Received February 15, 2020, accepted March 22, 2020, date of publication March 25, 2020, date of current version April 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983264

An Efficient and Fast Embedding Algorithm for the Virtual Networks

DEDONG HU AND ZHEN YANG
Beijing University of Posts and Telecommunications, Beijing 100876, China
Corresponding author: Zhen Yang (yangzhen@bupt.edu.cn)

This work was supported by the Key Research and Development Project of China under Grant 2018YFB2100300.

ABSTRACT Virtual network mapping is a hot issue over the past decade, which aims to map virtual networks to the underlying network as required. This procedure is a NP-hard question, therefore many previous algorithms just impose some constraints or use simple heuristic method to get the relative optimal result. Traditional researches almost focus on two independent stages: node embedding and link mapping. To address existing questions such as low acceptance ratio and high mapping time, we propose an efficient and fast embedding algorithm, named as AEF(An Efficient and Fast embedding algorithm), which is a two coordinated mapping stages algorithm. AEF uses some innovative strategies in two stages to improve the mapping time and ensure high performance. It has great advantages in large-scale complex networks or occasions with high real-time requirements because of faster mapping speed. Besides, it will also have more practical value in actual scenarios due to higher acceptance ratio and more balanced link load. The worst time complexity of our AEF is proved to be O($|N^S|^2 \cdot |N^V| \cdot (\log(|N^S|) + |N^V|)$). A large number of experimental results show that our algorithm is faster than most other algorithms and ensures the overall better efficiency.

INDEX TERMS Virtual network mapping, high performance, fast embedding, heuristic algorithm.

I. INTRODUCTION

With the development of internet, more and more various applications and traffic need to adapt to various service providers [1]. Network virtualization, a most promising technology, emerges as the times require. It allows heterogeneous virtual networks to coexist in the same shared underlying network [2]. However there are some challenges, such as, how to allocate resources efficiently and fast. It introduces the virtual network mapping, which aims to map the virtual networks to the underlying network as required. Nevertheless, virtual network mapping is a NP-hard question [3], which means the optima result is impossible to get in polynomial time.

Network virtualization attracts more and more researchers to develop related solutions in the past ten years [4], [5]. Previous algorithms can be classified into two categories, Uncoordinated VNE(virtual network embedding) and coordinated VNE [1]. The former is consisted of two separated stags [6]–[9], node embedding and link mapping. It finds substrate nodes for virtual network by greedy strategy and searches k-shortest paths, then picks the appropriate path that satisfies link attribute constraints in the subsequent stage. In fact, in link mapping phase, it can also be converted to MCF(multi-commodity flow) question. Such algorithm often leads to low acceptance ratio because of the insufficient search space. The latter adopts the two coordinated stages, which can be further subdivided into three major classes [1], two stages coordinated VNE [10]–[14], one stage coordinated VNE [15]–[17], interInP coordination [18] respectively. It makes consideration of link mapping in the node embedding phase or unifies the two stages. Such algorithm may spend too much time in virtual network mapping because of the more computation.

There are some questions according to the above description. On the one hand, low acceptance ratio shows a large number of invalid requests result in low revenue and lack of practicality. On the other hand, long virtual mapping time will cause the entire network to be paralyzed, because of lots of high concurrent virtual requests in the real-time scenes. Focusing on these issues, we propose An Efficient and Fast embedding algorithm, named as AEF, which is a two-stage coordination algorithm. Our main goal is to improve mapping time and make the other evaluation metrics at a higher level.

The associate editor coordinating the review of this manuscript and approving it for publication was Nabil Benamar.

Our main contributions in this paper are summarized as follows.

- We put forward a mapping rule, which considers whether neighbor links' resource of the mapped substrate node can support neighbor links' demand of the current virtual node in terms of bandwidth metric. This rule is equivalent to considering the next mapping in advance, called 1-lookhead. It ensures the higher probability of finding a path in subsequent link mapping stage when the mapping rule is satisfied. Thereby it can improve relatively the average acceptance ratio.
- Soft distance constraint is used. It differs from traditional concept [19]. This brenchmark is considered in the first time in node mapping phase. If it fails, which means we can not find substrate node to match in term of distance constraint, then we will ignore this restrict in the second time. It can ensure the set of mapped substrate nodes are as close as possible and enhance the whole revenue in these scenes constructed by Waxman model [20].
- Similar to the traditional algorithm, in the link mapping phase, we use BFS to find the solution and consider that the use of the underlying network topology resources may be unbalanced in this stage. For this reason, this paper makes a trade-off between the link load and the link mapping cost.
- We proof that the time complexity of our algorithm can be decreased compared with D-ViNE_SP, subgraph Isomorphism and conduct a large of simulation experiments. Simulation results also show that our AEF algorithm guarantees high performance and accelerates the mapping time. Therefore our AEF has great feasibility and advantages in theory.

The rest of this paper is organized as follows. We discuss the related work in Section II. Virtual network mapping problem and model are explained in Section III. We detail the process and implementation of AEF algorithm in Section IV. Numerous simulated comparative experiments are performed in Section V. At last, we summary the experimental result and draw conclusions.

II. RELEATED WORKS

The existing virtual network mapping algorithms can be roughly divided into three categories: two-stage independent algorithms, two-stage coordination algorithms and one-stage algorithms.

Two-phase independent algorithms simply fulfills node mapping phase and link mapping phase independently. Minlan Yuy *et al.* [6] put forward the baseline VN Embedding algorithm, which completes node mapping by means of greedy strategy and uses k-shortest paths algorithm to accomplish link mapping. In addition, the authors also propose innovative path splitting and path migration. The former adopts a strategy that can divide traffic for paths with insufficient bandwidth, so that a virtual link can be split into multiple underlying paths. The latter makes room for subsequent virtual requests by adjusting the path that has been mapped. However this paper simply adjusts the flow and does not optimize it, which makes process consume a lot of time. Lu *et al.* [4] and Zhu and Ammar [7] propose improvements, they use global labeling. Some links or nodes with high load are marked as "critical" and only these are adjusted so that the cost of regulation is not too high. In addition to adjusting the nodes with high load, another strategy can be used, which is to avoid using these high-load node resources when mapping. Razzaq *et al.* [9] introduce the NEL(Node Exhaustion Limit), they search a substrate node to match in order to make the resources of the underlying node minus the requirements of the virtual node must be greater than or equal to NEL. Jarray and Karmouch [16] use the pricing model through the auction mechanism to increase the minimum auction price for nodes with high loads. These two strategies effectively protect high-load nodes, thereby providing more options for subsequent virtual requests and improving the request acceptance ratio.

Two-stage coordination algorithms consider global resources when node mapping and link mapping, and the two phases depend on each other. This type of algorithm first defines node scales through certain topological properties. Traditional attributes include CPU of a node, degree of a node and bandwidth of its surrounding links. But for heuristic algorithms, these factors often do not fully reflect the overall topology and the mapping results are not very convincing. Zheng *et al.* [13] also consider the substrate nodes which have been mapped. Cao *et al.* [11], [12] have a comprehensive consideration of Node Centrality, Link Strength, Link Interference, Distance Between Two Nodes and so on. Then it uses the node scales to get all nodes ranking. Bianchi and Presti [14] use Markov model to iteratively calculate the importance of nodes according to previous node attributes, which is similar to [12]. Reference [4] compute it by means of LP(Linear Programming) model. Next it refers to these ranking to complete node mapping. Most algorithms do this step using L2S2(Large to Large and Small to Small) greedy strategy. However, [13] matches the underlying nodes based on whether the cost after matching is the smallest, and [14] mainly depends on whether the benefits after matching are the largest, that is the most rewards. At last, it accomplishes link mapping by MCF(Multi-Commodity Flow) or KSP(K-Shortest Path). But these algorithms may take more computation time, a common optimization trick is to remove links that do not meet bandwidth demand and simplify the network topology. In [13], the authors put forward an improved minimum cost flow method. In the latest research, Yao *et al.* [21] propose a model based on reinforcement learning to solve virtual network mapping, which processes network topology attributes and structures in a matrix form and shows better performance. Zhang *et al.* [26] introduce a reinforcement learning method to complete node mapping, and achieve optimization automatically with the historical data. This is the first to utilize historical requests data, and outperforms the most other algorithms.

One-stage algorithms is mainly to unify node mapping phase and link mapping phase, which completes them in one phase. Cordella *et al.* [15] use recursive one-stage algorithm to complete link mapping while node mapping with the help of five syntax rules and some semantic rules. In addition, it can use the ILP(Integer Linear Programming) model to complete the one-stage mapping. This model requires a lot of computing time to solve. Chowdhury *et al.* [19] relax the integer variable to get the LP model, and then use D-ViNE and R-ViNE to solve it. In [16], the authors use column generation technology to make the VN embedding problem be decomposed into a master problem(which includes constraints related to the availability of substrate resources) and a pricing problem (which includes the constraints related to the embedding of VN resources) and solve it by A-JNLE_CG_B&B and A-JNLE_CG_R algorithms. These strategies can greatly reduce more the running time than the original model.

Generally speaking, the calculation time of a one-stage algorithm is often large, even the improved scheme is not suitable in the occasions with high real-time requirements and two-stage independent algorithm itself is not optimal due to the independence of the two stages. In order to consider the efficiency and execution time of the algorithm, our algorithm is based on a two-stage coordination algorithm. In this paper, we first define the node scales similar to other algorithm. But our algorithm only uses the two attributes of node's CPU and bandwidth of its surrounding links. In fact, these two attributes play the main role. In the fast mapping scheme, the result does not need to be optimal, but the mapping is completed as quickly as possible, so it is reasonable to use these two attributes. ISP(Internet service provider) wants the physical nodes mapped by the virtual nodes to be as close together as possible, because it can reduce operating expenses and decrease network propagation delay [13]. We use the soft distance to achieve this requirement, and the result is that revenue/cost ratio will be improved. For load balancing, the optimization of nodes and the optimization of links cannot be satisfied at the same time [7], so this paper only does load balancing optimization for link mapping.

III. VNE PROBLEM MODEL

According to [1], our model is based on the center static concise classification. Similar to many previous algorithmic models [11], [19], we define our VNE problem by using graph theory. If not specified, superscripts S and V represent the underlying network and virtual network respectively, embedding and mapping has same meaning, referenced resources equals to node scale, link rule is equivalent to mapping rule, the current virtual node represents the virtual node participating in the node mapping at now in this paper.

A. NETWORK MODEL

1) SUBSTRATE NETWORK

Generally speaking, the underlying network is a relatively dense network topology. It can be describe by undirected weighted connected graph $G^S = (N^S, E^S)$, where N^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $n^S \in N^S$ is attached to CPU resource, labeled as $c(n^S)$. For two given substrate nodes i and j, the unique substrate link $e^S(i,j) \in E^S$ connects them. The bandwidth resource of the $e^S(i,j)$ is denoted by $b(e^S)$. In particular, the link bandwidth resource exists in both endpoints, it is to say, $e^S(i,j)$ and $e^S(j,i)$ share the bandwidth resource $b(e^S)$.

We use the notation $P^S(s,t)$ to represent all the paths from substrate node s to substrate node t. A path $p^S \in P^S(s,t)$ has a capacity, named as $S_E(p^S)$. It is defined in equation (1).

$$S_E(p^S) = \min_{e^S \in p^S} b(e^S) \tag{1}$$

Given the substrate node i, the referenced resource is denoted by $R(i)$, which is defined in equation (2). It provides a criteria for matching substrate nodes fast.

$$R(i) = c(i) + \sum_{e^s \in L^S(i)} b(e^s) \tag{2}$$

where $L^S(i)$ is the set of all adjacent substrate links of substrate node i. In the latest papers, for $R(i)$, factors such as node degree, link stress and interface between nodes are often considered. In fact, the CPU resources of the nodes and the bandwidth of the surrounding links account for the main components. This paper mainly implements fast mapping, which is not necessarily optimal for the solution, equation (2) is in line with the requirements of this paper.

We define the distance of two given substrate nodes by Euclidean distance. It is denoted by equation (3).

$$Distance(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \tag{3}$$

where i and j are any two substrate nodes, and i_x, i_y, j_x, j_y are the coordinate of i and j.

Fig. 1 describes a simple example of virtual network embedding. It is consisted of two parts. The left part in picture

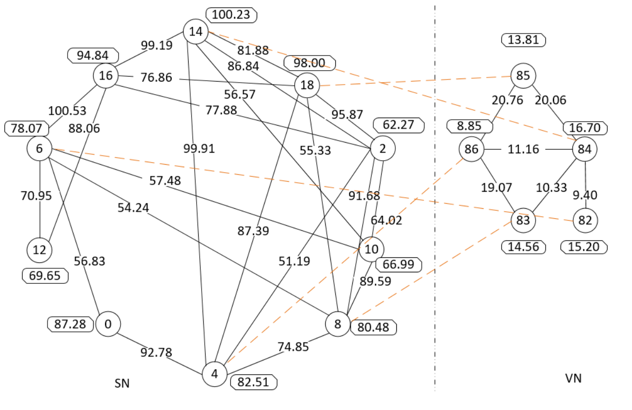

FIGURE 1. Virtual network embedding.

is the substrate network. Every substrate node has its label, for example, {0, 2, 4 ...}. The number in rectangle above node is its CPU resource. Every substrate link has a embedded number, which is its bandwidth resource. This topology is generated by Waxman model whose parameters are $\alpha = 0.1$, $\beta = 0.5$, because these parameters are consistent with the actual topology and more commonly used. An increase in the parameter α increases the probability of edges between any nodes in the graph, while an increase in β yields a larger ratio of long edges to short edges. Next we introduce the right part of picture.

2) VIRTUAL NETWORK

Similar to substrate network, we also use undirected weighted graph to characterize the virtual network by $G^V = (N^V, E^V)$, where N^V is the set of virtual nodes and E^V is the set of virtual links. Each virtual node $n^V \in N^V$ is associated with the CPU resource $c(n^V)$. Each virtual link $e^V(i,j) \in E^V$ between two virtual nodes i and j is attached to the bandwidth $b(e^V)$.

In Fig. 1, the right part is virtual network. The implication of its number is similar to the explanation of the substrate network. The model adopted is also Waxman model, and corresponding parameters are $\alpha = 0.5$, $\beta = 0.5$.

B. VIRTUAL NETWORK MAPPING

When a virtual request arrives or leaves, it needs to be mapped or freed. That is to allocate CPU resource and link bandwidth resource or revoke corresponding assigned resource in the substrate network. The whole process is completed in two coordinated stage, according to our AEF algorithm.

In Fig. 1, orange dotted lines denote node mapping procedure, the link mapping process is not special to show. For example, the figure depicts a node mapping and a link mapping. The node mapping is {85 -> 18, 86 -> 4, 84 -> 14, 83 -> 8, 82 -> 6} and the link mapping is {(85,86) -> (18,4), (85,84) -> (14,18), (84,86) -> (14,4), (83,86) -> (4,8), (83,84) -> (14,4)(4,8), (84,82) -> (14,16)(16,6)}.

1) NODE MAPPING STAGE

For each virtual request, a virtual node must be mapped to an unique substrate node. The node mapping function $F_N()$: $N^V \rightarrow N^S$ determines the assignment of virtual nodes.

$$F_N(M) \in N^S$$
$$F_N(M) \neq F_N(N), \quad \textit{if and only if } M = N$$
$$\arg\min_{r \in VRs} ExeTime({F_N}^r(\cdot)) \tag{4}$$
$$\text{subject to } c(M^V) \leq c(F_N(M^V)) \tag{5}$$
$$R(M^V) \leq R(F_N(M^V)) \tag{6}$$
$$b({e_i}^S) - \sum_{j=T_i}^{T_{i+1}} b({e_{k_j}}^V) \geq 0, \quad \exists k < |L^S(F_N(M))|, \quad \forall i \in [0,k] \tag{7}$$

where ${F_N}^r(\cdot)$ is node mapping for all node in one virtual request r, VRs is a set of all virtual requests. $ExeTime$ is a function of computation time that this phase will to take. k_j is the jth sequence subscript index with all links from $L^V(M)$ in full order, T_i is the interval sequence(subscript index) of selected elements(the virtual link after full permutation).

In above, equation (4) means that the node mapping time is as little as possible under the condition of satisfying the constraints. equation (5) and equation (6) indicate jointly that available CPU resource of the mapped substrate node $F_N(M^V)$ must be able to support CPU demand of the virtual node M^V. Equation (7) is the mapping rule in node embedding phase, which shows that the neighbor links' resource of the mapped substrate node $F_N(M^V)$ must can hold on all neighbor links' demand of the virtual node M_V. We must point out that all node CPU demand needs to be fulfilled, otherwise, this virtual request will be rejected directly.

2) LINK MAPPING STAGE

In this phase, every virtual link must be mapped to a shortest substrate path which satisfies the demand of this virtual link. The Link mapping function $F_L() : E^V \rightarrow P^S$ determines the assignment of virtual links.

$$F_L(e^V(M, M')) \in P^S(F_N(M), F_N(M'))$$
$$\arg\min_{r \in VRs} ExeTime({F_L}^r(\cdot)) \tag{8}$$
$$\text{subject to } b(e^V) <= S_E(p^S) \tag{9}$$

where ${F_L}^r(\cdot)$ is link mapping for all virtual links in one virtual request r, VRs is a set of all virtual requests. $ExeTime$ is a function to compute the time. Equation (8) shows that the link mapping time is as little as possible under the condition of satisfying the constraints, which is defined in equation (9). p^S is the shortest path in $P^S(F_N(M), F_N(M'))$. If p^S is empty, we reject this virtual request. Otherwise, we continue to process. If and only if all virtual links are embedded successfully, the link embedding stage is completed.

C. EVALUATION METRICS

In this paper, the most important and novel evaluation metric is the mapping time. We defined some metrics in equation (10) - equation (12).

$$TotalTime_k = \sum_{i=1}^{k} (ExeTime({F_N}^{r_i}(\cdot)) + ExeTime({F_L}^{r_i}(\cdot))) \tag{10}$$

where $TotalTime_k$ is the total time spent in previous k virtual requests embedding. r_i represents ith virtual request. In addition, the other important evaluation metric is the revenue to time, it shows how much revenue infrastructure provider can get.

$$Revenue(t) = \begin{cases} \sum c(n^V) + \sum b(e^V) & \text{mapped} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$RevenueToTime_k = \frac{\sum_{t=0}^{T} Revenue(t)}{T} \tag{12}$$

where $Revenue(t)$ is the revenue to map the virtual request at time t. If there is not virtual request at time t or this virtual request is not mapped successfully, the value is 0. T is current time, which has no scale and is just one unit. This unit can represent a minute, an hour, etc.

In order to measure the load of the underlying network links, we make a statistical analysis of the remaining bandwidths of all the substrate links, and use the standard deviation(denoted by Std) to measure whether the use of network bandwidth is balanced.

$$Std = \sqrt{\frac{\sum_{e^s \in E^S}(b(e^s) - x)^2}{n}} \tag{13}$$

where x is the average of all substrate link bandwidths.

The other two referenced evaluation metrics in this paper are the acceptance ratio and the cost to revenue ratio respectively. The detailed information can be seen in [1], [3], if you are interested in these respects.

IV. OUR AEF ALGORITHM

The proposed AEF algorithm is elaborated in this section. Our algorithm is based on heuristic search and aims to make mapping time as little as possible. The algorithm used in the node mapping phase of this paper looks like to other algorithms, such as [23], it adopts the measure of each node iteratively obtained by the page-rank algorithm, and then sorts according to this measure. However, in this paper, equation (2) is directly used for storing. In fact, there are many papers that use the page-rank algorithm, but they do not guarantee that the mapped results are optimal, and this process itself is time-consuming and not very suitable for time-sensitive occasions. At the same time, there is no clear improvement in the mapping cost. This paper considers the main factors of node mapping, and is complemented by the mapping rules and close mapping strategy to make node mapping fast and efficient. Firstly, we put forward innovative mapping rule to filter more redundant candidate nodes. Secondly the main algorithm is explained and described how to complete core mapping phase. We use some key strategies to speed up node mapping and link mapping. At last we analyze time complexity of the algorithm in many aspects. The detailed procedures are as following.

A. MAPPING RULE

A virtual node may be mapped to multiple candidate nodes in the substrate network. The actual number of candidate nodes are more than expected, even if the distance between substrate mapped nodes is required to be less than the given value DC(distance constraint) in our AEF algorithm. Therefore, how to filter these candidate nodes is a key issue. Generally speaking, the first step is to eliminate the substrate nodes that do not meet the CPU demand of the current virtual node. In fact, this method cannot work well in a resource-rich substrate network, although this way is also adopted in our algorithm. Thus, we propose a strategy to filter more redundant candidate nodes. It is a link rule, which means

Algorithm 1 Mapping Rule

```
Input: the virtual network $G^V$, the substrate network $G^S$, the
    virtual node $n^V$, the substrate node $n^S$
Output: the boolean result *mapped*
1: if $c(n^V) > c(n^S) || R(n^V) > R(n^S)$ then
2:     Return false
3: else
4:     Return whether or not equation (7) is staisfied
5: end if
```

that the adjacent links of the mapped substrate node must be able to accommodate all the adjacent links of the current virtual node. The detailed description refers to equation (7). If a substrate node satisfies this rule, it will be considered to be mapped. However, if there are multiple candidate nodes which satisfy link rule at the same time, we select the one with more referenced resource(defined in equation (2)).

Algorithm 1 implements the mapping rule in node embedding phase. The Mapping Rule procedure works as follows. Line 1 checks base conditions. If CPU resource or referenced resource(defined in equation (2)) of the mapped substrate node cannot accommodate demand of the current virtual node, we return false in Line 2, it means the mapped substrate node cannot match mapping rule. Otherwise, we judge this rule according to equation (7).

B. NODE MAPPING STAGE

In this phase, we use the mapping rule to consider the acceptance ratio of the subsequent link mapping and use the soft distance factor to get as many shorter paths as possible for the next stage.

1) Build Priority Queue: We construct two priority queues to host all substrate nodes and all virtual nodes respectively, named as pq^S and pq^V. All nodes are sorted by their referenced resource(detailed in equation (2)) in descending order.

2) Apply Soft Distance Factor: It must be noticed that the distance is not like other paper's concept. For example, in [11], [19], its distance constraint represents the relationship of virtual node and substrate node. However, our specified distance exists in substrate nodes. It is defined in equation (14).

$$DistanceFactor(i) = \frac{\sum_{n \in MN^S} Distance(i, n)}{|MN^S|} \tag{14}$$

where $Distance(i, n)$ is detailed in equation (3). MN^S is a collection of underlying nodes, which are mapped by the neighbor virtual nodes of the current virtual node. In general, the shorter the distance between the substrate nodes is, the smaller the path length between them is. Therefore, we can control the $DistanceFactor(i)$ to increase the cost to revenue ratio. In our algorithm, the default value of DC is 70.

$$DistanceFactor(i) \leq DC, \quad for\ i \in N^S$$

Soft Distance Factor is mainly to make all the underlying nodes mapped as close as possible. This goal has also been

Algorithm 2 Node Mapping

```
Input: the virtual network $G^V$, the substrate network $G^S$
Output: the boolean result $succ$
1:  build priority queue $pq^S$, $pq^V$
2:  while $pq^V \neq \emptyset$ do
3:      Pick out the first node from $pq^V$ named as $n^V$
4:      while $pq^S \neq \emptyset$ do
5:          Pick out the first node from $pq^S$ named as $n^S$
6:          Let $m$ = Mapping Rule($G^V$, $G^S$, $n^V$, $n^S$)
7:          if $m$ is true and $DistanceFactor(n^S) \leq D$ then
8:              Put them to nodeMapping Map
9:              Let $matched$ = true
10:             Break inner while loop
11:         end if
12:     end while
13:     if not $matched$ then
14:         Do not consider $DistaneFactor$, continue to repeat
            inner while loop. If we still cannot match the current
            virtual node, return false
15:     end if
16: end while
17: Return true
```

considered in other papers. In [5], authors use the BFS in the node mapping stage, taking the already mapped substrate node as the root node and performing hierarchical traversal to find the next substrate node that meets the constraints. In fact, the ultimate goal of this and that paper is to make a virtual link with fewer hops on the underlying path. However, this paper uses a heuristic method, which is based on the consideration of distance, the path length between the two substrate node that are closer is often shorter. The other algorithm uses the BFS, although the goal can also be achieved, but computation time will be greatly worse.

It is called soft distance because it is not required but optional. If we can find a substrate node which satisfies mapping rule above and distance factor simultaneously, it will be the best choice. Otherwise, the distance factor will be ignored to match substrate nodes at the second time.

The Node Mapping procedure works in the Algorithm 2 and the details are shown as follows. Line 1 builds two priority queues to save all substrate nodes and all virtual nodes respectively. Line 2 is outer loop, as long as the queue pq^V is not empty, we continue to execute, it will end until we process all virtual nodes. Line 3, we pick the element with more referenced resource since the pq^V has some elements. It says that we poll the first element named as n^V, because the two priority queues sort elements by their referenced resource value in descending order. Line 4 is inner loop, it checks whether the queue pq^S is empty. If pq^S has some elements, we pick out the first in Line 5 named n^S. Line 6 applies the mapping rule for n^V and n^S. If the mapping rule is matched, then we consider soft distance constraint(defined in equation (14)), which is required to be less or equal than DC. Once the two conditions are satisfied in Line 7, we complete node

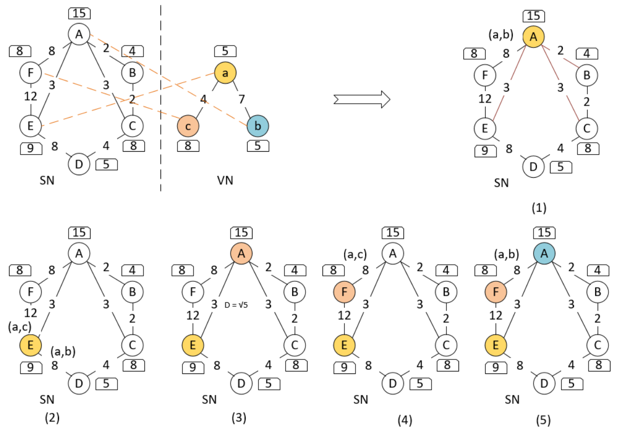

FIGURE 2. Node mapping.

mapping for the current virtual node and put the relationship into nodeMapping map in Line 8. We mark variable *matched* true and break inner loop in Line 9 - 10 at the same time. For a virtual node picked in Line 2, we cannot find any substrate node to match in Line 13, then we ignore soft distance constraint to find again from Line 4 - 12. If we cannot find mapped substrate node again, then we return false in Line 14. It means the link mapping is not successful. Otherwise, it works well in Line 17, we return true, which means the link mapping is completed successfully.

In Fig. 2, we make simply the position of the node as its coordinates. The upper case A - F represent substrate nodes and the lower case a - c denote virtual nodes. From substrate node A to substrate node F, their coordinates are $(1, 3)$, $(2, 2)$, $(2, 1)$, $(1, 0)$, $(0, 1)$, $(0, 2)$ repectively. According to Algorithm 2, we build a priority queue pq^S where all the substrate nodes are put. Then substrate nodes in pq^S are $\{A, E, F, C, D, B\}$. In addition, virtual nodes in pq^V are $\{a, c, b\}$. For line 2-3, we select virtual node a, then consider the substrate node A (seen at picture (1)) in line 4-5. When we check the Mapping Rule in line 6, (a,b) can be support by (A, F), but (a, c) cannot be satisfied by neighbor links (marked by brown line) of substrate node A. So we consider the next substrate node E (seen at picture (2)) in line 5. For substrate node E, (a, c) can be accommodated by (E, F) and (a, b) can be accepted by (E, D), therefore substrate node E satisfies the Mapping Rule. Notice we do not consider distance factor for the first mapped substrate node. We complete node mapping for virtual node a. Next, we pick out virtual node b to map in line 2-3. We also consider substrate node A firstly, because it has higher referenced resource in picture (3). We can verfy easily that the Mapping Rule is satisfied, but the distance from substrate node E to substrate node A is $\sqrt{5}$, which is large than DC(set to 1). Therefore we consider substrate node F in line 4-5 (seen in picture (4)) by skipping substrate node E, because substrate node E has been mapped. Obviously, substrate node F can satisfy all conditions. We complete node mapping for virtual node b. At last, we use the last node c in line 2-3. Except for substrate node E and substrate node F,

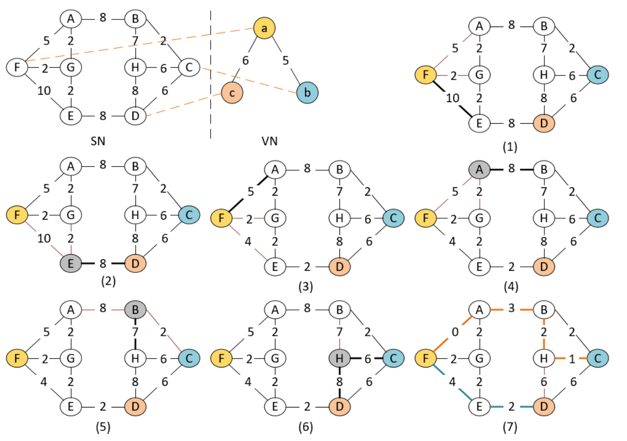

FIGURE 3. BFS baseline link mapping.

we cannot find any nodes which satisfy the distance factor, because the minimum distance is $\sqrt{2}$ which is large than DC. So we do not consider distance factor to execute again in line 4-12. Then we find substrate node A is a nice node which satisfies the Mapping Rule in picture (5). We complete node mapping for virtual node c.

C. LINK MAPPING STAGE

In this stage, we adopt a method to process virtual link mapping. This strategy guarantees the stability of our algorithm. Because the time complexity of traditional KSP(K-Shortest Path) algorithm [24] is related to k, there will be difference in the mapping time for every virtual link.

1) PRUNING THE SUBSTRATE NETWORK

According to equation (9), we can know that every substrate link resource in the final selected path must be more than the demand of the current virtual link. Therefore, we can pure some substrate links whose resource is less than the demand above.

2) FEASIBILITY ANALYSIS OF BFS

We use BFS strategy to find a shortest path from the start substrate node to the end substrate node, because BFS can find the actual shortest path in unweighted graph. The conclusion is explained at [24].

In Fig. 3, we ignore CPU resource and demand of nodes. A - H represent substrate nodes and a - c denote virtual nodes. Assume the node mapping is as shown in the figure. Then we prepare to start link mapping. For (a, c), we need to find a shortest path from substrate node F to substrate node D. In (1), we prune the neighbor links $\{(F, G), (F, A)\}$ of F, they are marked by thick brown line. The only link (F, E) which is marked by bold black line can support the demand of (a, c), so it can arrive next substrate node E, which is marked by gray circle. In (2), we prune the neighbor link (E, G), because its resource is less than demand. The only link (E, D) is eligible. Then it arrives at the destination substrate node D. So we find a path $(F, E)(E, D)$. We completed link mapping of (a, c). Before the next round, we need to update the bandwidth resource. In (3), we start to find another path from substrate node F to substrate node C. Because we update the network, we can only pass through (F, A). In (4), it arrives at A, then we prune link (A, G) and select (A, B) to go. In (5), it arrives at substrate node B. The link (B, C) will be removed, we pass through (B, H). In (6), it arrives at substrate node H. There is no link to be pruned, because all the neighbor links are suitable. We select (H, C) to reach the destination substrate node C and complete link mapping of (a, b). The corresponding shortest path is $(F, A)(A, B)(B, H)(H, C)$. Then we update bandwidth resource. In (7), we mark the corresponding path of link mapping with two colors.

Algorithm 3 Link Mapping Baseline

```
Input: the virtual network $G^V$, the substrate network $G^S$
Output: the boolean result $succ$
1: for all $e^V \in E^V$ do
2:     Prune the substrate network
3:     Use BFS to find a path $p$ satisfied equation (9) according to $e^V$
4:     if $p = null$ then
5:         Return false
6:     end if
7: end for
8: Return true
```

Algorithm 3 completes the link mapping stage by BFS. The Link Mapping procedure works as follows. Line 1 traverses all virtual links to complete mapping. Line 2 prunes some substrate links whose bandwidth resource is less than corresponding demand of the current virtual node. Line 3 uses BFS to find the shortest path from mapped start substrate node to mapped end substrate node. If we cannot find a path satisfied equation (9) in Line 4, it returns false in Line 5, which means the link mapping is unsuccessful. Otherwise, it works well in Line 8,and completes successfully the whole link mapping.

3) TRADE OFF LINK LOAD AND MAPPING COST

In fact, in the process of link mapping, the target is to use frequently the substrate link with higher bandwidth more than lower bandwidth ones. Because sensitive bandwidth links are reserved, more space can be provided for subsequent virtual request mappings, which indirectly improves the average acceptance ratio. To this end, we propose the following formula (define in equation (15)) to assign a cost value to each underlying link, and then the problem is converted to find a path with the minimum cost from the mapped start substrate node to the mapped end substrate node. We call it advanced strategy.

$$Cost(e^S) = \begin{cases} \frac{C}{b(e^S)}, & b(e^S) \geq d \\ Inf, & \text{else} \end{cases} \tag{15}$$

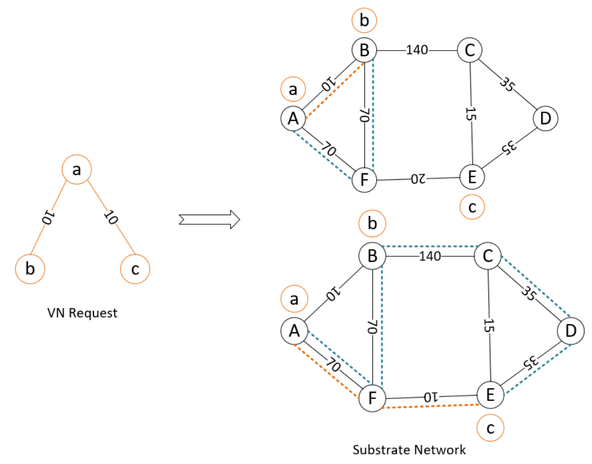

FIGURE 4. **Advance link mapping.**

where C is constant coefficient, its default value is 700. d is the current virtual link demand.

The aforementioned advanced strategy can ensure that the link with higher bandwidth is used as much as possible, but the path length found may be longer, which increases the cost of link mapping. In order to balance the link load and mapping cost, we need to ensure that the difference between the path calculated by the advanced strategy and the shortest path computed by BFS does not exceed SPL(Shortest Path Limit). If it exceeds, then we use the BFS strategy, otherwise we verify whether the path sought by the advanced strategy meets the requirements, and if it does, then we adopt the advanced strategy.

The notation of SPL can be understood as long as the difference between the path length and the shortest path length is less than this value is considered as an acceptable solution. It is an adjustable value that controls the balance between mapping cost and link load.

Fig. 4 is an example to explain the advanced strategy. In that Figure, the left part is a virtual request and the other is underlying network. The CPU resources or demands of the nodes are ignored. Assume that the mapping result of node mapping is $\{a \text{ -> } A,\ b \text{ -> } B,\ c \text{ -> } E\}$. The top right figure is the mapping of the virtual link ab. The orange-red dotted line is the path obtained using the BFS strategy and the blue dotted line is the path gotten by the advanced strategy. The corresponding path cost are $700/10 = 70$ and $700/70 + 700/70 = 20$ respectively. Since the difference between the two path length is 1, it is less than SPL(set to 2), we adopt the path (A -> F -> B). The bottom right of the figure is the mapping of the virtual link ac. As above, through calculation, we find that the cost of the orange-red dotted path is $700/70 + 700/10 = 80$ and the cost of the blue dotted path is $700/70 + 700/70 + 700/140 + 700/35 + 700/35 = 65$. However the difference between the two paths is 3, which is greater than the SPL. Therefore we adopt the path (A -> F -> E).

Algorithm 4 uses an advanced strategy to complete the link mapping, it trades off the link load and the mapping cost.

Algorithm 4 Link Mapping Advance

```
Input: the virtual network $G^V$, the substrate network $G^S$
Output: the boolean result $succ$
1:  for all $e^V \in E^V$ do
2:      Use BFS to find a path $p1$ satisfied (9) according to $e^V$
3:      Assign cost value to substrate links by equation (15)
        and Use Dijkstra to find a path $p2$ based on substrate
        link cost value.
4:      if $p1 = null$ then
5:          Return false
6:      else if $p2 \,!= null \ \& \ len(p2) - len(p1) \le SPL$ then
7:          Adopt $p2$
8:      else
9:          Adopt $p1$
10:     end if
11: end for
12: Return true
```

The Link Mapping Advance procedure works as follows. Line 1 and Line 2 find the two paths by the BFS strategy and advanced strategy respectively. Line 3 - 9 determines which path to accept. It considers link load based on the original algorithm and ensures that the minimum bandwidth in the underlying network is relatively large and introduces the concept of SPL to make mapping cost not too high.

D. TIME COMPLEXITY ANALYSIS

In this section, we will analyze the time complexity of the algorithm in detail.

1) AVERAGE TIME COMPLEXITY

Analysis of average time complexity needs to be based on a specific topology, we use Waxman model in this paper. The probability of edge is $prob$ defined as following.

$$prob = \alpha \cdot \exp^{-\frac{d(v,w)}{\beta \cdot L}} \tag{16}$$

where L is the maximum distance between all nodes.

Proposition 1: When $|N^S|$ is large enough, L in equation (16) *is close to $\sqrt{2}$, then $prob \approx \alpha \cdot \exp^{-\frac{1}{2\sqrt{2}\beta}}$*

A proof of Proposition 1 is provided in Appendix A.

Firstly, we compute the referenced resource(seen in equation (2)) for all substrate nodes and virtual nodes. In this process, we need to compute node CPU resource or demand and its neighbor links bandwidth resource or demand for every node. The time complexity is $\sum_{n^S \in N^S}(1 + L^S(n^S))$ for the substrate network, the same as the virtual network. Therefore the time complexity is approximate to $O(prob \cdot |N^S|^2)$.

Secondly, in Node Mapping Algorithm, line 2 and line 4 are two loops, every loop completes some operations at line 5-10. The main points of these operations are Mapping Rule Algorithm. It needs to verify equation (7), this process needs to put neighbor links of the substrate node and the virtual node to priority queue, it involves link sort. As a result, the time complexity is $O(prob \cdot |N^S| \cdot \log(prob \cdot |N^S|))$. In additional, in inner loop at line 4, the actual repeated time is k, where $k \leq |N^S|$.

Definition 1: The probability that substrate node satisfies the referenced resource condition is called Pr.

Definition 2: The probability that the distance of two substrate node less than DC is called Pd.

Proposition 2: Assume the substrate network has enough links and nodes, and the resources are randomly distributed. Then the actual number of executions in Algorithm 2 line 4 is $k = \min\{\frac{1}{Pr \cdot Pd}, |N^S|\}$.

A proof of Proposition 2 is provided in Appendix B.

Therefore, the average time complexity of Node Mapping Algorithm is $O(k \cdot |N^V| \cdot prob \cdot |N^S| \cdot \log(prob \cdot |N^S|))$.

Thirdly, in Link Mapping Baseline Algorithm, line 1 is only outer loop. The inner operation is mainly BFS. Because the prune operation can be combined into BFS process. BFS needs to traverse all substrate links. Thus, the average time complexity of this stage is $O(|N^V|^2 \cdot |N^S|^2)$. In fact, the Link Mapping Advance Algorithm has one more step than the Link Mapping Baseline Algorithm to find the path using Dijkstra, the average time complexity in this stage remains the same.

In summary, the average time complexity of whole algorithm is $O(prob \cdot |N^S|^2 + k \cdot |N^V| \cdot prob \cdot |N^S| \cdot \log(prob \cdot |N^S|) + |E^V| \cdot |E^S|)$, where $|E^V| = \frac{prob \cdot |N^V|^2}{2}$, $|E^S| = \frac{prob \cdot |N^S|^2}{2}$.

2) WORST TIME COMPLEXITY

We combine with the analysis of average time complexity, then we talk the worst time complexity of our algorithm. In fact, we just let $prob = 1$ and $k = |N^S|$ to conclude that the worst time complexity of AEF algorithm is $O(|N^S|^2 \cdot |N^V| \cdot (\log(|N^S|) + |N^V|))$, which applies to all topological networks.

E. PERFORMANCE ANALYSIS

Our algorithm has three main advantages(Execution Time, Acceptance Ratio and Link load) and performs well in other aspects.

1) EXECUTION TIME

Real time is the main goal of our algorithm. We optimize it in both node mapping stage and link embedding stage. In node mapping phase, we consider the referenced resource of node to match substrate node by L2S2 strategy, which is simple and fast. In link embedding stage, we use BFS to find a shortest path instead of KSP algorithm, because the latter is instable, we cannot know the suitable value of parameter k. For link mapping algorithm by BFS, the time complexity is $O(|E^V| \cdot |E^S|)$, however KSP is $O(k \cdot |E^V| \cdot |V^S|^2)$. High performance is referred to simulation results in next section.

2) ACCEPTANCE RATIO

This evaluation metric is very important to infrastructure provider. It is also the core point to judge the quality of an algorithm. In our algorithm, we consider to improve this evaluation metric in both two stages. In node mapping phase, we use Mapping Rule to ensure the mapped substrate node has more ability to complete subsequent link embedding. In link mapping stage, we use BFS to find path, it will find a shortest path as long as there exists some paths from the mapped start substrate node to the mapped end substrate node. Thus, AEF algorithm can improve greatly the acceptance ratio. This is also verified in Section IV.

3) NETWORK BANDWIDTH STANDARD DEVIATION(STD)

The cost of link setting in this paper is inversely proportional to the remaining bandwidth. The link with larger remaining bandwidth is selected first. Therefore, the standard deviation between all substrate link remaining bandwidths will be relatively small after link mapping, which can be clearly seen from the comparison between the AEF_Baseline algorithm and AEF_Advance algorithm in next section.

V. SIMULATION EXPERIMENT AND ANALYSIS

A. SIMULATION ENVIRONMENT DESCRIPTION

In the simulation, we adopt Waxman model to produce substrate network. We set the parameters $\alpha = 0.5, \beta = 0.1$. The environment settings are like [25]. The detailed parameter settings of substrate network is referred to Table 1 which describes some parameters for substrate network.

TABLE 1. Substrate network parameter settings.

Substrate Nodes	100
CPU Resource	U[50, 100]
Bandwidth Resource	U[50, 70]

For each virtual network, we also use Waxman model to generate it. We set the parameters $\alpha = 0.5, \beta = 0.25$. We make requested virtual networks arrive obey the Poisson distribution and set virtual networks arrival rate 10 very 100 time units. In addition, the lifetime for each virtual network obeys an exponentially distribution with a mean of 1000 time units. We run our simulation for 2000 time units. The detailed parameter settings of virtual network are referred to Table 2 which lists some parameters for virtual network.

TABLE 2. Virtual network parameter settings.

Virtual Nodes	U[5, 10]
CPU Demand	U[5, 20]
Bandwidth Demand	U[5, 20]

B. SIMULATION RESULT

In this subsection, we conduct the simulation experiment and show the results. The algorithms to be compared are

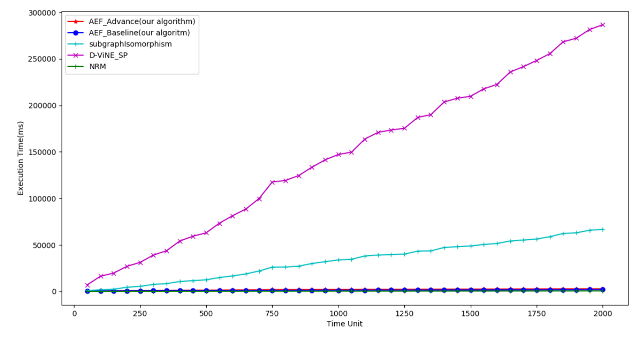

FIGURE 5. Total execution time.

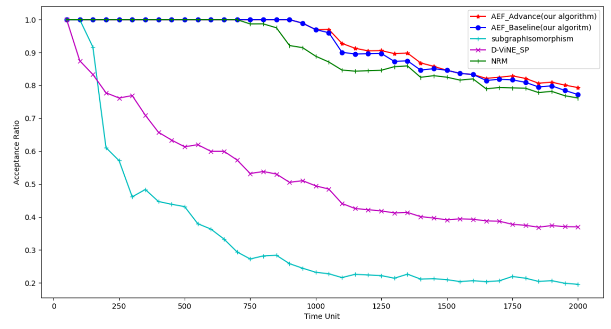

FIGURE 6. Average acceptance ratio.

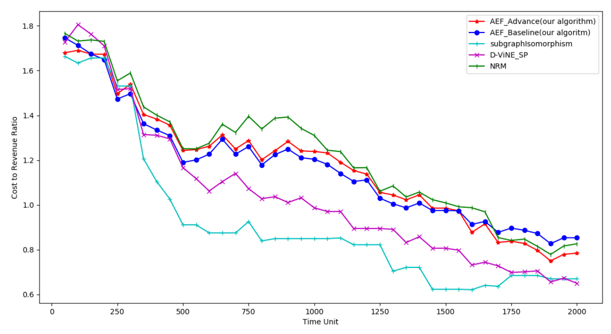

FIGURE 7. Average cost to revenue ratio.

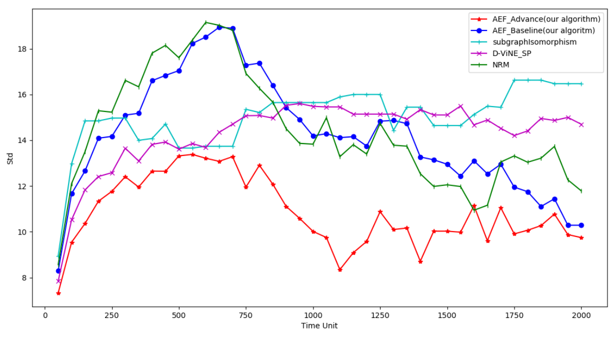

FIGURE 8. Substrate link bandwidths standard deviation.

TABLE 3. Compared algorithm description.

AEF_Baseline	our algorithm uses BFS strategy to complete link mapping
AEF_Advance	our algorithm uses Advanced strategy to complete link mapping
subgraphIsomorphism	One-stage algorithm based on graph matching
D-ViNE_SP	Deterministic Rounding Based Virtual Network Embedding Algorithm uses the shortest path algorithm to complete link mapping
NRM	Algorithm in latest paper is similar to greedy algorithm

NRM algorithm [26], subgraphIsomorphism algorithm and D-ViNE_SP algorithm. The detailed explanation refers to Table 3.

As is shown in Fig. 5, the curve is very steep for subgraphIsomorphism and D-ViNE_SP. They are often time-consuming in the process of mapping. The curve describing our algorithm is almost straight and the execution time is similar to NRM. NRM is a greedy algorithm without other optimization methods, so the execution time is the fastest. Its total execution time is 660ms. In the two versions of our algorithm, since the advanced strategy will call Dijkstra algorithm one more time during link mapping, the execution time will be longer than the baseline algorithm. For our algorithm, the execution time is relatively short, which is basically on the same order of magnitude as NRM. At last, our algorithm's execution time is 3.02% of subgraphIsomorphism and is 0.7% of D-ViNE_SP. The results imply that our algorithm has less execution time and is suitable for applications with real-time requirements.

As is show in Fig. 6, we analyze three algorithms, NRM, AEF_Advance, AEF_Baseline firstly. These three algorithms all maintain 100% acceptance ration in the early stage. Because the link node resources of the underlying network are sufficient in that stage, and generally they can be mapped successfully. With the arrival and completion of a large number of virtual requests, the underlying network resources have become scarce since then, which has led to a decline in the acceptance ration. At the same time, we can see that the AEF_Baseline has a higher acceptance ration than NRM. This is because AEF_Baseline considers mapping rules, called 1-lookhead, which can avoid virtual nodes mapping to wrong substrate nodes in advance. In addition, the AEF_Advance has a higher acceptance ration than the AEF_Baseline. This is because the former has optimized the link load, making the link bandwidth more balanced. Then we look at the other two algorithm, subgraphIsomorphism and D-ViNE_SP. These two algorithms show a rapid decline in the most of time and obvious low acceptance ratio. At last, our algorithm's acceptance ratio is 4.16 times that of the subgraphIsomorphism algorithm and is 2.13 times that of the D-ViNE_SP algorithm.

Fig. 7 shows that subgraphIsomorphism and D-ViNE make a better use of resources while bringing more benefits to InP. Meanwhile, our algorithm's performance in this regard is relatively well, although there are still some gaps between the above two algorithms. We mainly look at the remaining three algorithms. Most of the time, our algorithm performs better than the NRM in this respect, because our algorithm uses the soft distance factor in node mapping stage, ensuring that the mapped nodes are as close as possible, that is, the mapped path will be decrease accordingly.

In Fig. 8, the smaller the *Std* value, the more balanced the corresponding link load. It shows that AEF_Advance makes underlying link bandwidths more balanced, which is significantly better than other algorithms, because other algorithms do not consider link load optimization. At the same time, the two algorithms, AEF_Advance and AEF_Baseline, should be noticed. The difference between them is that the former considers the link load during link mapping phase, and uses the inverse function of remaining bandwidth of the link as link cost to find a backup path, so that the substrate link with the larger bandwidth is more likely to be selected, and the statistical standard deviation for the bandwidth will be smaller.

VI. CONCLUSION

To accelerate the mapping time and guarantee high performance, we propose the AEF algorithm. AEF is two coordinated stages algorithm, it is proved to solve the VNE issue in polynomial time, its time complexity is analyzed to O($|N^S|^2 \cdot |N^V| \cdot (\log(|N^S|) + |N^V|)$). Simulation Results show that our algorithm is faster than most other algorithms and ensures the overall better efficiency, especially in the acceptance ratio and link load. In addition, our algorithm's cost to revenue ratio is similar with the other compared algorithms. Overall, our algorithm can accelerate greatly mapping procedure and improve the average acceptance ratio under the precondition of guaranteeing the average revenue and link load on the whole.

APPENDIXES

APPENDIX A
PROOF PROPOSITION 1

Proof: Node v and Node w are randomly distributed at square with sides that are 1 unit in length. Therefore it equals to solve following formula.

$$\begin{aligned}
&\int_0^1\int_0^1\int_0^1\int_0^1 \sqrt{(v_x-w_x)^2+(v_y-w_y)^2}dw_y dv_y dw_x dv_x \\
&= \int_0^{\infty}\Big(\frac{erf(x)}{3x}\exp^{-x^2}+\frac{4}{15\sqrt{\pi}}\exp^{-2x^2} \\
&\quad +\frac{4}{15\sqrt{\pi}}\exp^{-x^2}\Big)dx \\
&= \frac{\sqrt{2}+2+5\sinh^{-1}(1)}{15} \\
&\approx 0.521 \qquad (17)
\end{aligned}$$

$$prob \approx \alpha \cdot \exp^{-\frac{1}{2\sqrt{2}\beta}} \qquad (18)$$

□

APPENDIX B
PROOF PROPOSITION 2

Proof: Assume the resource of substrate network obeys to U(0, γ), the demand of virtual network obeys to U(0, 1), it says the ratio is γ, which is constant value. Because substrate network owns more nodes and links, according to central limit theorem, every substrate node referenced value obeys to normal distribute, where $\mu = prob \cdot |N^S| \cdot \gamma$ and $\sigma^2 = \frac{prob \cdot |N^S| \cdot \gamma^2}{12}$.

$$f(x) = \frac{\sqrt{6}}{\gamma \cdot \sqrt{\pi \cdot prob \cdot |N^S|}}\exp^{-\frac{6(x-prob\cdot|N^S|\cdot\gamma)^2}{prob\cdot|N^S|\cdot\gamma^2}} \qquad (19)$$

For every virtual node, the average referenced resource is $\frac{prob \cdot |N^V|}{2}$. Therefore the probability that we select randomly one substrate node whose referenced resource is great than one virtual node is defined as following.

$$Pr = 1 - \Phi(\frac{prob \cdot |N^V|}{2}) \qquad (20)$$

The probability distribution of two distance is defined in (21).

$$\begin{aligned}
f(l) &= \begin{cases} 2l(l^2-4l+\pi), & 0 < l \le 1 \\ 2l(-2-\pi-l^2 \\ \quad +4\arcsin(\frac{1}{l})+4\sqrt{l^2-1}), & \text{note} \end{cases} \\
Pd &= \int_{-\infty}^{DC} f(l) \qquad (21)
\end{aligned}$$

where note is expressed by $1 < l \le \sqrt{2}$, DC is constant value and denotes the constraint to average distance of some substrate nodes. If the substrate node satisfies the referenced resource, it equals to satisfy the Mapping Rule. Therefore, we conclude following formula.

$$k = \min\{\frac{1}{Pr \cdot Pd}, \ |N^S|\} \qquad (22)$$

□

REFERENCES

[1] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[3] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of VNE: A survey," *China Commun.*, vol. 13, no. 6, pp. 48–62, Jun. 2016.

[4] M. Lu, Y. Lian, Y. Chen, and M. Li, "Collaborative dynamic virtual network embedding algorithm based on resource importance measures," *IEEE Access*, vol. 6, pp. 55026–55042, 2018.

[5] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on the degree and clustering coefficient information," *IEEE Access*, vol. 4, pp. 8572–8580, 2016.

[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[7] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. 25TH IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2006, pp. 1–12.

[8] S. Zhang, J. Wu, and S. Lu, "Virtual network embedding with substrate support for parallelization," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2615–2620.
[9] A. Razzaq, P. Sjodin, and M. Hidell, "Minimizing bottleneck nodes of a substrate in virtual network embedding," in *Proc. Int. Conf. Netw. Future*, Nov. 2011, pp. 35–40.
[10] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 783–791.
[11] H. Cao, Y. Guo, S. Wu, Z. Qu, H. Zhu, and L. Yang, "Location aware and node ranking value driven embedding algorithm for multiple substrate networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.
[12] H. Cao, L. Yang, and H. Zhu, "Novel node-ranking approach and multiple topology attributes-based embedding algorithm for single-domain virtual network embedding," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 108–120, Feb. 2018.
[13] Q. Zheng, J. Li, H. Tian, Z. Wang, and S. Wang, "A 2-layers virtual network mapping algorithm based on node attribute and network simplex," *IEEE Access*, vol. 6, pp. 77474–77484, 2018.
[14] F. Bianchi and F. L. Presti, "A Markov reward model based greedy heuristic for the virtual network embedding problem," in *Proc. IEEE 24th Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Sep. 2016, pp. 373–378.
[15] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.
[16] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.
[17] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding LC-VNE algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw*, vol. 24, no. 6, pp. 3648–3661, Dec. 2016.
[18] T.-H. Lee, S. Tursunova, and T.-S. Choi, "Graph clustering based provisioning algorithm for virtual network embedding," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 1175–1178.
[19] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.
[20] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.
[21] H. Yao, B. Zhang, P. Zhang, S. Wu, C. Jiang, and S. Guo, "RDAM: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding," *IEEE Trans. Emerg. Topics Comput.*, early access, Sep. 20, 2018.
[22] H. Yao, X. Chen, M. Li, P. Zhang, and L. Wang, "A novel reinforcement learning algorithm for virtual network embedding," *Neurocomputing*, vol. 284, pp. 1–9, Apr. 2018.
[23] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology awareness and optimization," *Comput. Netw.*, vol. 56, no. 6, pp. 1797–1813, Apr. 2012.
[24] T. H. Cormen, C. Stein, R. Rivest, and C. Leiserson, *Introduction to Algorithms*, 3rd ed. New York, NY, USA: McGraw-Hill, 2009, pp. 594–601.
[25] G. Wang, Z. Zhao, Z. Lu, Y. Tong, and X. Wen, "A virtual network embedding algorithm based on mapping tree," in *Proc. 13th Int. Symp. Commun. Inf. Technol. (ISCIT)*, Sep. 2013, pp. 243–247.
[26] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on computing, network, and storage resource constraints," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3298–3304, Oct. 2018.

DEDONG HU received the B.S. degree from the Beijing University of Posts and Telecommunications (BUPT), in 2017, where he is currently pursuing the master's degree in computer science technology. He has developed several software packages in Java. His research interests are in computer networks and new data mining.

ZHEN YANG received the B.S. degree in mathematics from Xiangtan University, in 1997, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, China, in 2007. He is currently an Associate Professor with the Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications. He has published more than 30 articles. His current researches focus on multimedia systems and networking, edge computing, and cloud computing.