A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks

Original

Availability:
This version is available at: 11583/2870934 since: 2021-07-26T10:16:18Z

Publisher:
Association for Computing Machinery (ACM)

Published
DOI:10.1145/3386367.3431670

Terms of use:
This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)
Poster: A Distributed Reinforcement Learning Approach for Energy and Congestion-Aware Edge Networks

Alessio Sacco  
aleSSio_sacco@polito.it  
Politecnico di Torino

Flavio Esposito  
flavio.esposito@slu.edu  
Saint Louis University

Guido Marchetto  
guido.marchetto@polito.it  
Politecnico di Torino

ABSTRACT
The abiding attempt of automation has also pervaded computer networks, with the ability to measure, analyze, and control themselves in an automated manner, by reacting to changes in the environment (e.g., demand) while exploiting existing flexibilities. When provided with these features, networks are often referred to as "self-driving". Network virtualization and machine learning are the drivers. In this regard, the provision and orchestration of physical or virtual resources are crucial for both Quality of Service guarantees and cost management in the edge/cloud computing ecosystem. Auto-scaling mechanisms are hence essential to effectively manage the lifecycle of network resources. In this poster, we propose Relevant, a distributed reinforcement learning approach to enable distributed automation for network orchestrators. Our solution aims at solving the congestion control problem within Software-Defined Network infrastructures, while being mindful of the energy consumption, helping resources to scale up and down as traffic demands fluctuate and energy optimization opportunities arise.

CCS CONCEPTS
* Networks → Network algorithms;  * Computer systems organization → Redundancy;  * Computing methodologies → Reinforcement learning.

KEYWORDS
reinforcement learning, self-driving networks, auto-scaling

ACM Reference Format:

1 INTRODUCTION
Recent advantages in artificial intelligence (AI) and machine learning (ML) are paving the path to autonomous and self-driving networks: networks that measure, analyze and control themselves in an automated manner, reacting to changes in the environment.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.  
CoNEXT ’20, December 1–4, 2020, Barcelona, Spain  
© 2020 Association for Computing Machinery.  
ACM ISBN 978-1-4503-7948-9/20/12 $15.00  
https://doi.org/10.1145/3386367.3431670

![Figure 1: System overview. The (logical) network controller receives in input traffic statistics and outputs new flow routes and power on/off commands.](image)
With the network partition, the controller supervises a subset of switches and communicates with the other controllers to obtain a consistent network view.

The decision logic is the fruit of a (self-)learning process, built upon the reinforcement learning (RL) framework. In our solution, each RL agent uses the one-step Q-learning algorithm [9]. Figure 1 shows the components and functionalities involved during the process. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The agent collects the statistics and combines them with previous historical data. Data are then passed to the optimizer module, which enforces the RL process and outputs the best possible strategy. The RL process. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The agent collects the statistics and combines them with previous historical data. Data are then passed to the optimizer module, which enforces the RL process and outputs the best possible strategy. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The decision logic is the fruit of a (self-)learning process, built upon the reinforcement learning (RL) framework. In our solution, each RL agent uses the one-step Q-learning algorithm [9]. Figure 1 shows the components and functionalities involved during the process. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The agent collects the statistics and combines them with previous historical data. Data are then passed to the optimizer module, which enforces the RL process and outputs the best possible strategy. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The decision logic is the fruit of a (self-)learning process, built upon the reinforcement learning (RL) framework. In our solution, each RL agent uses the one-step Q-learning algorithm [9]. Figure 1 shows the components and functionalities involved during the process. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.

The agent collects the statistics and combines them with previous historical data. Data are then passed to the optimizer module, which enforces the RL process and outputs the best possible strategy. The agent interacts with the underlying network and performs actions, where each action is associated with a reward. The reward function mimics the objective of reducing the energy cost and catering to the application requirements. We model this in the reinforcement learning problem, whose actual goal is to maximize the long-term discounted reward per action.