Hindawi
Discrete Dynamics in Nature and Society
Volume 2018, Article ID 4184805, 11 pages
https://doi.org/10.1155/2018/4184805

*Research Article*

# Online Self-Organizing Network Control with Time Averaged Weighted Throughput Objective

**Zhicong Zhang, Shuai Li, and Xiaohui Yan**

*Department of Industrial Engineering, Dongguan University of Technology, Dongguan, China*

Correspondence should be addressed to Zhicong Zhang; stephen1998@gmail.com

Received 16 June 2017; Revised 9 December 2017; Accepted 6 February 2018; Published 4 March 2018

Academic Editor: Francisco R. Villatoro



We study an online multisource multisink queueing network control problem characterized with self-organizing network structure and self-organizing job routing. We decompose the self-organizing queueing network control problem into a series of interrelated Markov Decision Processes and construct a control decision model for them based on the coupled reinforcement learning (RL) architecture. To maximize the mean time averaged weighted throughput of the jobs through the network, we propose a reinforcement learning algorithm with time averaged reward to deal with the control decision model and obtain a control policy integrating the jobs routing selection strategy and the jobs sequencing strategy. Computational experiments verify the learning ability and the effectiveness of the proposed reinforcement learning algorithm applied in the investigated self-organizing network control problem.

## 1. Introduction

Queueing network optimization problems widely exist in the fields of manufacturing, transportation, logistics, computer science, communication, healthcare [1], and so on. With the rapid development of the Internet of Things, large-scale logistics distribution network, wireless sensor network [2–4], new generation wireless communication network, and other network technologies, more and more new network structures and new network optimization problems emerge. Optimization of network control is an important factor to affect the efficiency of network operation.

Self-organizing networks are a kind of new queueing network system. In self-organizing networks, each station or node can establish a link with its adjacent stations or nodes, receive jobs from other stations or nodes, and transfer them to other stations or nodes. Due to the complex link relationship of stations or nodes, the paths and the sequence of the jobs to go through the network are very complicated. Consequently, the control problem of this kind of networks is very complicated. In literature, researchers concentrate on the control of multihop network, which is a kind of network with self-organizing characteristic. The research methods of multihop network control mainly include two categories. The first one is to decompose it into a series of single-station queueing problems or tandem queueing network problems [5]. The second kind of methods is to simplify the multihop network control problem into link scheduling problem [6] or queue management problem [7]. The main task of link scheduling is to establish a link between the stations and select the appropriate paths for job transferring. He et al. [8] proposed a load-based scheduling algorithm to optimize the link scheduling between stations so as to achieve the load balance of each station and reduce the degree of paths congestion. Pinheiro et al. [9] studied link scheduling and path selection by fuzzy control. Augusto et al. [10] simultaneously optimized link scheduling and routing planning. Nandiraju et al. [11] studied the problem of restricting the length of transmission path and improved the efficiency of long-path transmission. In order to enlarge the network capacity, Gupta and Shroff [12] optimized link scheduling and path selection by solving the maximum weighted matching problem subject to the $K$-hop interference constraints. The main task of queue management is to classify the jobs to the job groups and to determine the transmission order of the job groups. Fu and Agrawal [7] focused on the problem of jobs classification in queue management and improved the efficiency by batch processing of the jobs. Nieminen et al. [13] and Wang et al. [14] studied

optimization of energy management and queue management in multihop networks. Liu et al. [15] reduced the transmission delay and shortened the queue length by modeling and analysis based on Markov chain. Kim et al. [16] considered the fairness of customer services and improved the efficiency of the network while reducing the difference of customers' waiting time. Vučević et al. [17] and Zhou et al. [18] used a reinforcement learning (RL) algorithm to optimize queue management that allocates the data packets to the queues.

In this paper, we study an online multisource multisink queueing network control problem limited by the queue length. We consider the inherent self-organization characteristic of the queueing network problem, transform the problem into Markov Decision Processes (MDP), and then construct an RL system to deal with them. An optimized control strategy and a global optimized solution are obtained by the proposed RL system. The rest of this paper is organized as follows: we introduce the self-organizing queueing network control problem in Section 2, formulate the problem into an RL model in Section 3, present the detailed RL algorithm in Section 4, conduct computational experiments in Section 5, and draw conclusions in Section 6.

## 2. Problem Statement

The online self-organizing network control problem concerned in this paper is described as follows. There are $m$ stations in the network and $n$ types of jobs arrive at the network. Let $E = \{e_i \mid 1 \le i \le m\}$ denote the set of stations in the network and let $J_{j,h}$ $(1 \le j \le n)$ denote the $h$th job of type $j$. Take the network in Figure 1 as an example. As shown in Figure 1, the self-organizing queueing network is composed of three types of stations. The first type of stations is arrival stations. Each type of jobs has a specific arrival station. The specified arrival station for type $j$ jobs is $b_j \in E_a$ $(E_a \subset E)$, where $E_a$ denotes the set of arrival stations. The second type of stations is transfer stations, which receive jobs and send them to other transfer stations or destination stations. The set of transfer stations is denoted by $E_t = \{N_i \mid 1 \le i \le m_t\}$, where $N_i$ denotes the $i$th transfer station, $m_t$ denotes the number of transfer stations, and $E_t \subset E$. The third type of stations is destination stations. Each type of jobs has a specific destination station and the jobs of the same type aim to arrive at the same destination station. The specified destination station for type $j$ jobs is $d_j \in E_d$, where $E_d$ $(E_d \subset E)$ denotes the set of destination stations. Once a job is processed by its specified destination station, it passes through the entire network.

Jobs of type $j$ arrive at their arrival station $b_j$ following Poisson process with rate parameter $\lambda_j$. The arriving jobs wait in the queue of station $b_j$ for transferring. Let $\Omega(e_i)$ $(\Omega(e_i) \subset E)$ denote the set of stations visible to station $e_i$. Specifically, each arrival station corresponds to a set of visible transfer stations. $\Omega(b_j)$ $(\Omega(b_j) \subset E_t)$ denotes the set of transfer stations visible for the arrival station $b_j$ of type $j$ jobs. Each transfer station also corresponds to a set of visible stations. $\Omega(N_i)$ $(\Omega(N_i) \subset E_t \cup E_d)$ denotes the set of stations visible for transfer station $N_i$. $\Omega(N_i)$ contains one or more transfer stations or destination stations. Transfer station $N_i$ is qualified to transfer the jobs of the types in set $J(N_i)$ $(J(N_i) \subseteq \{1, 2, \ldots, n\})$. One station transfers only one job at a time. From the arrival station to the destination station, each job needs to pass through at least one transfer station. Under certain conditions, the arrival station $b_j$ $(1 \le j \le n)$ can establish a link with transfer station $N_i$ if $N_i \in \Omega(b_j)$ and then send a type $j$ job to station $N_i$. The job waits in the queue to be transferred to another station. The arrival station $b_j$ $(1 \le j \le n)$ cannot send a job to transfer station $N_i$ if $N_i \notin \Omega(b_j)$. Similarly, transfer station $N_i$ can establish a link with station $e_i$ if $e_i \in \Omega(N_i)$, select a job from its queue, and send this job to station $e_i$. If station $e_i \in E_d$, then transfer station $N_i$ sends a type $j$ job to station $e_i$ only if $e_i$ is the destination station for type $j$ jobs, that is, station $d_j$.

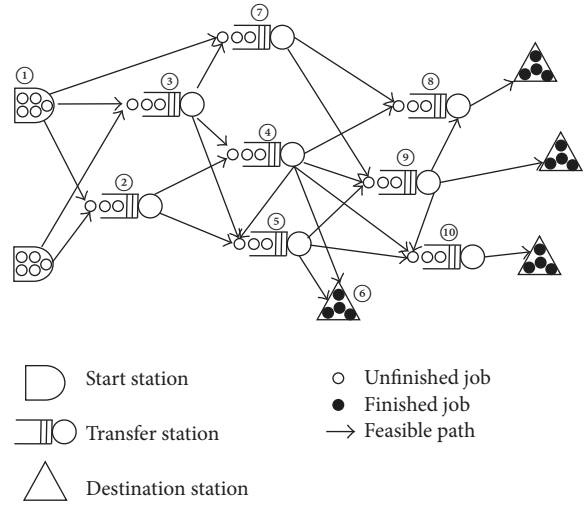


Figure 1: An example of self-organizing network with multitype stations.

There exist a lot of feasible paths for a job from its arrival station to its destination station. Take the network in Figure 1 as an example. Assume that the arrival station for type $j$ jobs is station $e_1$ and the destination station for type $j$ jobs is station $e_6$. The sets of visible stations of stations $e_1$~$e_5$ are $\{e_2, e_3, e_7\}$, $\{e_4, e_5\}$, $\{e_4, e_5, e_7\}$, $\{e_6, e_8, e_9, e_{10}\}$, and $\{e_6, e_9, e_{10}\}$, respectively. Thus, many paths are feasible for type $j$ jobs from the arrival station to the destination station, such as $e_1 \to e_2 \to e_4 \to e_6$, $e_1 \to e_2 \to e_5 \to e_6$, $e_1 \to e_3 \to e_4 \to e_6$, and $e_1 \to e_3 \to e_5 \to e_6$.

The queue capacity of each transfer station is limited, which is denoted by $C_q$. The maximum number of simultaneous transferring jobs for each station is also limited; that is, the number of jobs being transferred to this station cannot exceed a predetermined quantity $C_L$. Though a station can be linked by more than one upstream station, it is allowed to link to at most one downstream station. In order to establish a link between two stations to send jobs, the following conditions must be met: (1) the downstream station is visible for the upstream station; (2) the number of stations transferring jobs to the downstream station is less than the predetermined maximum number; (3) the queue length of the downstream station has not reached the upper limit.

A station is not allowed to send a job to another station if a link is not established between the two stations. An upstream station is allowed to send one or more jobs after establishing a link with a downstream station until it establishes a new link with another downstream station. Assume that a station can only send a job to another station at a time. The time required for establishing a link between two stations is a random variable. Let $\eta(e_i, e_k)$ denote the time required for establishing a link between stations $e_i$ and $e_k$ $(1 \le i, k \le m)$, which follows exponential distributed random variable $\exp(\lambda_\eta(e_i, e_k))$ with parameter $\lambda_\eta(e_i, e_k)$. The time consumed in transferring a job depends on the station and the job type. The transferring time of a type $j$ job by stations $e_i$ is denoted by $t_{i,j}$, which follows exponential distributed random variable $\exp(\lambda_t(e_i, j))$ with parameter $\lambda_t(e_i, j)$.

The task of network control is to control the routes and the transferring sequence of the jobs. Based on the dynamic status of the queueing network, each station selects an appropriate job from its queue and sends it to an appropriate transfer station or its destination station. The control objective function is to maximize the time averaged weighted throughput (i.e., the weighted throughput rate) of the jobs across the network, which is defined as

$$\max \frac{1}{T} \sum_{p=1}^{n_T} w_p, \tag{1}$$

where $T$ is the running time, $n_T$ is the total number of jobs of all types passing across the network by time $T$, and $w_p$ $(1 \le p \le n_T)$ is the weight of the $p$th job passing through the network.

The problem addressed above is of a new queueing network problem with the following characteristics. (1) The first one is that the problem is an online dynamic control problem for multisource multisink networks with limited queue length. (2) The second one is self-organization characteristics of the jobs' transferring paths. There are multiple kinds of jobs with different destination stations. For arbitrary job, many alternative paths exist from the arrival station to its destination station. The most suitable path is not necessarily the shortest one or the one with the fewest transfer stations. Moreover, the more complex the network structure is, the more flexible the path selection is. Network control needs to be conducted considering the factors such as the global situation, the transferring time of each job on each station, the efficiency of each station, and the length of each station's queue. (3) The third one is the self-organization characteristics of network structure. The topological structure of the network is complex and may be dynamic; that is, the location and the number of stations and the relationship among the stations may vary over time. The control approach for the queueing network should be able to adapt to the changes of network topology structure.

In the following sections, an RL model is constructed to depict the above network control problem and an RL algorithm is proposed to deal with it.

## 3. The Reinforcement Learning Model

To depict the size of the feasible solution space and the difficulty of the self-organizing network control problem, we use a tandem queueing network control problem as an extremely simple example. This tandem queueing network is composed of $m$ tandem stations $(e_1, e_2, \ldots, e_m)$ and $n$ jobs of different types that need to be processed on each station in the order of $e_1, e_2, \ldots, e_m$. Suppose that each station processes only one job at a time and the processing sequences of the jobs on different stations are independent. For each station, there are $n!$ possible permutations of the jobs. Thus the number of feasible solutions to this $m$-station network control problem is $(n!)^m$. If $m = 7$ and $n = 4$, then $(n!)^m$ is an enormous figure much larger than $10^9$. Moreover, the general online self-organizing network control problem is much more complicated than the above tandem queueing network control problem with the same number of stations and job types. Due to the large scale of the self-organizing network, it is difficult to formulate the whole system as a unified model and solve it. We formulate the RL model of the self-organizing queueing network problem described in the previous section. According to the characteristics of the self-organizing queueing network control problem and following the decomposition-association strategy, the whole queueing network is decomposed into a number of closely connected small-scale subnetworks and a Markov Decision Process (MDP) model is constructed for each subnetwork. That is, the whole queueing network control problem is converted into a plurality of interconnected MDP problems. The subnetworks are connected by the coupling mechanism. By using this method, we can enhance the adaptability and robustness of the model and make it more adaptive to the changes of the topology structure of self-organizing networks so as to reduce the size of the problem and keep the essential structure of the original problem.

Construct a subnetwork for each station in the self-organizing queueing network. For each station, its corresponding subnetwork is centered on this station and contains its adjacent stations linked with this station. Each subnetwork corresponds to an RL subsystem which is used to solve the MDP model formulating the control problem for this subnetwork. The state transition of an RL subsystem directly causes the state variation of its adjacent RL subsystems, thus the adjacent RL subsystems are coupled in state transition.

Reinforcement learning (RL) is a machine learning method proposed to solve large-scale multistage decision problems or Markov Decision Processes with incomplete probability information. In the following we convert the self-organizing queueing network control problems of the subnetworks into RL problems, mainly including representation of states, construction of actions, and definition of the reward function. In this section we also introduce the coupling mechanism of the RL subsystems.

*3.1. States Representation.* State variables describe the primary characteristics of the RL subsystem. The $u$th $(u = 0, 1, 2, \ldots)$ state of the $k$th $(1 \le k \le m)$ RL subsystem

is represented by vector $s_{k,u}$, which is composed of state variables and defined as

$$s_{k,u} = \left[ j_{k,u}; q_{j,k,u} \ (1 \le j \le n); e_{D,k,u}; n_{E,k,u} \right], \quad (2)$$

where $j_{k,u}$ denotes the type of the job being transferred from the central station of the $k$th RL subsystem at the $u$th state ($j_{k,u}$ equals zero if the central station is idle), $q_{j,k,u}$ $(1 \le j \le n)$ denotes the number of type $j$ jobs waiting in the queue of the central station of the $k$th RL subsystem at the $u$th state, $e_{D,k,u}$ denotes the downstream station to which the central station of the $k$th RL subsystem is linking at the $u$th state, and $n_{E,k,u}$ denotes the number of upstream stations linking the central station of the $k$th RL subsystem at the $u$th state. The trigger events for state transition are arrival of a job and completion of transferring a job on a station.

*3.2. Actions.* When the central station of an RL subsystem is idle and a trigger event occurs, the RL subsystem selects an action. Since the task of the queueing network control is to control the routes and the transferring sequence of the jobs, an action of an RL subsystem contains two decisions: one decision is to determine which station this central station is going to connect to and the other decision is to select a job from the jobs waiting in its queue to transfer. For the $k$th subsystem, the number of available actions is $\sum_{e_i \in \Omega(e_k)} |\Psi(e_k, e_i)|$, where $\Omega(e_k)$ denotes the set of stations visible to station $e_k$, $\Psi(e_k, e_i)$ denotes the set of qualified job types that station $e_k$ may select and transfer to station $e_i$, and $|\Psi(e_k, e_i)|$ denotes the cardinality of set $\Psi(e_k, e_i)$. For an RL subsystem (e.g., the $k$th subsystem), where a trigger event occurs, a feasible action for this subsystem is to select station $e_i$ to link with and send a type $j$ job if all the following conditions are satisfied: (1) $e_i \in \Omega(e_k)$; that is, station $e_i$ is a visible station for station $e_k$; (2) the number of upstream stations currently transferring jobs station $e_i$ is less than the predetermined number $C_L$; (3) the queue length of station $e_i$ is less than $C_q - C_L$; (4) job type $j$ is a qualified job type for station $e_k$ to select and transfer to station $e_i$. Trivially, a null action (i.e., selecting no job) is selected if no job is waiting in the queue of station $e_k$ or one of the above conditions is not satisfied.

*3.3. The Reward Function.* The reward function indicates the instant and long-term impact of an action; that is, the immediate reward indicates the instant impact of an action and the average reward indicates the optimization of the objective function value. Thus, the whole RL system receives larger time averaged reward for larger time averaged weighted throughput. Let $\tau_{k,u}$ denote the time at the $u$th decision-making epoch of the $k$th RL subsystem, that is, the time when the state of the $k$th RL subsystem transfers from $s_{k,u-1}$ into $s_{k,u}$. Let $r(s_{k,u}, a_{k,u})$ denote the reward that the $k$th RL subsystem selects action $a_{k,u}$ at state $s_{k,u}$ and receives reward at time $\tau_{k,u+1}$. Without loss of generality, assume that the central station $e_k$ of the $k$th RL subsystem transfers a type $j$ job during time interval $(\tau_{k,u}, \tau_{k,u+1}]$ and completes transferring the job at time $\tau_{k,u+1}$; then $r(s_{k,u}, a_{k,u})$ is defined as

$$r(s_{k,u}, a_{k,u}) = \frac{w_j \left[ L_j(e_k) - L_{j,2} \right]}{D_j}, \quad (3)$$

where $w_j$ is the weight of the $j$th job type, $L_j(e_k)$ is the label of the central station $e_k$ for the $j$th job type, $L_{j,2}$ is the label of the downstream station to which the type $j$ job just transferred from station $e_k$, and $D_j$ is the length of the shortest path from the arriving station to the destination station of a type $j$ job (i.e., the least amount of flow time for a type $j$ job across the whole network from its arrival station to its destination station). The label of a station for the $j$th job type is defined as the length of the shortest path from the $k$th station to the destination station of the $j$th job type, that is, the least amount of flow time for a type $j$ job from the current station to the destination station. The immediate reward $r(s_{k,u}, a_{k,u})$ represents the progress of a job's passing through the network during the time between two state transitions. In the following we show the property of the reward function.

**Lemma 1.** *For a type $j$ job $(1 \le j \le n)$, assume that this job attains a transfer station $N_i$ with label $L_j(N_i)$. Whichever path this job attains transfer station $N_i$ through, the accumulated reward $R(N_i)$ of all RL subsystems caused by this job is*

$$\frac{w_j \left[ D_j - L_j(N_i) \right]}{D_j}, \quad (4)$$

*where $w_j$ is the weight of the $j$th job type and $D_j$ is the length of the shortest path from the arriving station to the destination station of the $j$th job type.*

*Proof.* Without loss of generality, assume that the job starts from its arrival station $b_j$ and before it attains station $N_i$, it is transferred successively by transfer stations $N_{i,1}, N_{i,2}, \ldots, N_{i,h}$. Let $L_j(N_{i,k})$ $(1 \le k \le h)$ denote the label of station $N_{i,k}$ for the $j$th job type. Then the reward caused by the type $j$ job during the process of being transferred from station $b_j$ to station $N_{i,1}$ is

$$\frac{w_j \left[ L_j(b_j) - L_j(N_{i,1}) \right]}{D_j} = \frac{w_j \left[ D_j - L_j(N_{i,1}) \right]}{D_j}. \quad (5)$$

Similarly, the reward caused by the job during the process of being transferred from station $N_{i,k}$ $(1 \le k \le h-1)$ to station $N_{i,k+1}$ is

$$\frac{w_j \left[ L_j(N_{i,k}) - L_j(N_{i,k+1}) \right]}{D_j}. \quad (6)$$

Consequently, the accumulated reward caused by the job during the process of being transferred from station $b_j$ to station $N_i$ is

$$R(N_i) = \frac{w_j \left[ D_j - L_j(N_{i,1}) \right]}{D_j} + \frac{w_j \left[ L_j(N_{i,1}) - L_j(N_{i,2}) \right]}{D_j}$$

$$+\frac{w_j\left[L_j\left(N_{i,2}\right)-L_j\left(N_{i,3}\right)\right]}{D_j}+\cdots$$
$$+\frac{w_j\left[L_j\left(N_{i,h-1}\right)-L_j\left(N_{i,h}\right)\right]}{D_j}$$
$$+\frac{w_j\left[L_j\left(N_{i,h}\right)-L_j\right]}{D_j}=\frac{w_j\left[D_j-L_j\left(N_i\right)\right]}{D_j}. \tag{7}$$

By Lemma 1 we obtain the following lemma. □

**Lemma 2.** *For a type $j$ job ($1 \le j \le n$), assume that this job attains its destination station $d_j$. Whichever path this job passes through the network from its arrival station to its destination station, the accumulated reward of all RL subsystems caused by this job during the whole process is $w_j$.*

*Proof.* By Lemma 1, the accumulated reward caused by this job during the whole process of passing through the network is

$$R\left(d_j\right)=\frac{w_j\left[D_j-L_j\left(d_j\right)\right]}{D_j}, \tag{8}$$

where $L_j(d_j)$ denotes the label of station $d_j$. Since $d_j$ is the destination station of type $j$ jobs, by the definition of a station's label we get $L_j(d_j) = 0$. Hence,

$$R\left(d_j\right)=\frac{w_j\left(D_j-0\right)}{D_j}=w_j. \tag{9}$$

A state transition takes place when a new job arrives at the network or a job is completely transferred by any station. Without loss of generality, we assume that the sojourn time of any arriving job in the network is finite. Hence, the total number of jobs staying in the network at any time is finite. According to Lemmas 1 and 2 we prove the following theorem. □

**Theorem 3.** *If there exists a positive integer $U$ such that the total number of jobs staying in the network is less than or equal to $U$, then maximizing the time averaged weighted throughput of the network (i.e., the control objective function (1)) is equivalent to maximizing the time averaged reward of all RL subsystems over infinite time.*

*Proof.* Assume that the jobs arriving at the network are divided into two sets $\varphi_1$ and $\varphi_2$. $\varphi_1$ is the set of jobs having passed through the network by time $T$ and $\varphi_2$ is the set of jobs still staying in the network at time $T$. Let $J_p$ denote the $p$th arriving job. Thus, according to Lemmas 1 and 2, the accumulated reward of all RL subsystems by time $T$ is

$$R_T=\sum_{J_p\in\varphi_1}w_p+\sum_{J_p\in\varphi_2}\frac{w_p\left(D_p-L_p\right)}{D_p}, \tag{10}$$

where $w_p$ is the weight of job $J_p$, $D_p$ is the length of the shortest path from the arrival station of job $J_p$ to the destination station of job $J_p$, and $L_p$ denotes the label of the station at which job $J_p$ is at time $T$. By definition, the time averaged weighted throughput by time $T$ is

$$\overline{P_T}=\frac{1}{T}\sum_{J_p\in\varphi_1}w_p. \tag{11}$$

It follows from (10) and (11) that

$$\overline{P_T}=\frac{1}{T}\left[R_T-\sum_{J_p\in\varphi_2}\frac{w_p\left(D_p-L_p\right)}{D_p}\right]. \tag{12}$$

Because the total number of jobs staying in the network is less than or equal to $U$, that is, $|\varphi_2| \le U$, we have

$$\sum_{J_p\in\varphi_2}\frac{w_p\left(D_p-L_p\right)}{D_p}\le\sum_{J_p\in\varphi_2}w_p\le U\max_{j\in\{1,2,\ldots,n\}}\left\{w_j\right\}. \tag{13}$$

It follows from (12) and (13) that

$$\frac{1}{T}\left(R_T-U\max_{j\in\{1,2,\ldots,n\}}\left\{w_j\right\}\right)\le\overline{P_T}\le\frac{R_T}{T}. \tag{14}$$

Since $U\max_{j\in\{1,2,\ldots,n\}}\{w_j\}$ is a constant, we have

$$\lim_{T\to+\infty}\frac{1}{T}\left(R_T-U\max_{j\in\{1,2,\ldots,n\}}\left\{w_j\right\}\right)=\lim_{T\to+\infty}\frac{R_T}{T}. \tag{15}$$

It follows from (14) and (15) that

$$\lim_{T\to+\infty}\overline{P_T}=\lim_{T\to+\infty}\frac{R_T}{T}. \tag{16}$$

Consequently, maximizing the time averaged weighted throughput is equivalent to maximizing the time averaged reward over infinite time. This links the long-term average reward of the RL system and the optimization of the objective function value for the network control problem. □

*3.4. The State Transition Mechanism and the Coupling Mechanism.* The trigger events for state transition in an RL subsystem are completion of transferring a job to the central station of this subsystem and completion of transferring a job from the central station of this subsystem. Take the $k$th RL subsystem as an example to illustrate the state transition mechanism. Currently the $k$th RL subsystem is at the $u$th decision-making state $s_{k,u}$. This subsystem takes an action $a_{k,u}$ and it transfers to an interim state $s^*_{k,u}$ immediately. When a trigger event for the $k$th RL subsystem takes place, the system transfers to a new decision-making state $s_{k,u+1}$ and receives a reward $r_{u+1}$, which is computed due to $s_{k,u}$ and $a_{k,u}$. The above procedure is repeated until a terminal state is attained; that is, all the jobs reach their destination stations. An episode is a trajectory from the initial state to a terminal state of a schedule horizon. With the states representation defined as (2), the decision process is a Markov Decision Process.

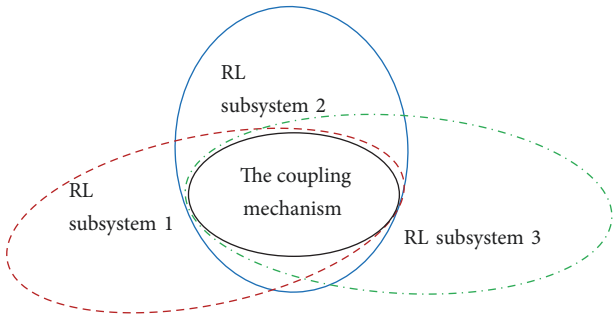


Figure 2: The constitution of the RL architecture.

An RL subsystem is coupled with another RL subsystem if a link is established between the central stations of these two subsystems. For example, for any two stations $e_i$ and $e_k$, if there is a link from $e_i$ to $e_k$ or from $e_k$ to $e_i$, then the two RL subsystems with central stations $e_i$ and $e_k$ are coupled with each other. The coupling mechanism of the RL subsystems (as shown in Figure 2) mainly contains the following two aspects. (1) The first one being the coupled correlation of the state transitions of coupling RL subsystems. One action in a subsystem can change the state of its corresponding coupling subsystem; that is, the state transition of a subsystem directly causes the variation of state variables of its corresponding coupling subsystem. (2) The second one being broadcast mechanism of reward signals in the coupling RL subsystems and the coupled iteration of state values of RL subsystems.

To describe the coupling mechanism more precisely and explain the overlap among subsystems, we give an illustrative example. Suppose that $e_i$ is a visible station to station $e_k$ and they are the central stations of the $k$th RL subsystem and the $i$th RL subsystem. Currently the $k$th RL subsystem is at the $u$th decision epoch and station $e_k$ is idle. The $k$th RL subsystem is at the $u$th decision-making state $s_{k,u}$ and the $i$th RL subsystem is at the $v$th state $s_{i,v}$, where $s_{k,u} = [0; q_{j,k,u}\ (1 \le j \le n); e_{D,k,u}; n_{E,k,u}]$ and $s_{i,v} = [j_{i,v}; q_{j,i,v}\ (1 \le j \le n); e_{D,i,v}; n_{E,i,v}]$. At this decision epoch, the $k$th subsystem takes an action $a_{k,u}$ which selects a job of type $j^*$ waiting in its queue, establishes a link with station $e_i$, and transfers the job to station $e_i$. The $k$th subsystem transfers to an interim state $s^*_{k,u} = [j^*; q_{j,k,u}\ (1 \le j < j^*), q_{j^*,k,u} - 1, q_{j,k,u}\ (j^* < j \le n); e_i; n_{E,k,u}]$ immediately. The difference of states $s^*_{k,u}$ and $s_{k,u}$ is that the job type being transferred on station $e_k$ is $j^*$, the number of type $j^*$ jobs waiting the queue of station $e_k$ decreases by one, and the downstream station to which station $e_k$ is linked is station $e_i$. The $i$th subsystem transfers to an interim state $s^*_{i,v} = [j_{i,v}; q_{j,i,v}\ (1 \le j \le n); e_{D,i,v}; n_{E,i,v}+1]$ immediately. The difference of states $s^*_{i,v}$ and $s_{i,v}$ is that the number of upstream stations linking station $e_i$ increases by one. When station $e_k$ completes transferring the type $j^*$ job, the $k$th subsystem receives a reward $r(s_{k,u}, a_{k,u})$ and it transfers to the next decision-making state $s_{k,u+1} = [0; q_{j,k,u}\ (1 \le j < j^*), q_{j^*,k,u} - 1, q_{j,k,u}\ (j^* < j \le n); e_i; n_{E,k,u}]$. The $i$th subsystem transfers to the next decision-making state $s_{i,v+1} = [j_{i,v}; q_{j,i,v}\ (1 \le j < j^*), q_{j^*,k,u} + 1, q_{j,i,v}\ (j^* < j \le n); e_{D,i,v}; n_{E,i,v}+1]$ simultaneously. For each RL subsystem, the state transition process continues as above until all the jobs reach their destination stations.

For the $k$th RL subsystem, when its central station completes transferring a type $j$ job at the $(u+1)$th decision epoch, the state value of the $k$th RL subsystem is updated with its immediate reward $r(s_{k,u}, a_{k,u})$ following the proposed RL algorithm. The state values of all the subsystems coupled with the $k$th subsystem are also updated. Assume that the $i$th subsystem is coupled with the $k$th subsystem; then the virtual reward, denoted by $r(s_{k,u}, a_{k,u}, i)$, for updating the state value of the $i$th subsystem is defined as

$$r(s_{k,u}, a_{k,u}, i) = \frac{D_j - t_{k,j}}{D_j} r(s_{k,u}, a_{k,u}), \tag{17}$$

where $D_j$ is the shortest path from the arrival station to the destination station of the $j$th job type and $t_{k,j}$ is the transfer time of a type $j$ job from station $e_k$. The detailed computation procedure of the RL algorithm is shown in Section 4.

## 4. A Reinforcement Learning Algorithm with Time Averaged Reward

The online self-organizing queueing network control problem is converted into an RL problem in the previous section. We apply reinforcement learning to solve the RL problem and use $\varepsilon$-greedy policy to balance exploration and exploitation. $\varepsilon$-greedy policy means that the algorithm selects the greedy action with probability $1 - \varepsilon$ and selects an available action randomly with probability $\varepsilon$, where $\varepsilon$ $(0 \leqslant \varepsilon \leqslant 1)$ is usually a small positive number.

We propose the following reinforcement learning algorithm (Algorithm 4), where $K$ is the number of RL subsystems, $S_k$ denotes the state space of the $k$th RL subsystem, $V_k(s_{k,u})$ denotes the value of state $s_{k,u}$, $\alpha$ is the learning rate for state values, $\rho_k$ denotes the estimated reward rate of the $k$th RL subsystem, $\beta$ is the learning rate for the estimated reward rates, $N$ is the total number of jobs required to pass through the network, and $c$ is the number of jobs currently having passed the network.

*Algorithm 4* (a reinforcement learning algorithm with time averaged reward for online self-organizing queueing network control problem).

*Step 1.* For each job type $j$ $(1 \le j \le n)$, create a network $G_j$ composed of the stations qualified to transfer type $j$ jobs. For each station $e_i$ $(1 \le i \le m)$ in network $G_j$, compute its label $L_j(e_i)$.

*Step 2.* Set parameters $\alpha$, $\beta$, $\varepsilon$, and $N$ to the predetermined values and initialize $c$ to be zero. For each $k$ $(1 \le k \le K)$, set $u = 0$ and current time $\tau_{k,u} = 0$ and initialize $\rho_k$ to be one. For each $k$ $(1 \le k \le K)$, set current state $s_{k,u}$ as the initial state $s_{k,0}$ and initialize $V_k(s_k) = 1$ for all $s_k$ $(s_k \in S_k)$; that is, initialize the state value function for all states of all RL subsystems.

*Step 3.* Determine the station where the trigger event occurs (e.g., station $e_k$). Determine the current state $s_{k,u}$ for station

$e_k$ and the set of available actions for the $k$th RL subsystem at state $s_{k,u}$.

*Step 4*. Select action $a_{k,u}$ based on the current state value $V_k(s_{k,u})$ of the $k$th RL subsystem following the $\varepsilon$-greedy control policy.

*Step 5*. Implement action $a_{k,u}$ and determine the next time $\tau_{k,u+1}$, the time at the $(u+1)$th decision-making epoch of the $k$th RL subsystem, following the state transition mechanism. Determine the state $s_{k,u+1}$ at time $\tau_{k,u+1}$ and calculate reward $r(s_{k,u}, a_{k,u})$ by (3).

*Step 6*. Update the state value $V_k(s_{k,u})$ as

$$\begin{aligned} V_k(s_{k,u}) = V_k(s_{k,u}) + \alpha \Big\{ & r(s_{k,u}, a_{k,u}) \\ & - \rho_k(\tau_{k,u+1} - \tau_{k,u}) \\ & + \max_{a'} \left\{ r(s_{k,u+1}, a') - \rho_k(\tau' - \tau_{k,u+1}) + V_k(s') \right\} \\ & - V_k(s_{k,u}) \Big\}. \end{aligned} \tag{18}$$

Update $\rho_k$ as

$$\begin{aligned} \rho_k = \rho_k + \frac{\beta}{\tau_{k,u+1} - \tau_{k,u}} \Big\{ & r(s_{k,u}, a_{k,u}) \\ & - \rho_k(\tau_{k,u+1} - \tau_{k,u}) \\ & + \max_{a'} \left\{ r(s_{k,u+1}, a') - \rho_k(\tau' - \tau_{k,u+1}) + V_k(s') \right\} \\ & - \max_{a} \left\{ r(s_{k,u}, a) - \rho_k(\tau - \tau_{k,u}) + V_k(s) \right\} \Big\}, \end{aligned} \tag{19}$$

where $a'$ denotes an available action which may be taken at state $s_{k,u+1}$, $s'$ denotes the state to which if action $a'$ is taken at state $s_{k,u+1}$, then the $k$th subsystem is transferred, $\tau'$ denotes the time when the next state transition takes place if action $a'$ is taken at state $s_{k,u+1}$, $a$ denotes an available action which may be taken at state $s_{k,u}$, $s$ denotes the state to which if action $a$ is **taken** at state $s_{k,u}$, then the $k$th subsystem is transferred, and $\tau$ denotes the time when the next state transition takes place if action $a$ is taken at state $s_{k,u}$.

*Step 7*. Update the RL subsystems coupled with the $k$th RL subsystem as follows. For each RL subsystem coupled with the $k$th RL subsystem (e.g., the $i$th subsystem at state $s_{i,v}$), compute $r(s_{k,u}, a_{k,u}, i)$ by (17) and then update $V_i(s_{i,v})$ and $\rho_i$ as

$$\begin{aligned} V_i(s_{i,v}) = V_i(s_{i,v}) + \alpha & \left\{ 1 - \frac{\rho_i(\tau_{k,u+1} - \tau_{k,u})}{r(s_{k,u}, a_{k,u})} \right\} \\ & \cdot r(s_{k,u}, a_{k,u}, i), \end{aligned}$$

$$\begin{aligned} \rho_i = \rho_i + \frac{\beta}{\tau_{k,u+1} - \tau_{k,u}} & \left\{ 1 - \frac{\rho_i(\tau_{k,u+1} - \tau_{k,u})}{r(s_{k,u}, a_{k,u})} \right\} \\ & \cdot r(s_{k,u}, a_{k,u}, i). \end{aligned} \tag{20}$$

*Step 8*. Set $u = u + 1$ and adjust the current time by $\tau_{k,u+1}$ for the $k$th subsystem. Update the current states of all subsystems following the state transition mechanism.

*Step 9*. If the trigger event is that a job finishes passing through the network, then $c = c + 1$. If the number of jobs across the network is $N$, then the algorithm terminates; otherwise go to Step 3.

## 5. Computational Experiments

In this section, we conduct computational experiments to examine the learning ability and the performance of the proposed reinforcement learning algorithm (Algorithm 4). We first use a queueing network with four types of jobs as the test bed. A test problem specifies the number of jobs to be scheduled and each problem generates 50 instances. To verify the convergence of the state value function during the learning process, a test problem with 10000 jobs is used for each instance. An instance is the whole process of generating a schedule for the 10000 jobs in an instance from the initial state to the state when all the jobs have passed through the network. The weights of all types of jobs $w_j$ $(1 \le j \le n)$ are in the range $[0.1, 1]$ and the parameters of transferring times $\lambda_t(e_i, j)$ and link establishing times $\lambda_\eta(e_i, e_k)$ are in the range $[1, 10]$. The transferring times and link establishing times are exponentially distributed. The parameters of the proposed algorithm are set with $\alpha = 0.01$, $\beta = 0.05$, $\varepsilon = 0.01$, and $\rho_k = 1$ $(1 \le k \le K)$. The maximum size of the queues of transfer stations $C_q$ is set to be 5.

To investigate the convergence of the state value function, we examine the variation of the state values during the learning process. The experiment results in this section take the average over 50 instances. Let $N_{s,k}$ denote the number of states in the state space of the $k$th subsystem and $V_s$ denote the mean value of all states. $V_s$ is defined as (21), where $S_k$ denotes the state space of the $k$th subsystem and $V_k(s_k)$ denotes the value of state $s_k$ $(s_k \in S_k)$. Figure 3 shows the variation of $V_s$ with respect to the number of jobs having crossed the network. When the number of finished jobs is larger than 3000, $V_s$ decreases slowly and gradually converges to $-20.53$.

$$V_s = \frac{1}{\sum_{k=1}^{K} N_{s,k}} \sum_{k=1}^{K} \sum_{s_k \in S_k} V_k(s_k). \tag{21}$$

Let $V_p(s_k)$ denote the value of state $s_k$ $(s_k \in S_k)$ at the time when the $p$th job completely passes through the network. Let $\mathrm{ADV}_s$ denote the average value of the absolution of the difference of values of all states between the two time points when the successive two jobs completely cross the network.

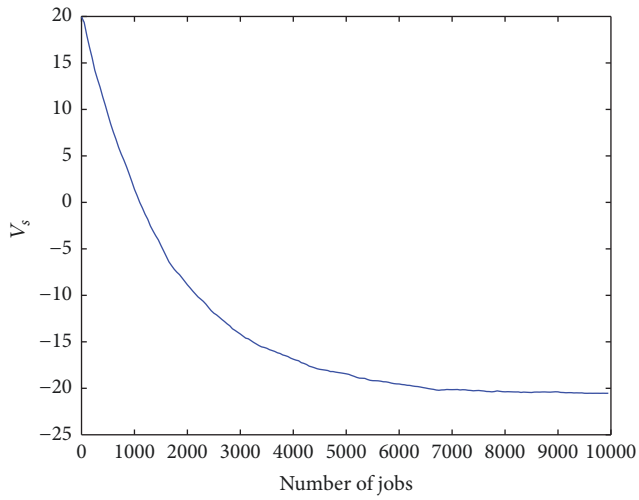


Figure 3: Variation of $V_s$ with respect to the number of finished jobs.

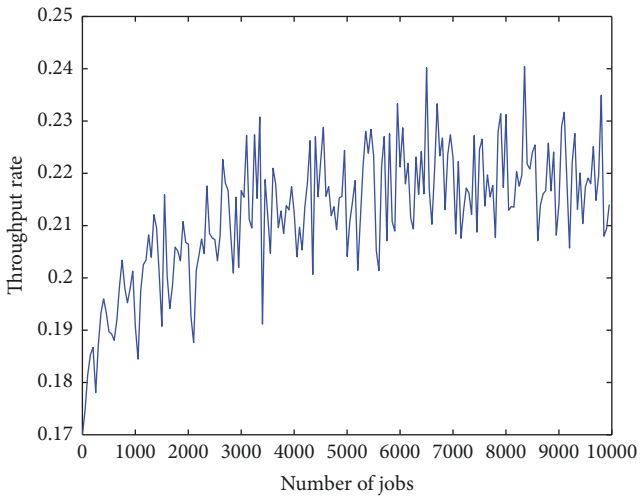


Figure 5: The Learning Curve of weighted throughput rate.

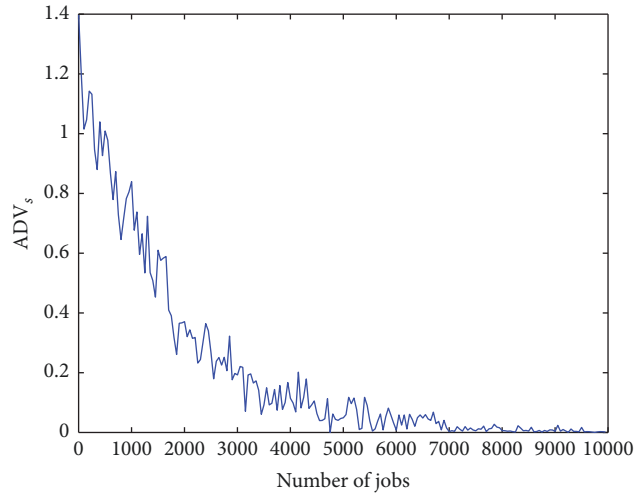


Figure 4: Variation of $ADV_s$ with respect to the number of finished jobs.

For example, at the time when the $p$th job completely passed through the network, $ADV_s$ is defined as

$$ADV_s = \frac{1}{\sum_{k=1}^{K} N_{s,k}} \sum_{k=1}^{K} \sum_{s_k \in S_k} \left| V_p(s_k) - V_{p-1}(s_k) \right|. \quad (22)$$

Figure 4 shows the variation of $ADV_s$ with respect to the number of finished jobs. Although both the curves in Figures 3 and 4 converge asymptotically, the shape of the curve in Figure 4 is not so smooth as Figure 3. When the number of finished jobs is larger than 3000, $ADV_s$ is less than 0.2. Figures 3 and 4 show that when the number of finished jobs increases, $ADV_s$ asymptotically converges to zero which indicates that the state values are gradually stable.

For a given problem, we can draw a "Learning Curve" to examine the learning ability of the proposed reinforcement learning algorithm. In a Learning Curve, the objective function values are averaged over 50 instances. As shown in Figure 5, $X$-coordinate represents the number of finished jobs and $y$-coordinate represents the time averaged weight throughput from the initial time to the current time. For example, the point $(2000, 0.196)$ on the Learning Curve indicates that the weighted throughput rate from the initial time to the time when the 2000th job is finished is 0.196. The Learning Curve increases asymptotically and rapidly in the first 3000 jobs and then fluctuates in the latter jobs. This curve shows that RL system learns quickly through interaction and finds a good policy in the former 3000 jobs. Thereafter, the improvement of the control policy gradually slows down in the latter jobs.

To validate the adaptability of Algorithm 4 for various problems and examine the effectiveness of Algorithm 4, more extensive test problems are also randomly generated and conducted to demonstrate the performance of Algorithm 4. We consider the networks with different topology structures corresponding to the test problems with different numbers of stations ($m$ takes 10, 15, 20, 25, 30, 35, and 40, resp.). For a specific number of stations, the values of the relative arrival intensity index (RAII) is, respectively, 0.50, 0.75, 1.00, 1.50, and 2.00. The RAII index indicates the number of arrival jobs at the arrival stations in these extensive test problems relative to the above test problem. The larger RAII index is, the more jobs arrive at the arrival stations. Each extensive test problem generates 50 instances. We use three approaches, the $\varepsilon$-greedy approach (Algorithm 4), the completely random routing (CRR) approach, and the purely greedy routing (PGR) approach, to solve the test problems. The CRR approach and the PGR approach correspond to $\varepsilon = 1$ and $\varepsilon = 0$, respectively. Table 1 shows the time averaged weighted throughput (AWT) of the jobs across the network, respectively, obtained by three comparative approaches when 10000 jobs have passed through the network. The experiment results in Table 1 are also averaged over 50 instances. As shown in Table 1, the AWT index increases with respect to the growth of the RAII index. When the RAII index is larger than one, the growth rate of the AWT index is slower than the case that the RAII index is less than one, since high intensity

Table 1: The time averaged weighted throughput (AWT) of the extensive test problems.

| RAII | | $m$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 10 | 15 | 20 | 25 | 30 | 35 | 40 |
| 0.50 | CRR | 0.076 | 0.079 | 0.091 | 0.102 | 0.117 | 0.131 | 0.147 |
| | PGR | 0.089 | 0.101 | 0.112 | 0.127 | 0.143 | 0.161 | 0.175 |
| | $\varepsilon$-greedy | 0.092 | 0.104 | 0.115 | 0.131 | 0.147 | 0.166 | 0.183 |
| 0.75 | CRR | 0.105 | 0.117 | 0.130 | 0.153 | 0.174 | 0.198 | 0.230 |
| | PGR | 0.129 | 0.142 | 0.165 | 0.185 | 0.215 | 0.242 | 0.276 |
| | $\varepsilon$-greedy | 0.134 | 0.147 | 0.170 | 0.192 | 0.222 | 0.251 | 0.286 |
| 1.00 | CRR | 0.171 | 0.195 | 0.221 | 0.258 | 0.282 | 0.304 | 0.338 |
| | PGR | 0.209 | 0.248 | 0.275 | 0.317 | 0.348 | 0.389 | 0.419 |
| | $\varepsilon$-greedy | 0.218 | 0.257 | 0.284 | 0.329 | 0.363 | 0.403 | 0.431 |
| 1.50 | CRR | 0.228 | 0.255 | 0.285 | 0.344 | 0.383 | 0.428 | 0.469 |
| | PGR | 0.285 | 0.323 | 0.357 | 0.424 | 0.481 | 0.537 | 0.597 |
| | $\varepsilon$-greedy | 0.295 | 0.335 | 0.373 | 0.439 | 0.502 | 0.561 | 0.618 |
| 2.00 | CRR | 0.267 | 0.296 | 0.324 | 0.398 | 0.446 | 0.503 | 0.571 |
| | PGR | 0.338 | 0.375 | 0.415 | 0.493 | 0.574 | 0.642 | 0.706 |
| | $\varepsilon$-greedy | 0.352 | 0.391 | 0.433 | 0.511 | 0.598 | 0.669 | 0.735 |

Table 2: The relative AWT value of the three approaches.

| RAII | | $m$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 10 | 15 | 20 | 25 | 30 | 35 | 40 |
| 0.50 | CRR | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | PGR | 1.178 | 1.278 | 1.226 | 1.239 | 1.219 | 1.229 | 1.206 |
| | $\varepsilon$-greedy | 1.212 | 1.314 | 1.260 | 1.280 | 1.252 | 1.268 | 1.241 |
| 0.75 | CRR | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | PGR | 1.231 | 1.215 | 1.269 | 1.208 | 1.234 | 1.222 | 1.199 |
| | $\varepsilon$-greedy | 1.277 | 1.256 | 1.310 | 1.255 | 1.276 | 1.268 | 1.241 |
| 1.00 | CRR | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | PGR | 1.218 | 1.271 | 1.245 | 1.230 | 1.234 | 1.303 | 1.266 |
| | $\varepsilon$-greedy | 1.272 | 1.318 | 1.285 | 1.275 | 1.287 | 1.351 | 1.315 |
| 1.50 | CRR | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | PGR | 1.252 | 1.267 | 1.256 | 1.231 | 1.256 | 1.255 | 1.246 |
| | $\varepsilon$-greedy | 1.296 | 1.312 | 1.310 | 1.275 | 1.310 | 1.311 | 1.292 |
| 2.00 | CRR | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| | PGR | 1.269 | 1.268 | 1.281 | 1.238 | 1.286 | 1.277 | 1.237 |
| | $\varepsilon$-greedy | 1.320 | 1.321 | 1.336 | 1.283 | 1.341 | 1.330 | 1.287 |

of arrival jobs leads to congestion of the networks. For each test problem, $\varepsilon$-greedy approach obtains larger AWT index than the CRR approach and the PGR approach. Table 2 lists the relative AWT values of the $\varepsilon$-greedy approach and the PGR approach. The relative AWT value of a problem for an approach is defined as the AWT index obtained by this approach divided by the AWT index obtained by the CRR approach.

As shown in Table 2, the relative AWT values obtained by the PGR approach range from 1.178 to 1.303 and the relative AWT values obtained by the $\varepsilon$-greedy approach range from 1.212 to 1.351. For the test problems with the RAII index taking 0.50, 0.75, 1.00, 1.50, and 2.00, respectively, the average relative AWT values obtained by the PGR approach are 1.223, 1.226, 1.249, 1.256, and 1.265, respectively, and the average relative AWT values obtained by the $\varepsilon$-greedy approach are 1.262, 1.269, 1.291, 1.305, and 1.316, respectively. Compared with the CRR approach, the PGR approach and the $\varepsilon$-greedy approach improve the weighted throughput rate with an average proportion of 24.4% and 28.8%, respectively. Experiment results show that the $\varepsilon$-greedy approach is superior to both the CRR approach and the PGR approach. Experiment results validate the adaptability and robustness of the proposed algorithm for test problems of various scales and different topology structures. Experiment results also indicate that the reinforcement learning system learns to select an appropriate action on different occasions, links stations and schedules jobs flexibly in online environment, and obtains optimized results.

## 6. Conclusions

We decompose the investigated online self-organizing queueing network control problem with time averaged weighted throughput objective into a series of cross-correlated Markov Decision Processes and convert them into a coupled reinforcement learning model. In the reinforcement learning system, maximizing the time averaged weighted throughput of all jobs across the network is equivalent to maximizing the time averaged reward of all subsystems. Online reinforcement learning algorithm with time averaged reward is adopted to solve the reinforcement learning model and obtains a control policy integrating the jobs routing selection strategy and the jobs sequencing strategy. Computational experiments verify the convergence of the state value function through the learning process and the effectiveness of the proposed algorithm applied in the self-organizing queueing network control problem. In the test problems, the proposed algorithm improves the weighted throughput rate of the networks with a remarkable average proportion through online learning process. The experiment results show that the reinforcement learning system is adaptive to different network topology structures and it learns an optimized policy through interaction with the control process.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper. The mentioned received funding in the "Acknowledgments" did not lead to any conflicts of interest regarding the publication of this manuscript.

## Acknowledgments

This work is supported by Natural Science Foundation of Guangdong Province (no. 2015A030313649, no. 2015A030310274), Science and Technology Planning Project of Guangdong Province, China (no. 2015A010103021), Key Platforms and Scientific Research Program of Guangdong Province (Featured Innovative Projects of Natural Science, no. 2015KTSCX137), and National Natural Science Foundation of China (Grant no. 61703102).

## References

[1] A. K. Nandi and H. R. Medal, "Methods for removing links in a network to minimize the spread of infections," *Computers & Operations Research*, vol. 69, pp. 10–24, 2016.

[2] F. Carrabs, R. Cerulli, C. D'Ambrosio, M. Gentili, and A. Raiconi, "Maximizing lifetime in wireless sensor networks with multiple sensor families," *Computers & Operations Research*, vol. 60, article no. 3734, pp. 121–137, 2015.

[3] M. E. Keskin, I. K. Altinel, and N. Aras, "Combining simulated annealing with Lagrangian relaxation and weighted Dantzig-Wolfe decomposition for integrated design decisions in wireless sensor networks," *Computers & Operations Research*, vol. 59, pp. 132–143, 2015.

[4] M. Rebai, M. Le Berre, H. Snoussi, F. Hnaien, and L. Khoukhi, "Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks," *Computers & Operations Research*, vol. 59, pp. 11–21, 2015.

[5] A. A. Hanbali, R. Haan, R. J. Boucherie et al., "A tandem queueing model for delay analysis in disconnected ad hoc networks," in *Proceedings of the International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pp. 189–205, 2008.

[6] A. Chattopadhyay and A. Chockalingam, "Past queue length based low-overhead link scheduling in multi-beam wireless mesh networks," in *Proceedings of the 2010 International Conference on Signal Processing and Communications, SPCOM '10*, IEEE, Bangalore, India, July 2010.

[7] W. Fu and D. P. Agrawal, "Effective radio partitioning and efficient queue management schemes in a wireless mesh network," in *Proceedings of the IEEE Global Telecommunications Conference*, pp. 1–5, 2008.

[8] W. He, S. Yang, D. Teng et al., "A link level load-aware queue scheduling algorithm on mac layer for wireless mesh networks," in *Proceedings of the International Conference on Wireless Communications and Signal Processing*, 2009.

[9] B. Pinheiro, V. Nascimento, R. Gomes et al., "A multimedia-based fuzzy queue-aware routing approach for wireless mesh networks," in *Proceedings of the 20th International Conference on Computer Communications and Networks, ICCCN '11*, pp. 1–7, IEEE, Hawaii, Hawaii, USA, August 2011.

[10] C. H. P. Augusto, C. B. Carvalho, M. W. R. Da Silva, and J. F. De Rezende, "REUSE: a combined routing and link scheduling mechanism for wireless mesh networks," *Computer Communications*, vol. 34, no. 18, pp. 2207–2216, 2011.

[11] N. S. Nandiraju, D. S. Nandiraju, D. Cavalcanti et al., "A Novel Queue Management Mechanism for Improving Performance of Multihop Flows in IEEE 802.11s based Mesh Networks," in *Proceedings of the 25th IEEE International Performance, Computing, and Communications Conference*, pp. 161–168, IEEE, Arizona, Ariz, USA, April 2006.

[12] G. R. Gupta and N. B. Shroff, "Practical scheduling schemes with throughput guarantees for multi-hop wireless networks," *Computer Networks*, vol. 54, no. 5, pp. 766–780, 2010.

[13] J. Nieminen, H. Paloheimo, and R. Jäntti, "Energy-adaptive scheduling and queue management in wireless LAN mesh networks," in *Proceedings of the 5th Annual International Wireless Internet Conference, WICON '10*, pp. 1–9, March 2010.

[14] K. Wang, C. F. Chiasserini, J. G. Proakis, and R. R. Rao, "Joint scheduling and power control supporting multicasting in wireless ad hoc networks," *Ad Hoc Networks*, vol. 4, no. 4, pp. 532–546, 2006.

[15] Y. Liu, T. Wu, and X. Xian, "Queue control based delay analysis and optimization for wireless mesh networks," in *Proceedings of the 2nd International Conference on Education Technology and Computer, ICETC '10*, pp. 104–108, IEEE, Shanghai, China, 2010.

[16] M. Kim, V. K. S. Iyer, and P. Ning, "MrFair: misbehavior-resistant fair scheduling in wireless mesh networks," *Ad Hoc Networks*, vol. 10, no. 3, pp. 299–316, 2012.

[17] N. Vučević, J. Pérez-Romero, O. Sallent, and R. Agustí, "Reinforcement learning for active queue management in mobile all-ip networks," in *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'07*, IEEE, Athens, Greece, September 2007.

[18] Y. Zhou, M. Yun, T. Kim, A. Arora, and H.-A. Choi, "RL-based queue management for QoS support in multi-channel multi-radio mesh networks," in *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications, NCA '09*, pp. 306–309, IEEE, Massachusetts, Mass, USA, July 2009.