SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2021, Vol. 64 182302:1–182302:21
https://doi.org/10.1007/s11432-020-3094-6

Energy-efficient URLLC service provisioning in softwarization-based networks

Mengjie LIU[1], Gang FENG[1*] & Weihua ZHUANG[2]

[1]*National Key Laboratory on Communications, University of Electronic Science and Technology of China, Chengdu* 610054*, China;*
[2]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo* N2L 3W8*, Canada*

Received 23 May 2020/Revised 27 August 2020/Accepted 20 October 2020/Published online 9 July 2021

Abstract Software defined networking (SDN) and network function virtualization (NFV) as new technologies have shown great potential in improving the flexibility of resource management for network service provisioning. As traffic dynamics may cause violation of rigid service requirements, especially for ultra-reliability and low-latency communication (URLLC) service, it is essential yet challenging to dynamically allocate an appropriate amount of resources (including computation, transmission, and energy) to network functions (NFs) in softwarization-based networks. Meanwhile, with the explosion of high resource-demanding applications, the energy efficiency of communication networks deserves significant attention. In this paper, we investigate the dynamic network function resource allocation (NFRA) problem with aim to minimize long-term energy consumption while guaranteeing the requirements of URLLC services in softwarization-based networks. To cater for efficient on-line NFRA decisions, we design a distributed dynamic NF resource allocation (DDRA) algorithm based on dynamic value iteration (DVI). The convergence of the DDRA algorithm is proved. We conduct simulation experiments based on real-world data traces for performance evaluation. The numerical results demonstrate that the proposed DDRA algorithm achieves around 25% and 20% energy consumption reduction when compared with two benchmark algorithms, respectively.

Keywords softwarization-based networks, SDN/NFV, URLLC, energy efficiency, dynamic resource allocation, distributed value iteration

Citation Liu M J, Feng G, Zhuang W H. Energy-efficient URLLC service provisioning in softwarization-based networks. Sci China Inf Sci, 2021, 64(8): 182302, https://doi.org/10.1007/s11432-020-3094-6

1 Introduction

Ultra-reliability and low-latency communications (URLLC) as a new service category has the potential to enable a broad range of real-time services, such as e-healthcare, self-driving cars, mission critical applications, industrial automation, and augmented reality (AR) [1]. However, the quality enhancement of URLLC services (i.e., reliability and latency) over the limited network resources necessitates new technical solutions.

Software defined networking (SDN) and network function virtualization (NFV) as new technologies facilitate network function (NF) deployment for service providers. SDN, which is very suitable to work with NFV, bears the responsibility of managing applications running on common commodity hardware instead of proprietary hardware. SDN could greatly reduce the cost and increase the flexibility and programmability of NFs in the networks due to the separation of control from the data forwarding [2]. A specific network service can be provisioned by an ordered sequence of required NFs, called service function chain (SFC) [3]. For example, the network service to support AR applications may need a sequence of NFs such as firewall, filtering, security, image, and voice processing. An SFC is orchestrated on a subset of nodes connected by links in the physical/substrate network. NF instances created for an SFC are allocated a suitable amount of physical resources on substrate nodes (including computation and transmission resources), and interconnected by specifying forwarding rules, so that the packets of service flows are processed following the SFC.

* Corresponding author (email: fenggang@uestc.edu.cn)

On the other hand, energy efficiency has proved to be a cost-effective deployment goal in the communication networks, as it determines the cooling costs, financial gains, processing load balancing, and node reliability, of the commodity servers in the substrate network [4]. To meet the low-latency and ultra-reliability requirements of URLLC services, sufficient resources for computing and transmission should be allocated, which may conflict with energy-efficient network deployment targets. Actually, the trade-off between the requirements of URLLC services and the energy efficiency of the system has received extensive attention in wireless networks [5]. The wired core networks are responsible for the major parts of the consumed energy in the communication networks, because the core networks with stronger computation and storage capability than wireless networks install and run a large amount of NFs demanding high computation capacity [6]. In this study, we consider striking the balance between the transmission requirements of URLLC services and energy efficiency in the softwarization-based core networks.

Specifically, network function resource allocation (NFRA) for URLLC services faces two main challenges. One is how to analyze the reliability of URLLC services in the core networks and design an analytical model. Packet loss may happen when incoming packets overflow from the waiting queue of an NF, which could influence the reliability of URLLC services. Hence, the packet loss rate should also be taken into account in the core networks. The dynamics of incoming packets increase the complexity of analysis. The other is that the computational complexity of the centralized NFRA algorithms becomes extremely high with an explosive increase in the number of users and data usage [7–9]. As we know, the distributed algorithm can efficiently decrease the time complexity of algorithms. However, the convergence of a distributed algorithm is hard to be guaranteed, which needs careful corporation between distributed computing nodes.

In this paper, we propose a distributed dynamic NFRA algorithm (DDRA) by investigating the dynamic NFRA problem which is formulated as an infinite horizon Markov decision process (MDP) with the aim to minimize long-term average energy consumption in softwarization-based networks. The convergence of DDRA is proved in this paper. To improve the latency performance of URLLC services, we consider a general SFC framework [10], where parallel NF processing is allowed. Our main contributions can be summarized as follows:

• By modeling the dynamic traffic as a Markov chain model, we propose the packet loss rate model of URLLC services. We derive the closed-form expression of packet loss rate, which is used as an indicator to measure the service reliability. Specifically, we define the packet loss rate at an NF module as the probability that the queue length takes on a value larger than the maximum queue size [11, 12].

• A global optimal algorithm based on a value iteration algorithm is derived. As low-complexity on-line algorithms are of key importance in a dynamic environment, we further develop DDRA that is distributed value iteration (DVI) based. The convergence property of DDRA is proved to guarantee the effectiveness of DDRA as an on-line resource allocation mechanism.

• We conduct simulations to evaluate the performance of DDRA. Numerical results, based on the real-world trace from Google data center, demonstrate that the proposed DDRA algorithm achieves high energy efficiency in comparison with two benchmarks, whose solutions refer to CoordVNF [13] and BFDSU [14], respectively.

The remainder of this paper is organized as follows. We present a brief overview of related studies in Section 2. Section 3 describes the system model and the problem formulation. Section 4 presents the proposed solutions to the problem. We provide the numerical results as well as discussions in Section 5, and finally conclude this study in Section 6.

2 Related work

Performing dynamic NFRA is necessary for an SDN/NFV based network by flexibly scaling resources up or down for SFCs, as the packet arrival rate increases or decreases respectively. Meanwhile, the need for high resource utilization and low energy consumption in the system motivates research for dynamic NFRA mechanisms, which roughly fall into two categories based on whether the approach is reactive or proactive to a dynamic environment [8]. In the following, we provide a summary of on-line resource allocation mechanisms in these two categories.

The reactive algorithms make instantaneous decisions at a specific time based on detected workload [9, 14–17]. The resource reconfiguration and NF function migration need to be performed when the quality of service (QoS) requirements are violated. In this situation, the energy consumption for

migrating NF modules needs to be taken into account, which can be comparable to or even dominant in the energy consumption for packet forwarding and packet processing. Nguyen et al. [15] proposed a novel market-based resource allocation framework with the aim of computing a market equilibrium solution and allocating multiple types of resources of fog nodes to services in a fair and efficient manner. Zhou et al. [16] proposed a dynamic energy-efficient resource allocation scheme to make instantaneous resource allocation decisions in wireless networks at specific time instants rather than over a long period. The aforementioned algorithms are implemented by redeploying/migrating virtual network function modules to guarantee QoS requirements of services in a reactive way, which may lead to a high migration energy consumption.

The proactive NFRA mechanisms allocate network resources for SFCs with packet rate (resource demand) prediction [8,9,18] or without this prediction [7,19] in the near future based on historic data. In this paper, we consider proactive NFRA with packet rate prediction based on a real-world dataset. Various centralized on-line proactive approaches have been investigated in [7–9, 19]. However, the centralized algorithms suffer from high computational complexity when the number of SFCs is moderately large due to an explosive increase in the number of users and data usage.

To reduce the computational complexity, Han et al. [18] proposed an approximate MDP-based dynamic virtual machine (VM) management algorithm to minimize the long-term resource cost with the consideration of VM migration. However, the relationship between NFs in an SFC is not considered. In this paper, we propose a distributed NFRA algorithm to reduce the computation complexity and consider the general SFC framework with parallelizable NFs, whereas in the traditional SFC framework, NFs are chained in a sequential way. Furthermore, existing studies on resource allocation for NFs mainly focus on QoS requirements in terms of latency [20]. Different from existing studies, we consider meeting QoS requirements in terms of both end-to-end delay as well as packet loss rate.

3 System model and problem formulation

In this section, we describe the network model, traffic model (packet rate prediction model), energy consumption model, end-to-end delay model, and packet loss rate estimation model. Based on these models, we formulate the dynamic NFRA problem. Time is partitioned into intervals of constant duration. For clarity, we list the major parameters and variables in Table 1.

3.1 Network model

We consider an SDN/NFV based network model, as illustrated in Figure 1, where a control platform lies between the services and the substrate network. Based on the information collected from the substrate network and services (QoS demands, resource capability, and traffic information), the controller(s) in the control platform computes an NFRA solution of a control policy which determines the assignment of processing cores, the NF mapping, and the link mapping policies for each service. A virtual node in an SFC is mapped onto a substrate node. A virtual link is mapped onto a substrate link or a set of substrate links. In the substrate network, incoming packets look up a cascade of flow switch tables in the main flow switch, the NF switch, and the physical switch. The main flow switch table is preconfigured as the first table of the data path to forwarding data packets coming from different nodes. The NF switch table forwards data for the packet queue of different NFs. The physical switch delivers data on specific physical channels or substrate network links. In this way, packet forwarding and routing decisions are completely separated.

We model the substrate network as a weighted directed graph $G = (\boldsymbol{V}, \boldsymbol{L})$, where $\boldsymbol{V} = \{v|1 \leqslant v \leqslant V\}$ and $\boldsymbol{L} = \{l|1 \leqslant l \leqslant L\}$ are two sets containing substrate nodes and substrate links respectively. The set of switches is denoted by $\boldsymbol{C} = \{c\,|\,1 \leqslant c \leqslant C\}$. Each node has a switch used for packet forwarding, i.e., $V = C$. Once an NF is mapped onto node v, the physical switch in node v will be active for forwarding packets.

Moreover, we consider a general framework of SFCs, called generalized SFC (GSFC), in the dynamic NFRA problem to meet rigid delay requirements of services. We define the connection relationship among the required NFs by a GSFC as a service function graph. Specifically, in the service function graph of a GSFC, the NF pairs that have no dependency can be executed in parallel, e.g., Firewall and Monitor in Figure 2. Actually, at the service layer, 41.5% of existing NF pairs have no dependency and can be executed in parallel without causing extra resource overhead [21]. In this paper, we consider a

Table 1 Parameters and variables

Notation	Parameter
$B_{n,t}$	Bandwidth cost of virtual link ι_n in time-interval t
D_l	Propagation of one packet traversing substrate link $l \in \boldsymbol{L}$
E_v	Maximum available energy provided by node $v \in \boldsymbol{V}$ in each time-interval
$\mathcal{F}_i$	Set of NFs of GSFC i.
$\mathcal{G}_i$	A weighted directed graph of general SFC (GSFC) i
I_n	Maximum queue size of NF F_n
L_l	Maximum available bandwidth of substrate link $l \in \boldsymbol{L}$
$\mathcal{L}_i$	Set of virtual links of GSFC i
M_n	Set of feasible cores for NF F_n
N_r	Number of NFs in sub-chain $P_{i,j}^r$
Q	Set of all possible packet arrival rate
R_i	Maximum packet loss rate of GSFC i
S_1	Energy consumption of an active port of a switch
S_2	Energy consumption of the chassis of a switch
$S_{i,n}^t$	Event that the buffer of NF F_n in GSFC i is not full when a packet arrives at the buffer of F_n in time-interval t
T_i	Maximum end-to-end latency of GSFC i
$U_{i,n}^t$	Event that there are U packets in the buffer of NF $F_n \in \mathcal{F}_i$ in time-interval t
X	Number of GSFCs in the network
Y	Number of NFs in the network
ν_n	Migration volume (memory size) of NF F_n
ϑ	Energy consumption per GB of migration NF volume
Notation	**Variable**
m_t	Core assignment action in time-interval t
k_t	Virtual link mapping action in time-interval t
ω_t	NF mapping action in time-interval t
π	Control policy
$\varsigma_{n,t}$	Binary variable indicating whether NF F_n migrates to another substrate node or not in time-interval t
$\xi_{l,t}$	Binary variable indicating whether substrate link $l \in \boldsymbol{L}$ is active or not in time-interval t
$\zeta_{c,t}$	Binary variable indicating whether switch $c \in \boldsymbol{C}$ is active or not in time-interval t

general model for NF, which covers any type of NF. The GSFC model thus covers any type of URLLC applications. Let i be the index of GSFCs. In GSFC i, let $\mathcal{F}_i$ and $\mathcal{L}_i$ be the set of NFs and the set of virtual links respectively. We model GSFC i as a weighted directed graph $\mathcal{G}_i$ which is an ordered sequence of sub-graphs as shown in Figure 2, i.e., $\mathcal{G}_i = \{P_{i,j} \,|\, 1 \leqslant j \leqslant J_i\}$, where $P_{i,j}$ is the jth sub-graph of $\mathcal{G}_i$, and J_i represents the maximum number of sub-graphs in $\mathcal{G}_i$. Sub-graph $P_{i,j}$ is composed of R_{ij} sub-chains, i.e., $P_{i,j} = \{P_{i,j}^r | 1 \leqslant r \leqslant R_{ij}\}$. Sub-chain $P_{i,j}^r$ consists of an ordered sequence of NFs and virtual links, i.e., $P_{i,j}^r = \{F_1, \iota_1, \ldots, F_n, \iota_n, \ldots, F_{N_r-1}, \iota_{N_r-1}, F_{N_r} | F_n \in \mathcal{F}_i, \iota_n \in \mathcal{L}_i\}$, where N_r denotes the number of NFs in sub-chain $P_{i,j}^r$. Note that the start and end NFs of all sub-chains in the same sub-graph are always equipped with a merging module and are usually called merging-NFs. The backups/copies of the same packet processed by the parallel sub-chains merge at the merging-NF into one packet, which is then sent to the next node/NF [21]. In the example of Figure 2, VPN and Transcoder are the Merging-NFs of sub-chain (VPN, Firewall, Transcoder) and sub-chain (VPN, Monitor, Transcoder). Without loss of generality, sub-graph $P_{i,j}$ can represent one sub-chain (e.g., $P_{1,2}$ in Figure 2), such that the conventional SFC can be considered as a special case of the GSFC.

3.2 Traffic model

As in some existing studies [18, 22, 23], Google cluster data traces can be used to simulate real long-term data traces of VMs and NFs. Here, we use the workload trace of a task in Google cluster to model the workload trace of an NF in the substrate network. Google cluster trace records the resource utilization of the machines in Google's data-center in the form of jobs and task events. A job consists of multiple tasks which are interconnected by specific forwarding rules. Each task represents a program to run on a single machine. As the relationship between an SFC and its NFs is similar to that between a job and its tasks, we use the workload trace of a task to model the workload trace of an NF [18, 24].

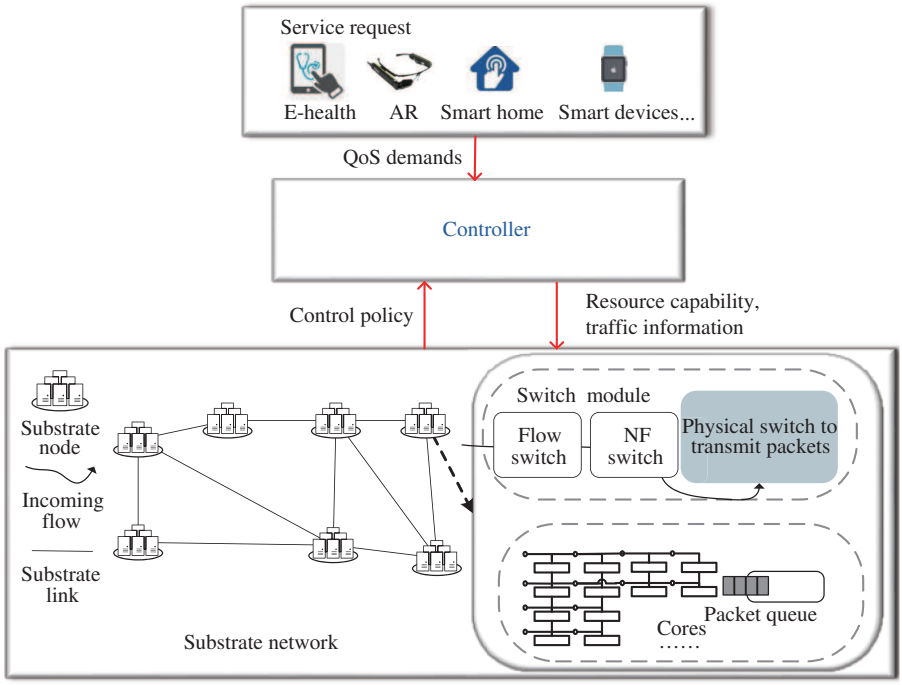

Figure 1 (Color online) An overview of the system model.

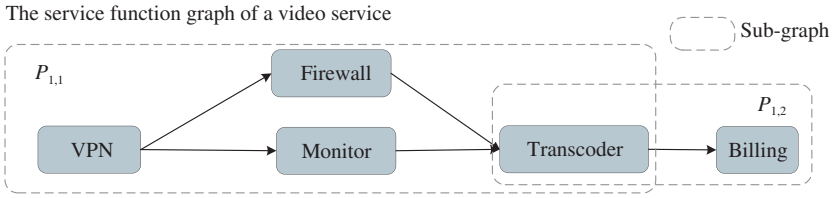

Figure 2 (Color online) An example of the service function graph of a video service.

By analyzing the characteristics of the trace in Google cluster [24], we use the finite-state stationary Markov chain model to estimate the temporal correlation of the packet arrival rate of an NF. A Markov chain is a stochastic model in which among a set of possible events the probability of each event depends only on the state in the previous time-interval. Note that we define the packet arrival rates at NFs for a short time interval. While the packet arrival rates at NFs may not strictly follow a Markov chain, it shares some of the common features of a Markov chain [18]. In Subsection 5.1, we provide experimental results to validate the Markov chain model used as an approximation to capture dynamics of the real-world trace.

Based on this observation, we define the transition matrix of workload traces (the packet arrival rates) of all NFs. Let X denote the number of SFCs in the system, $\lambda'_{i,n}$ (packet/s) the packet arrival rate of NF F_n and $\lambda'_i = (\lambda'_{i,1}, \ldots, \lambda'_{i,Y}) \in Q$, where $Y = \sum_{i=1}^{X} |\mathcal{F}_i|$ is the number of NFs in the system and Q the set of all possible λ'_i. Note that λ'_i indicates a state of the packet arrival rates at NFs in the system, and there are in total $|Q|$ possible such states, where $|\cdot|$ denotes the cardinality of a set. We can obtain the following transition matrix of the packet arrival rates at NFs in time-interval t:

$$\mathcal{T}_t = \begin{bmatrix} \mathcal{T}^t_{1,1} & \mathcal{T}^t_{1,2} & \cdots & \mathcal{T}^t_{1,|Q|} \\ \mathcal{T}^t_{2,1} & \mathcal{T}^t_{2,2} & \cdots & \mathcal{T}^t_{2,|Q|} \\ \vdots & \vdots & & \vdots \\ \mathcal{T}^t_{|Q|,1} & \mathcal{T}^t_{|Q|,2} & \cdots & \mathcal{T}^t_{|Q|,|Q|} \end{bmatrix},$$

where $\mathcal{T}^t_{k,l}$ represents the transition probability (kernel) from state λ'_l in time-interval t to state λ'_k in

time-interval $t+1$. Based on the assumption that the packet arrival rate at one NF follows a Markov chain mode, the event that the average number of packets arriving at the NF in one time slot is dependent on that of the same NF in the previous time slot. These kinds of events (state transitions) are independent of each other since they are considered in the time dimension based on the Markov chain model, no matter for the same NF or different NFs. As the service flows in different SFCs are isolated via service function chains, the packet flow rates of different SFCs are independent of each other. Above all, the packet arrival rate of one NF at time-interval t only depends on the packet arrival rate of this NF at time-interval $t-1$. According to Burke's theorem [25], the output of an M/M/1 queue with arrival rate λ is a Poisson process of rate λ. Note that the length of the output packet of the NFs can be scaled up or down linearly by some NFs (e.g., an encryptor or a filter). Then, we can assume that the packet arrival rates at the NFs in an SFC also follow a Markov chain. Hence, the transition kernel is given by

$$\begin{aligned}\mathcal{T}_{k,l}^{t} &= \Pr\left(\boldsymbol{R}_{t+1} = \lambda_k' | \boldsymbol{R}_t = \lambda_l'\right) \\ &= \prod_{n=1}^{Y} \Pr\left(\lambda_{n,t+1} = \lambda_{k,n}' \middle| \lambda_{n,t} = \lambda_{l,n}'\right) \\ &= \prod_{n=1}^{Y} \mathcal{T}_{k,l,n}^{t}, \end{aligned} \tag{1}$$

where $\Pr(\cdot)$ denotes the probability mass function; $\boldsymbol{R}_t = (\lambda_{1,t}, \ldots, \lambda_{Y,t})$ denotes the vector of packet arrival rates of NFs; and $\lambda_{n,t}$ denotes the packet arrival rate of NF F_n in time-interval t. We denote by $R = \{r_1, r_2, \ldots, r_a\}$ the set of all possible packet arrival rates, i.e., $\lambda_{n,t} \in R$.

3.3 Energy consumption model

Without loss of generality, we assume that the length of the time interval is 1 unit of time. Hence, the time interval is omitted in the following energy consumption model. The consumed energy of the system mainly consists of computational, forwarding, and migration energy [26]. A virtual NF module in a GSFC can be mapped to one core or multiple cores to implement multi-core processing in a substrate node [10,27]. We denote the core assignment action for all NFs in time-interval t by set $m_t = \{m_{n,t} | 1 \leqslant n \leqslant Y\}$, where $m_{n,t} \in M_n$ denotes the number of cores assigned to NF F_n in time-interval t and M_n denotes the set of feasible cores for NF F_n. Recall that Y denotes the number of NFs in the system. Specifically, the computational energy consumption in time-interval t is modelled as

$$P_c(m_t) = \sum_{n=1}^{Y} p_c(m_{n,t}), \tag{2}$$

where the approximation model $p_c(m_{n,t}) = m_{n,t}(S_{\text{idle}} + (S_{\max} - S_{\text{idle}}) d_n(m_{n,t})\lambda_{n,t})$ [18], widely adopted in related studies, is used to evaluate the computational energy consumption of NF F_n; S_{idle} and $S_{\max}$ denote the power consumption when the core is in the idle state and when the core in the fully-loaded state (100% utilization of CPU), respectively; Function $d_n(\cdot)$ maps the number of cores m to $d_n(m)$ which is the time duration of NF F_n processing a packet on m cores. Specifically, $d_n(m) = d_n(1)/S_n(m)$, where speed function $S_n(m)$ describes how much faster NF F_n processing a packet on m cores when compared with $d_n(1)$ according to Amdahl's law [27]. More details about $S_n(\cdot)$ are given in Section 5. Moreover, $d_n(m_{n,t})\lambda_{n,t}$ indicates the CPU utilization of NF F_n in time-interval t.

The forwarding energy consumption is mainly determined by the chassis in switches and by the number of active ports [28]. As substrate link $l_{s,v} \in \boldsymbol{L}$ requires two ports between substrate node s and node v, the total energy consumption of the switches in the system in time-interval t is given by

$$P_s(\boldsymbol{\xi}_t, \boldsymbol{\zeta}_t) = 2S_1 \boldsymbol{\xi}_t \cdot \boldsymbol{e}^{\mathrm{T}} + S_2 \boldsymbol{\zeta}_t \cdot \boldsymbol{e}^{\mathrm{T}}, \tag{3}$$

where S_1 is the amount of energy consumption of each active port of a switch; S_2 is the amount of energy consumption of the chassis of a switch; $\boldsymbol{e}$ is an all-ones row vector of dimension L; $\boldsymbol{\xi}_t = [\xi_{l,t}]_{1\times L}$ is a vector of binary variable where $\xi_{l,t} = 1$ when link $l \in \boldsymbol{L}$ is active in time-interval t, and $\xi_{l,t} = 0$ otherwise; and $\boldsymbol{\zeta}_t = [\zeta_{c,t}]_{1\times C}$ is a vector of binary variables where $\zeta_{c,t} = 1$ when switch $c \in \boldsymbol{C}$ is active in time-interval t, and $\zeta_{c,t} = 0$ otherwise.

As the NF migration energy increases with the migrated traffic volume of the NF [26], the migration energy consumption of NFs in the system in time-interval t is given by

$$P_m(\boldsymbol{\varsigma}_t) = \vartheta \boldsymbol{\nu} \cdot \boldsymbol{\varsigma}_t^{\mathrm{T}}, \tag{4}$$

where ϑ is the energy consumption per GB of the migration NF volume; $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_Y)$ is the vector of migration traffic volume where ν_n is the memory size of NF F_n; and $\boldsymbol{\varsigma}_t = (\varsigma_{1,t}, \ldots, \varsigma_{Y,t})$ is a vector of binary variable where $\varsigma_{j,t} = 1$ indicates NF F_j migrates to another different substrate node in time-interval t, and $\varsigma_{j,t} = 0$ otherwise.

The energy consumption in node v in time-interval t according to (2) and (3) is given by

$$e_{v,t}(m_t, \omega_t) = \sum_{n=1}^{Y} p_c(m_{n,t}) \omega_{n,v,t} + S_1 a_v + S_2 \left(1 - \prod_{n=1}^{Y} (1 - \omega_{n,v,t})\right), \tag{5}$$

where the NF mapping action in time-interval t is represented by set $\omega_t = \{\omega_{n,v,t} \,|\, 1 \leqslant n \leqslant Y\}$ with $\omega_{n,v,t} = 1$ if NF F_n is mapped to substrate node $v \in \boldsymbol{V}$ in time-interval t, and $\omega_{n,v,t} = 0$ otherwise; a_v is the number of active links connecting node v and is determined by $\boldsymbol{\xi}_t$. Note that $\boldsymbol{\xi}_t$ is directly determined by the link mapping action $k_t = \{k_{i,t} | 1 \leqslant i \leqslant X\}$ in time-interval t, i.e.,

$$\xi_{l,t} = 1 - \prod_{i=1}^{X} \prod_{\iota_n \in \mathcal{L}_i} (1 - k_{i,n,l,t}),\ l \in \boldsymbol{L}, \tag{6}$$

where the link mapping action for GSFC i in time-interval t is represented by a set of binary variable, i.e., $k_{i,t} = \{k_{i,n,l,t} | \iota_{n,n+1} \in \mathcal{L}_i\}$ in which $k_{i,n,l,t} = 1$ indicates that virtual link $\iota_{n,n+1} \in \mathcal{L}_i$ is mapped to substrate link $l \in \boldsymbol{L}$ in time-interval t, and $k_{i,n,l,t} = 0$ otherwise.

3.4 Delay model

Let the end-to-end delay of a packet traversing a GSFC be the duration between its generation at the source node and its arrival at its sink node, which consists of propagation, queuing, and processing delays. An accurate end-to-end delay model is analyzed in [3], where the traffic multiplexing is considered for achieving multiplexing gain. For simplicity, the multiplexing procedure is omitted here. In GSFC, as the copies of packets processed by the parallel sub-chains merge at the Merging-NF, the maximum delay of all sub-chains of one sub-graph decides the delay of the sub-graph. Recall that J_i is the number of sub-graphs in $\mathcal{G}_i$ and R_{ij} is the number of sub-chains in sub-graph $P_{i,j}^r$. Thus, the delay is given by

$$\begin{aligned} t_i(m_{i,t}, k_{i,t}) = \sum_{j=1}^{J_i} \max_{1 \leqslant r \leqslant R_{ij}} \left\{ \sum_{\iota_n \in P_{i,j}^r} D_l k_{i,n,l,t} + \sum_{n=1}^{N_r - 1} (d_n(m_{i,n,t}) + q_n(m_{i,n,t})) \right\} \\ + d_{|\mathcal{F}_i|}(m_{i,|\mathcal{F}_i|,t}) + q_{|\mathcal{F}_i|}(m_{i,|\mathcal{F}_i|,t}), \end{aligned} \tag{7}$$

where the core assignment action for GSFC i in time-interval t is represented by set $m_{i,t} = \{m_{i,n,t} | F_n \in \mathcal{F}_i\}$ in which $m_{i,n,t}$ indicates the number of cores assigned to NF $F_n \in \mathcal{F}_i$ in time-interval t; the queuing delay of a packet in the waiting queue of NF F_n assigned with m cores is given by $q_n(m)$; the propagation delay of one packet traversing substrate link l is fixed and denoted by D_l; $d_{|\mathcal{F}_i|}(m)$ and $q_{|\mathcal{F}_i|}(m)$ denote the processing delay and queuing delay of one packet at the last NF $F_{|\mathcal{F}_i|}$ assigned with m cores in GSFC i, respectively.

In addition, the queuing delay is affected by the time-varying traffic load. For simplicity, we consider Poison packet arrivals at NFs in each time interval. The transition probability function of Poison packet arrival rate at an NF in two adjacent time intervals is provided by the Markov chain model (1) and the method of predicting this probability is shown in (15). We model the queuing delay for one packet arriving at NF F_n as the expected value

$$q_n(m) = \frac{\lambda_{i,n,t} d_n(m)}{\lambda_{i,n,t}(1 - \lambda_{i,n,t} d_n(m))}, \tag{8}$$

where $\lambda_{i,n,t}$ is Poison packet arrival rate at NF F_n in time-interval t and $\lambda_{i,n,t} d_n(m) = \rho_n(m)$ is the traffic intensity. Furthermore, to ensure the processing system stability, we have $\rho_n(m) < 1$.

3.5 Estimation of packet loss rate

An NF module in the substrate node serves incoming packets one at a time from the front of the queue based on the first-come first-served discipline. According to the most popular queue management scheme, i.e., Droptail [11, 12], an arrival packet is dropped when the buffer is full, as the buffer size (storage capacity) of each NF module is limited in the substrate nodes. Let I_n be the buffer size of NF F_n. For clarify, let $S_{i,n}^t$ denote the event that the buffer of NF F_n in GSFC i is not full when a packet arrives at the buffer of F_n in time-interval t and $U_{i,n}^t$ the event that there are $U_{i,n}^t$ packets in the buffer of NF $F_n \in \mathcal{F}_i$ in time-interval t.

Let $\mathbb{P}(\cdot)$ be a probability measure. We denote by $r_{i,t}(m_{i,t})$ the loss rate of a packet traversing GSFC i in time-interval t, which can be expressed as

$$r_{i,t}(m_{i,t}) \overset{\text{(a)}}{=} 1 - \mathbb{P}\left(\bigcap_{F_n \in \mathcal{F}_i} S_{i,n}^t\right) \overset{\text{(b)}}{=} 1 - \prod_{F_n \in \mathcal{F}_i} \mathbb{P}(S_{i,n}^t). \tag{9}$$

In (9), the labelled equalities follow from the facts that: (a) the packet loss happens when the queue of NF $F_n \in \mathcal{L}_i$ is full and a packet arrives at the queue of F_n in time-interval t; (b) $S_{i,n}^t, F_n \in \mathcal{L}_i$ are disjoint events.

According to the M/M/1 queue model, $U_{i,n}^t$ is geometrically assigned with parameter $1 - \rho'$ where $\rho' = \rho_n(m_{i,n,t}) = \lambda_{i,n,t} d_n(m_{i,n,t})$ and $\lambda_{i,n,t}$ is the packet arrival rate of NF $F_n \in \mathcal{F}_i$ in time-interval t. Then we have

$$\mathbb{P}(U_{i,n}^t) = (1 - \rho')\,(\rho')^{U_{i,n}^t}. \tag{10}$$

Based on $\mathbb{P}(U_{i,n}^t)$, we obtain the cumulative distribution function:

$$\mathbb{P}(S_{i,n}^t) = \mathbb{P}(U_{i,n}^t \leqslant I_n) = \sum_{k=0}^{I_n} (1 - \rho')\rho'^k = 1 - \rho'^{I_n+1}. \tag{11}$$

Therefore, once the core assignment action $(m_{i,t})$ for GSFC i in time-interval t is determined, the packet loss rate of GSFC i can be obtained from (9)–(11). Note that the methods of fast link/node failure recovery have been studied in previous work, such as [29]. Therefore, in this study, we only consider the potential packet congestion in NF modules in the packet loss rate estimation model, without considering any potential link failure and node failure in the substrate network.

3.6 Problem formulation

Based on the aforementioned models, we now formulate the dynamic NFRA problem as an MDP. The MDP can be modelled as a 5-tuple $(\mathbb{S}, \mathbb{A}^X, \Pr, R, \mathcal{A})$, where $\mathbb{S}$ is a finite set of states as given in Definition 1; $\mathbb{A}^X$ is a finite set of actions; $\Pr(S'|S, A) = \Pr(S_{t+1} = S'|S_t = S, A_t = A)$ is the transition probability that action A in state S_t will lead to state S_{t+1}; R is the instantaneous cost function as $S \to \mathbb{R}$; and $\mathcal{A}(S_i)$ is the finite set of actions available from state S_i.

Definition 1. The system state in time-interval t is uniquely specified by $S_t = (\boldsymbol{R}_t, \boldsymbol{\Upsilon}_t) \in \mathbb{S}$ where $\boldsymbol{R}_t$ determines the packet arrival rates at the NFs in the system and $\boldsymbol{\Upsilon}_t = \{\Upsilon_{v,t}^1, \Upsilon_{v,t}^2, \Upsilon_{v,t}^3 | v \in \boldsymbol{V}\}$ determines the number of active cores $(\Upsilon_{v,t}^1)$, active ports $(\Upsilon_{v,t}^2)$, and active chassis $(\Upsilon_{v,t}^3)$ in switches.

A solution to an MDP is a control policy which maps states to actions to minimize the long-term average cost. Let A_t denote the NFRA action set at time interval t, which consists of the core assignment, virtual link mapping and NF mapping actions at time interval t in the system, i.e., $A_t = (m_t, k_t, \omega_t)$.

Definition 2. The control policy π is a mapping from the system states to NFRA action set A_t, i.e., $\pi(S_t) = A_t = (m_t, k_t, \omega_t) \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of actions available at state S_t.

Aiming to minimize the long-term average energy consumption while guaranteeing end-to-end delay and packet loss rate requirements of each SFC, we formulate a dynamic NFRA problem as Problem 1 (P1).

$$\text{P1}: \ \min_{\pi} P(\pi) = \lim_{T \to \infty} E_\pi \left[\frac{1}{T} \sum_{t=1}^{T} p(t)\right] \tag{12}$$

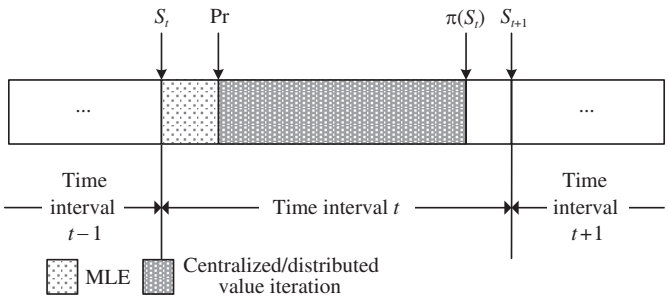

Figure 3 The illustration of the solution to P1.

$$\text{s.t. } \rho_n(m_{n,t}) \leqslant 1,\ \forall t, n, \tag{12a}$$
$$t_i(m_{i,t}, k_{i,t}) \leqslant T_i,\ \forall t, i, \tag{12b}$$
$$r_{i,t}(m_{i,t}) \leqslant R_i,\ \forall t, i, \tag{12c}$$
$$b_{l,t}(k_t) \leqslant L_l,\ \forall t, l, \tag{12d}$$
$$c_{v,t}(m_t, \omega_t) \leqslant C_v,\ \forall t, \tag{12e}$$
$$e_{v,t}(m_t, \omega_t) \leqslant E_v,\ \forall t, \tag{12f}$$
$$k_{i,n,l,t}\omega_{i,n,v,t}\omega_{i,n+1,v',t} = 1,\ l \in \{l_{v,v_1}, \ldots, l_{v_k,v'}\}^q,\ \forall t, i, \tag{12g}$$
$$m_{n,t} \in M_n, k_{n,t}, \omega_{n,t} \in \{0, 1\},\ \forall F_n,$$

where $P(\pi)$ is the long-term average energy consumption under control policy π; $p(t)$ is the instantaneous energy consumption in time-interval t, with $p(t) = P_c(m_t) + P_s(\boldsymbol{\xi}_t, \boldsymbol{\zeta}_t) + P_m(\boldsymbol{\varsigma}_t)$; Eq. (12a) guarantees the stability of processing system (core utilization needs to be less than 1); Eq. (12b) guarantees the delay requirement for GSFC i; Eq. (12c) guarantees the packet loss rate requirement for GSFC i and the maximum packet loss rate required by GSFC i is denoted by $R_i \in (0, 1)$. As the capacity of the substrate nodes and links is limited, we have (12d)–(12f) where L_l, C_v, and E_v denote the maximum available bandwidth, the maximum number of cores and the maximum available energy of substrate link l and node v in each time-interval respectively. The bandwidth consumption of substrate link l is given by $b_{l,t}(k_t) = \sum_{i=1}^{X} \sum_{\iota_n \in \mathcal{L}_i} B_{n,t} k_{n,l,t}$, where $B_{n,t}$ is the bandwidth consumption of virtual link ι_n in time-interval t. The core consumption of substrate node v is given by $c_{v,t}(m_t, \omega_t) = \sum_{i=1}^{X} \sum_{n=1}^{Y} m_{i,n,t}\omega_{i,n,v,t}$, where $\omega_{i,n,v,t} = 1$ if NF F_n of GSFC i is mapped to substrate node $v \in \boldsymbol{V}$ in time-interval t, and $\omega_{i,n,v,t} = 0$ otherwise. Constraint (12g) ensures the connection of adjacent NFs in the same SFC, where $\{l_{v,v_1}, \ldots, l_{v_k,v'}\}^q$ indicates the qth feasible path between substrate node v and node v'. The set of feasible paths can be found by the depth-first searching algorithm [30].

4 On-line algorithm design and analysis

4.1 Overview of the solutions

In this subsection, we first present an optimal solution to the dynamic NFRA problem which is a non-convex optimization problem. The high complexity of the optimal solution by using the centralized value iteration inspires us to develop a low-complexity distributed dynamic NFRA (DDRA) algorithm which is DVI based. The convergence property of the DVI is proved to guarantee the effectiveness of DDRA. Figure 3 shows the main idea of the solution to P1, where each time interval includes two time windows. In the first time window, the maximum likelihood estimation sliding window method (MLE) [31] uses historical traffic data to generate the transition probability matrix of system states, i.e., $\Pr \in \mathbb{R}^{|\mathbb{S}| \times |\mathbb{S}|}$. In the second time window, the centralized/distributed value iteration method makes use of Pr generated by MLE to calculate the control policy, $\pi(S_t)$. Note that an appropriately short time interval shall be set to catch the dynamics of traffic. The centralized value iteration with an exponential time complexity to compute the utility function and thus cannot be used as an on-line algorithm. In contrast, the DDRA algorithm with low complexity to compute the utility function can be used as an on-line mechanism to match the short time interval.

4.2 Optimal value iteration

As P1 is formulated as a model-based infinite horizon average MDP, the optimal policy can be obtained by solving the equivalent Bellman's equation [32]. Dynamic programming is a very powerful tool in solving optimal control problems. Considering the discrete variables in this paper, we use the value iteration as a dynamic programming algorithm to effectively solve Bellman's equation, which is the base for obtaining the optimal solution to an MDP, such as P1.

Lemma 1 (Equivalent Bellman's equation)**.** If scalar η and a vector of the utility values, i.e., $\mathcal{U} = (\mathcal{U}(S_1),\mathcal{U}(S_2),\ldots,\mathcal{U}(S_{|\mathbb{S}|}))$ satisfy the Bellman's equation (13) for P1, where the instantaneous power consumption $p(S_i)$ is obtained when the system is at state S_i, then η is the optimal average energy consumption for all $S_i \in \mathbb{S}$, i.e., $\eta = \min_\pi P(\pi) = P(\pi^*)$. Furthermore, policy π^* is optimal if the minimum is obtained in the R.H.S of (13).

$$\eta + \mathcal{U}(S_i) = \min_{A\in\mathcal{A}(S_i)} \left\{ p(S_i) + \sum_{S_j\in\mathbb{S}} \Pr(S_j|S_i,A)\mathcal{U}(S_j) \right\}. \tag{13}$$

Proof. See [32].

Given control policy π, we need to learn the one-step transition probability matrix $\Pr \in \mathbb{R}^{|\mathbb{S}|\times|\mathbb{S}|}$, where element $\Pr(S_t|S_{t-1},\pi(S_{t-1}))$ at row m and column l is derived as follows:

$$\begin{aligned}
\Pr(S_t|S_{t-1},\pi(S_{t-1})) &= \Pr(\boldsymbol{R}_t,\boldsymbol{\Upsilon}_t|\boldsymbol{R}_{t-1},\boldsymbol{\Upsilon}_{t-1},\pi(S_{t-1})) \\
&= \Pr(\boldsymbol{R}_t|\boldsymbol{R}_{t-1})\Pr(\boldsymbol{\Upsilon}_t|\boldsymbol{\Upsilon}_{t-1},\pi(S_{t-1})) \\
&\overset{\text{(a)}}{=} \Pr(\boldsymbol{R}_t = \lambda_i'|\boldsymbol{R}_{t-1} = \lambda_l') \\
&= \mathcal{T}_{m,l}^t \\
&= \prod_{n=1}^{Y} \mathcal{T}_{m,l,n}^t.
\end{aligned} \tag{14}$$

In (14), label (a) follows the fact that $\Pr(\boldsymbol{\Upsilon}_t|\boldsymbol{\Upsilon}_{t-1},\pi(S_l)) = 1$ according to the deterministic NF migration, core allocation and link allocation actions.

Using MLE [31] to estimate the transition probability of the packet arrival rate of NF F_n, we have

$$\mathcal{T}_{m,l,n}^t = \frac{\sum_{k=t-e}^{t} 1[r_n(k-1) = \lambda_m'|r_n(k) = \lambda_l']}{\sum_{k=t-e}^{t} 1[r_n(k-1) = \lambda_m']}, \tag{15}$$

where $1[\cdot]$ is the indicator function which is equal to 1 if the condition holds, and 0 otherwise; e is the length of the sliding window of MLE.

After generating transition probability matrix Pr, we can compute the optimal values of utility vector $\mathcal{U}^* = (\mathcal{U}^*(S_1),\mathcal{U}^*(S_2),\ldots,\mathcal{U}^*(S_{|\mathbb{S}|}))$ by using the value iteration with a dynamic programming mapping T as $\mathcal{U}^{k+1}(S_i) = T\mathcal{U}^k(S_i)$, $\forall S_i \in \mathbb{S}$, which can be expressed as

$$T\mathcal{U}^k(S_i) = \min_{A\in\mathcal{A}(S_i)} \left\{ p(S_i) + \sum_{S_j\in\mathbb{S}} \Pr(S_j|S_i,A)\mathcal{U}^k(S_j) \right\}. \tag{16}$$

In (16), the utility value $\mathcal{U}$ will eventually converge to the optimal utility value $\mathcal{U}^*$ which satisfies the Bellman's equation, i.e., $\eta + \mathcal{U}^* = T\mathcal{U}^*$ [32].

The Bellman's equation [18] is NP-hard by using the dynamic programming mapping T, as it needs an exponential time to compute the utility function. Specifically, T in (16) takes $O(|\mathbb{S}|)$ operations for calculating the R.H.S of (16), where $|\mathbb{S}|$ is the number of states in $\mathbb{S}$. Then, it takes $O(|\mathbb{A}^X|\log|\mathbb{A}^X|)$ for the sorting to get the minimal utility value. Hence, it takes $O_1 = O(|\mathbb{S}||\mathbb{A}^X|\log|\mathbb{A}^X|)$ operations to obtain the minimal utility values, where $|\mathbb{S}| = X^3Y^3|R||\boldsymbol{L}|$ and $|\mathbb{A}| = X^2Y^2|\boldsymbol{V}||M_n|\sum_{i=1}^{Y} C_{Y-1}^{i-1}C_{N_g}^{i}$. The number of mapping solutions is $\sum_{i=1}^{Y} C_{Y-1}^{i-1}C_{N_g}^{i}$ for a GSFC with Y NFs on one feasible path with N_g nodes [10]. Unfortunately, O_1 is high even for a problem with moderate number of SFCs. For example,

if the number of GSFCs X equals 5, and $|\mathbb{A}|$ equals 20, we have $|\mathbb{A}^X| = (|\mathbb{A}|)^X = 20^5$. Therefore, the dimension of the action space increases exponentially with the number of GSFCs and needs exponential time to calculate the utility function. This inspires us to further design an efficient algorithm to obtain a suboptimal coordination policy of all GSFCs in the following.

4.3 Distributed value iteration mechanism

As mentioned before, the solution of Bellman's equation corresponds to the global optimal solution of Problem 1. However, the centralized value iteration method cannot avoid high-dimensional action space of optimal control problems. In [33], a distributed iterative dynamic programming algorithm is proposed, in which the control policy of only one user is updated to optimize the cooperation policy of a multiple-users system. Inspired by the approach, we propose a distributed value iteration method, called DVI. We intuitively explain the key idea of DVI as follows. In our framework, the control policy of individual GSFCs is updated in a distributed dynamic iterative manner. Each GSFC maintains and updates a current global policy by policy sharing with all other GSFCs to ensure the feasibility and global optimality of the selected actions.

There exist essential differences between our proposed DVI and the algorithm in [33]. The latter requires constructing initial utility values for all states to ensure the monotonically decreasing property of the utility value, which is required to ensure the convergence of the algorithm in [33]. However, it is rather time-consuming to find such initial utility values for all states. To overcome this obstacle, inspired by the idea of relative value iteration [34], DVI randomly chooses a reference state, S_r, which can be any fixed state in space $\mathbb{S}$. In each iteration, utility value $\mathcal{U}(S)$ for state S is replaced by $\mathcal{U}(S) - \mathcal{U}(S_r)$. The convergence property of DVI can be proved by setting the initial utility values for all states to zero.

Before giving the distributed iteration functions, we define some notations. Let $\mathbb{X} = \{1, \ldots, X\}$ be the set of SFCs in the system. For $i \in \mathbb{X}$, let $a^i = (m_{i,t}, k_{i,t}, \omega_{i,t}) \in \mathcal{A}_i$ be the action for GSFC i, where $\mathcal{A}_i$ is the available action set for SFC i. The control policy is given by $A = (a^1, \ldots, a^X) \in \mathbb{A}^X$. Let $\overline{A}_{y_0} = \{a^{\overline{y}} \,|\, \overline{y} \in \mathbb{X}, \overline{y} \neq y_0\}$. Let $\widetilde{T}$ denote the dynamic programming mapping as $R \to R$. Let function $\delta(S, A)$ be defined as $\mathbb{S} \times \mathbb{A}^X \to \mathbb{S}$ with $\delta(S, A) = S'$.

Specifically, we modify the value function (16) into a set of sub-functions, i.e., $\{u_t^i, i \in \mathbb{X}\}$. Given state S_r, for $S \in \mathbb{S}$, $\theta_0 \in \mathbb{X}$ and $\iota = 0, 1, \ldots$, the DVI is given by

$$a_\iota^{\theta_0}(S) = \arg \min_{a^{\theta_0} \in \mathcal{A}_0(S)} \left\{ p(S) + \sum_{S' \in \mathbb{S}} \Pr\left(S' \middle| S, a_\iota^{\theta_0}(S), \overline{A}_{\theta_0}(S)\right) u_\iota^{\theta_0}(S') \right\}, \tag{17}$$

$$u_{\iota+1}^{\theta_0}(S) = \widetilde{T} u_\iota^{\theta_0}(S) = \min_{a^{\theta_0} \in \mathcal{A}_0(S)} \left\{ p(S) + \sum_{S' \in \mathbb{S}} \Pr\left(S' \middle| S, a^{\theta_0}, \overline{A}_{\theta_0}(S)\right) u_\iota^{\theta_0}(S') \right\}, \tag{18}$$

$$u_{\iota+1}^{\theta_0}(S) = u_{\iota+1}^{\theta_0}(S) - u_{\iota+1}^{\theta_0}(S_r), \ \forall S \in \mathbb{S}, \tag{19}$$

where $\mathcal{A}_0(S)$ is the set of feasible actions of SFC θ_0 at state S and $S' = \delta(S, a_\iota^{\theta_0}(S), \overline{A}_{\theta_0}(S))$. Then, we have $u_{\iota+1}^{\theta_0}(S) = p(S) + \sum_{S' \in \mathbb{S}} \Pr(S'|S, a_\iota^{\theta_0}(S), \overline{A}_{\theta_0}(S))$.

For $\iota \to \infty$, we define $u_\infty^{\theta_0}(S) = \lim_{\iota \to \infty} u_\iota^{\theta_0}(S)$ and $a_\infty^{\theta_0}(S) = \lim_{\iota \to \infty} a_\iota^{\theta_0}(S)$. Then, we construct a control sequence as the current global policy $A_{\theta_l}(S) = [a_l^1(S), \ldots, a_l^X(S)]$, where

$$a_l^y(S) = \begin{cases} a_{l-1}^y(S), & y \neq \theta_l, \\ a_\infty^y(S), & y = \theta_l. \end{cases} \tag{20}$$

For $\forall l = 1, 2, \ldots$, $\theta_l \in \mathbb{X}$ and $\iota = 0, 1, \ldots$, the DVI is given by

$$a_\iota^{\theta_l}(S) = \arg \min_{a^{\theta_l} \in \mathcal{A}_l(S)} \left\{ p(S) + \sum_{S' \in \mathbb{S}} \Pr\left(S' \middle| S, a_\iota^{\theta_l}(S), \overline{A}_{\theta_l}(S)\right) u_\iota^{\theta_l}(S') \right\}, \tag{21}$$

$$u_{\iota+1}^{\theta_l}(S) = \widetilde{T} u_\iota^{\theta_l}(S) = \min_{a^{\theta_l} \in \mathcal{A}_l(S)} \left\{ p(S) + \sum_{S' \in \mathbb{S}} \Pr\left(S' \middle| S, a^{\theta_l}, \overline{A}_{\theta_l}(S)\right) u_\iota^{\theta_l}(S') \right\}, \tag{22}$$

$$u_{\iota+1}^{\theta_l}(S) = u_{\iota+1}^{\theta_l}(S) - u_{\iota+1}^{\theta_l}(S_r), \ \forall S \in \mathbb{S}, \tag{23}$$

where $S' = \delta(S, a_\iota^{\theta_l}(S), \overline{A}_{\theta_l}(S))$. For $\iota \to \infty$, we define $u_\infty^{\theta_l}(S) = \lim_{\iota\to\infty} u_\iota^{\theta_l}(S)$ and $a_\infty^{\theta_l}(S) = \lim_{\iota\to\infty} a_\iota^{\theta_l}(S)$. We can construct a control sequence as the current global policy $A_{\theta_{l+1}}(S) = [a_{l+1}^1(S), \ldots, a_{l+1}^X(S)]$ according to (20).

Remark 1. (1) For $\forall i \in \mathbb{X}$, let $a_0^i(S)$ be the initial action at state S and $A_{\theta_0}(S) = (a_0^1(S), \ldots, a_0^Y(S))$ be the initial control sequence. The initial control sequence, $A_{\theta_0}(S)$, satisfies the delay and stability constraints (12a)–(12c); (2) Note that a^{θ_l} should also satisfy constraints (12d)–(12f). If $a^{\theta_l} = a' \in \mathcal{A}_l(S)$ violates any of (12d)–(12f), we denote the instantaneous energy consumption in time-interval t by $p_t(S') = M$, where $S' = \delta(S, a', \overline{A}_{\theta_l}(S)), M \in \mathbb{R}^+$, and M is a very large number which satisfies $M \gg E_s$; (3) For given state S_r, we have $u_\iota^{\theta_l}(S_r) = 0$; (4) We set $u_{\iota=0}^{\theta_0}(S) = 0$ for $\forall S \in \mathbb{S}$.

Based on the above preparations, we now summarize the DDRA algorithm as Algorithm 1. In DDRA for the time slotted system as illustrated in Figure 3, the controller initializes the utility function, $u_{\iota=0}^{\theta_0}(S) = 0$, for all individual GSFCs. Algorithm 1 mainly consists of two steps, i.e., MLE and DVI. At the beginning of a time interval, the controller makes use of the MLE and the historical data to predict the transition kernel. Next, the DVI makes use of the MLE result to compute the control policy for each GSFC in time-interval t and determine the control action for each SFC. The lower bound of the minimum number of iterations for sub-function u_t^i in each time-interval is determined by the computation precision $\xi > 0$.

Algorithm 1 DDRA Algorithm

1: **Initialization:**
2: Set the computation precision $\xi > 0$ and the number of total time-intervals D;
3: Fix a state $S_r \in \mathbb{S}$;
4: Set $l = 0$ and $u_{\iota=0}^{\theta_0}(S) = 0$, $\theta_0 \in \mathbb{X}$ for $\forall S \in \mathbb{S}$;
5: **while** $t < D$ **do**
6: In time-interval t:
7: Each NF updates the transition probabilities according to Eq. (15);
8: Let action $a_\iota^{\theta_l}(S)$ be updated by (21) and value function $u_{\iota+1}^{\theta_l}(S)$ be updated by (22) and (23);
9: If $|u_{\iota+1}^{\theta_l}(S) - u_\iota^{\theta_l}(S)| \leqslant \xi$, then let $\iota = 0$, $u_0^{\theta_{l+1}}(S) = u_{\iota+1}^{\theta_l}(S)$ and go to the next step. Otherwise, let $\iota = \iota + 1$ and goto step 8;
10: If $|u_0^{\theta_{l+1}}(S) - u_0^{\theta_l}(S)| \leqslant \xi$, go to the next step. Otherwise, let $l = l + 1$ and $\iota = 0$, go to step 9;
11: Construct control action under the current system state S_t, which is $A_t(S_t) = [a_{l+1}^1(S_t), \ldots, a_{l+1}^X(S_t)]$ according to (20). Set $t = t + 1$;
12: **end while**
13: Return $P^*(\pi) = u_0^{\theta_l}(S_r)$ and optimal control policy $\pi = [A_1(S_1), A_2(S_2), \ldots]$.

The DVI is developed to avoid the huge dimension of action space. In the DVI, only the action of each SFC is used to update the value functions. Then, the computational complexity is dramatically reduced from $O_1 = O(|\mathbb{S}||\mathbb{A}^X| \log |\mathbb{A}^X|)$ to $O_2 = O(|\mathbb{S}||\mathbb{A}| \log |\mathbb{A}|)$. We thus have $O_2 \ll O_1$.

4.4 Convergence of DVI

Before analyzing the local and global convergence property of the DVI, we have Lemma 2.

Lemma 2. Let P_{a_j} denote the transition probability matrix when taking action a_j. There exists a positive integer m such that for every admissible policy $\pi = \{a_0^{\theta_l}, a_1^{\theta_l}, \ldots, a_m^{\theta_l}\}$, where $a_m^{\theta_l} = (a_m^{\theta_l}(S_1), \ldots, a_m^{\theta_l}(S_{|\mathbb{S}|}))$, there exists a state S_r and $\epsilon > 0$, such that

$$\left[P_{a_m^{\theta_l}} P_{a_{m-1}^{\theta_l}} \cdots P_{a_1^{\theta_l}}\right]_{ir} \geqslant \epsilon,\ 1 \leqslant i \leqslant Y, \tag{24}$$

$$\left[P_{a_{m-1}^{\theta_l}} P_{a_{m-2}^{\theta_l}} \cdots P_{a_0^{\theta_l}}\right]_{ir} \geqslant \epsilon,\ 1 \leqslant i \leqslant Y, \tag{25}$$

where $[\cdot]_{ir}$ denotes the element of the ith row and rth column of the corresponding matrix.

Proof. For $i = 1, \ldots, Y$ and $0 \leqslant j \leqslant m$, according to (14), we have

$$\begin{aligned}\left[P_{a_m^{\theta_l}} P_{a_{m-1}^{\theta_l}} \cdots P_{a_0^{\theta_l}}\right]_{ij} &= \Pr\left(S_j | S_i, a_i^{\theta_l}(S_i), \overline{A}_{\theta_l}(S_i)\right)\\ &= \Pr(\boldsymbol{R}_j, \boldsymbol{\Upsilon}_j | \boldsymbol{R}_i, \boldsymbol{\Upsilon}_i, a_i^{\theta_l}(S_i), \overline{A}_{\theta_l}(S_i))\\ &= \Pr\left(\boldsymbol{R}_j | \boldsymbol{R}_i\right) \Pr(\boldsymbol{\Upsilon}_j | \boldsymbol{\Upsilon}_i, a_i^{\theta_l}(S_i), \overline{A}_{\theta_l}(S_i))\\ &= \Pr\left(\boldsymbol{R}_j = \lambda_p' | \boldsymbol{R}_i = \lambda_q'\right)\end{aligned}$$

$$= \mathcal{T}_{p,q} \quad \text{(The transition kernel (1))}. \tag{26}$$

According to MLE, in each time interval t, there exists a state, λ'_p, such that $\mathcal{T}_{p,q} > 0$. This concludes the proof.

Based on Lemma 2, we prove the local convergence of DVI in Theorem 1. Theorem 1 shows that $u_\iota^{\theta_l}(S)$ converges to a limit, $u_\infty^{\theta_l}(S)$. However, we know that the optimization by individual SFCs cannot guarantee $u_\iota^{\theta_l}(S)$ to converge to the solution of (13). Hence, the global convergence should be validated. Based on Theorem 1, the global convergence of DVI is proved in Theorem 2.

Theorem 1 (Local convergence)**.** At each time interval t, for SFC $\theta_l \in \mathbb{Y}$, $S \in \mathbb{S}$, and $l = 0, 1, \ldots$, $\iota = 0, 1, \ldots$, if $\iota \to \infty$, the value function $u_\iota^{\theta_l}(S)$ converges to a limit, $u_\infty^{\theta_l}(S)$, satisfying

$$u_\infty^{\theta_l}(S_r) + u_\infty^{\theta_l}(S) = \widetilde{T} u_\infty^{\theta_l}(S). \tag{27}$$

Proof. See Appendix A.

Theorem 2 (Global convergence)**.** For $\iota = 0, 1, \ldots$, value function $u_\iota^{\theta_l}(S)$ is a Cauchy sequence if $l \to \infty$. As $l \to \infty$ and $\iota \to \infty$, $u_\iota^{\theta_l}(S)$ converges to a limit, $u^\infty(S)$, and $u_\iota^{\theta_l}(S_r)$ converges to η in (13), i.e., $\eta = u^\infty(S_r)$.

Proof. See Appendix B.

Next, we analyze the lower bound of the minimum number of iterations of u_t^i in time-interval t. To obtain this lower bound, we first present Lemma 3. Notice we use $\|\cdot\|$ to denote the L_∞-norm.

Lemma 3. $\widetilde{T}$ is a contraction mapping of modulus γ with respect to L_∞-norm scaled by vector $(w_1, w_2, \ldots, w_{|\mathbb{S}|})$, i.e.,

$$\|\widetilde{T} u_\tau^i - u^*\|^w \leqslant \gamma \|\widetilde{T} u_{\tau-1}^i - u^*\|^w, \tag{28}$$

where $\|u\|^w = \|u(S_1)/w_1, u(S_2)/w_1, \ldots, u(S_{|\mathbb{S}|})/w_{|\mathbb{S}|}\|$, $u^* = (u_1^*, u_2^*, \ldots, u_{|\mathbb{S}|}^*)$ (i.e., $u^* = \widetilde{T} u^*$) and $0 < \gamma < 1$ (More details about γ can be found in [35]).

Proof. cf. Lemma 3 in [35] for proof.

Based on Lemma 3, we can obtain Theorem 3.

Theorem 3. The lower bound of the minimum number of iterations of u_t^i in time-interval t is $n = \lceil \log(\xi / \|c - u^*\|) / \log(\gamma) \rceil$ where $\xi > 0$ is the computation precision, c is the initial values of u_τ^i i.e., $c = u_0^i$.

Proof. Based on Lemma 3, the error $\|\widetilde{T} u_\tau^i - u^*\|^w$ can be bounded as follows:

$$\|\widetilde{T} u_\tau^i - u^*\|^w \leqslant \gamma \|\widetilde{T} u_{\tau-1}^i - u^*\|^w \leqslant \gamma^2 \|\widetilde{T} u_{\tau-2}^i - u^*\|^w \leqslant \gamma^n \|c - u^*\|.$$

Next, the analysis follows the idea of bounding $\gamma^n \|c - u^*\|$ by ξ, i.e., $\gamma^n \|c - u^*\| \leqslant \xi$. Then, we obtain the lower bound on the minimum number of iterations $n = \lceil \log(\xi / \|c - u^*\|) / \log(\gamma) \rceil$.

Note that a larger ξ value results in faster convergence, while it may lead to lower detection accuracy. The value of ξ should be carefully specified.

5 Numerical results and discussion

In this section, we first validate the feasibility of the Markov chain model as shown in (1) by examining real data trace from Google data centres. We implement both the optimal value iteration in (16) (Optimal VI) and the DDRA algorithm (Algorithm 1) to obtain the corresponding solution of P1. Then, we conduct simulation experiments to validate the effectiveness of the DDRA algorithm by comparing it with two benchmarks. Specifically, we evaluate the performance of optimal value iteration (Optimal VI), DDRA, Benchmarks 1 and 2 with different delay (T_i) and packet loss rate (R_i) requirements.

5.1 Validation of Markov chain model

We adopt the Markov chain model to capture the temporal correlation of packet arrival rates of tasks in Google data trace approximately. To validate the feasibility of the Markov chain model, we randomly choose a task among around 4000 tasks. As mentioned previously, we use the workload trace of a task in

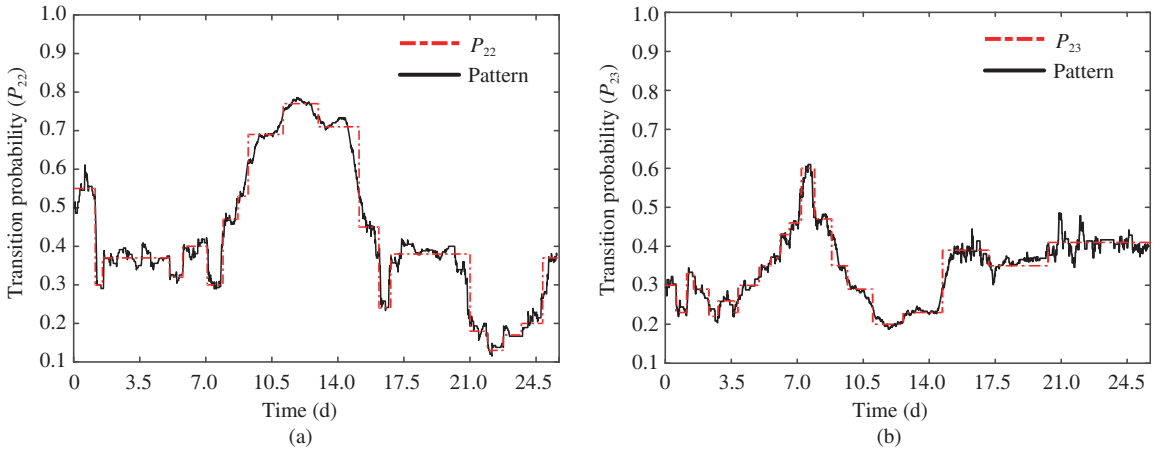

Figure 4 (Color online) Transition probability of packet arrival rate of a data trace from Google data center. (a) Transition probability of P_{22} vs. time; (b) transition probability of P_{23} vs. time.

Google cluster to simulate the dynamic packet arrival rate of an NF. We divide the value of the packet arrival rate into 10 levels. The transition probability of the packet arrival rate is calculated by MLE given in (15) with a sliding window of 2 days.

Figures 4(a) and (b) show the variation of P_{22} and P_{23} over 25 days, where P_{22} indicates the transition probability that the packet arrival rate lingers in level-2 in the preceding time-interval and lingers in level-3 in the next time-interval. Since all transition probabilities have similar characteristics, we only show P_{22} and P_{23} as examples. We can see that transition probabilities P_{22} and P_{23} always linger around a value for several minutes or hours. This validates that the transition probability of packet rates is quasi-static for a short term [18]. Hence, we can use the Markov chain as an approximation to calculate the transition probabilities.

5.2 Performance evaluation of DDRA

Network topology. We consider two scenarios: (a) a small-scale network in which the number of nodes is set to $V = 5$ and the number of SFCs is set to $X = 4$; and (b) a large-scale network with $V = 12$ and $X = 10$ [10, 36]. The small-scale network is used to validate the performance upper bound provided by the optimal solution of Optimal VI and examine the difference between the optimal performance and that of the DDRA algorithm. The large-scale network is used to evaluate the performance gain of the DDRA algorithm. Owing to the high complexity of Optimal VI, we only evaluate it for the small-scale network.

SFC requests. The length of GSFCs varies from 3 to 6 units. The source and sink nodes of GSFCs are randomly specified. We choose the speed-up factor for the NFs with general form $1/(1-p+p/m)$ where $0 < p < 1$, $m \in M_n$, as used in previous work [10]. There are three NFs that can work in parallel in service layer in each GSFC. The service packet rate is discretized into 10 levels. Recall that the bandwidth consumption of virtual link ι_n (the end NF of virtual link ι_n is NF F_n) in time interval t is denoted by $B_{n,t}$, which can be estimated by $B_{n,t} = \lambda_{n,t} L_p$, where L_p denotes the average packet length. For simplicity, all the packets have equal length, i.e., $L_p = 1024$ bytes for all SFCs in the network. Specifically, when the system is at state $\boldsymbol{R}_t = (\lambda_{1,t}, \ldots, \lambda_{Y,t})$ in time-interval t, the packet arrival rate of NF F_n in time-interval t can be estimated by $\lambda_{n,t} = \sum_{j=1}^{10} \mathcal{T}_{m,j,n}^{t-1} \lambda'_{m,n}$. $\mathcal{T}_{m,j,n}^{t-1}$ is calculated according to (15).

In the simulations, the duration of the time-slot is set to 10 min according to [18] and other parameters are listed in Table 2 [10, 37]. Throughout the performance evaluation, we compare the proposed DDRA algorithm with two other heuristic benchmarks referring to solutions in CoordVNF [13] and BFDSU [14]. Both of these mechanisms are reactive to a dynamic environment and make instantaneous decisions at the beginning of each time interval by using the detected workload. We assume that Benchmark 1 (B1) and Benchmark 2 (B2) use multiple cores as the computation model as in our system model.

• B1 is a heuristic mechanism aiming to minimize the energy consumption by solving the embedding problem for SFCs. The core allocation scheme for an NF is fixed by not obeying the delay and reliability

Table 2 Simulation parameters

Parameters	Values
$S_{\max}$, S_{idle}	25 W, 5 W
S_1, S_2	50 W, 25 W
ν, ϑ	20–50 GB, 0.512 W
$d_n(1)$, D_l	0.8–1.2 ms, 0.3–0.5 ms
L_l, E_v, C_v, λ	6 Mbps, 1.5 kW, 64, 100–1000 packets/s
I_n, ξ	50 (packet), 10^{-1}

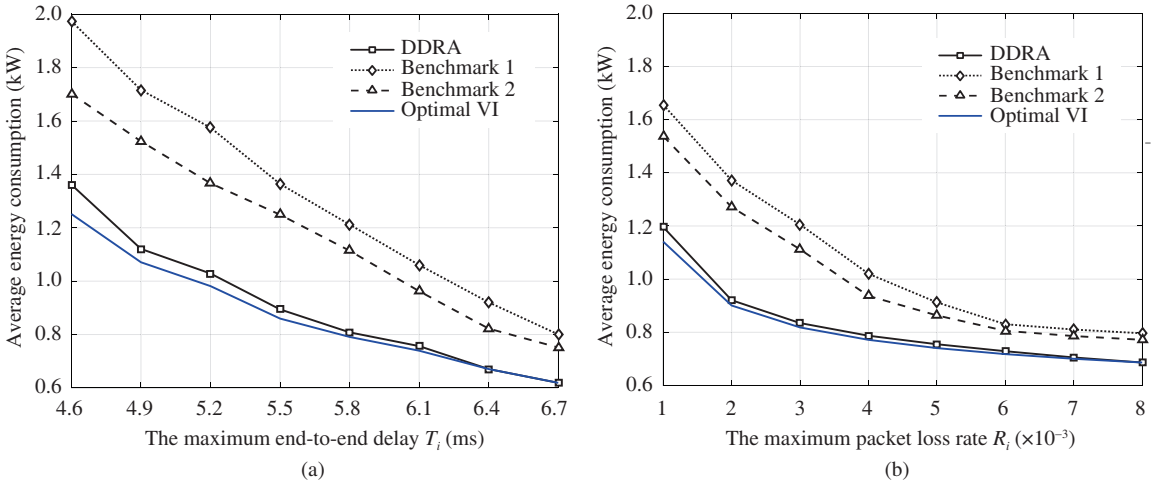

Figure 5 (Color online) Average energy consumption under (a) $R_i = 6 \times 10^{-3}$ and (b) $T_i = 6.5$ ms in the small-scale network.

constraint of an SFC. The first NF instance is mapped at the candidate substrate nodes in close proximity to the source node that is fixed. After an NF instance has been mapped to a substrate node and then to embed the next NF, B1 examines the candidate substrate nodes in close proximity to this previously mapped substrate node. The maximum distance between the candidate substrate nodes and the previously mapped node is limited. If it fails in finding a substrate node to embed the NF, it discards the last embedding steps and iteratively tries to embed the NFs on alternative nodes. Repeat these steps until the last NF is embedded;

• B2 is a priority-driven weighted mechanism with objective of minimizing the energy consumption. The core allocation scheme for an NF is fixed by not obeying the delay and reliability constraint of an SFC. B2 sorts all computing nodes in an increasing order of the remaining amount of resources of each computing node and sorts all NFs in a descending order of their total resource demand. The most resource-demanding NF is first placed at the node with minimal remaining resources. Then, resort all computing nodes in an increasing order of the remaining amount of resources and place the second resource-demanding NF at the node with minimal remaining resources. Repeat these steps until the last NF is embedded.

The complexity of B1 and B2 is $O(N^2M)$ and $O(NM) + O(N) + O(M)$, respectively, where N is the number of substrate nodes and M is the number of virtual NFs of all SFCs. More specifically, the worst case of searching (sorting) procedure in B1 takes $O(N)$ and the worst case is that there are total $O(NM)$ times for all NFs of searching procedure due to the infeasibility of the substrate node found in each step for embedding an NF instance. Thus, the complexity of B1 is $O(MN^2)$. The sorting procedure of B2 takes $O(N)$ and $O(M)$ for substrate nodes and all NFs, respectively. The worst case of feasible mapping procedure takes $O(NM)$. Thus, the complexity of B2 is $O(NM) + O(N) + O(M)$.

In the experiments, one time interval is set to 10 min. We use the following performance metrics over 3 days (432 time-intervals in total): (1) average energy consumption as given by $P(\pi)$ to evaluate the total energy efficiency, (2) average computational energy consumption, (3) average network forwarding energy consumption, and (4) average migration energy consumption. The definition of the last three metrics is similar to that of average energy consumption.

First, we demonstrate the performance upper bound of the DDRA algorithm in the small-scale network in Figures 5(a) and (b). Figure 5(a) shows that the average energy consumption in the system of all

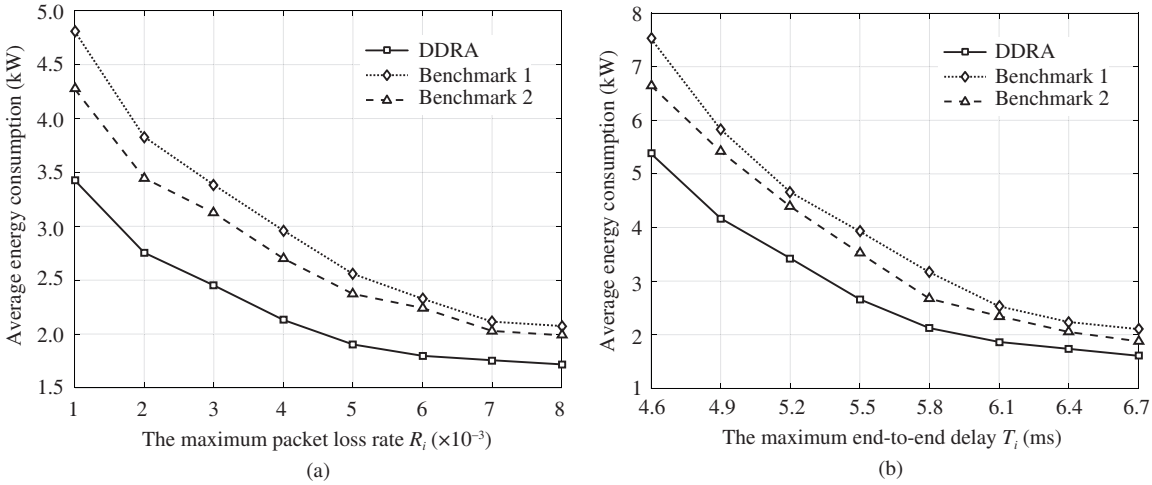

Figure 6 Average energy consumption under (a) $T_i = 6.5$ ms and (b) $R_i = 6 \times 10^{-3}$ in the large-scale network.

four mechanisms monotonically decreases with the maximum end-to-end delay, and the average energy consumption of the DDRA algorithm is always lower than that of B1 and B2. This is because the DDRA algorithm allocates appropriate computational resources for individual NFs, which allows to use fewer cores and even fewer nodes (switches) to provision the services. In addition, the prediction method in the DDRA algorithm captures the regularities of traffic in individual SFCs, so as to efficiently allocate appropriate resources for NFs and decrease the number of NF migrations in each time-interval. Therefore, the DDRA algorithm can achieve both computational energy gain and migration energy gain. In contrast, B1 and B2 choose the core assignment policy in a greedy way according to the current traffic load of NFs. Moreover, the average energy consumption of the DDRA algorithm is slightly higher than that of the optimal value iteration method when $T_i < 6.1$ ms, but can converge to that of the optimal one with T_i. This is because the DVI may abandon the actions violating the constraints of the maximum capacity of nodes and links as mentioned in Remark 1. In this way, the DVI may exclude an optimal solution and converge to a suboptimal solution. A lower T_i (larger computational cores) implies a higher probability of violating the constraints, such as the maximum capacity of nodes and end-to-end delay. The gap between the curves of the DDRA and the theoretical optimal solution increases as the maximum end-to-end delay decreases. This is because that a lower delay requirement may cause a big penalty value which leads to a poor convergence of the algorithm. Naturally, this in turn has a bad impact on the algorithm to converge to an optimal solution. Figure 5(b) shows the average energy consumption in the system of each mechanism decreases with the maximum packet loss rate. The average energy consumption using the DDRA algorithm is always lower than that of B1 and B2 due to the same reasons in Figure 5(a). Therefore, the DDRA algorithm can efficiently improve energy efficiency for service provisioning in the network and benefit from the low packet loss requests when compared with B2 and B1 in the small-scale network.

Next, we demonstrate the performance improvement of the DDRA algorithm in the large-scale network as shown in Figures 6(a) and (b). Owing to the high complexity of the optimal value iteration, we only evaluate DDRA, B1, and B2 in the large-scale network. Figure 6(a) shows the average energy consumption of three mechanisms using the trace from Google data center with R_i in the large-scale network. We can see that the average energy consumption monotonically decreases with packet loss rate R_i. This is as expected as the lower packet loss rate of an SFC always implies the higher computation capacity of NFs in the SFC. The average energy consumption in Figure 6(a) follows DDRA $<$ B1 $<$ B2, because the DDRA allocates appropriate cores for individual NFs aiming at decreasing the computational energy consumption. Using less computational cores for each NF means that more NFs can be mapped onto the same node. In this way, more switching resources can be saved. In addition, DDRA needs fewer NF migrations than BFUSU and B1 by capturing the traffic dynamics. Therefore, DDRA as an on-line mechanism can significantly improve energy efficiency when compared with B1 and B2 in the large-scale network. Figure 6(b) shows the average energy consumption as a function of end-to-end delay requirements of the three mechanisms. The average energy consumption of the DDRA algorithm is always lower than that of B1 and B2 due to the same reasons as in Figure 6(a). In Figure 6(b), we can

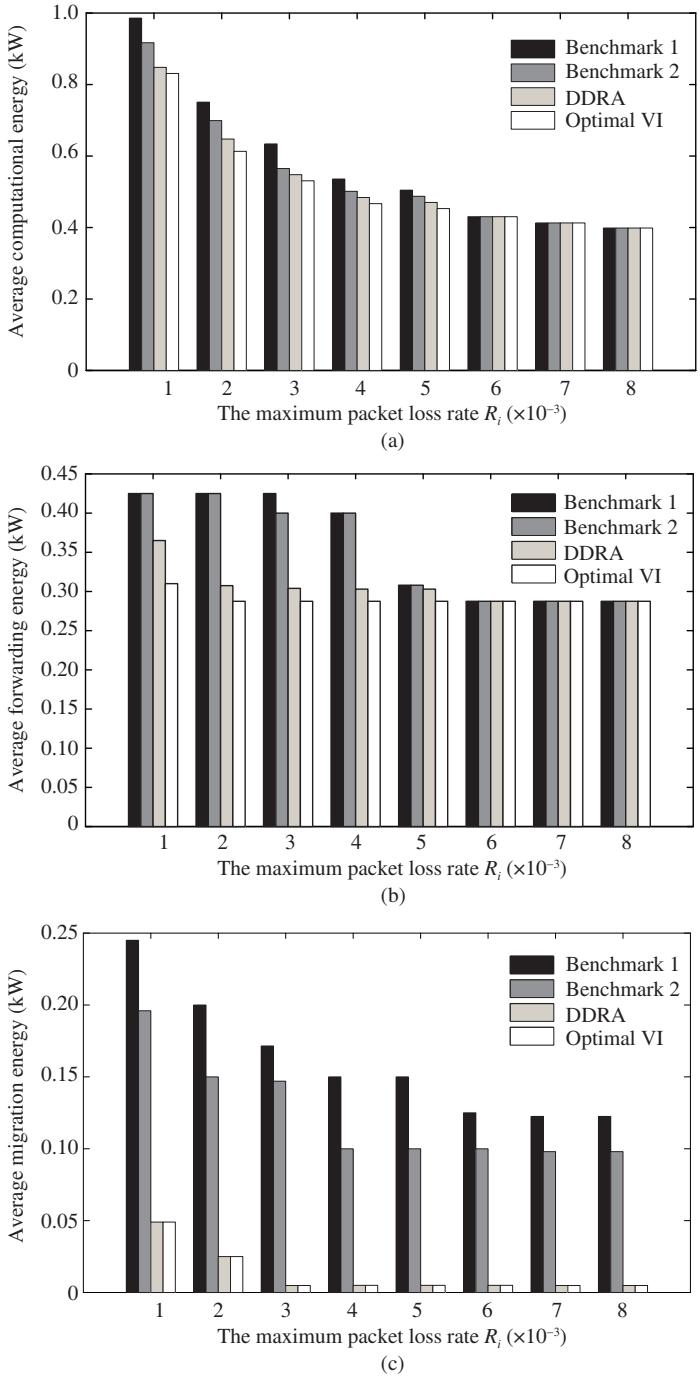

Figure 7 Average energy consumption under under $T_i = 6.5$ ms in the small-scale network. (a) Average computational energy consumption; (b) average forwarding energy consumption; (c) average migration energy consumption.

see that DDRA can reduce the average energy consumption approximately 25% and 20% when compared with B1 and B2 respectively. Thus, DDRA can effectively reduce much more energy consumption than B1 and B2 in a long term. This validates that DDRA also saves more energy for the low latency requests when compared with B1 and B2 in the large-scale network.

Figure 7 shows the average computational, forwarding and migration energy consumption in the small-scale network. As demonstrated in Figure 7(a), when the maximum packet loss rate increases, the computational energy consumption decreases in all four mechanisms. The DDRA can utilize core resources more efficiently than B2 and CoorVNF. Moreover, the difference between DDRA and optimal VI becomes smaller when the maximum packet loss rate increases. Figure 7(b) shows that, when $R_i \leqslant 4 \times 10^{-3}$,

the DDRA has higher forwarding energy consumption than optimal VI, as the DDRA may obtain a suboptimal virtual link and NF mapping action by using a distributed calculation method. Further, the DDRA has lower forwarding energy consumption than B2 and CoorVNF, because the DDRA maps more NFs onto a same substrate node by using less cores compared with B2 and CoorVNF. As shown in Figure 7(c), the DDRA performs less NF migrations than B2 and CoorVNF, because it tries to flexibly allocate appropriate computational and forwarding resources to individual NFs and thus decreasing the number of NF migrations, while B2 and CoorVNF are implemented by redeploying/migrating virtual network function modules for improving resource utilization and guaranteeing QoS requirements. Although the DDRA may not strictly capture the packet arrival rates sometimes, the results show that it still outperforms B1 and B2 by effectively reducing the long-term average energy consumption.

6 Conclusion

In this paper, we have investigated the dynamic NF resource allocation as a large-scale MDP for minimizing the long-term average energy consumption, while guaranteeing the packet loss rate and the end-to-end delay in SDN/NFV based networks. We first derive an optimal algorithm which has high computational complexity. To achieve a low computational complexity, we propose a distributed dynamic NFRA algorithm, which can be used as an on-line mechanism. The convergence property of the proposed the DDRA algorithms has been proved. Numerical results based on real world data traces demonstrate that the DDRA algorithm performs well and significantly outperforms two benchmark mechanisms. These results clearly illustrate that DDRA as an on-line mechanism improves the energy efficiency for services in the next generation softwarization networks.

Energy-efficient network function resource allocation for a large number of users in dynamic environments is technically challenging. The transition probability function is model-based in this work. It is interesting to extend the current work by considering a model-free solution in future research when the historical data is not achievable.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61871099, 61631004) and China Postdoctoral Science Foundation (Grant No. 2019M663476). We gratefully acknowledge the many helpful suggestions made by Shaoe LIN, Qihao LI, Weizhang TING, Junlin LI, Nan CHEN, and anonymous referees. We also thanks to the support of joint training public postgraduates of Chinese Scholarship Council (CSC).

References

1 Series M. Framework and overall objectives of the future development of imt for 2020 and beyond. Recommendation ITU-2083, 2015. https://www.itu.int/rec/R-REC-M.2083/en
2 Ye Q, Li J L, Qu K G, et al. End-to-end quality of service in 5G networks: examining the effectiveness of a network slicing framework. IEEE Veh Technol Mag, 2018, 13: 65–74
3 Ye Q, Zhuang W H, Li X, et al. End-to-end delay modeling for embedded VNF chains in 5G core networks. IEEE Int Things J, 2019, 6: 692–704
4 Zhang S Q, Wu Q Q, Xu S G, et al. Fundamental green tradeoffs: progresses, challenges, and impacts on 5G networks. IEEE Commun Surv Tut, 2017, 19: 33–56
5 Mukherjee A. Energy efficiency and delay in 5G ultra-reliable low-latency communications system architectures. IEEE Netw, 2018, 32: 55–61
6 Dharmaweera M N, Parthiban R, Sekercioglu Y A. Toward a power-efficient backbone network: the state of research. IEEE Commun Surv Tut, 2015, 17: 198–227
7 Chen Y, Zhang N, Zhang Y C, et al. Energy efficient dynamic offloading in mobile edge computing for Internet of Things. IEEE Trans Cloud Comput, 2019. doi: 10.1109/TCC.2019.2898657
8 Fei X C, Liu F M, Xu H, et al. Adaptive VNF scaling and flow routing with proactive demand prediction. In: Proceedings of IEEE INFOCOM, Honolulu, 2018. 486–494
9 Yu B, Han Y N, Wen X M, et al. An energy-aware algorithm for optimizing resource allocation in software defined network. In: Proceedings of IEEE GLOBECOM, Washington, 2016. 1–7
10 Liu M J, Feng G, Zhou J H, et al. Joint two-tier network function parallelization on multicore platform. IEEE Trans Netw Serv Man, 2019, 16: 990–1004
11 Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE ACM Trans Netw, 1993, 1: 397–413
12 Bolot J-C. End-to-end packet delay and loss behavior in the Internet. In: Proceedings of ACM SIGCOMM Computer Communication Review, 1993. 289–298
13 Beck M T, Botero J F. Coordinated allocation of service function chains. In: Proceedings of IEEE Global Communications Conference (GLOBECOM), 2015. 1–6
14 Zhang Q X, Xiao Y K, Liu F M, et al. Joint optimization of chain placement and request scheduling for network function virtualization. In: Proceedings of IEEE ICDCS, 2017. 731–741
15 Nguyen D T, Le L B, Bhargava V K. A market-based framework for multi-resource allocation in fog computing. IEEE ACM Trans Netw, 2019, 27: 1151–1164
16 Zhou Z Y, Dong M X, Ota K, et al. Energy-efficient resource allocation for D2D communications underlaying cloud-RAN-based LTE-A networks. IEEE Int Things J, 2016, 3: 428–438

17 Lyu X C, Tian H, Ni W, et al. Energy-efficient admission of delay-sensitive tasks for mobile edge computing. IEEE Trans Commun, 2018, 66: 2603–2616
18 Han Z H, Tan H S, Chen G H, et al. Dynamic virtual machine management via approximate Markov decision process. In: Proceedings of IEEE INFOCOM, San Francisco, 2016. 1–9
19 Chen L X, Zhou S, Xu J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE ACM Trans Netw, 2018, 26: 1619–1632
20 Ye Q, Li J L, Qu K G, et al. End-to-end quality of service in 5G networks: examining the effectiveness of a network slicing framework. IEEE Veh Technol Mag, 2018, 13: 65–74
21 Sun C, Bi J, Zheng Z L, et al. NFP: enabling network function parallelism in NFV. In: Proceedings of ACM Conference of Special Interest Group on Data Communication, 2017. 43–56
22 Chen L H, Shen H Y. Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters. In: Proceedings of IEEE INFOCOM, 2014. 1033–1041
23 Zhang Q, Zhani M F, Boutaba R, et al. Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans Cloud Comput, 2014, 2: 14–28
24 Reiss C, Wilkes J, Hellerstein J L. Google cluster-usage traces: format+schema. 2011. https://scholar.google.com.hk/scholar?q=google+cluster-usage+traces:format%2Bschema&hl=zh-CN&as_sdt=0&as_vis=1&oi=scholart
25 Burke P J. The output of a queuing system. Oper Res, 1956, 4: 699–704
26 Sharkh M A, Jammal M, Shami A, et al. Resource allocation in a network-based cloud computing environment: design challenges. IEEE Commun Mag, 2013, 51: 46–52
27 Sun X H, Ni L M. Another view on parallel speedup. In: Proceedings of IEEE/ACM Conference on Supercomputing, 1990. 324–333
28 Mahadevan P, Sharma P, Banerjee S, et al. A power benchmarking framework for network devices. In: Proceedings of International Conference on Research in Networking, 2009. 795–808
29 Wen R H, Feng G, Tang J H, et al. On robustness of network slicing for next-generation mobile networks. IEEE Trans Commun, 2019, 67: 430–444
30 Korf R E. Depth-first iterative-deepening: an optimal admissible tree search. Artif Intell, 1985, 27: 97–109
31 Myung I J. Tutorial on maximum likelihood estimation. J Math Psychol, 2003, 47: 90–100
32 Bertsekas D P. Dynamic Programming and Optimal Control. Belmont: Athena Scientific, 1995
33 Wei Q L, Liu D R, Shi G, et al. Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming. IEEE Trans Ind Electron, 2015, 62: 4203–4214
34 Bertsekas D. Distributed dynamic programming. IEEE Trans Autom Control, 1982, 27: 610–616
35 Tseng P. Solving H-horizon, stationary Markov decision problems in time proportional to log(H). Oper Res Lett, 1990, 9: 287–297
36 Wen R H, Feng G, Tan W, et al. Protocol stack mapping of software defined protocol for next generation mobile networks. In: Proceedings of IEEE International Conference on Communications (ICC), 2016. 1–6
37 Liu H K, Xu C Z, Jin H, et al. Performance and energy modeling for live migration of virtual machines. In: Proceedings of ACM International Symposium on High Performance Distributed Computing, 2011. 171–182

Appendix A Proof of Theorem 1

We first show that $\{u_\iota^{\theta_l}(S)\}$ is a Cauchy sequence when fixing θ_l. Then, from (23), $u_\infty^{\theta_l}(S_r)+u_\infty^{\theta_l}(S)=\widetilde{T}u_\infty^{\theta_l}(S)$ holds for all $S\in\mathbb{S}$.

First, denoting

$$\nu_\iota^{\theta_l}=\widetilde{T}u_\iota^{\theta_l}(S_r),$$

we have

$$\begin{aligned}\boldsymbol{u}_{\iota+1}^{\theta_l}&=\boldsymbol{p}+P_{a_\iota^{\theta_l}}\boldsymbol{u}_\iota^{\theta_l}-\nu_\iota^{\theta_l}\boldsymbol{e}\leqslant\boldsymbol{p}+P_{a_{\iota-1}^{\theta_l}}\boldsymbol{u}_\iota^{\theta_l}-\nu_\iota^{\theta_l}\boldsymbol{e},\\ \boldsymbol{u}_\iota^{\theta_l}&=\boldsymbol{p}+P_{a_\iota^{\theta_l}-1}\boldsymbol{u}_{\iota-1}^{\theta_l}-\nu_{\iota-1}^{\theta_l}\boldsymbol{e}\leqslant\boldsymbol{p}+P_{a_\iota^{\theta_l}}\boldsymbol{u}_{\iota-1}^{\theta_l}-\nu_{\iota-1}^{\theta_l}\boldsymbol{e},\end{aligned}$$

where $\boldsymbol{e}$ is an all-ones vector of dimension $|\mathbb{S}|$. Setting $\delta_\iota^{\theta_l}=\boldsymbol{u}_{\iota+1}^{\theta_l}-\boldsymbol{u}_\iota^{\theta_l}$, we have

$$P_{a_\iota^{\theta_l}}\delta_{\iota-1}^{\theta_l}+(\nu_{\iota-1}^{\theta_l}-\nu_\iota^{\theta_l})\boldsymbol{e}\leqslant\delta_\iota^{\theta_l}\leqslant P_{a_{\iota-1}^{\theta_l}}\delta_{\iota-1}^{\theta_l}+(\nu_{\iota-1}^{\theta_l}-\nu_\iota^{\theta_l})\boldsymbol{e}.$$

By iterating, we have

$$P_{a_\iota^{\theta_l}}\cdots P_{a_{\iota-m+1}^{\theta_l}}\delta_{\iota-m}^{\theta_l}+(\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l})\boldsymbol{e}\leqslant\delta_\iota^{\theta_l}\leqslant P_{a_{\iota-1}^{\theta_l}}\cdots P_{a_{\iota-m}^{\theta_l}}\delta_{\iota-m}^{\theta_l}+(\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l})\boldsymbol{e}.\tag{A1}$$

According to Lemma 2 and the fact $\delta_{\iota-m}^{\theta_l}(S_r)=0$, the R.H.S of (A1) yields

$$\delta_\iota^{\theta_l}(S_i)\leqslant\sum_{j=1}^{|\mathbb{S}|}\left[P_{a_{\iota-1}^{\theta_l}}\cdots P_{a_{\iota-m}^{\theta_l}}\right]_{ij}\delta_{\iota-m}^{\theta_l}(S_j)+\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l}\leqslant(1-\epsilon)\max_j\delta_{\iota-m}^{\theta_l}(S_j)+\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l},$$

which implies that

$$\max_j\delta_\iota^{\theta_l}(S_j)\leqslant(1-\epsilon)\max_j\delta_{\iota-m}^{\theta_l}(S_j)+\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l}.$$

Similarly, from the L.H.S of (A1), we have

$$\min_j\delta_\iota^{\theta_l}(S_j)\geqslant(1-\epsilon)\min_j\delta_{\iota-m}^{\theta_l}(S_j)+\nu_{\iota-m}^{\theta_l}-\nu_\iota^{\theta_l}.$$

By subtracting the last two relations, for $\forall S_j \in \mathbb{S}$ and $\iota \geqslant m$, we have

$$\max_j \delta_\iota^{\theta_l}(S_j) - \min_j \delta_\iota^{\theta_l}(S_j) \leqslant (1-\epsilon)\left(\max_j \delta_{\iota-m}^{\theta_l}(S_j) - \min_j \delta_{\iota-m}^{\theta_l}(S_j)\right).$$

Then, for $A > 0$ and all $\iota \geqslant m$, we have

$$\max_j \delta_\iota^{\theta_l}(S_j) - \min_j \delta_\iota^{\theta_l}(S_j) \leqslant A(1-\epsilon)^{\iota/m}.$$

As $\delta_\iota^{\theta_l}(S_r) = 0$, it follows that for $\forall S_i \in \mathbb{S}$,

$$\left|u_{\iota+1}^{\theta_l}(S_i) - u_\iota^{\theta_l}(S_i)\right| = \left|\delta_\iota^{\theta_l}(S_i)\right| \leqslant \max_j \delta_\iota^{\theta_l}(S_j) - \min_j \delta_\iota^{\theta_l}(S_j) \leqslant A(1-\epsilon)^{\iota/m}.$$

Thus, for every $r > 1$, we have

$$\begin{aligned}\left|u_{\iota+r}^{\theta_l}(S_i) - u_\iota^{\theta_l}(S_i)\right| &\leqslant \sum_{l=0}^{r-1}\left|u_{\iota+1+l}^{\theta_l}(S_i) - u_{\iota+l}^{\theta_l}(S_i)\right| \\ &\leqslant A(1-\epsilon)^{\iota/m}\sum_{l=0}^{r-1}(1-\epsilon)^{l/m} \\ &= \frac{A(1-\epsilon)^{\iota/m}(1-(1-\epsilon)^{r/m})}{1-(1-\epsilon)^{1/m}}.\end{aligned}$$

According to Cauchy convergence, the value function $u_\iota^{\theta_l}(S)$ converges to a limit $u_\infty^{\theta_l}(S)$.

Appendix B Proof of Theorem 2

The theorem can be proved in three steps.

(1) We show that for $\forall \iota = 0, 1, \ldots$, $\{u_\iota^{\theta_l}(S)\}$ is a Cauchy sequence and convergent as $l \to \infty$.

First, letting

$$\nu_\iota^{\theta_l} = \widetilde{T} u_\iota^{\theta_l}(S_r),$$

we have

$$\begin{aligned}\boldsymbol{u}_\iota^{\theta_{l+1}} &= \boldsymbol{p} + P_{a_{\iota-1}^{\theta_{l+1}}} \boldsymbol{u}_{\iota-1}^{\theta_{l+1}} - \nu_{\iota-1}^{\theta_{l+1}}\boldsymbol{e} \leqslant \boldsymbol{p} + P_{a_{\iota-1}^{\theta_l}} \boldsymbol{u}_{\iota-1}^{\theta_{l+1}} - \nu_{\iota-1}^{\theta_{l+1}}\boldsymbol{e}, \\ \boldsymbol{u}_\iota^{\theta_l} &= \boldsymbol{p} + P_{a_\iota^{\theta_l}-1} \boldsymbol{u}_{\iota-1}^{\theta_l} - \nu_{\iota-1}^{\theta_l}\boldsymbol{e} \leqslant \boldsymbol{p} + P_{a_\iota^{\theta_l}} \boldsymbol{u}_{\iota-1}^{\theta_l} - \nu_{\iota-1}^{\theta_l}\boldsymbol{e}.\end{aligned}$$

We set $\boldsymbol{q}_\iota^{\theta_l} = \boldsymbol{u}_\iota^{\theta_l+1} - \boldsymbol{u}_\iota^{\theta_l}$, resulting in

$$P_{a_{\iota-1}^{\theta_{l+1}}} \boldsymbol{q}_{\iota-1}^{\theta_l} + (\nu_{\iota-1}^{\theta_l} - \nu_{\iota-1}^{\theta_{l+1}})\boldsymbol{e} \leqslant \boldsymbol{q}_\iota^{\theta_l} \leqslant P_{a_{\iota-1}^{\theta_l}} \boldsymbol{q}_{\iota-1}^{\theta_l} + (\nu_{\iota-1}^{\theta_l} - \nu_{\iota-1}^{\theta_{l+1}})\boldsymbol{e}.$$

According to (20), by iteration, we obtain

$$P_{a_{\iota-1}^{\theta_{l+1}}} \cdots P_{a_0^{\theta_{l+1}}} \cdots P_{a_\iota^{\theta_l}} \boldsymbol{q}_\iota^{\theta_{l-1}} + (\xi_\iota^{\theta_{l-1}} - \xi_\iota^{\theta_l})\boldsymbol{e} \leqslant \boldsymbol{q}_\iota^{\theta_l} \leqslant P_{a_{\iota-1}^{\theta_l}} \cdots P_{a_0^{\theta_l}} \cdots P_{a_\iota^{\theta_{l-1}}} \boldsymbol{q}_\iota^{\theta_{l-1}} + (\xi_\iota^{\theta_{l-1}} - \xi_\iota^{\theta_l})\boldsymbol{e},$$

where $\xi_\iota^{\theta_l} = \sum_{i=0}^{\iota-1} \nu_i^{\theta_{l+1}} + \sum_{i=\iota}^{\infty} \nu_i^{\theta_l}$. Letting $P'_{\theta_l} = P_{a_{\iota-1}^{\theta_{l+1}}} \cdots P_{a_0^{\theta_{l+1}}} \cdots P_{a_\iota^{\theta_l}}$, by iteration, we then have

$$P'_{\theta_l} \cdots P'_{\theta_{l-m+1}} \boldsymbol{q}_\iota^{\theta_{l-m}} + (\xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l})\boldsymbol{e} \leqslant \boldsymbol{q}_\iota^{\theta_l} \leqslant P'_{\theta_{l-1}} \cdots P'_{\theta_{l-m}} \boldsymbol{q}_\iota^{\theta_{l-m}} + (\xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l})\boldsymbol{e}. \tag{B1}$$

According to Lemma 2 and the fact $q_\iota^{\theta_{l-m}}(S_r) = 0$, the R.H.S of (B1) yields

$$q_\iota^{\theta_l}(S_i) \leqslant \sum_{j=1}^{|\mathbb{S}|}\left[P'_{\theta_{l-1}} \ldots P'_{\theta_{l-m}}\right]_{ij} q_\iota^{\theta_{l-m}}(S_j) + \xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l} \leqslant (1-\epsilon)\max_j q_\iota^{\theta_{l-m}}(S_j) + \xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l},$$

which implies that

$$\max_j q_\iota^{\theta_l}(S_j) \leqslant (1-\epsilon)\max_j q_\iota^{\theta_{l-m}}(S_j) + \xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l}.$$

Similarly, from the L.H.S of (B1) we have

$$\min_j q_\iota^{\theta_l}(S_j) \geqslant (1-\epsilon)\min_j q_\iota^{\theta_{l-m}}(S_j) + \xi_\iota^{\theta_{l-m}} - \xi_\iota^{\theta_l}.$$

By subtracting the last two relations, for $\forall S_j \in \mathbb{S}$ and $l \geqslant m$ we have

$$\max_j q_\iota^{\theta_l}(S_j) - \min_j q_\iota^{\theta_l}(S_j) \leqslant (1-\epsilon)\left(\max_j q_\iota^{\theta_{l-m}}(S_j) - \min_j q_\iota^{\theta_{l-m}}(S_j)\right).$$

The rest proof follows the proof of Theorem 1.

(2) We show that the limit of the iterative value function $\boldsymbol{u}_\iota^{\theta_l}$ satisfies the Bellman equation, as $l \to \infty$.

First, we define some notations. For $\forall \iota = 0, 1, \ldots$, let

$$u_\iota^\infty(S) = \lim_{l\to\infty} u_\iota^{\theta_l}(S),$$
$$u^\infty(S) = \lim_{\iota\to\infty} u_\iota^\infty(S).$$

For $\forall \iota = 0, 1, \ldots$, we define $\Gamma_i = \{l|\theta_l = i, 1 \leqslant i \in \mathbb{Y}\}$, and let n_i be the number of elements in Γ_i. Let $\sigma_1, \sigma_2, \ldots, \sigma_{n_i}$ be the elements in Γ_i that satisfy $\sigma_1 < \sigma_2 < \cdots < \sigma_{n_i}$, i.e., $\Gamma_i = \{\sigma_j | j = 1, 2, \ldots, n_i\}$, where $n_i \to \infty$.

As $n_i \to \infty$, the policy of SFC i is updated infinite times. Then, the optimal policy for SFC i can be defined as

$$u_\iota^i(S) = \arg\min_{a^i}\left\{p(S) + \sum_{S'\in\mathbb{S}} \Pr\left[S'|S, a^i, \overline{A}_i(S)\right] u_\iota^\infty(S')\right\}.$$

According to (27), when $\iota \to \infty$, we have

$$u^\infty(S_r) + u^\infty(S) = \min_{a^i}\left\{p(S) + \sum_{S'\in\mathbb{S}} \Pr\left[S'|S, a^i, \overline{A}_i(S)\right] u^\infty(S')\right\}.$$

For $\forall i \in \mathbb{Y}$, we have $n_i \to \infty$ and

$$u^\infty(S_r) + u^\infty(S) = \min_{a^1}\left\{p(S) + \sum_{S'\in\mathbb{S}} \Pr\left[S'|S, a^1, \overline{A}_1(S)\right] u^\infty(S')\right\},$$
$$\vdots$$
$$u^\infty(S_r) + u^\infty(S) = \min_{a^Y}\left\{p(S) + \sum_{S'\in\mathbb{S}} \Pr\left[S'|S, a^Y, \overline{A}_Y(S)\right] u^\infty(S')\right\},$$

which means that

$$u^\infty(S_r) + u^\infty(S) = \min_{A}\left\{p(S) + \sum_{S'\in\mathbb{S}} \Pr\left[S'|S, A\right] u^\infty(S')\right\}. \tag{B2}$$

(3) We show that value function $u^\infty(S_r)$ equals to η.

According to the definition of η in (13) and (B2), we can obtain

$$\eta = u^\infty(S_r).$$