2020. Adaptive federated learning and digital twin for industrial internet of things. IEEE Trans. Ind. Inform. 17 (8), 5605–5614.

Sun, C., Wu, X., Li, X., Fan, Q., Wen, J., Leung, V.C., 2021. Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic. IEEE Trans. Netw. Sci. Eng. 1. http://dx.doi.org/10.1109/TNSE.2021.3076795 (Early Access).

Tang, Q., Xie, R., Yu, F.R., Chen, T., Zhang, R., Huang, T., Liu, Y., 2022. Collective deep reinforcement learning for intelligence sharing in the internet of intelligence-empowered edge computing. IEEE Trans. Mob. Comput. (Early Access).

Tao, F., Zhang, M., Liu, Y., Nee, A., 2018. Digital twin driven prognostics and health management for complex equipment. CIRP Ann. 67 (1), 169–172. http://dx.doi.org/10.1016/j.cirp.2018.04.055, URL: https://www.sciencedirect.com/science/article/pii/S0007850618300799.

Tao, F., Zhang, H., Liu, A., Nee, A.Y.C., 2019. Digital twin in industry: State-of-the-art. IEEE Trans. Ind. Inform. 15 (4), 2405–2415. http://dx.doi.org/10.1109/TII.2018.2873186.

Wu, J., Wei, Z., Li, W., Wang, Y., Li, Y., Sauer, D.U., 2020. Battery thermal-and health-constrained energy management for hybrid electric bus based on soft actor-critic DRL algorithm. IEEE Trans. Ind. Inform. 17 (6), 3751–3761.
Wu, Y., Zhang, K., Zhang, Y., 2021. Digital twin networks: a survey. IEEE Internet Things J. 8 (18), 13789–13804.
Yu, F.R., 2021. From information networking to intelligence networking: Motivations, scenarios, and challenges. IEEE Netw. 35 (6), 209–216.
Zhang, J., Liu, Y., Qin, X., Xu, X., 2021. Energy-efficient federated learning framework for digital twin-enabled industrial internet of things. In: Proc. 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC, pp. 1160–1166.
Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J., 2019. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proc. IEEE 107 (8), 1738–1762.
Zhu, Y., Yao, H., Mai, T., He, W., Zhang, N., Guizani, M., 2022. Multiagent reinforcement-learning-aided service function chain deployment for Internet of Things. IEEE Internet Things J. 9 (17), 15674–15684.

Zhongwei Huang is currently pursuing the Ph.D. degree in Artificial Intelligence with the School of Computer Science and Engineering, Macau University of Science and Technology, Macau, China. He is also a visiting scholar at the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), 2022. His current research interests include Network Function Virtualization (NFV), Deep Reinforcement Learning (DRL), and its application in the Internet of Things (IoTs).

Dagang Li is currently an assistant professor at Macau University of Science and Technology, Macau SAR, China. He received his Bachelor's degree from Huazhong University of Science and Technology in Wuhan, China, and Ph.D. from Katholieke Universiteit Leuven in Leuven, Belgium. His research interests include artificial intelligence and its applications in future networks.

Jun Cai is currently a professor and dean of the School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou, China. He received the B.S. degree from Hunan Normal University, Changsha, China, the M.S. degree from Jinan University, Guangzhou, China, and the Ph.D. degree from Sun Yat-Sen University, China in 2003, 2006, and 2012, respectively. He is interested in the research of network function virtualization (NFV), software-defined networks (SDN), and complex networks.

Hua Lu, Director of Network Technology Innovation Center, Guangdong Communication & Network Institute. Distinguished Professor of Shenzhen Research Institute, Beijing University of Posts and Telecommunications. Distinguished Expert of "Technology China" Future Network Professional Technology Service Team. Research directions include: 5G/6G core network, edge computing, new network architecture, software-defined network, P4 programmable, virtualization, etc.