Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 193 (2021) 228–238

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Young Scientists Conference on Computational Science

Multi-Agent Deep Reinforcement Learning-Based Algorithm For Fast Generalization On Routing Problems

Ibraheem Barbahan[a,*], Vladimir Baikalov[a], Valeriy Vyatkin[b], Andrey Filchenkov[a]

[a] *ITMO University, 197101, 49 Kronverksky pr., St Petersburg, Russia*
[b] *Aalto University, 02150 Espoo, Finland*

Abstract

We propose a fast generalization method for DQN-Routing, an algorithm based on multi-agent deep reinforcement learning that suffers from generalization problem when introduced to new topologies even if it was trained on a similar topology. The proposed method is based on the wisdom of crowds and allows the distributed routing algorithm, DQN-Routing, to generalize better to new topologies that were not seen before during training. The proposed method also aims to decrease the solution search time as the original DQN-Routing algorithm takes a long time to converge, and to increase the overall performance by minimizing the mean delivery time and total power consumption and the number of collisions. The experimental evaluation of our method proved that is capable to generalize to new topologies and outperform the DQN-Routing algorithm.

Keywords: DQN-Routing ; Distributed routing problems ; Fast generalization ; Multi-agent deep reinforcement learning; Routing problem ; DQN

1. Introduction

The interest in developing methods and algorithms for routing problems has been increasing along with the emergence of the routing problem, especially in the logistics domain. The class of routing problem is huge as constraints and conditions related to the nature of the problem or to the environment in which the problem needs to be solved must be taken in account. This leads to high variety of problem statements as well as the high number of different algorithms applicable only to certain conditions.

The distributed approach for solving routing problems is more relevant to real world problems especially when the layout grows bigger, and the problem gets more complicated due to more constraints and conditions that affect the routing decision making process [1]. It becomes more important to include more elements in this process and

[*] Corresponding author. Tel.: +7 931 398-95-23
E-mail address: ibraheembarbahan@hotmail.com

collect as many detailed information possible about the layout at a specific point in time to be able to find the most accurate routing decision. Many algorithms were developed and standardized to solve the distributed routing problems especially in the computer network domain [2, 3]. However as more complicated tasks appeared in the logistics domain, some standard routing algorithms became not the best choice to find an optimal solution to the routing problem.

And as these problems become more computationally expensive, machine learning techniques were often seen as a good candidate to tackle these problems [4, 5, 6] due to replacing heavy computation by a fast approximation and efficient exploration of the search space and to learn new routing policies. Among machine learning-based approaches, **multi-agent reinforcement learning** is considered to have a great potential solving the distributed routing problem due to the distributed structure of the agents and the decision-making process based on interactions with the environment [7, 8, 9].

The DQN-Routing Algorithm [10] is a distributed algorithm that views the routing problem as a reinforcement learning problem.

It is based on deep multi-agent reinforcement learning networks combined with link-state protocol.

It runs on a network built from interconnected routers and treats the routers as learning agents, hence representing the distributed routing problem as multi-agent reinforcement learning problem. Each router is modeled as a deep neural network which allows each router to calculate and consider heterogeneous data about its environment, and that allows a better optimization of a more complex cost function.

The algorithm was shown to be adaptive for small changes in topology and the load in network. This is **local adaptiveness**. However, the algorithm does not adaptive to big changes that may occur because of network reconstruction or major shift in behavior of objects being routed. In other words, it does not show **global adaptiveness**, so the algorithm should be entirely retrained to adjust for a new environment, which takes time. This drawback is essential since natural topologies are a subject of change with time. A road network has new roads build and some old roads closed temporary for reparation. Such changes can be very frequent. Therefore, it is important to speed up the process of learning the algorithm in new environment.

The goal of our work is to **develop a new deep multi-agent reinforcement learning-based approach that can be easily adopted to work on new topologies it has never seen before**. To do so, we treat each graph as an instance, so we want our algorithm to **generalize** its behavior on new unseen instances. The ability to generalize to a new data after convergence on the training data is one of the most important challenges that faces machine learning algorithms. Generalization is a measure of how the model performs on predicting unseen data. So, it is important to come up with the best-generalized model to give better performance against future data, a lot of practices on the data processing and models building are already solving the problem of generalization.

The rest of the paper is organized as follows. In Section 2 we overview related work. In Section 3 we analyze the reasons why DQN-routing does not show the global adaptiveness and present ... In Section 4 we describe experimental setup and three scenarios we used to test DNQ-Routing and FastGe and compare them. The results of this experiments and their discussion is presented in Section 5. Section 6 summarizes the paper and outlines the future work.

2. Related Work

2.1. DQN-Routing

The DQN-Routing algorithm takes as an input one topology layout representing the conveyor network, specifying the sources and the sinks and length of the conveyor along with diverters that creates new conveyors. With the topology there is another part of the input which is a group of settings for the scenario on that topology, the settings file contains information about the bags that will travel on the topology, the number of bags is specified as group of bags, each group can travel from source to one or more sink, with specific delay that can accrue. Along with setting related to the scenario, the setting file contains other information and values about the DQN-Routing algorithm itself, such as conveyor environment parameters like speed and energy consumption, some parameters related to the router reward weight and learning rate, the optimizer and the activation function for the neural networks, batch size and embedding size and embedding type and others, moving forward we will denote the two files for input as X the input state of the DQN-Routing algorithm where $X = x_1, x_2, ..., x_n$.

Based on the input graph, a graph embedding is calculated and then the pre-training process that produce a pre-trained model for that specific input, we will denote the pre-training function as $f(x)$. The pre-trained model is then used to train the DQN-Routing algorithm on the scenario specified in the setting file, we will denote the training function as $g(f(x))$. For each run, the DQN-Routing algorithm calculates the number of collisions and outputs two graphs, one is the mean delivery time and the other is the mean power consumption at each unit of simulation time.

The DQN-Routing algorithm converges on one topology and its setting, and as mentioned before it outperforms the state-of-the-art routing algorithms. However, the DQN-Routing lacks the capability of generalization to a new topology. This means for any new input, the DQN-Routing algorithm should run again from scratch to give its good result. And for small changes in the topology layout or the scenario setting of an already trained input, the DQN-Routing algorithm will only run again the training process, but it will not give the same good results as if it would when run from scratch on the new input changes, the two ways of running on anew topology is shown in Eq. 1 where $t_i = f(x_i)$.

$$y_i = \left\{ \begin{array}{ll} g(t_i) & \text{when fully trained.} \\ g(t_j)\ \ j \neq i & \text{when generalized from trained model.} \end{array} \right\} \tag{1}$$

This problem of generalization holds back the DQN-Routing from being applied in new applications of the logistics domain as the whole training process for every new input is not time efficient because the solution search time is already long and if not trained from scratch the DQN-Routing will stop giving the state-of-the art result.

2.2. Generalization problem

In some fields of machine learning such as deep reinforcement learning, generalization remains a challenging problem for such algorithms, which are often trained and tested on the same set of deterministic problem environments. When test environments are unseen and perturbed, but the nature of the task remains the same. Solving the generalization problem in a deep reinforcement algorithm largely depends on the nature of the tasks and how similar they are, the policy and the training process, the environmental conditions and that makes the task of solving generalization problem almost unique for each algorithm and the DQN-Routing is no exception.

The problem of generalization in routing tasks using reinforcement learning was tackled recently in specific use cases where a deep reinforcement learning algorithm that operates on networks represented as graphs fails to generalize to new graphs not seen during training. In SDN-based Optical Transport Network (OTN) routing problem which is solved by deep reinforcement learning method proposed by [11], it was proposed that the reason behind the generalization failure is the fact that the deep reinforcement learning methods for routing, use a traditional neural network architecture such as fully connected networks or Convolutional Neural Networks CNN which are not suitable to model graph-structured information. Instead, it was proposed to use Graph Neural Network to operate over graphs and learn relational reasoning and finds relation between graph elements and how it was composed. graph neural network based on Message-passing Neural Networks was used to capture useful information about the links in the graph and the relations between them and the data passed through them.

This approach named DQN+GNN was a success in SDN-based Optical Transport Network, however this cannot be used in the DQN-Routing case due to many reasons. The most important of them is that the reinforcement learning approach in DQN+GNN has a single centralized agent, and the method depends largely on this fact in the way it operates, while in the DQN-Routing algorithm the deep reinforcement learning is a multiagent approach and it depends largely on this fact in the way it makes routing decisions. Besides that, the DQN+GNN is applied to a problem with much simpler constraints than what the DQN-Routing algorithm has and altering the DQN+GNN to match the constraints of DQN-Routing would lead to some computational expensive operations that will consume more time and the whole process will not be efficient.

Looking at the generalization problem from a different point of view, where it is caused by the huge amount of time required on each time to run and that is caused by the need of a big number of interactions between agents and the environment to learn the suitable policy [12, 13]. And this is a problem related to the nature of the reinforcement learning agent that has no prior knowledge about the environment and no pre-existing data that can be used and therefore there must be a considerable amount of time devoted to exploration. Based on this, an approach of domain adaptation was presented [14], to use the idea of transfer learning in the reinforcement learning field. As a subsection of transfer learning, the domain adaptation techniques aim to transfer the knowledge from a source domain to a target domain, where the source domain is a problem that the algorithm solved efficiently, and the target domain is a problem similar to the problem in the source domain but somehow different and the algorithm applied in the source can probably solve it as well. So instead of running the algorithm from the source domain on the target domain data, domain adaptation techniques transfer the knowledge from the source domain to the target domain. To do so, an alignment should be done between the source domain and the target domain, this alignment should have a kind of encoders to make the source and target domain data similar in structure as much as possible and a discriminator should be built to distinguish which data is coming from the source domain and which is coming from the target domain and that is to test how good the encoding process is working.

There are a lot of approaches for the alignment and almost all of them are related to the problems of both source and target domains. The advanced approaches in the domain adaptation that show the power of it are applied to tasks that are similar but not identical, as an example of this is the Atari games where domain adaptation technique achieved a great result and stood out the state-of-the-art results when trying to transfer knowledge from a trained game of Pong to the games of Breakout and Tennis without the need to train the agents again on the policy. All those games have similar tasks, but the tasks are not identical, and the domain adaptation approach depends on this fact that there is a need to encode the input data in the source and target domains. In the DQN-Routing algorithm, and in terms of the domain adaptation approach as explained before, it will not be applicable, because DQN-Routing does not have different tasks between the source domain and the target domain. Actually the task is exactly the same, the only thing that changes is the input data representing the topology and its settings but that does not create a different task, so what the DQN-Routing algorithm has instead can be considered as domain shifts, and this shift is either a big or a small shift depending on how two input topologies are similar.

Instead, a basic domain adaptation technique called Adversarial Discriminative Domain Adaptation [15] can be used to solve such generalization problem where the difference between the source domain and the target domain is considered as a domain shift. However, this approach still needs an alignment of two domains and in those alignments, there should be a way to represent the input data and it should be trainable for the discriminator, for the DQN-Routing to describe the input graph the algorithm only uses graph embeddings. Unfortunately, the current version of the DQN-Routing algorithm can only calculate the embeddings but not train them. That makes the standard Domain Adaptation technique not applicable for the DQN-Routing generalization problem.

3. FastGe: a proposed solution for fast generalization to new topologies

3.1. Why DQN-Routing fails in generalization?

First a deeper analyzing should be conducted, and a better understanding should be obtained to determine what is the main reason that is preventing the DQN-Routing algorithm from generalizing to a new topology.

When the DQN-Routing algorithm operates on a topology and its settings as an input, it calculates the embeddings of the graph topology. These embeddings are the only way how an input layout is represented in the whole pipeline of the DQN-Routing algorithm. After that, the pre-training process starts based on the calculated embeddings as an input along with some algorithm related parameters from the input setting file, as a result of which a pre-trained model is built and saved. This pre-trained model is built based on a very specific information related strictly to the input topology and setting. The next step is to train the algorithm based on the pre-trained model and running the scenario selected in the input setting file. Eq. 1 describes the two ways of running the DQN-Routing on a new topology.

The way this pipeline is built gives us an idea on why the generalization is failing when a new topology is introduced to the DQN-Routing algorithm. If we do not need the algorithm to run from scratch, and we need it to generalize from previously obtained knowledge. The DQN-Routing algorithm will load the saved pre-trained model of an old input topology and setting, and will continue the process from the training point. That does not only eliminate the firsts steps of calculating the embeddings related to the new input topology, it will take more time just for the training process because in this case the training is trying to learn a scenario based on pretrained model of another topology.

In other words, the training process is split to two parts. The first one is pre-training on a topology and settings and the second one is to continue the training process on another topology. This clearly explains why running the

DQN-Routing algorithm from scratch on the new topology will lead to better result than trying to generalize from a trained one. Running from scratch means having the same topology layout and setting in all stages of the algorithm while in the other case any changes in the new topology compared to the old one will confuse the algorithm as it does not know about such layout or setting.

3.2. FastGe idea and scheme

Based on the analysis we conducted above and on the specific way that the DQN-Routing algorithm is built, we propose a solution based on the **wisdom of crowd principle**, even though that this principle is used in problems of machine leaning different form the routing problems like general classification [16] or in some convolutional applications like the immune algorithm [17]. The wisdom of crowds basically means that making decisions based on multiple sources of data is more efficient than just using a single source. In the DQN-Routing a single pre-trained model contains information about a single case, a single topology with a specific number of nodes and length of conveyors and specific parameters, and unless the new topology is completely identical to the topology which the pre-trained model is built on, the DQN-Routing algorithm will fail to generalize. That leads us to a solution in which we create a new pipeline that will build a combined model as the pre-trained model, based on pre-trained models of several topologies and settings where those topologies are different in size and scenarios and embeddings, the proposed solution was called FastGe and is presented in Listing 1.

Algorithm 1: FastGe Algorithm

```
initialization:
$f(x)$ is pre-training function;
$g(x)$ is training function;
$X_{train}$ a set of training input for the DQN-Routing algorithm;
$X_{test}$ a set of testing input for the DQN-Routing algorithm;
foreach training topology $x_i \in X_{train}$ do
    return $f(x_i)$;
end
combine all the produced pre-trained models;
return $f(X_{combined})$;
foreach test topology $x_j \in X_{test}$ do
    skip pre-training;
    $Y = g(f(X_{combined}))$;
    return $Y$;
end
```

3.3. FastGe training

To build FastGe, we prepared a set of random connected graphs representing the conveyors networks, then we choose 20 graphs with sizes between 15 and 100 nodes . We made sure that our selection is also based on the most non similar embeddings of the graphs. We took 10 of those topologies and run the DQN-Routing algorithm on them, we saved the produced pre-trained models. Next, we built the core of FastGe, which is combining those models into one model, a new class was created, this class will take the 10 pre-trained models and merge all of them in one model, this model will contain features trained on different layouts and settings, while a single model of the pre-trained models focuses on only one layout and settings. Each pre-trained model as a weak model with low complexity and that means lower variance error and combining all of them will leave the variance error lower when compared to an individual model with the same capabilities.

After building the combined model, we no longer need the pre-training phase of the DQN-Routing algorithm, and that alone will save a lot of time in the run time. Based on our solution, any new topology will load the combined model and then train based on it considering it to be the pre-trained model of this topology. But it is now better than it because now it has features from a wide range of topologies layouts and settings and that will make the whole training process faster and more efficient.

4. Experiments

We took two samples of our input topology. The first one is quite a small topology of 18 nodes showed in Appendix A. We will refer to it as **S-topology**. And the second one is a large topology of 70 nodes showed in Appendix B. We will refer to it as **L-topology**.

For each of those two, we created a setting file with scenarios suitable for the size of the topology and number of sources and sinks.

All experiments were conducted on a machine with four separate NVIDIA GeForce GTX 1080 Ti, a 16 threads Intel i5-6600K processor and 128 GB RAM memory.

We ran three series of experiments: 1) on DQN-Routing generalization; 2) on FastGe performance; 3) on FastGe generalization.

Experiments on DQN-Routing generalization. The first experiment is to fully train the DQN-Algorithm on S-topology and then try to generalize it to L-topology. Then we experimented the opposite by training the DQN-Routing algorithm on L-topology and then try to generalize it to the S-topology. The goal of this experiment is to investigate how well DQN-routing generalizes on new topologies.

Experiments on FastGe performance. For following experiments, we used 10 topologies to create the combined model, and another five topologies for testing including S-topology and L-topology. The second experiment is to test the efficiency of the proposed solution for fast generalization in comparison of the generalization results from first experiment for S-topology and L-topology.

Experiments on FastGe generalization. A third experiment is conducted to compare the original performance of DQN-Routing algorithm when run from scratch on new topologies and running our FastGe to generalize to the new topologies without running from scratch. We used the same topologies as in the previous experiment. We refer to running the DQN-Routing algorithm from scratch as FULL-RUN, and to the generalization experiments by GENERALIZATION.

5. Results

5.1. Results on DQN-Routing generalization

The results of those experiments are shown listed in Table 1 and Fig. 1 show a comparison of the mean delivery time of L-topology when fully trained and when generalized from S-topology. Fig. 2 shows a comparison of the mean power consumption for that topology in both cases. These experiments show the lack of generalization capability of the DQN-Routing algorithm. This means that it has to be retrained from the scratch for each new topology it sees.

Table 1. DQN-Routing generalization performance

Experiment	Run time (*HH:MM:SS*)	Collision number	Mean delivery time	Mean power consumption
S-topology full train	00:05:44	25	232	2.7
L-topology full train	01:12:48	9	363	2.3
S-topology generalized from L-topology	00:09:04	26	261	2.8
L-topology generalized from S-topology	01:23:17	22	373	3.8

5.2. Results on FastGe performance

Results on performance of FastGe are presented in Table 2. Fig. 3 shows the mean delivery time for S-topology before and after using the proposed method, and Fig. 4 shows the mean power consumption.

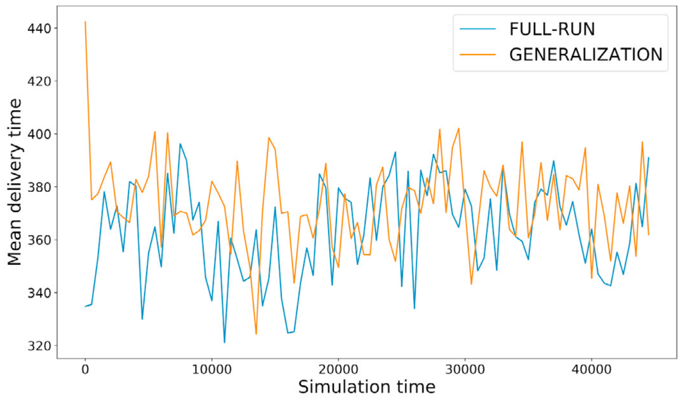


Fig. 1. Mean delivery time for L-topology when full train vs when generalized from S-topology

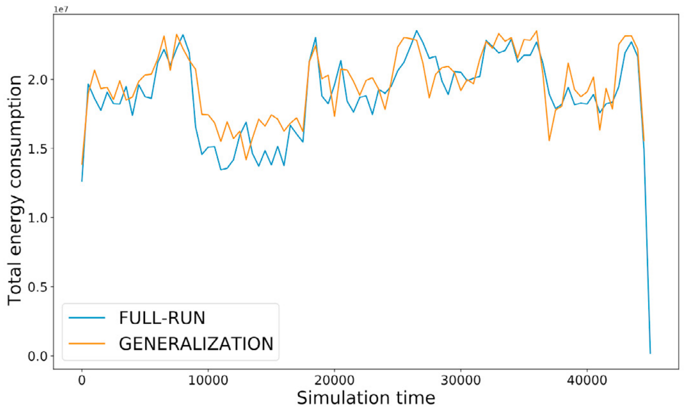


Fig. 2. Mean power consumption for L-topology when full train vs when generalized from S-topology

Table 2. FastGe generalization performance

Experiment	Run time (*HH:MM:SS*)	Collision number	Mean delivery time	Mean power consumption
S-topology full train	00:05:44	25	232	2.7
L-topology full train	01:12:48	9	363	2.3
S-topology generalized with FastGe	00:04:04	17	218	2.6
L-topology generalized with FastGe	00:33:17	1	274	1.8

5.3. Results on FastGe generalization

The results are listed in table 3 for the five testing topologies.

6. Conclusion

In this paper, we focus on the multi-agent-based routing. Our work explains the peculiarity of the generalization problem in the DQN-Routing algorithm with a review on related possible solutions and how the specific nature of DQN-Routing algorithm prevents using common solution. A proposed solution based on the wisdom of crowds principle was developed and explained to boost the generalization capability of the DQN-Routing algorithm. All ex-

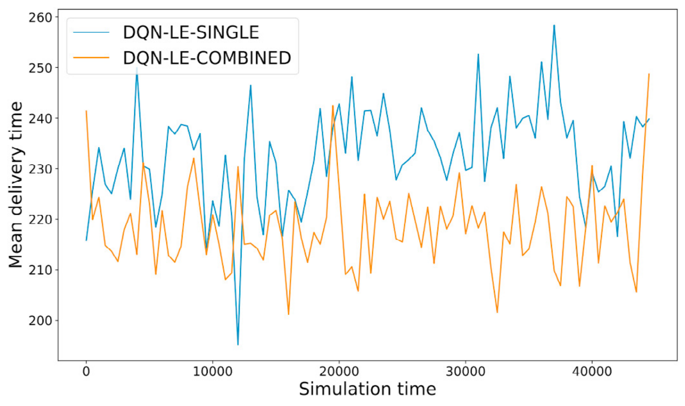


Fig. 3. Mean delivery time for S-topology DQN-Routing old generalization (single) VS mean delivery time for S-topology using FastGe (combined)

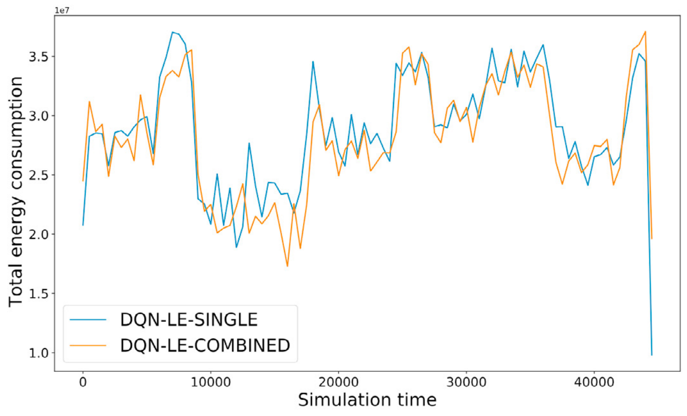


Fig. 4. Mean power consumption for S-topology DQN-Routing old generalization (single) VS S-topology FastGe generalization (combined)

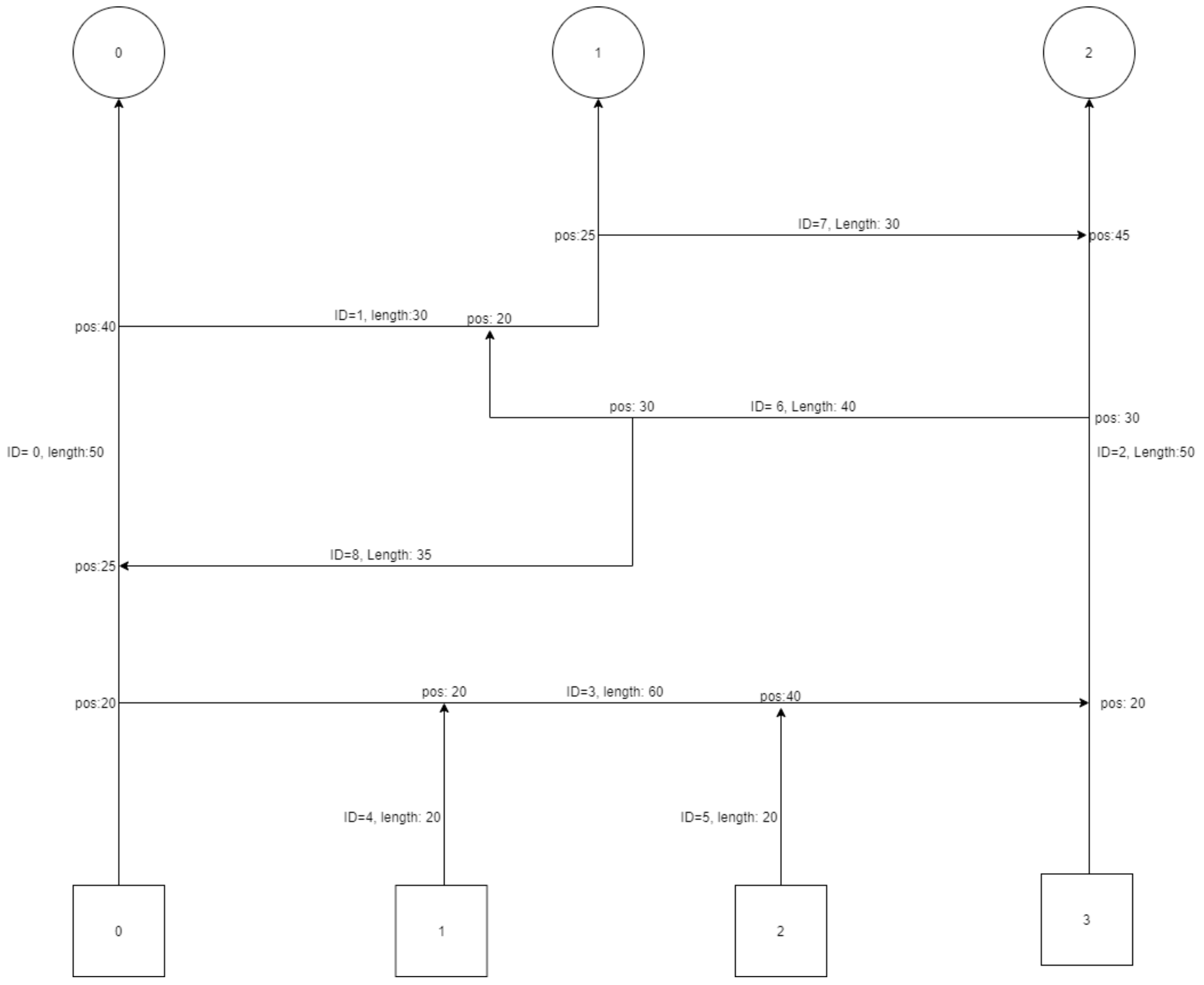
Table 3. Results of second experiment. All values are for FastGe / DQN-Routing

Topology	Run time (*HH:MM:SS*)	Collision number	Mean delivery time	Mean power consumption
S-topology (18 nodes)	00:04:04/00:05:44	17/25	218/232	2.6/2.7
L-topology (70 nodes)	00:33:17/01:12:48	1/9	274/363	1.8/2.3
topology-1 (44 nodes)	00:29:41/00:39:47	1/7	180/290	1.5/1.5
topology-2 (32 nodes)	00:22:38/00:34:09	2/5	255/300	1.6/1.5
topology-3 (20 nodes)	00:10:29/00:18:56	0/7	110/155	1.1/1.2

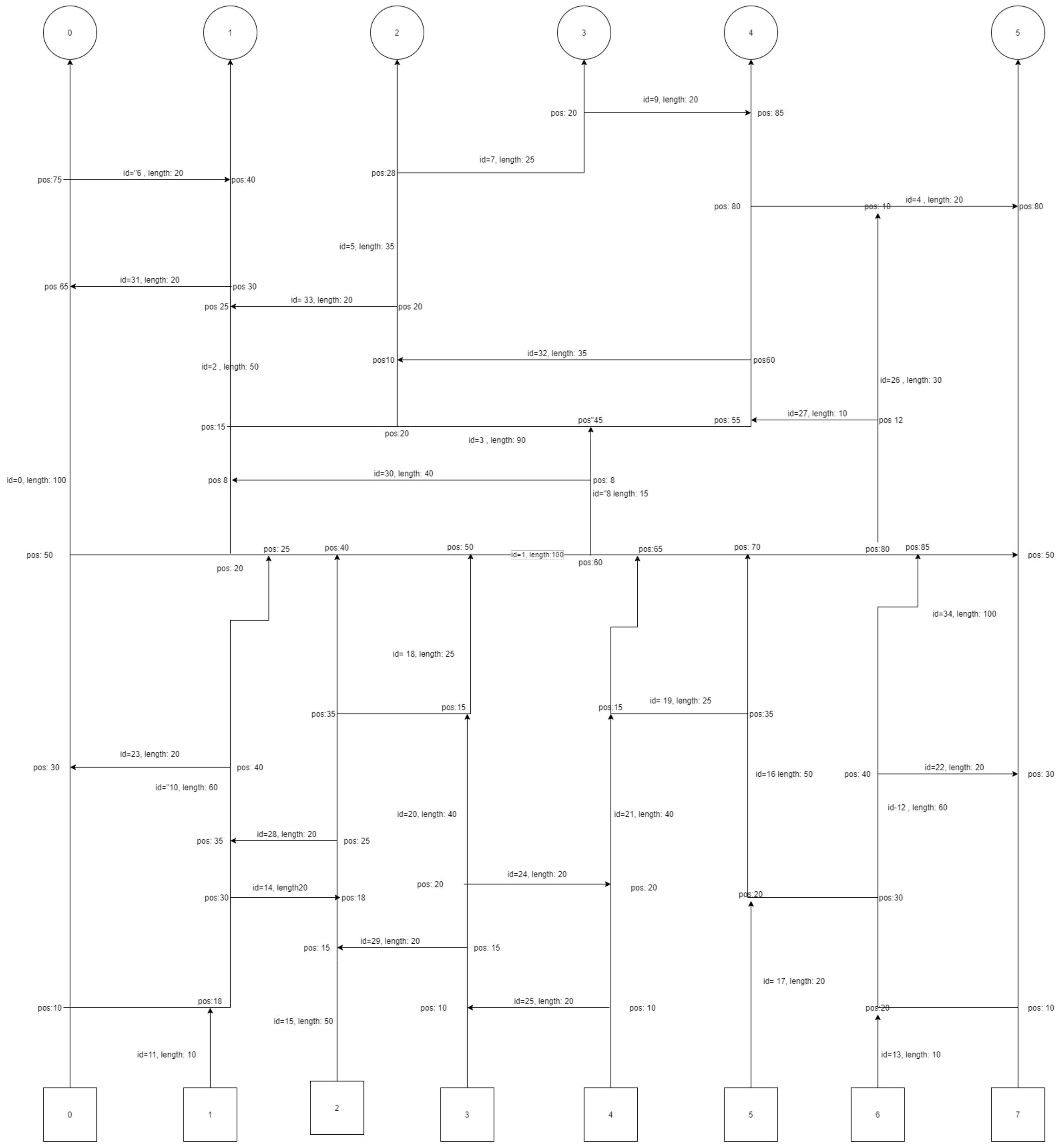
periments that were conducted show that the proposed solution led to a general better performance as well as solving the generalization problem. Mean delivery time and mean power consumption was reduced and the solution run time is faster after applying our solution. FastGe outperformed the DQN-Routing algorithm solution for the distributed routing problem of baggage handling system, and it solved the generalization problem for new topologies by giving results better than running from scratch and in less time. This work is considered as a solid base for potential other solutions and enhancement of the DQN-Routing algorithm as the process of developing the algorithm goes on.

Acknowledgements. The work was financially supported by the Russian Science Foundation (Project 20-19-00700).

Appendix A. S-topology of 18 nodes



Appendix B. L-topology of 70 nodes



References

[1] H. Xin. (2011) “Introduction of centralized and distributed routing protocols.” *2011 International Conference on Electronics, Communications and Control (ICECC)*

[2] Liu, J., Wan, J., Wang, Q. et al. (2016) “A survey on position-based routing for vehicular ad hoc networks.” *Telecommun Syst* **62** : 15–30.

[3] Adamu Murtala Zungeru, Li-Minn Ang, Kah Phooi Seng. (2012) “Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison.” *Journal of Network and Computer Applications* **35** : 1508-1536.

[4] Yong Wang, Margaret Martonosi, and Li-Shiuan Peh. (2006) “A Supervised Learning Approach for Routing Optimizations in Wireless Sensor Networks.” *roceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality* : 79-86.

[5] Valadarsky, Asaf and Schapira, Michael and Shahaf, Dafna and Tamar, Aviv. (2017) “A machine learning approach to routing.” *arXiv preprint arXiv:1708.03074*

[6] Duraipandian, M. (2019) “Performance evaluation of routing algorithm for Manet based on the machine learning techniques.” *Journal of trends in Computer Science and Smart technology (TCSST)* **1** : 25-38.

[7] Al-Rawi, Hasan AA and Ng, Ming Ann and Yau, Kok-Lim Alvin. (2015) “Application of reinforcement learning to routing in distributed wireless networks: a review.” *Artificial Intelligence Review* **43** : 381-416.

[8] Sun, Penghao and Hu, Yuxiang and Lan, Julong and Tian, Le and Chen, Min. (2019) “TIDE: Time-relevant deep reinforcement learning for routing optimization.” *Future Generation Computer Systems* **99** : 401-409.

[9] Grondman, Ivo and Busoniu, Lucian and Lopes, Gabriel AD and Babuska, Robert. (2012) “A survey of actor-critic reinforcement learning: Standard and natural policy gradients.” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **42** : 1291-1307.

[10] Dmitry Mukhutdinov, Andrey Filchenkov, Anatoly Shalyto, Valeriy Vyatkin. (2019) “Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system.” *Future Generation Computer Systems* **94** : 587-600.

[11] Almasan P, Suárez-Varela J, Badia-Sampera A, Rusek K, Barlet-Ros P, Cabellos-Aparicio A. (2019) “Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case.” *arXiv preprint arXiv:1910.07421*.

[12] Cobbe K, Klimov O, Hesse C, Kim T, Schulman J. (2019) “Quantifying generalization in reinforcement learning.” *International Conference on Machine Learning* : 1282-1289.

[13] Wang H, Zheng S, Xiong C, Socher R. (2019) “On the generalization gap in reparameterizable reinforcement learning.” *International Conference on Machine Learning* : 6648-6658.

[14] Carr, Thomas and Chli, Maria and Vogiatzis, George. (2018) “Domain adaptation for reinforcement learning on the atari.” *arXiv preprint arXiv:1812.07452*.

[15] Tzeng, Eric and Hoffman, Judy and Saenko, Kate and Darrell, Trevor. (2017) “Proceedings of the IEEE conference on computer vision and pattern recognition.” *Future Generation Computer Systems* : 7167-7176.

[16] Zhou, Denny and Platt, John C and Basu, Sumit and Mao, Y. (2012) “Learning from the wisdom of crowds by minimax entropy.” ıtMicrosoft Research, Redmond, WA .

[17] Cohen, Irun R and Efroni, Sol. (2019) “The immune system computes the state of the body: crowd wisdom, machine learning, and immune cell reference repertoires help manage inflammation.” *Frontiers in immunology* **10** .