Neurocomputing 284 (2018) 1–9

Contents lists available at ScienceDirect

Neurocomputing

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

A novel reinforcement learning algorithm for virtual network embedding

Haipeng Yao[a,*], Xu Chen[a], Maozhen Li[b], Peiying Zhang[a], Luyao Wang[c]

[a] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecom, Beijing, P.R. China
[b] Department of Electronic and Computer Engineering, Brunel University London, Uxbridge UB8 3PH, UK
[c] Beijing Advanced Innovation Center for Future Internet Technology, Beijing University of Technology, Beijing, P.R. China

A R T I C L E I N F O

Article history:
Received 1 November 2017
Revised 6 December 2017
Accepted 6 January 2018

Communicated by Dr. Ma Lifeng Ma

Keywords:
Virtual network embedding
Reinforcement learning
Policy network
Policy gradient

A B S T R A C T

Network virtualization enables the share of a physical network among multiple virtual networks. Virtual network embedding determines the effectiveness of utilization of network resources. Traditional heuristic mapping algorithms follow static procedures, thus cannot be optimized automatically, leading to sub-optimal ranking and embedding decisions. To solve this problem, we introduce a reinforcement learning method to virtual network embedding. In this paper, we design and implement a policy network based on reinforcement learning to make node mapping decisions. We use policy gradient to achieve optimization automatically by training the policy network with the historical data based on virtual network requests. To the best of our knowledge, this work is the first to utilize historical requests data to optimize network embedding automatically. The performance of the proposed embedding algorithm is evaluated in comparison with two other algorithms which use artificial rules based on node ranking. Simulation results show that our reinforcement learning is able to learn from historical requests and outperforms the other two embedding algorithms.

1. Introduction

The combination of network virtualization and software defined networks is considered as the foundation towards the next generation of Internet architecture [1,2]. Network virtualization enables the coexistence of multiple heterogeneous virtual networks on a shared network [3–6]. For Internet Service Providers (ISPs), it enables new business models of hosting multiple concurrent network services on their infrastructures. Decisions for embedding are challenging problems for ISPs since it determines the effectiveness of utilization of network resources. A sub-optimal embedding algorithm will decrease the overall capacity of the infrastructure and lead to cost of revenue for ISPs. A virtual network consists of several virtual nodes (e.g. virtual routers), connected by a set of virtual links. The purpose of virtual network embedding is to map virtual networks to a shared physical network while providing the requests with adequate computing and bandwidth resources.

However, the virtual network embedding problem has been proved to be NP-hard [7]. As a result, a large number of heuristic algorithms have been proposed [8–11], but most of them rely on artificial rules to rank nodes or make mapping decisions. The parameters in these algorithms are always fixed and cannot be optimized, making the embedding decisions sub-optimally. On the other hand, in prior works, the information about substrate network and the knowledge about virtual network embedding hidden in historical network request data have always been overlooked. Historical network requests are a good representation of temporal distribution and resource demands in the future.

In recent years, big data, machine learning and artificial intelligence have exciting breakthroughs achieving state of the art results such as natural language understanding and object detection. Machine learning algorithms process a large amount of data collected during a period and automatically learn the statistical information from the data to give classification or prediction. Reinforcement learning, as a widely-used technique in machine learning, has shown a great potential in dealing with complex tasks, e.g., game of go [12], or complicated control tasks such as auto-driving and video games [13]. The goal of a reinforcement learning system (or an agent) is to learn better policies for sequential decision making problems with an optimal cumulative future reward signal [14].

* Corresponding author.
E-mail address: yaohaipeng@bupt.edu.cn (H. Yao).

https://doi.org/10.1016/j.neucom.2018.01.025
0925-2312/© 2018 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license.
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

In this paper, we introduce reinforcement learning into the problem of virtual network embedding to optimize the node mapping process. Similar to earlier works [8,9], our work is based on the assumption that all network requests follow an invariable distribution. We divide our network request data into a training set and a testing set, to train our reinforcement learning agent (RLA) and evaluate its performance respectively. We devise an artificial neural network called policy network as the RLA, which observes the status of substrate network and outputs node mapping results. We train the policy network with historical network request data using policy gradient through back propagation. An exploration strategy is applied in the training stage to find better solutions, and a greedy strategy is applied in evaluation to fully evaluate the effectiveness of the RLA. Extensive simulations show that the RLA is able to extract knowledge from historical data and generalize it to incoming requests. To the best of our knowledge, this work is the first to utilize historical network requests data and policy network based reinforcement learning to optimize virtual network embedding automatically. The RLA outperforms two representative embedding algorithms based on node ranking in terms of long-term average revenue and acceptance ratio, while making a better utilization of network resources.

The rest of this paper is organized as follows. Section 2 assesses a number of related works. Section 3 introduces the virtual network embedding problem and the proposed network model. Section 4 presents the design and implementation of the reinforcement learning agent together with its training and testing process. Section 5 evaluates the performance of the RLA, and Section 6 concludes the paper.

2. Related work

Virtual network embedding involves two stages—node mapping and link mapping. Some works, e.g., [10,11], solve the problem using a one-stage approach and assign virtual nodes and links coordinately using linear programming or mixed integer programming (MIP). For example, a rounding-based approach is applied in R-ViNE and D-ViNE algorithms [10] to achieve a linear programming relaxation of the MIP. However, these methods demand certain additional constraints such as location requirements to extend the network topology to an augmented graph so that the computing space can be greatly reduced. In other works [8,9,15], node mapping and link mapping are solved independently. First, the substrate nodes are ranked based on their availability measured with certain rules. Then, a greedy node mapping strategy is applied where the priority of mapping is decided by rank results. Substrate nodes with more available resources will be considered first in the node mapping stage. Finally, the virtual links are mapped to the shortest path that has enough bandwidth resources between fixed nodes. Yu et al. [8] focus on path splitting and migration in link mapping problem, which means a virtual link may be mapped to several substrate links and existing link mapping may change according to the condition of substrate network. However, those approaches require the support of the substrate network and might not be available. Inspired by PageRank [16] that ranks the relative importance of Web pages, the authors of [9] proposed an algorithm based on Markov random walk to solve the problem of node ranking and mapping. The availability of each substrate node as well as its neighbors is considered in node ranking. However, the node ranking methods mentioned above follow invariable procedures. Thus no automatic optimization can be performed, which leads to sub-optimal ranking and embedding results. Furthermore, the updating process of node ranking takes a rather long time to run and may not converge. The aforementioned virtual network embedding algorithms are carried out in a centralized manner, which means that a centralized controller is responsible for gathering information about substrate network and making mapping decisions. In [17], a multi-agent based approach and a distributed protocol are proposed to ensure distributed negotiation and synchronization between substrate nodes. In addition, many works [18,19] also consider the energy efficiency of VNE.

Haeri and Trajkovic [20] combined reinforcement learning and virtual network embedding. But different from our work, they employ the Markov decision process to solve the node mapping problem and use Monte-Carlo tree search (MCTS) as action policies. As a result, the MCTS has to be applied every time when a virtual request arrives, which requires a great amount of computing power and makes it less time-efficient. The works presented in [21,22] also employ reinforcement learning. However, they focus on the dynamic resource management among virtual networks. Mijumbi et al. [21] applied a q-learning based reinforcement learning agent to build a decentralized resource management system, which takes the role to increase or decrease the resources allocated to a certain virtual network. Mijumbi et al. [23] employed an artificial network to make resources reallocation decisions and train the network with a q-table from reference [21]. In [22], a reinforcement learning based neuro-fuzzy algorithm is proposed. The aforementioned works apply machine learning and reinforcement learning approaches to achieving dynamic resource management among virtual networks which are already embedded in the substrate network. Our work differs from these works in two ways. One is that we utilize a policy network based reinforcement learning method and apply policy gradient to train the policy network. Another is that we aim to improve the efficiency of the virtual network embedding process instead of dynamic resource management among virtual networks after embedding.

Table 1
Frequently used notations.

G^S	Substrate network
N^S	Nodes of substrate network
L^S	Links of substrate network
A_N^S	Node attribute of substrate network
A_L^S	Link attribute of substrate network
G^V	Virtual network of a certain virtual request
N^V	Nodes of a virtual network
L^V	Links of a virtual network
A_N^V	Constraints of substrate nodes
A_L^V	Constraints of substrate nodes

3. Network modeling

In this section, we present a network model and formulate the virtual network embedding problem with description of its components. The notations used in this section are shown in Table 1.

Fig. 1 shows the mapping process of two different virtual network requests. A substrate network is represented as an undirected graph $G^S = (N^S, L^S, A_N^S, A_L^S)$, where N^S denotes the set of all the substrate nodes, L^S denotes the set of all the substrate links, A_N^S and A_L^S stand for the attributes of substrate nodes and links respectively. In consistency with earlier works [8,9], in this paper we consider computing capability as node attribute and bandwidth capacity as link attribute. Let P^S denote the set of all the loop-free paths in substrate network. Fig. 1(c) shows an example of a substrate network, where a circle denotes a substrate node, and a line connecting two circles denotes a substrate link. The number in a square box denotes the CPU (computing) capacity of that node, and the number next to a substrate link denotes the bandwidth of that link. Similarly, we also use an undirected graph $G^V = (N^V, L^V, C_N^V, C_L^V)$ to describe a virtual network request, where N^V denotes the set of all the virtual nodes in the request, L^V denotes the set of all the virtual links in the request, C_N^V and C_L^V stand