energies

MDPI

Article

Machine Learning Requirements for Energy-Efficient Virtual Network Embedding

Xavier Hesselbach * and David Escobar-Perez

Department of Network Engineering, Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3, 08034 Barcelona, Spain; david.escobar@estudiantat.upc.edu

* Correspondence: xavier.hesselbach@upc.edu; Tel.: +34-93-401-59-87

Abstract: Network virtualization is a technology proven to be a key enabling a family of strategies in different targets, such as energy efficiency, economic revenue, network usage, adaptability or failure protection. Network virtualization allows us to adapt the needs of a network to new circumstances, resulting in greater flexibility. The allocation decisions of the demands onto the physical network resources impact the costs and the benefits. Therefore it is one of the major current problems, called virtual network embedding (VNE). Many algorithms have been proposed recently in the literature to solve the VNE problem for different targets. Due to the current successful rise of artificial intelligence, it has been widely used recently to solve technological problems. In this context, this paper investigates the requirements and analyses the use of the Q-learning algorithm for energy-efficient VNE. The results achieved validate the strategy and show clear improvements in terms of cost/revenue and energy savings, compared to traditional algorithms.

Keywords: energy efficiency; virtual network embedding; revenue; artificial intelligence; Q-learning; energy savings

check for updates

Citation: Hesselbach, X.; Escobar-Perez, D. Machine Learning Requirements for Energy-Efficient Virtual Network Embedding. *Energies* **2023**, *16*, 4439. https://doi.org/10.3390/en16114439

Academic Editor: Valentina E. Balas

Received: 22 April 2023
Revised: 26 May 2023
Accepted: 29 May 2023
Published: 31 May 2023

1. Introduction

Network virtualization [1] involves the revolutionary concept of decoupling the underlying physical network infrastructure from the services and applications running on it. It enables the creation of multiple virtual networks, each possessing its own unique set of resources and functionalities, onto a shared physical network infrastructure.

One of the biggest challenges of network virtualization is finding the most effective way to embed the demands of different virtual networks on a physical network trying to make the best use of the resources. This problem is called Virtual Network Embedding (VNE).

In this context, the business model includes the network services provider, the network operator and the infrastructure provider. Virtual operators handle a part of a partition of the physical resource, not affected by others, like a real independent network, with its own resources, protocols and strategies. Virtual operators demand resources to build their virtual networks (nodes and links) from the substrate infrastructure (physical network) constructed by the infrastructure providers.

Virtual network embedding [1,2] is the efficient assignment of a set of Virtual Network Requests (VNR) onto a physical network. These networks are composed of a set of nodes interconnected by links. The demands or resources can be of any type, such as processing capacity (CPU), energy consumption, bandwidth capacity (bit rate), delay and so on.

Many algorithms have been proposed recently in the literature to solve the VNE problem for different targets [3] and regarding the strategies in terms of allocation method (online and online), coordination (coordinated or uncoordinated), type of constraints (CPU, bandwidth, delay, availability, etc), lifetime, rollback options and many other issues.

In computational complexity, the VNE problem is included in the NP-hard problem group [4], as it aims to achieve optimal resource embedding for each virtual network. The

NP-type problem can be solved with a non-deterministic Turing machine using a number of polynomial computation times [5].

VNE is a critical issue for network operators in terms of the cost and the revenue, so regarding the utilization of available resources and therefore the energy consumed. The energy consumed is especially critical in big networks, from the resources selected and the energy required for the computational effort, impacting the monetary benefits of the operator.

Regarding consumption, modern devices are not only more efficient but also more powerful than their previous generation's counterparts. This increased efficiency and power make the previous generation's devices obsolete. Consequently, in heterogeneous networks, powerful devices are typically prioritized and selected first. However, in this work, we focus on homogeneous networks, although our analysis is not limited to them.

While the consumption model is an essential aspect to consider, it is not thoroughly examined in this paper. Nevertheless, we have taken it into account when calculating the cost and revenue metrics.

Each VNR is processed in general in two stages: The one that handles the virtual nodes and the one that handles the virtual links. In some works, both stages are processed altogether in a coordinated way. In this case, usually the complexity is bigger [6].

In [3], the authors propose an evaluation model related to coordinated mapping. In [7], the mechanism proposed is a good application example for shared and federated cloud services.

In the Virtual Node Embedding stage, each virtual node on the same network must be assigned to a different physical node. Therefore, this assignment consists of the following function: M_N (): $N^V \rightarrow N^S$. Where N^V refers to a virtual node and N^S refers to a substrate node. M_N (M) $\in N^S$. M_N (M) = M_N (N), only if M = N, whenever it is possible to satisfy the demands of the virtual nodes with the resources of the physical network. The demands that arrive can be CPU, energy consumption, storage, capacity of the node, etc.

In the Virtual Link Embedding stage, each virtual link can be embedded on a single physical link or on more than one. Virtual link embedding is implemented in the following manner: M_L (): $L^V \rightarrow P^S$ for all virtual links. P^S (M_N (M), M_N (N)), where P^S represents the path which starts with the node M_N (M) until the node M_N (N). Thus, it is needed to be satisfied: M_L (M,N) $\subseteq P^S$ (M_N (M), M_N (N)), as long as the demands of the virtual links are fulfilled. The demands that are made in a VNR regarding this scenario are usually link capacity, delay and energy consumption at the interfaces (network interface card) in the link.

The use of Artificial Intelligence (AI) and learning strategies based on ML [8] as a means to solve complex technological problems is growing exceptionally fast. This opens up a range of possibilities to be able to tackle problems such as VNE, which had not been considered taking into account the AI-aware algorithms.

When reinforcement learning [9] is focused on the VNE problem, the algorithms of this group can be used since the environment required can be provided by the scenarios of the physical network. Another advantage of this type of learning is that it does not need any kind of input data, unlike other types of learning. It learns with a state-action mechanism receiving feedback from the environment itself.

The main objective of this paper is to underscore the remarkable advantages that AI brings to the forefront when addressing VNE (Virtual Network Embedding) problems. This research endeavors to introduce groundbreaking perspectives and advancements by precisely defining the fundamental evaluation metrics, showcasing the potential benefits through an AI strategy built upon Q-learning and extensively delving into the specific objectives that can be effectively addressed through the utilization of AI algorithms. By doing so, this work not only sheds light on the immense potential of AI in the realm of VNE but also lays the groundwork for transformative progress and pioneering developments in the field. The proposal is based on [9].

The rest of the paper is organized as follows: In Section 2, the methods required in this work to solve the VNE problem are reviewed. In Section 3 the performance of our proposed algorithm is compared with typical VNE algorithms. In Section 4 a discussion is presented based on the results obtained. Finally, conclusions are drawn in Section 5.

2. Materials and Methods

In this section, the methodology and tools required to solve the VNE problem of this work are presented.

2.1. Development Tool

A solid environment is essential to develop and implement the algorithm, to compare the results with other algorithms. ALEVIN [10,11], coded using Java, manages various types of offline algorithms and arbitrary parameters of resources and demands. With the help of the user manual included in the downloadable software [12], it easily enables developers or researchers to add their own algorithms, parameters or metrics to the software. It has a graphical interface that allows us to view physical and virtual networks as directed graphs in order to make the simulations more intuitive. Therefore, ALEVIN has been chosen in this work as the environment to develop, implement and compare the algorithm.

2.2. Construction of the Algorithm

The environment definition is a critical part of the implementation of the algorithm. The environment consists of a physical network with different resources in links and nodes, formed mainly by bandwidth and CPU resources, and also several requests in the form of virtual network embeddings with different demands of resources.

The structure of reinforcement learning algorithms (Figure 1) consists of an agent and an environment, exchanging information through states, actions and rewards. The agent chooses an action and the environment generates a reward depending on the chosen action and returns a new state to the agent. In the strategy, for each virtual network to be embedded, an agent must be initialized who will have access to the resources of the physical network.

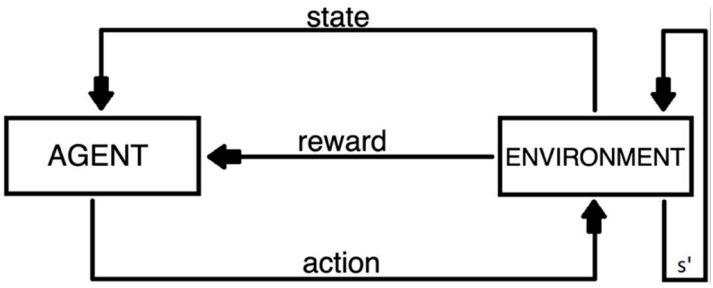

Figure 1. Basic reinforcement learning model.

The actions that an agent will allocate refer to the nodes of the physical network so that for each state the agent will have a set of actions: $A = \{0 \ldots N^f\}$, where N^f will be equivalent to the amount of total physical network nodes that can be chosen to make the embedding in the same state. By defining the actions as the set of nodes of the physical network, the states will be the nodes of the virtual network request that will be embedded and, therefore, the agent will also have a set of states $S = \{0 \ldots N^v\}$, where N^v will refer to the total number of virtual network nodes that will define the states.

With the definitions of states and actions, the node embeddings can be processed.

Figure 2 illustrates a simple example with 2 VNRs and a physical network. It showcases a plausible and simplified allocation of the three nodes from VNR1 onto the physical network, providing a visual representation of the concept being discussed, such that the virtual network 1 embedding would be written with the following notation:

{0, 0}, {1, 1}, {2, 2}, where the tuples are pairs of action states {S, A}.

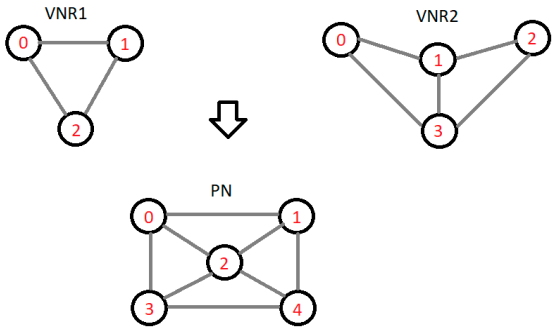

Figure 2. VNE example.

In this algorithm, a matrix Q is defined, storing the results of every Q value of each state-action pair. By definition, this matrix has as columns the set of actions that the agent can take and as rows the set of states that the agent can choose (Figure 3).

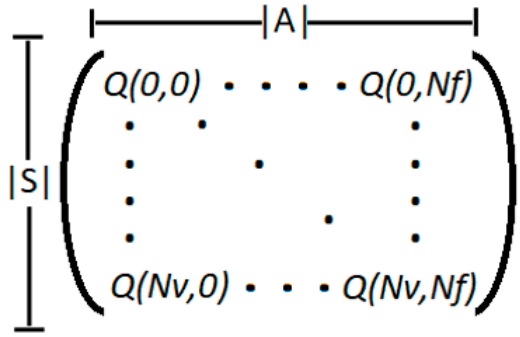

Figure 3. Structure of Q matrix.

The number of rows in this array is |S| and the number of actions is |A|. As we explained, the states represent the set of nodes in the virtual network to be embedded and the actions represent the set of nodes in the physical network. Therefore, each element $Q(S_i, A_i)$ represents the quality of that state-action pair, which means that a large value of Q will correspond to a good state-action pair and a small value of Q will correspond to a bad state-action couple.

The procedure is the following:

At each choice of state-action pair, the value that belongs to it must be calculated, according to the equation that calculates the Q value:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha\left(R(s,a) + \gamma max_{a'} Q(s',a') - Q_{t-1}(s,a)\right) \quad (1)$$

where $Q_{t-1}(s,a)$ is the current value in the table corresponding to the state-action choice, α is the learning value that controls how quickly the agent adopts the random changes imposed by the environment, $R(s,a)$ is the reward provided to the agent for taking the corresponding state-action pair, γ is a discount factor corresponding to the affectation of future states to the current, $max_{a'} Q(s',a')$ is the maximum value Q corresponding to the new state.

As the algorithm in the problem has been suggested, the choice of states made by the environment does not depend on the action chosen in the previous state, as it does not make sense to choose the next virtual node to embed from one action selected in the previous state. This peculiarity makes the dependence between states zero, forcing a series of adaptations to the equation that calculates the Q value in each state by deleting the factor that calculates the incidence of future state-action pairs in the current:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) - Q_{t-1}(s,a)) \quad (2)$$

This equation allows the algorithm to explore the environment and learn strategies, considering past experiences or values regardless of future states.

In the VNE problem of this work, the node mapping stage and the link mapping stage are considered. There has already been talk of how states are defined as virtual nodes and actions as physical nodes, leaving the node mapping scenario ready. In some works, it is done in a coordinated way [6], expecting better results but increasing the complexity.

To obtain a correlation between node mapping and link mapping, it has been decided to implement a reward function that takes into account link mapping. At each node embedding choice, this reward function will find the virtual links between the already embedded virtual nodes and the node regarding the current state and map the links between them to the corresponding physical paths.

For better resource optimization, the paths in the physical network where the virtual links are to be embedded will be the shortest paths. That is, an algorithm will be used to find the shortest paths suitable for embedding depending on whether they can satisfy resource demands or not.

The shortest paths will be found using Dijkstra's algorithm. This, once given a couple of nodes, returns the shortest path; it is included in the ALEVIN software thanks to the JUNG2 library [13].

The reward function can be defined as follows:

$$R(s,a) = \sum_{i=0}^{N^{Ve}} \frac{1}{S_i} M(link_i) \tag{3}$$

where N^{Ve} is the number of nodes that have been embedded in the pre-current states, S_i the number of hops in the physical path where the $link_i$ is to be embedded so the chosen paths are penalized with many hops and $M(link_i)$ returns a value according to the mapping of the $link_i$ link with the Dijkstra algorithm. It can be a big number if the mapping was possible or a low number if it was hardly possible due to the restrictions set by the current resources.

With the target of a more optimal embedding and a faster convergence, a function has been added: In the last state of each embedding in each iteration, the Q values corresponding to the choice of each state-action are updated. Thus, if the virtual network embedding has been successful, a constant is added to each chosen Q value, but if the embedding has not been successful, a constant is subtracted from each chosen Q value.

In order to obtain a stable Q matrix, where all state-action combinations have been seen, an exploration-exploitation rule must be followed. The agent, at the beginning of the training, has no information and all its Q values are zero, so a model capable of balancing the use of exploration and exploitation in the algorithm is required, boosting exploration in the first iterations to discover new Q values and exploitation in the last ones choosing the best Q values. Therefore, the Epsilon-Greedy model is used in this work: The Epsilon-Greedy model is a simple method to choose between exploration and exploitation in a random way, with probability ε per exploration and $1 - \varepsilon$ per exploitation. In order to obtain more stable values of Q and better convergence, it has been decided to give ε dynamic values, in such a way that it decreases as the algorithm adds iterations. This forces the algorithm to explore at the beginning when the Q table is empty and to explode at the end when all possible combinations have been discovered. The equation that computes ε in each iteration is as follows:

$$\varepsilon = min_\varepsilon + \frac{1 - min_\varepsilon}{e^{-\lambda * num_{iter}}} \tag{4}$$

where min_ε is ε minimal so that the function is never 0, λ a factor that serves to attenuate the function and takes longer to decrease more iterations and num_{iter} is the number of iterations at the present time. In this way, for each iteration, the value of epsilon will decrease with the intention of prioritizing the exploitation in future iterations.

The main idea of this algorithm is to continuously update the Q matrix with the Q values obtained from the state-action choice and their corresponding rewards. In the long

term, after a number of iterations, the Q matrix begins to gain good stability and this results in an optimal virtual network embedding.

The Algorithm 1 consists of a main loop that repeats the code with the number of iterations indicated with the variable *max_iter*. Within this loop, another loop **for** iterates over all the states, over the nodes of the virtual network. For each virtual node if ε it is smaller than num_random, it will choose the action with the best Q value in that state, as long as it satisfies the limitations imposed on the nodes. If ε is greater than *num_random*, the algorithm will take a random action of the possible ones in that state. Once an action is decided in its current state, the reward for choosing that action or physical node is calculated, and that reward depends on the mapping of the virtual links that links that virtual node to the virtual nodes already embedded. Once the reward is calculated, the Q value is updated with the Q value update formula and this Q value is stored in the Q table.

Once the **for** loop reaches the last state or the last virtual node when being embedded, after updating the Q tale with the Q value corresponding to the choice of the action, it enters the last **if**, since it satisfies the condition. Within this, if the embedding has been satisfactory, the corresponding Q values are updated with a value higher than they had before, adding a constant and, in case of the contrary, they are updated with a lower value than they had before, subtracting the same constant, in order to achieve a stable Q table in fewer iterations.

Algorithm 1 Q-learning-based VNE algorithm

```
while (num_iter < max_iter){
  for (i = 0; i< N^v; i++){
    if (num_random > ε){
      Choose action A_i with a better Q value for this state
      This Q value has to satisfy the restrictions imposed on the nodes
    }else{
      Chooses random action for this state
    }
    Calculate the reward R with the Equation (3)
    Calculate the Q value with the Equation (2)
    Update the Q table with the new Q value
    If (i == N^v){
      Update all the Q values chosen in the actual embedding with a new reward
depending on whether embedding has been completed or not}
  }
  num_iter++;
ε = min_ε + (1−min_ε) / e^(−λ*num_iter)
Return an embedding based on the Q table
}
```

3. Results

This section presents the metrics, the algorithms used to compare with the AI proposal of this work, the justification of the scenarios considered and the results from the simulations.

3.1. Algorithms Used to Compare

3 algorithms have been selected as a reference, according to the relevance of their performance shown in other works:

Detection of subgraph isomorphisms (SID) [14]: This algorithm proposes a solution regarding graph theory. It is a heuristic algorithm based on finding an isomorphic graph that represents a VNR that satisfies the demands within the physical network. One of the features of this proposal is to try to perform node and link embedding in the same scenario, which helps to take better decisions on link embedding. This algorithm has been chosen to compare with the algorithm implemented in terms of runtime, as its developers say

that mappings are very fast because everything is done in the same scenario. They also claim that it is very suitable for mapping large virtual networks, perfect for evaluating the execution time of algorithms.

Basic VN [15]: This algorithm aims to optimize link and node stress independently by dividing it into two smaller problems to solve it sequentially. Therefore, a virtual network embedding is proposed considering the stress of nodes and links throughout the virtual network embedding process. This algorithm has been chosen because its developers ensure that it has a very good performance in networks with a large number of conditions, they also ensure a very good performance in networks that are not very connected as is the case of those that are formed randomly, adding that divides the problem into two smaller ones to optimize the stress level will be a good comparison in terms of runtime.

BFS node ranking Coordinated mapping [16]: This algorithm uses a backtracking strategy when the exploration of potential paths fails, to embed the nodes and virtual links in the same stage. This algorithm has been chosen to compare with the algorithm implemented in terms of virtual network acceptance and long-term revenue, as its developers ensure a very high performance of the algorithm in these two metrics at the expense of runtime due to the high level of computing needs.

3.2. Metrics Used for the Evaluation of the Algorithm

To evaluate the algorithm and compare the results obtained, the most relevant metrics used in the literature to evaluate the performance of the algorithms that solve the VNE problem have been chosen:

- Cost: Refers to the total physical resources required to map virtual networks. It is determined by taking into account all the resources of the physical network that have been used by the VNRs.
- Revenue: Refers to the sum of the virtual resources requested by the virtual networks that have been successfully mapped.
- Cost/Revenue: This metric will be used to evaluate how the algorithm optimizes the resources of the physical network to find embeddings. A high value of this metric means that a lot of resources are needed to embed virtual networks. Optimally, Cost/Revenue = 1.
- Runtime: This metric measures the time it takes the algorithm to complete the embeddings. This is relevant in terms of the mathematical complexity of the problem and the heuristics that are taken into consideration.
- Acceptance of VNRs: This metric will be used to assess the ability of the algorithm to host different VNs on the same physical network, taking into account the use of network resources.

Three scenarios have been defined to evaluate the performance. The main features are detailed below.

3.3. Simulation Scenario 1

This scenario consists of a physical network with nine nodes forming a grid topology. This topology is chosen in order to obtain a type of network similar to the mesh networks that are so much used to improve the performance of a network. What is intended in this scenario is, from the same physical network, to perform different simulations by increasing the size of the virtual network to be embedded in order to evaluate the cost/revenue metric to get a good performance in terms of reducing costs in embedding, as all nodes in the network they are interconnected with each other.

In this scenario, the physical network nodes and links have 100 CPU resources and bandwidth, respectively. The VN increases the number of nodes by two in each subscenario, with the VN demands ranging from 0 to 90 for each link and node in terms of bandwidth and CPU, respectively.

It has already been clarified in previous sections that the cost/revenue metric refers to the division between the physical network resources required for embedding and the

resources demanded by the virtual network. The optimal value to be obtained should be close to 1, meaning low cost and high revenue.

The graph in Figure 4 shows that all algorithms are decreasing in terms of cost/revenue as the embedding of the VNR is larger. In terms of stability, it should be noted that the implemented algorithm is obtaining better values in each embedding, highlighting the first value obtained when the algorithms were executed to embed a three-node network. This means that this algorithm takes great care of the way the physical network resources are used and one could say that one of its strengths is the efficiency with which the network resources are treated in each embedding, obtaining the most optimal possible solutions.

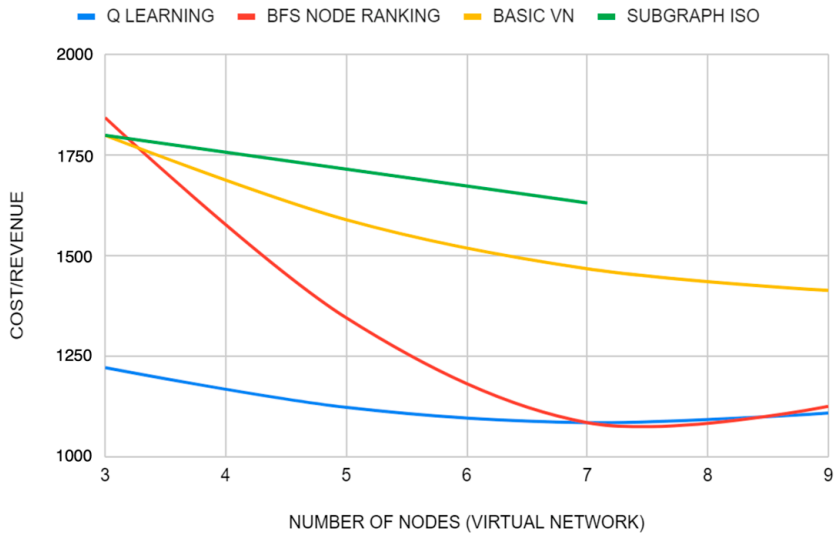

Figure 4. Cost/revenue simulation 1.

3.4. Simulation Scenario 2

This scenario consists of a physical network with ten nodes forming a topology in random shape with the Waxman generator ($\alpha = 1$, $\beta = 0.5$). The main goal is to evaluate the acceptance metric of virtual networks. So, different sub-scenarios with the same physical network will be considered, and in each, increasing in steps of one the number of virtual networks that must be embedded, starting with the first one with one VNR, the second one with two VNRs, up to the sixth one with six VNRs, in order to see the number of VNRs accepted in each algorithm and compare it with the metric obtained with Q-learning.

The generation is made using a Waxman generator. The physical network consists of a maximum of 250 resources on each CPU node and a maximum of 250 bandwidth resources on each link, which means that each node and physical link respectively will get a number of resources between 0 and 250 at random. Each VN contains 3 nodes and 3 to 5 links each, a maximum of 70 CPU resources for each node and a maximum of 70 bandwidth resources for each link, obtaining a number of resources between 0 and 70 randomly for each element of the virtual network.

In this way, the ability of algorithms to embed different networks on the same physical network where resources are randomly distributed can be evaluated.

The graph of Figure 5 shows how the implemented algorithm, Q-learning, remains constant at each execution embedding all VNRs; however, the other algorithms do not show this constancy, as they are not able to embed all VNRs in some of the executions. Therefore, Q-learning is the winner.

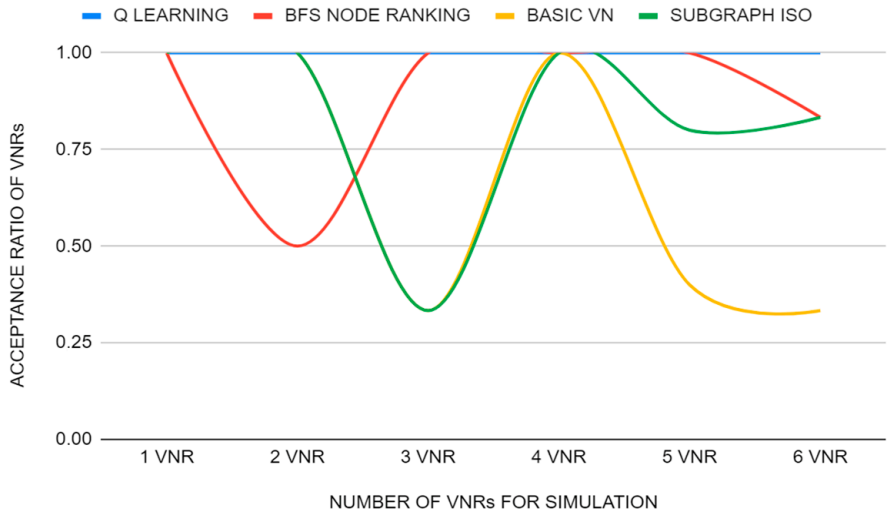

Figure 5. Acceptance ratio of VNRs simulation 2.

3.5. Simulation Scenario 3

This scenario consists of a physical network with 30 nodes forming a random network topology with the Waxman generator ($\alpha = 1$, $\beta = 0.5$). The aim is to make different simulations from the same physical network, increasing the size of the virtual network to be embedded, in order to evaluate the runtime metric. As the VNE is an NP-hard problem, the performance of the algorithm in terms of complexity with medium networks will be evaluated using the random topologies that both are used in intra-domain networks, becoming a challenge for different algorithms to embed random networks in another one of very large size.

Each sub-scenario will consist of only one VNR and the physical network so that in each sub-scenario the number of nodes will increase from the first sub-scenario with 3 nodes to the last with 10 nodes.

Physical network resources are randomly distributed on each node and linked with a value between 0 and a maximum of 300 resources. Those of the VNR are distributed in each node and linked with a value between 0 and a maximum of 50 resources.

As can be seen in Figure 6, basic VN and SID get very good execution times and a projection that in large scenarios this will not increase so abruptly. On the other hand, BFS node ranking coordinated VNE and Q-learning get worse times, making Q-learning slightly worse than BFS node ranking.

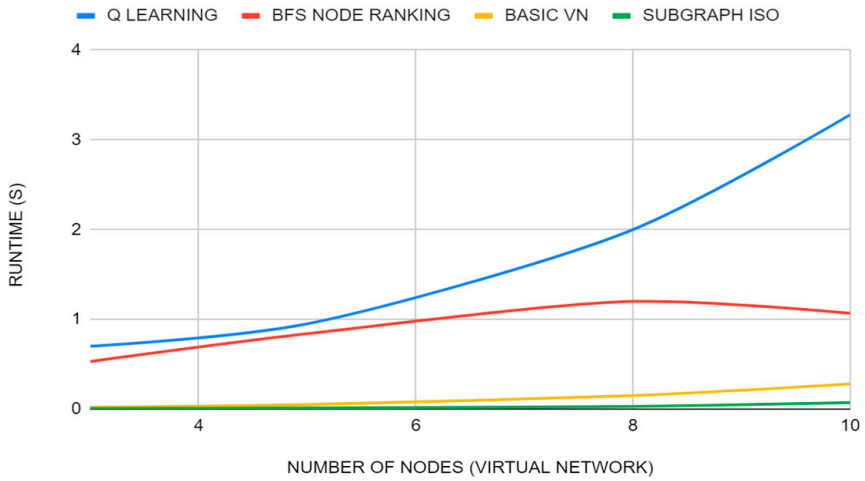

Figure 6. Runtime simulation 3.

4. Discussion

Three algorithms have already been used to compare the results with the algorithm implemented, that three already implemented in ALEVIN, which is unexpected the low runtime of the SID and basic VN algorithms, although these do not show good stability in terms of acceptance of virtual networks and have a poor cost/revenue ratio compared to the other two algorithms.

The BFS node ranking coordinated mapping algorithm obtains execution times slightly shorter than the implemented algorithm, but longer than those of SID and basic VN. On the other hand, it gets better results in the other two metrics with respect to SID and basic VN, but worse results than the AI algorithm, which does not make it stand out in any way.

The modeling of the VNE problem with an algorithm from AI opens up new roads for studying the efficiency of traditional algorithms used to solve the problem, as these present worse metrics in terms of cost/revenue and acceptance of virtual networks.

The good results obtained by Q-learning are not a coincidence. Throughout this work it has been shown how this algorithm is focused on improving these two metrics, thanks to the training method implemented with the corresponding rewards obtained with the virtual link embedding and with the incentives given at the end of each iteration depending on whether the embedding has been a success or not. One must take into account, however, the large number of iterations that the algorithm needs to find the most optimal solution to the problem, but also the best metrics of the comparison in cost/revenue and in acceptance of virtual networks has been reached. So, in terms of efficiency and performance, it is the best strategy.

In the current state proposed, Q-learning is not designed to solve problems in very large environments, as it would result in a table with thousands of states and actions, but it is a promising strategy. Finding a way to improve the algorithm in terms of runtime is presented as a difficult but exciting challenge. In order to get better performance in terms of runtime, two ways could be explored:

- Try a new AI algorithm that solves the VNE problem, which means that all the progress made in terms of cost/revenue and acceptance of VNRs may change.
- Improve the algorithm with new tools to be able to obtain better results in terms of execution time, with good results in the other metrics.

The recommendation is to explore these AI tools that are presented as an update to the implemented algorithm:

Deep Q-learning [17] is a great opportunity to improve the implemented algorithm. This algorithm is an extension of the classical Q-learning algorithm, it uses deep neural networks to approximate Q values, where states form the input and the Q value of all possible actions is generated as the output. Figure 7 maps a neural network where the input is the states with value [1–4] and the output is represented by the actions with their corresponding Q value. Using neural networks to the detriment of the Q tables to store the Q value of each state-action pair makes the computation time required to execute the algorithm shorter.

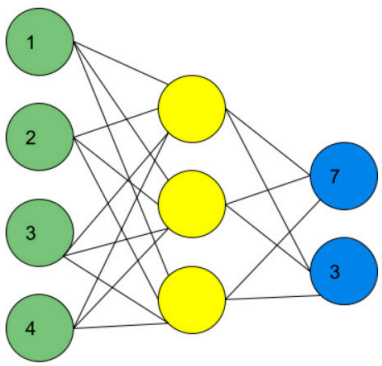

Figure 7. Mapping of a neural network.

Including this extension in the classic Q-learning algorithm would improve performance in terms of runtime and it is expected to maintain good results in terms of cost/revenue and acceptance of VNRs, as the body of the algorithm would remain the same because it would only change the way the information is stored.

5. Conclusions

Machine Learning requirements for energy-efficient Virtual Network Embedding have been studied and defined. The primary focus of this paper has been to emphasize the compelling advantages offered by AI when addressing VNE problems. This research aims to introduce novel insights and advancements by defining the essential metrics necessary for evaluation, demonstrating the potential benefits through an innovative AI strategy based on Q-learning, and extensively exploring the specific objectives that can be effectively tackled using AI algorithms.

Based on the initial findings presented in this paper, the proposed strategy emerges as an exceptionally promising approach to significantly enhance the effectiveness of embedding in small and medium virtual networks. In terms of both VNR acceptance rates and cost/revenue considerations, this strategy showcases notable advantages over existing methods documented in the literature. These early results strongly suggest that the proposed approach holds substantial potential for achieving superior performance and achieving notable advancements in the field of virtual network embedding.

Hence, this work serves as a pivotal foundation for the development of the next generation of coordinated or uncoordinated VNE strategies, with a specific focus on optimizing energy efficiency. By laying the groundwork and presenting groundbreaking insights, this research paves the way for innovative approaches that will significantly advance the field and foster substantial improvements in energy-efficient VNE methodologies.

As this work is an initial exploration into harnessing the promising advantages of AI in tackling Virtual Network Embedding problems, it is imperative to conduct comprehensive tests encompassing diverse strategies and parameters. This approach is vital in order to evaluate the performance and effectiveness of the proposed methodology, paving the way for further advancements.

Author Contributions: Conceptualization, X.H.; Software, D.E.-P.; Formal analysis, X.H.; Investigation, X.H.; Writing—original draft, X.H. and D.E.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Agencia Estatal de Investigación of Ministerio de Ciencia e Innovación of Spain under project PID2019-108713RB-C51 MCIN/AEI/10.13039/501100011033.

Data Availability Statement: Data sharing not applicable. No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fischer, A.; Botero, J.F.; Beck, M.T.; De Meer, H.; Hesselbach, X. Virtual Network Embedding: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1888–1906. [CrossRef]
2. Cao, H.; Wu, S.; Hu, Y.; Liu, Y.; Yang, L. A survey of embedding algorithm for virtual network embedding. *China Commun.* **2019**, *16*, 1–33. [CrossRef]
3. Fan, W.; Xiao, F.; Lv, M.; Han, L.; Wang, J.; He, X. Node Essentiality Assessment and Distributed Collaborative Virtual Network Embedding in Datacenters. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 1265–1280. [CrossRef]
4. Andersen, D.G. Theorical Approaches to Node Assignment. 2002. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.119.1332 (accessed on 29 September 2021).
5. NP (Complexitat)—Viquipèdia, L'enciclopèdia Lliure. Available online: https://ca.wikipedia.org/wiki/NP_(Complexitat) (accessed on 29 September 2021).
6. Hesselbach, X.; Amazonas, J.R.; Villanueva, S.; Botero, J.F. Coordinated node and link mapping VNE using a new paths algebra strategy. *J. Netw. Comput. Appl.* **2016**, *69*, 14–26. [CrossRef]
7. Sermakani, A.M.; Paulraj, D. Effective Data Storage and Dynamic Data Auditing Scheme for Providing Distributed Services in Federated Cloud. *J. Circuits Syst. Comput.* **2020**, *29*, 2050259. [CrossRef]

8. Andriy Burkov, B. The Hundred-Page Machine Learning. 2019. Available online: http://themlbook.com/ (accessed on 23 September 2021).
9. Qiang, W.; Zhongli, Z. Reinforcement Learning Model, Algorithms and Its Application. In Proceedings of the IEEE 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, China, 19–22 August 2011; pp. 1143–1146. [CrossRef]
10. Fischer, A.; Botero, J.F.; Duelli, M.; Schlosser, D.; Hesselbach, X.; De Meer, H. ALEVIN—A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* **2011**, *37*. [CrossRef]
11. Duelli, M.; Schlosser, D.; Botero, J.F.; Hesselbach, X.; Fischer, A.; de Meer, H. VNREAL: Virtual Network Resource Embedding Algorithms in the Framework ALEVIN. In Proceedings of the 7th EURO-NGI Conference on Next Generation Internet Networks, Kaiserslautern, Germany, 27–29 June 2011. [CrossRef]
12. Duelli, M.; Fischer, A.; Botero, J.F.; Diaz, L.; Till, M.; Schlosser, D.; Singeorzan, V.; Hesselbach, X. Alevin2/Wiki/Home. Available online: https://sourceforge.net/p/alevin/wiki/home/ (accessed on 26 September 2021).
13. JUNG—Java Universal Network/Graph Framework. Available online: http://jung.sourceforge.net/ (accessed on 27 September 2021).
14. Messmer, B.T.; Bunkem, H. Efficient subgraph isomorphism detection: A decomposition approach. *IEEE Trans. Knowl. Data Eng.* **2000**, *12*, 307–323. [CrossRef] [PubMed]
15. Zhu, Y.; Ammar, M. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In Proceedings of the 25TH IEEE International Conference on Computer Communications, Barcelona, Spain, 23–29 April 2006. [CrossRef]
16. Cheng, X.; Su, S.; Zhang, Z.; Wang, H.; Yang, F.; Luo, Y.; Wang, J. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 38–47. [CrossRef]
17. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A Theoretical Analysis of Deep Q-Learning. Available online: http://arxiv.org/abs/1901.00137 (accessed on 10 January 2019).