He et al. EURASIP Journal on Wireless Communications and Networking (2018) 2018:150
https://doi.org/10.1186/s13638-018-1170-x

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Multi-objective virtual network embedding algorithm based on Q-learning and curiosity-driven

Mengyang He, Lei Zhuang*, Shuaikui Tian, Guoqing Wang and Kunli Zhang

Abstract

Network virtualization is a vital technology that helps overcome shortcomings such as network ossification of the current Internet architecture. However, virtual network embedding (VNE) involving the allocation of resources for heterogeneous virtual network requests (VNRs) on the substrate network (SN) is considered as NP-hard problem. VNE process may involve conflicting objectives, including energy saving and VNR acceptance rate as the most critical. In this paper, we propose a virtual network multi-objective embedding algorithm based on Q-learning and curiosity-driven (Q-CD-VNE) for improving the performance of the system by optimizing conflicting objectives, namely energy saving and acceptance rate. The proposed algorithm employs Q-learning and curiosity-driven mechanism by considering other non-deterministic factors to avoid falling into a local optimum. The major contributions of this work involve (1) modeling of the multi-objective deterministic factors as binary (0, 1) integer programming problem, (2) formulating the virtual node mapping problem using the Markov decision process (MDP), (3) solving the VNE problem using Q-learning algorithm, (4) mining non-deterministic factors using curiosity-driven mechanism for avoiding prematurely falling into the Exploration-Exploitation dilemma and local optimal. Experimental results in comparison with representative researches in the field prove that the proposed algorithm can reduce energy consumption, improve the request acceptance rate, and improve the long-term average income.

Keywords: Network virtualization, Q-learning, Curiosity-driven, Energy-aware

1 Introduction

Network virtualization is considered as an important technology of next-generation Internet for addressing the growing problem of network ossification [1]. In the recent past, the energy-aware virtual network embedding (EEVNE) remained as a focus of the current research community in the field for tackling this problem. EEVNE refers to shifting the virtual network embedding (VNE) problem from the utility to the power consumption of the substrate networks (SNs). The main aim is to reduce energy consumption of SNs in the mapping process for slicing the cost of VNE operation and maintenance. Request acceptance rate is one of the most important indicators in the VNE processes. It describes success rate of VNRs. Therefore, quality of the request acceptance rate directly reflects the quality of the VNE algorithm.

Most of the existing energy-efficient strategies usually search the subset of resources in the whole SNs for the VNs. Whereas, resource consolidation achieves the minimization of energy consumption by switching off or hibernating as many physical infrastructures as possible like physical servers and fiber optic links. However, this process may lead to the hotspots among SNs. Thus, it can result in a sharp decrease in request acceptance rate. Here, these objectives are conflicting in nature. Therefore, comprehensive consideration of these conflicting objectives has become a critical problem that requires immediate attention for resolving it.

In addition, there exist multiple factors known as non-deterministic factors that affect VNE performance, such as load balancing, mapping duration, and fragmentation rate [2–6]. These factors should be taken into consideration in specific scenarios for obtaining an accurate solution to the problem.

* Correspondence: ielzhuang@zzu.edu.cn
School of Information and Engineering, Zhengzhou University, Zhengzhou, China

In this work, we propose a VNE method based on improved Q-learning algorithm, which considers the factor of energy saving and VNR acceptance rate as the main optimization objectives (termed deterministic factors). We also explored non-deterministic factors for ensuring guaranteed performance of deterministic factors. The proposed method is a two-phase embedding algorithm that maps nodes followed by mapping of the links. It employs the Q-learning algorithm of reinforcement learning and a curiosity-driven mechanism to map the virtual nodes and shortest-path algorithm for mapping virtual links.

In summary, the proposed method firstly performs multi-objective modeling of deterministic factors as binary (0–1) integer programming problem. Then, it formalizes the virtual node mapping problem using the Markov decision process (MDP), and the node mapping algorithm is taken as an intelligent agent to aware the substrate environment. Finally, it employs Q-learning algorithm to solve the model, in combination with a curiosity-driven mechanism for mining non-deterministic factors. In this work, the optimization of the deterministic factors is used as an exploitation value, and the optimization of non-deterministic factors are used as an exploration value. The proposed method generates a trade-off solution to avoid prematurely falling into the Exploration-Exploitation dilemma and local optimal with a limited number of requests.

Major contributions of this study are as described below:

1. Modeling of the multi-objective deterministic factors as binary (0, 1) integer programming problem.
2. Formulating the virtual node mapping problem using the MDP.
3. Solving the VNE problem using Q-learning algorithm.
4. Mining non-deterministic factors using curiosity-driven mechanism for avoiding prematurely falling into the Exploration-Exploitation dilemma and local optimal.

The remainder of this paper is organized as follows. Section 2 summarizes the state of the art highlighting the significant researches in the field of VNE. Section 3 describes the methods for modeling the VNE problem as binary (0–1) integer programming and use of MDP. Section 4 details the proposed method for solving VNE problem using Q-learning and the curiosity-driven mechanism. Section 5 presents a performance evaluation of the proposed method and presented experimental results followed by their discussion. Finally, Section 6 concludes the proposed work at the end of this paper.

2 Related research

Numerous studies have been conducted for solving VNE problem by considering its different perspectives. Yu et al. [7] proposed an effective competition algorithm for tackling VNE problem of non-division of path. They authors claimed the optimization of request acceptance rate of the virtual network mapping. Triki and Kara [8] proposed a solution to energy-saving mapping problem of virtual networks and provided an analysis of energy consumption, request priority, and requested distance constraints. Guan and Choi [9] studied data centers from both computing resources and network resources. Chen et al. [10] proposed the construction of dictionary library through historical data in the early stage of virtual network mapping and deployed the virtual network in a small-scale nodes and link set, thereby achieved saving in energy.

It can be noticed from the above cited work that researchers focused on the optimization of a single goal. Most of the researches lack in simultaneous consideration of both deterministic factors along with non-deterministic factors.

In recent years, machine learning techniques, particularly reinforcement learning technique, have been widely used to deal with decision-making issues. Karimpanal and Wilhelm [11] employed off-policy learning to address multiple objectives using adaptive clustering Q-learning algorithm. The authors proposed that agents can effectively use their own exploration behavior by identifying the possible goals in the environment to find effective strategies in the case of unknown goals. Vamvoudakis et al. [12] proposed the use of Q-learning technique for continuous-time-based graphical games on large networks with completely unknown linear system dynamics. They used the model-free formula of Q-learning function to model a large network to meet the user-defined distributed optimization criteria and used it for the agent having no information about the leader. The Q function parameterizes each neighborhood tracking error of the agent and results in a better strategy.

Curiosity has always been considered as one of the basic attributes of intelligence, and giving curiosity to machines is also an important research objective in the field of computer science. Hester et al. [13] attempted to make the machine curious by using reinforcement learning technique. They proposed an intrinsically motivating model-based reinforcement learning algorithm that allows agents to explore themselves. Pathak and Agrawal [14] proposed a self-supervised prediction algorithm based upon curiosity-driven exploration. In many real-world scenarios, curiosity can be used as an intrinsic reward signal when the external rewards are scarce or absent that allows the agents to explore the external environment and learning. Kompella and Stollenga [15] proposed a curiosity-driven approach to enable the agents to link intrinsic rewards with the improvement of

the external world model for acquiring new skills in absence of external guidance.

Findings of the literature cited above indicate that reinforcement learning method based on the curiosity-driven has great advantages and feasibility in discovering new skills and actions. Therefore, it is necessary to further analyze the problem of mining non-deterministic factors in multi-objective embedding of virtual networks using reinforcement learning method.

3 Problem formulation

3.1 Optimization model of deterministic factors

In this work, we proposed an optimization model for deterministic factors, namely energy saving and VNR acceptance rate, using binary (0–1) integer programming method. The proposed model considers the minimization of energy consumption and maximization of request acceptance rate. The objective function can be mathematically represented as shown in Eq. (1).

$$\max(\omega \text{acceptance} - (1-\omega)\cdot f(\text{energy})) \tag{1}$$

Subject to:

$$\left(\forall i \in N^V\right)\left(\forall j \in N^S\right) : f_j^i \cdot \text{ReqCPU}(i) \le \text{CPU}(j) \tag{2}$$

$$\left(\forall l_{uw} \in L^V\right)\left(\forall l_{mn} \in L^S\right) : f_{mn}^{uw} \cdot \text{ReqBWL}(l_{uw}) \le \text{BWL}(l_{mn}) \tag{3}$$

$$\left(\forall i \in N^V\right) : \sum\nolimits_{j \in N^S} f_j^i = 1 \tag{4}$$

$$\left(\forall j \in N^S\right) : \sum\nolimits_{i \in N^V} f_j^i \le 1 \tag{5}$$

$$\left(\forall i \in N^V\right)\left(\forall j \in N^S\right) : f_j^i \in \{0,1\} \tag{6}$$

$$\left(\forall l_{uw} \in L^V\right)\left(\forall l_{mn} \in L^S\right) : f_{mn}^{uw} \in \{0,1\} \tag{7}$$

where ω is a weighting factor that is used to adjust the weight of VNR acceptance rate and energy consumption. It is applied to meet different requirements in different scenarios.

acceptance describes VNR's acceptance rate, as defined in Eq. (8).

$$\text{acceptance} = \frac{A_T}{C_T} \tag{8}$$

where A_T indicates the number of virtual network requests successfully mapped during the T period, and C_T indicates the total number of virtual network requests reached in the T period.

energy represents the energy consumption of the substrate network which consists of physical nodes and links, as defined in Eqs. (9) and (10), respectively.

$$P_j^i = \begin{cases} P_{\text{idle}} + \left(P_{\text{busy}} - P_{\text{idle}}\right)\cdot\phi, \\ \text{if } i \text{ virtual nodes sucessfully mapped} \\ \text{on the } j \text{ physical nodes} \\ 0, \text{otherwise} \end{cases} \tag{9}$$

$$P_{mn}^{uw} = \begin{cases} P_{\text{linkidle}}, \text{if um vitual link sucessfully mapped} \\ \text{on the mn physical link} \\ 0, \text{otherwise} \end{cases} \tag{10}$$

where P_{idle} is the basic power consumption of physical node, P_{busy} is the full-load power consumption of physical node, ϕ represents the CPU load rate of substrate node, and P_{linkidle} indicates the link energy consumption, which is generally constant. Therefore, energy consumption of substrate network can be computed as per Eq. (11).

$$\text{energy} = \sum\nolimits_{j \in N^S} P_j^i + \sum\nolimits_{m,n \in N^S} P_{mn}^{uw} \tag{11}$$

The proposed work considers two-phase mapping Algorithm involving mapping of nodes first. Therefore, only the node energy consumption is considered for using QCD VNE algorithm and updated as shown in Equation below:

$$\text{energy} = \sum\nolimits_{j \in N^S} P_j^i$$

where f(energy) in Eq. (1) represents data normalization processing which is logarithmic function conversion. The purpose is to eliminate the data difference caused by the dimensions of energy saving and VNR rate. It facilitates subsequent data processing and speeds up program convergence, as defined in Eq. (12).

$$f(\text{energy}) = \frac{\log_{10}(\text{energy})}{\log_{10}\left(\max_{\text{energy}}\right)} \tag{12}$$

where $\max_{\text{energy}}$ is the maximum value of substrate network energy consumption.

Equations (2) and (3) represent the capacity constraints, where ReqCPU(i) indicates the CPU resource requests for i virtual node, CPU(j) represents the total CPU resource for j physical node, ReqBWL(l_{uw}) is the BW resource requests for l_{uw} virtual link, and BWL(l_{mn}) denotes the total BW resource for l_{mn} physical link. Equation (4) restricts a virtual node to map to only one physical node. Equation (5) indicates that the same virtual node cannot be mapped to the same physical node, NN_o is the number of virtual nodes. Equations (6) and (7) represent variable constraints. If i virtual node is successfully mapped on j physical node, $f_j^i = 1$; otherwise, $f_j^i = 0$. Similarly, if l_{uw} virtual link is successfully mapped on l_{uw} physical link, $f_{mn}^{uw} = 1$; otherwise, $f_{mn}^{uw} = 0$.

3.2 Virtual network embedding as Markov decision process

We use the MDP to model the virtual network with an assumption that the arrival and departure of VNRs obey Poisson distribution. If a flow of events is a Poisson event flow, it needs to satisfy the stationary, non-post-effect, and generality [16]. For the virtual network mapping problem, the occurrence of events is independent of each other, and its probability depends only on the time interval, only one virtual request enters or leaves in a unit time. Therefore, the virtual network mapping problem can be modeled as a Markov decision process as a quadruplet: $\{S、A、P、R\}$, where S represents state set with the state S_ρ denoted as the final state. The value of reward in the final state is denoted as R_ρ. A is the action set, P is the state transition probability, and $R(s,a)$ is the reward of the action a at state S.

A MDP is called an episode from the initial state to the final state. A successful mapping process of each virtual network termed as one episode. In each episode, the agent starts execution from a randomly selected state until it reaches a final state. At the end of the episode, the agent is randomized to a new initial state and begins the next episode.

Assume that in a given state S_t, the agent selects one physical node $n^s \in N^{\psi_s}$ for mapping the virtual node $n^v \in N^{\psi_v}$ and then enters the next state S_{t+1}, where N^{ψ_s} is a set of all physical nodes that can carry virtual nodes n^v, and N^{ψ_v} is all non-mapped virtual nodes. The state of M at time t is defined as:

$$S_t = \left(\left(N_t^{\psi_v} = N_{t-1}^{\psi_v} \backslash \left\{n_{t-1}^v\right\}\right), \left(N_t^{\psi_s} = N_{t-1}^{\psi_s} \backslash \left\{n_{t-1}^s\right\}\right)\right)$$

where n_{t-1}^v is a physical node carrying the previous virtual node n_{t-1}^s. In the initial state, since there is no node that has been mapped, $N_1^{\psi_v} = N^{\psi_v}$, $N_1^{\psi_s} = N^{\psi_s}$.

The action of agent selection node $n_t^s \in \{N_t^{\psi_s} \cap N_t^{\psi_s}(n_t^v)\}$ is defined as:

$$A_t = \{\varepsilon\} \cup \left\{\left(n_t^v \_ n_t^s\right) : \forall n_t^s \in \left\{N_t^{\psi_s} \cap N_t^{\psi_s}\left(n_t^v\right)\right\}\right\}$$

where ε indicates an arbitrary action that can reach the terminal state. When the agent selects the physical node n_t^s for the current virtual node n_t^v, it transits to the next state S_{t+1}. Therefore, the probability of state transition that the agent selects action A_t transits to the next state S_{t+1} in state S_t is defined as:

$$P_r\left(S_{t+1} | n_t^v \_ n_t^s, S_t\right) = 1$$

where total reward consists of an external reward R_t^e and an internal signal R_t^i. The external reward R_t^e is calculated from the deterministic factor optimization model as shown in Eq. (13). The internal signal R_t^i is calculated by the curiosity-driven mechanism as described in Section 4.2.

$$R_t^e = \max(\omega \text{acceptance} - (1-\omega) f(\text{energy})) \quad (13)$$

$$= \begin{cases} R_\rho, & R_t < R_\rho \\ R_t, & \text{otherwise} \end{cases}$$

If the total reward calculated in the state S_t is smaller than it in the state R_ρ, then the virtual request reaches the final state, and its value will be replaced by R_ρ, otherwise it will remain unchanged.

4 Virtual network embedding based on reinforcement learning and curiosity-driven

In this section, we defines Q-learning and curiosity-driven for their application to VNE problem, and presents a description of the virtual network multi-objective embedding algorithm based on Q-learning and curiosity-driven (Q-CD-VNE).

4.1 VNE based on Q-learning algorithm

We employed Q-learning algorithm of reinforcement learning to solve the MDP process. It allows agents to automatically determine the ideal behavior within a specific environment for maximizing its performance. Simple reward feedback is required for the agent to learn its behavior. The agent's living environment in each episode is described as a state set S, which can perform any possible action described as action set A as depicted in Fig. 1 Each time an action a_t is performed in the state s_t, the agent receives a reward r_t, thus generating a series of states s_i, a set of actions a_i and reward r_i till the end of the episode. It finds the action sequence π iteratively for maximizing all reward values as an optimal strategy as shown in Eq. (14).

$$\pi^*(\mathrm{S}) = \arg\max_a [r(s,a) + \gamma v^*(\delta(s,a))] \quad (14)$$

For finding optimal strategies, the approximate behavioral value function Q is usually used. After successful mapping of a virtual node, the system deliverse a Q value to the agent. The Q value matrix is obtained by approximating the behavior value function $Q(s_t, a_t)$ gradually. This matrix can be used in the next episode to find the node with the highest reward value quickly for embedding,

where the Q value update strategy function is expressed as:

$$Q(s,a) = R(s,a) + \gamma \max \mathrm{Q}(\delta(s,a), a')$$

The estimate of Q is expressed as:

$$\hat{Q}(s,a) \leftarrow R(s,a) + \gamma \max_{a'} \hat{Q}(s', a') \quad (15)$$

The convergence of $\hat{Q}$ to Q has been proven [17]. Here, $R(s,a)$ is the reward value of the action a in the state s, $Q(\delta(s,a), a')$ is the Q value of all the next actions

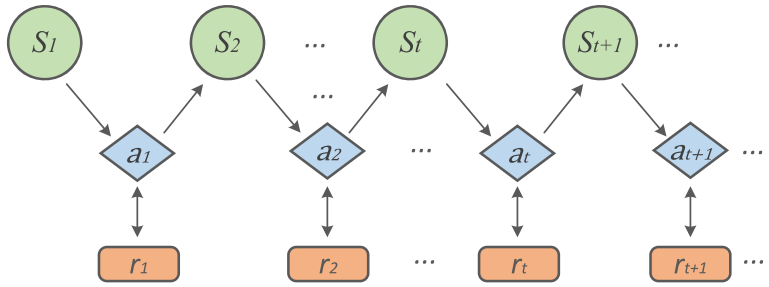

Fig. 1 Reinforcement learning model

a' after the action a is performed in the state s, and γ is the conversion factor which is used to weigh the relationship between the current reward value and the subsequent reward value.

At a time t, the Q value update process of the virtual network node mapping is shown in Fig. 2:

The solid lines with arrows in Fig. 2 represent the sequence of actions that have been mapped successfully, with the dashed arrows pointing to the selectable mapping nodes. This figure represents a network diagram for selecting the next action, that is, embedding the virtual node b when the virtual node a is successfully mapped to the physical node A. The detailed process is described below.

$$S_1 = \left(\left(N_t^a, N_t^b\right), \left(N_t^A, N_t^B, N_t^C, N_t^D\right)\right)$$

After the action $n_t^a n_t^A$ is performed in state S_1, the agent arrivals at state S_2.

$$S_2 = \left(\left(N_t^b\right), \left(N_t^B, N_t^C, N_t^D\right)\right)$$

At state S_2, the mappable action set $A_2\{n_t^b n_t^B, n_t^b n_t^C, n_t^b n_t^D\}$ satisfies the resource of VNRs. The Q matrix is initialized to an all-zero matrix. The Q values $\hat{Q}(S_2, n_t^b n_t^B)$, $\hat{Q}(S_2, n_t^b n_t^C)$, and $\hat{Q}(S_2, n_t^b n_t^D)$ are calculated according to the Q value update strategy formula (13) when the system selects the action which is as follows.

$$\begin{aligned}\hat{Q}(S_2, A_2) &\leftarrow R(S_2, A_2) + \gamma \max_{a'} \hat{Q}(s', a') \\ &= \max\left(R_{S_2}^e\right) + \gamma \max_{a'} \hat{Q}(s', a') \\ &= \max\left(n_t^b\_n_t^B, n_t^b\_n_t^C, n_t^b\_n_t^D\right) + \max(0, 0, 0) \\ &= 1 - \min(f(\text{energy}))\end{aligned}$$

For simplicity, we ignored the values of ω, γ, and internal signals, where $\max_{\text{energy}}$ is calculated as follows.

$$\max_{\text{energy}} = \max\left(P_{\text{idle}} + (\text{CPU}(i) - P_{\text{idle}})\frac{\text{ReqCPU}(i)}{\text{CPU}(i)}\right)$$

where $\text{CPU}(i) \in [a, b]$, and $\text{ReqCPU}(i) \in [c, d.]$

We draw this functional image in MATLAB and find the maximum value of this function. This function has an extreme in the interval for $\text{ReqCPU}(i) = c$ and $\text{CPU}(i) = b$. We can obtain the action with the largest reward value using above cited expressions. Following this action, the reward value is recorded in the Q matrix as the physical node evaluation standard of the mapping, and the next state S_3 is started. This

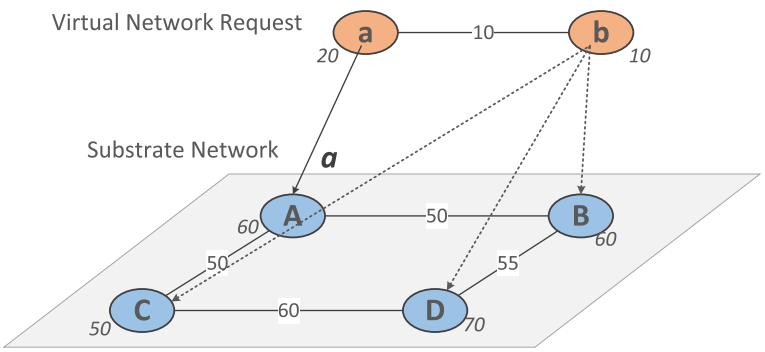

Fig. 2 The Q value update process of a virtual network embedding

is repeated until the Q-value matrix reaches the convergence state, and the system can select the next action A as per the current Q-value matrix by considering energy saving and VNR acceptance rates.

4.2 The reward calculation method based on curiosity-driven

In this section, we present a curiosity-driven mechanism for generating internal signals and passing them to external reward generators for mining other non-deterministic factors. This method can achieve a trade-off between exploration and exploitation and avoids falling into local optimum.

The traditional reinforcement learning method lacks the prediction of unknown environment for making decisions with the highest reward called exploitation-only. This causes a mismatch between the exploitation value and the exploration value and eventually leads to fall into a local optimum. For example, in the virtual network mapping process, the agent tends to repeatedly select nodes that perform well in the previous training process after much iteration. It may lead to deterioration of the substrate network connectivity and magnify the fragmentation rate of substrate network. Thus, it can be concluded that only considering the performance of deterministic factors can easily ignore the performance of non-deterministic factors. Therefore, it is suggested to consider prediction to the environment for selecting action a, so that the agent can adjust the mapping strategy before the formation of fragment in substrate network.

As shown in Fig. 3, the agent joins a curiosity-driven mechanism, by adding internal signals for external rewards and predicting the results of its own actions in a self-monitoring manner. In the iterative process, the internal curiosity mechanism will motivate agents to explore their own predictions of action by exploring the environment.

The curiosity-driven mechanism consists of external reward generator and internal signal generator as described in following paragraphs.

4.2.1 External reward generator

It is used to calculate rewards from deterministic factors, such as energy saving and VNR acceptance rate. It is called external reward r_t^e and calculated using Eq. (13).

4.2.2 Internal signal generator

Agent trains its proficiency by responding to the changes in the environment by predicting movements and training familiarity with the surrounding environment through predictions of the next environment, so that corresponding changes can be made before the environment changes. Then, they explore other factors that affect the performance of VNE to generate signals which guide the generation of total rewards, denoted as r_t^i. The calculation process is as described below.

4.2.3 Step 1

It predict the action $\hat{a}_t$ that causes the changes in states by training a neural network amounts to learning function $g(\cdot)$. The current state is denoted as $\varnothing(s_t)$, take as inputs $\varnothing(s_t)$ and $\varnothing(s_{t+1})$ and predict action $\hat{a}_t$ makes the agent from state s_t reach state s_{t+1} which defined as

$$\hat{a}_t = g(s_t, s_{t+1}; \theta_I)$$

where $\hat{a}_t$ is the predicted value of the action a_t, θ_I is trained by $\min_{\theta} L_I(\hat{a}_t, a_t)$, and L_I is the softmax loss

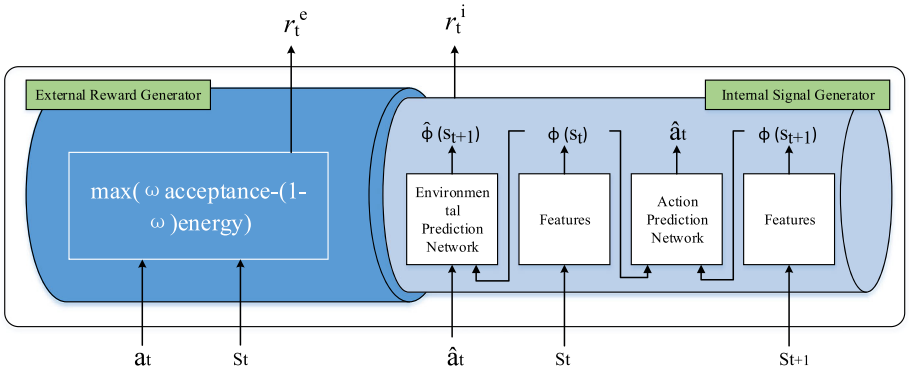

Fig. 3 Curiosity-driven mechanism

function measures the difference between predicted behavior and actual behavior.

4.2.4 Step 2

It predicts the next state $\hat{\varnothing}(s_{t+1})$ generated by the action $\hat{a}_t$ under the current environment $\varnothing(s_t)$by training another neural network amounts to learning function $f(\cdot)$.The state at $t+1$ is predicted by taking inputs as $\hat{a}_t$ and $\varnothing(s_t)$.

$$\hat{\varnothing}(s_{t+1}) = f(\varnothing(s_t), \hat{a}_t; \theta_F)$$

where $\hat{\varnothing}(s_{t+1})$ is the predictor of $\varnothing(s_t)$ and θ_F is minimized by the loss function L_F.

$$\min_{\theta_F} L_F(\varnothing(s_t), \hat{\varnothing}(s_{t+1})) = \frac{1}{2}\|\hat{\varnothing}(s_{t+1}) - \varnothing(s_{t+1})\|_2^2.$$

4.2.5 Step 3

It calculates the internal signal r_t^i by predicting the next state:

$$r_t^i = \begin{cases} 0, if\ \hat{\varnothing}(s_{t+1}) \text{ causes SN}'\text{s connectivity changes} \\ e^{-\frac{(\hat{\varnothing}(s_{t+1}) - \varnothing(s_{t+1}))^2}{(\varnothing(s_{t+1}) - \hat{\varnothing}(s_{t+1}))^2}}, \quad \text{otherwise} \end{cases}$$

This assignment method is reference [18]. After computing the values for internal signal and the external reward generated by the deterministic factors, it is superimposed as a new reward to guide the agent to make a decision. Therefore, r_t is defined as

$$r_t = r_t^i + r_t^e$$

In summary, the proposed method involves the agent to obtain the current VNE environment such as the situation of the substrate network resources, the link connection status, and the request volume of the virtual network through the MDP model. At the beginning of each episode, the first mapped virtual node is randomly transported to a physical node in the set of executable actions, then the curiosity-driven mechanism obtains the total reward value (composed of internal signals and external rewards), recorded in the Q matrix, and then moves to the next state s_{t+1}. Through this adaptive learning scheme, the potential impact factors can be mined by taking into account energy saving and VNR acceptance rates, for obtaining a global optimal mapping method.

4.3 Algorithm description

It can be concluded from the description cited process that Q-learning algorithm mixed with curiosity-driven mechanism can ensure the energy saving and VNR acceptance rate performance and avoid falling into local optimum. So, the proposed algorithm is described as below.

Algorithm 1 Multi-Objective Embedding Algorithm for Virtual Network Based on Q-learning and Curiosity-Driven

1: initialization： MDP model: state set S, action set A, transition matrix P. Initialization $\hat{Q}(S_t, A_t)$ matrix as an all zero matrix.
2: Establish a binary (0-1) integer programming.
3:if $t < \rho + 1$
4: $step = 0$
5: while $(step < \max\_step)$
6: if $a_t \neq \emptyset$ then
7: for each $a_t \in A$ do
8: $R_t = R_t^e \cdot R_t^i$
9: Calculate the $\hat{Q}$ value and include it in the $\hat{Q}$ matrix
10: end for
11: else reject
12: $s_t \leftarrow s_{t+1}$
13: $step = step + 1$
14: end while
15:else
16: if $t = \rho$
17: Using Shortest Path Algorithm to Solve Virtual Network Link Embedding
18: else
19: reject
20: end if
21:end if

5 Performance evaluation

This section describes the performance evaluation of the proposed method. It provides the details of the simulation environment, performance metrics, and experimental results followed by their discussion.

5.1 Simulation environment

The proposed method is implemented and executed on a PC having configuration as CPU: 3.4 GHz and 4G of memory. We used GT-ITM model and NS2 software to generate the substrate network and virtual network request topology [19]. The number of the

substrate network nodes pre-designed in this experiment is 100, and the probability of the nodes connecting to each other is assumed as 0.5. The distribution of the substrate network and virtual network resources is as follows.

The value distribution of the CPU resource value of the physical nodes is [50, 100] and is subject to uniform distribution. The value range of the bandwidth resources of physical links is [50,100] and is uniformly distributed. The value distribution range of the virtual node CPU resource request amount is [0, 14] and obeys the uniform distribution. The value range of the virtual link bandwidth resource requirement is [0, 34] and follows the uniform distribution. An average of 100 time units can reach 20 virtual network requests among all requests, and these requests obey the Poisson distribution. The number of requests to reach the virtual network in the experiment statistics is 2000, and the number of runtime units is about 14,000. The constant values for node and link energy consumption are set as follows: $P_l = 150$, $P_b = 150$, $P_n = 15$, $\omega = 0.5$, the number of iterations is 100.

5.2 VNE performance metrics

The comprehensive quality of VNE problem can be judged in terms of following metrics.

1. Average number of open nodes (ANON):

$$\mathrm{ANON} = \frac{\sum_{i=1}^{N_T} NO_i}{N_T}$$

where N_T represents the number of all valid time periods from 0 to T, NO_i represents the number of the physical nodes that are active in the effective period i.

2. Average number of open links (ANOL):

$$\mathrm{ANOL} = \frac{\sum_{i=1}^{N_T} LO_i}{N_T}$$

where LO_i indicates the number of the physical links that are active during the valid period i.

3. Average utilization of CPU (AUCPU):

$$\mathrm{AUCPU} = \frac{\sum_{i=1}^{N_T} NRU_i}{N_T}$$

where NRU_i indicates the CPU utilization of the node resources in the effective time unit i.

4. Average utilization of BW (AUBW):

$$\mathrm{AUBW} = \frac{\sum_{i=1}^{N_T} LRU_i}{N_T}$$

where LRU_i indicates the bandwidth utilization of the link resources in the effective time unit i.

5. Average amount of energy consumption (AAEC):

$$\mathrm{AAEC} = \frac{\sum_{i=1}^{N_T} E_i}{N_T}$$

where E_i is the consumption of the physical resources within the effective period i.

6. Average ratio of revenue and cost (ARRC):

$$\mathrm{ARRC} = \frac{\sum_{i=1}^{A_T} \mathrm{Revenue}_{R_i}}{\sum_{i=1}^{A_T} \mathrm{Cost}_{R_i}}$$

where A_T represents the number of VNRs accepted successfully from time 0 to time T. $\mathrm{Revenue}_{R_i}$ and Cost_{R_i} represent the revenue and cost of a successful virtual network request R_i, respectively.

7. Average acceptance ratio (AAR):

$$\mathrm{AAR} = \frac{A_T}{C_T}$$

where C_T represents the total number of VNRs in the time period from 0 to T.

8. Substrate network fragmentation (SNF):

$$\mathrm{SNF}(t) = 1 - \frac{\sum_{i=1}^{m} \left(\mathrm{Residual}\left(\left(N^S, L^S\right), t\right)\right)^q}{\left(\sum_{i=1}^{m} \mathrm{Residual}\left(\left(N^S, L^S\right), t\right)\right)^q}$$

where m represents the number of fragments in the substrate network, $\mathrm{Residual}((N^S, L^S), t)$ is the substrate residual resources, and q is a positive integer number greater than 1 to reduce the influence of the small negligible fragments as long as one large fragment exits [20].

9. Node load balance (NLB):

$$\mathrm{NLB} = \sqrt{\sum_{n^s \in N_S} \left(\mathrm{CPU}(n^s) - \frac{\sum \mathrm{CPU}\,(n^s)}{|N_S|}\right)^2}$$

5.3 Comparative analysis of experimental results

The proposed method is executed in a simulation environment as described above, and reported results are compared with the representative researches in the field in terms of identified performance metrics. In order to compare the performance of the proposed method, we choose the EAVNE algorithm [21] that aims to save energy as well as our algorithm, the

GSOVNE algorithm [22], because both algorithms use intelligent algorithms to embed virtual requests and the classic greedy algorithm SP [23]. These three algorithms have similarities with our algorithm, so they are used for comparison experiments. Ensure that our algorithmic experimental results are more credible.

Figures 3, 4, 5, 6, and 7 show the comparative results of the proposed method and representative methods in terms of average number of open nodes, average number of open links, average node resource utilization, average link resource utilization, and average amount of energy consumption.

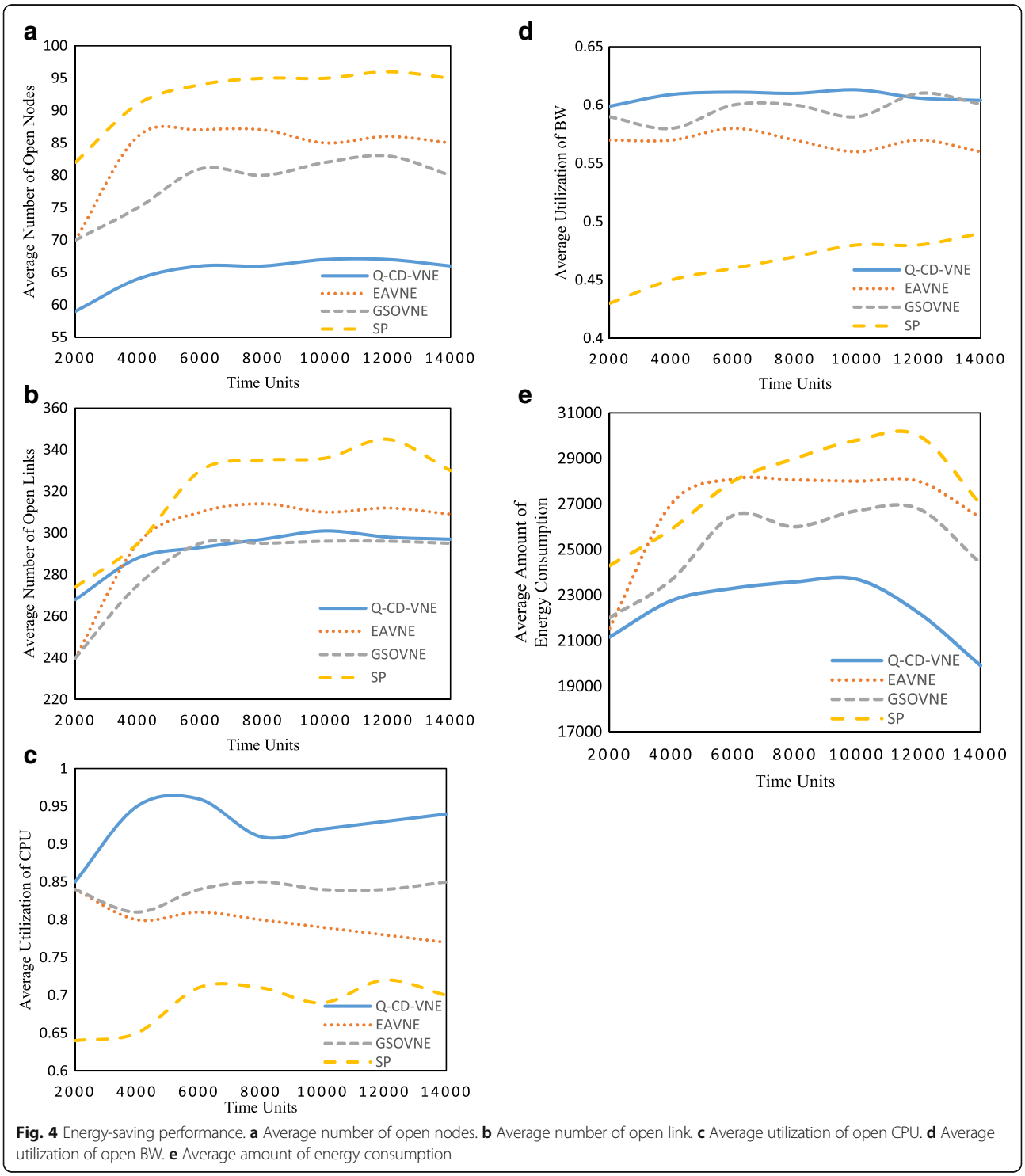

Fig. 4 Energy-saving performance. **a** Average number of open nodes. **b** Average number of open link. **c** Average utilization of open CPU. **d** Average utilization of open BW. **e** Average amount of energy consumption

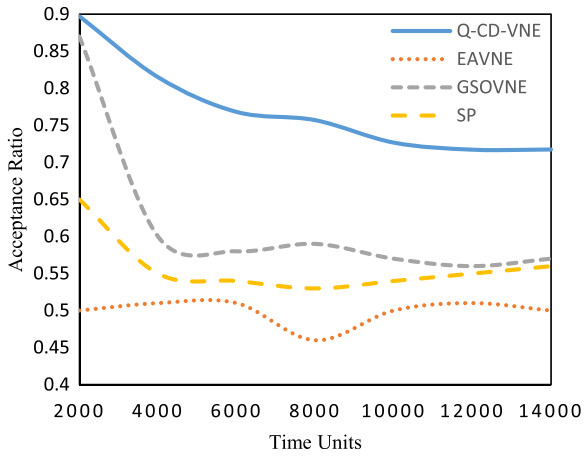

Fig. 5 Acceptance ratio

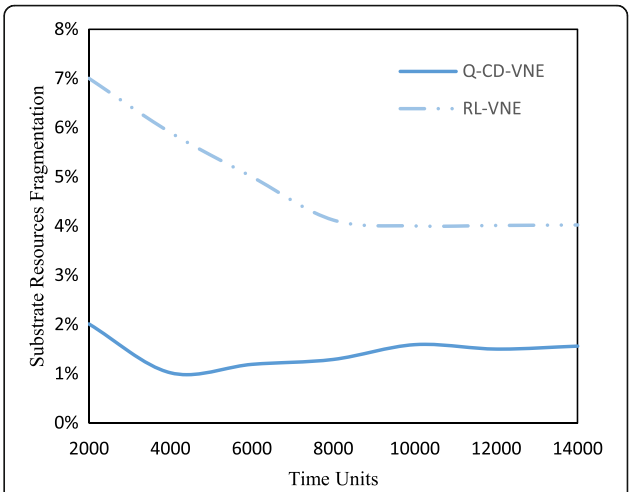

Fig. 7 Substrate resources fragmentation

It can be observed from Fig. 4a that ANON of the Q-CD-VNE algorithm is lower than the EAVNE, GSOVNE, and SP algorithms by 18, 13, and 32, respectively. It can be seen that it is significantly lower than that of the SP algorithm. This is because the SP algorithm does not consider energy saving during the mapping process. As depicted in Fig. 4b, ANOL of the Q-CD-VNE algorithm is 6 and 29 lower than the EAVNE algorithm and the SP algorithm, respectively, and about 5 more links than the GSOVNE algorithm. The performance of this algorithm is almost the same as that of EAVNE algorithm and the GSOVNE algorithm in terms of link opening amount. This is due to that these algorithms are all two-phase mapping algorithms, and the link-based processing uses the greedy-based shortest path algorithm, so no significant difference has been observed. It can be concluded from Fig. 4c that the AUCPU of the Q-CD-VNE algorithm is increased by 12.4, 8.3, and 23.3%, respectively, in comparison to the EAVNE algorithm, the GSOVNE algorithm, and the SP algorithm. Figure 4d shows that the AUBW of the Q-CD-VNE algorithm is 3.8, 0.1, and 13.8% higher than the EAVNE, GSOVNE, and SP algorithms, respectively. The improvement of the AUCPU is more significant than AUBW. It happens because the algorithm involves a two-phase mapping, and no more optimized algorithm is used for mapping the links. Figure 4e shows the performance of algorithms in terms of AAEC because the Q-CD-VNE algorithm has a good energy-saving effect and the energy consumption of the proposed algorithm is the lowest among the four algorithms in comparison. Compared with the EAVNE algorithm, the GSOVNE algorithm, and the SP algorithm, they have reduced 4335.54, 2276.78, and 5338.83 W, respectively.

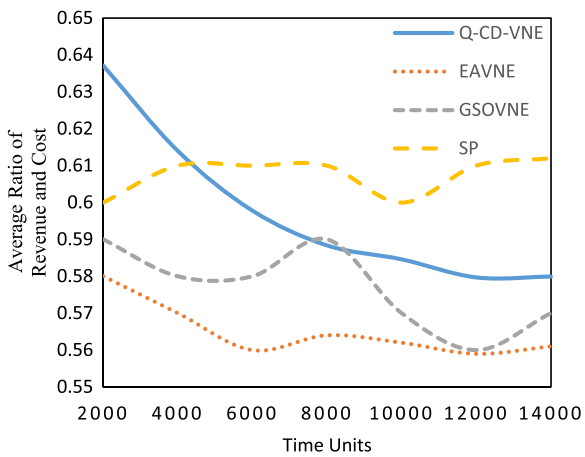

Fig. 6 Average ratio of revenue and cost

Figure 5 shows that the VNR acceptance rate of the Q-CD-VNE algorithm as 27.27, 15.12, and 21.13% higher than that of EAVNE, GSOVNE, and SP algorithms, respectively. This is because the algorithm regards energy saving and VNR acceptance rate as external rewards and considers both performances in the mapping process. Therefore, both the energy saving and the VNR acceptance rate have been improved.

Figure 6 shows that the average ratio of revenue and cost of Q-CD-VNE algorithm is slightly higher than that of the comparison energy-saving algorithm EAVNE and the intelligent algorithm GSOVNE, and almost the same value as that of SP algorithm. It shows that the performance of the algorithm has not decreased while considering other factors.

Figures 7 and 8 show the substrate fragmentation rate and node load rate performance after adding a curiosity-driven mechanism. Here, Q-VNE is an algorithm with no curiosity mechanism. Figure 7 shows that the substrate fragmentation rate is reduced by 3–5 percentage points in comparison to the algorithm without the curiosity mechanism. Figure 8 shows that the load

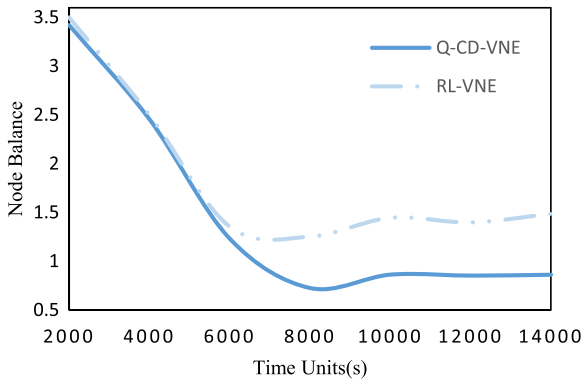

Fig. 8 Node balance

imbalance is alleviated after the program runs for about 5000 time units. This is due to the fact that after the agent has been involved in the curiosity mechanism for a period of time, when the node is fully loaded, we feed back this information to the agent, prompting agent not to select actions which degrade the SN's connectivity, thus avoiding the increase of node full load rate. This will inevitably reduce the fragmentation rate and ease the imbalance of the load.

6 Conclusions

In this paper, a virtual network embedding scheme is proposed on the basis of Q-learning algorithm, MDP, and curiosity-driven technology. The scheme addressed the multi-objective trade-off problem in VNE. The experimental results show that the algorithm can find a good trade-off between conflicting objectives. The comparative results prove that the proposed method can improve the performance of the system in terms of conflicting objectives by reducing energy consumption, improving request acceptance rate, and improving the long-term average income.

In the future, we will consider making the fragmentation rate and the load rate as deterministic factors. Meanwhile, we will consider improving the curiosity mechanism in order to explore more non-deterministic factors that can be used as optimization goals and incorporate them into deterministic factors.

Abbreviations

AAEC: Average amount of energy consumption; AAR: Average acceptance ratio; ANOL: Average number of open links; ANON: Average number of open nodes; ARRC: Average ratio of revenue and cost; AUBW: Average utilization of BW; AUCPU: Average utilization of CPU; EEVNE: Energy-aware virtual network embedding; MDP: Markov decision process; NLB: Node load balance; SN: Substrate network; SNF: Substrate network fragmentation; VNE: Virtual network embedding; VNRs: Virtual network requests

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (no. 61379079), The Science and Technology Key Project of Henan Province (no. 172102210478), and The International Cooperation Program of Henan Province (no. 152102410021). The Key Scientific Research Project of Higher Education of Henan (no.17A520057).

Authors' contributions

MH is in charge of the major theoretical analysis, algorithm design, numerical simulations, draft of the article, and critical revisions. ST is in charge of part of the algorithm design and experimental design. LZ, GW, and KZ provided critical revision and final approval of the version to be published. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 24 April 2018 Accepted: 30 May 2018
Published online: 15 June 2018

References

1. A Fischer, JF Botero, BM Till, et al., Virtual network embedding: a survey. IEEE Commun. Surv. Tutorials **15**(4), 1888–1906 (2013)
2. XW Zheng, B Hu, DJ Lu, et al., A multi-objective virtual network embedding algorithm in cloud computing. J. Internet Technol. **17**(4), 633–642 (2016)
3. Z Zhang, S Su, Y Lin, et al., Adaptive multi-objective artificial immune system based virtual network embedding. J. Netw. Comput. Appl. **53**(3), 140–155 (2015)
4. P Zhang, H Yao, C Fang, et al., Multi-objective enhanced particle swarm optimization in virtual network embedding. EURASIP J. Wirel. Commun. Netw. **2016**(1), 167–175 (2016)
5. AA Shahin, Memetic multi-objective particle swarm optimization-based energy-aware virtual network embedding. Int. J. Adv. Comput. Sci. Appl. **6**(4), 35–46 (2015)
6. I Houidi, W Louati, D Zeghlache, Exact multi-objective virtual network embedding in cloud environments. Comput. J. **58**(3), 403–415 (2015)
7. JJ Yu, CM Wu, et al., Design and analysis of virtual network mapping competitive algorithms. Comput. Sci. **42**(2), 33–38 (2015)
8. N Triki, N Kara, ME Barachi, et al., A green energy-aware hybrid virtual network embedding approach. Comput. Netw. Int. J. Comput. Telecommun. Netw. **91**(C), 712–737 (2015)
9. XJ Guan, BY Choi, S Song, Energy efficient virtual network embedding for green data centers using data center topology and future migration. Comput. Commun. **69**(C), 50–59 (2015)
10. XH Chen, CZ Li, LY Chen, et al., Energy efficient virtual network embedding based on actively hibernating substrate nodes and links. J. Softw. **25**(7), 1416–1431 (2014)
11. TG Karimpanal, E Wilhelm, Identification and off-policy learning of multiple objectives using adaptive clustering. Neurocomputing **263**(8), 39–47 (2017)
12. KG Vamvoudakis, Q-learning for continuous-time graphical games on large networks with completely unknown linear system dynamics. Int. J. Robust Nonlinear Control **27**(16), 2900–2920 (2017)
13. T Hester, P Stone, Intrinsically motivated model learning for developing curious robots. Artif. Intell. **247**, 170–186 (2017)
14. D Pathak, P Agrawal, AA Efros, T Darrell, in *ICML*. Curiosity-driven exploration by self-supervised prediction (2017)
15. VR Kompella, M Stollenga, M Luciw, et al., Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. Artif. Intell. **247**, 313–335 (2017)
16. L Xia, Mean-variance optimization of discrete time discounted Markov decision processes. Automatica **88**, 76–82 (2018)
17. C Watkins, P Dayan, Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
18. C Bu, XW Wang, M Huang, Adaptive routing service composition: modeling and optimization. J. Softw. **28**(9), 2481–2501 (2017)
19. E Zegura, K Calvert, S Bhattacharjee, in *INFOCOM '96. Conference on Computer Communications*. How to model an Internet work (IEEE, San Francisco, 1996), pp. 594–602
20. J Gehr, J Schneider, in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID 09*. Measuring fragmentation of two-dimensional resources applied to advance reservation grid scheduling (2009), pp. 276–283

21. JF Botero, X Hesselbach, M Duelli, et al., Energy efficient virtual network embedding. IEEE Commun. Lett. **16**(5), 756–759 (2012)
22. ZH Chen, XW Zheng, DJ Lu, *A Study of a Virtual Network Embedding Algorithm. IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing* (IEEE, Las Vegas, 2015), pp. 814–818
23. M Yu, Y Yi, J Rexford, et al., Rethinking virtual network embedding: substrate support for path splitting and migration. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 19–29 (2008)