An enhanced algorithm based on paths algebra strategy to solve the VNE problem

Canhui WANG, Fangjin ZHU*, Qijia ZHANG
School of Computer Science and Software Engineer, Shandong University, Jinan, PR China
Corresponding author: zhufj@sdu.edu.cn

***Abstract*—Network virtualization is widely considered to be one of the main paradigms to solve the Internet ossification problem. One of the most challenging works in this paradigm will be the efficient use of the substrate resources, which is known as virtual network embedding (VNE) problem. The VNE problem is known to be an *NP-hard* problem which needs some heuristic and approximate algorithms to be solved. In this paper, the VNE is decomposed into two stages: virtual node and virtual link mapping. In the node mapping stage, the breadth first search algorithm is employed to construct loop-free tree for each substrate node, then we explored different hops sufficient capacity of each SN node via the loop-free tree, and analyze the best value of hops that contributes to gain better performance for the algorithm. And the link mapping stage, it can be seen as multi-constraint routing, which is known to be an *NP-hard* problem. The paths algebra framework was adopted to address this problem, for its convergence for multi-constraint routing problem and a large solution space that the algorithm can gain an enhanced optimization performance.**

Keywords—network virtualization; virtual network embedding; sufficient capacity; multi-constraint routing; paths algebra framework

I. INTRODUCTION

An increasing numbers of applications with various Quality of service (QoS) have been developed on the Internet. However, the problem is that, it is difficulty for the traditional TCP/IP to support the different data delivery mechanism for these QoS requirements, which is known as Internet ossification [1]. Network virtualization is widely considered to be one of the main paradigms to solve the Internet ossification problem. One of the most challenging works in this paradigm will be the efficient use of the substrate resources, which is known as virtual network embedding (VNE) problem.

The VNE problem includes both node mapping and link mapping stage. It is well known that the problem of *NP-hard* multiple separator can be reduced to the VNE problem [2]. Therefore, the VNE problem is known to be an *NP-hard* problem, and both node and link mapping can be an *NP-hard* problem too. So, solving this problem needs some heuristic and approximate algorithms.

In the node mapping stage, several related works [3-6] have been proposed. In these works, the cost of VN request is generally gained by the sum of CPU costs of peer nodes mapped and bandwidth costs of all mapped paths. A common node-mapping solution is to find the substrate nodes that are closer to each other, and search the shortest path between the peer nodes mapped [7-8], which indicates that we need to find some sufficient capacity nodes which have more neighboring nodes and links, in other words, more CPU and bandwidth resources in pursuit of more opportunities and load balances to embed the future VN request. However, most of the paper proposed [3-6] did not consider the influence of different hops and definition for each substrate node sufficient capacity on the performance of node mapping, which would affect the efficiency of link mapping.

In the link mapping stage, there are two methods that have been proposed to solve the VNE problem: single-path and multi-path approaches [9]. The multi-path approaches indicate that the virtual link request can be split into the substrate possible paths, which can reduce the computational complexity of the VNE problem. However, this approach causes some problem such that, the VN Providers may find it hard to discover and select the available substrate resources for each VN request for splitting with limited knowledge of substrate topology and resource. And the common approach for VN request splitting is based on a highly abstract view of the substrate network topologies, which can result in inefficient embedding [10]. Considering the routing protocol used, or the operability of managing virtual request, therefore, the single-path virtual link mapping approach is adopted in this paper. The virtual link mapping stage can be seen as multiple multi-constraint routing problems [9]. However, the multi-constraint routing is *NP-hard* problem [14]. Many solutions proposed have been verified that their merits restricted to a universe whose validity is often defined either by the size of the networks or by their topology, and their intuitive portability does not work for other applications [9]. Analyzing this problem, we need to design an algorithm to ensure the routing convergence for different types of QoS metrics or QoS metrics composition, in which this problem can be addressed by the paths algebra framework [16], which allow validating the proposed solutions independently from network topology or implementation details [9].

In this paper*, our work is to explore different hops sufficient capacity of each SN node via the loop-free tree, and analyze the best value of hops that contributes to gain better performance for the node mapping algorithm, which directly affects the efficiency of link mapping.

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

The link mapping stage, it can be seen as multi-constraint routing, which is known to be an *NP-hard* problem. The paths algebra framework was adopted to address this problem, for its convergence for multi-constraint routing problem and a large solution space that the algorithm can gain an enhanced optimization performance. Our work is to find the optimization solution for each virtual request via the paths algebra, the optimal objectives would be discussed in section III.

After this introduction, section II shows the VNE background and some notations. The algorithm based on paths algebra to solve the VNE problem includes two stages: virtual node and virtual link mapping. The node mapping will be discussed in Section III, link mapping will be discussed in Section IV. Section V presents the simulation results and analysis. Section VI concludes the paper and indicates the future work.

II. Background and Notations

The VNE problem describes how to map the virtual requirements onto the substrate network. The virtual request's nodes and links are annotated with the value of the corresponding request, and the substrate nodes and links are annotated with the resource value.

A. Background and network notations

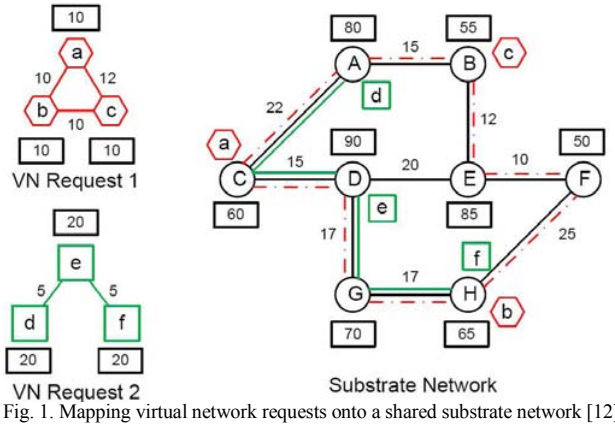

Fig. 1. Mapping virtual network requests onto a shared substrate network [12].

Let $G=(V,E)$ be a weighted and undirected graph, where V is the set of nodes, and $E \in V \times V$ is the set of links. Substrate Network is modeled as $G^S=(V^S,E^S)$ and Virtual Network is modeled as $G^R=(V^R,E^R)$. The available CPU resource of substrate node v is defines as $CPU(v)$. Similarly, the available bandwidth resource of the substrate link (v,u) is defined as $BW(v,u)$. The symbol of $\rho(v)$ is defined as the Euclidean distance between one node and its neighbors, $Length(v,u)$ is defined as the value of hops length between node v and u. And the notation $Stress(v)$ indicates the number of virtual requires that have mapped onto the substrate node v. Fig.1 shows the mapping results of VN request 1 and 2, for VN request 1, the virtual node a, b and c is mapped to the substrate node C, H, B. The virtual link (a, b), (a, c), (b, c) is mapped to substrate link (C, D, G, H), (C, A, B), (B, E, F, H). The CPU resources of these substrate nodes satisfy the CPU request of virtual nodes and the bandwidth resources of these substrate links satisfy the demand of virtual links.

III. Node Mapping

A. Definition of sufficient capacity

As far as node mapping, when ρ =1 hop, the sufficient capacity for each substrate node can be considered as a star topology, which is simple and has none loops. However, when ρ is greater or equal to 2, the loops for the topology of ρ hops sufficient capacity for each substrate node must be taken into account, for the loops in SN will confuse the definition of ρ hops sufficient capacity of each substrate node. So, taking one of the substrate node B's sufficient capacity given in Fig.1 as an example, the ρ =1 neighboring nodes A and E have a greater influence on node B's sufficient capacity than the ρ =2 neighboring nodes C, D and F. In other words, the nodes have smaller influence on root sufficient capacity, when they are far away from the root node. Therefore, the breadth first search algorithm to construct loop-free tree, and nodes in different tree level mean different types of influence they have.

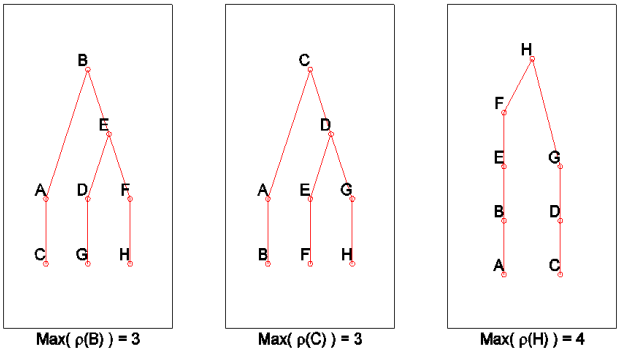

Fig. 2. BFS tree of substrate node given in Fig.1.

Fig.2 shows us the BFS tree of substrate node B, C and H given in Fig.1, In terms of network topology and the richness of CPU and bandwidth resources, the sufficient capacity of each substrate node is diversity, so we need some methods to measure the sufficient capacity for each substrate node.

The relationship strength coefficient reflects the relationship between the root node and its ρ -hops neighbor nodes, where ρ can take the values of 1, 2, 3, etc. With the increase of the height of the BFS tree, the number of the ρ hops neighboring nodes shows an exponential growth, while the relationship between the ρ hops neighboring nodes and the root node becomes weaker and weaker, therefore, the relationship strength coefficient is defined as:

$$\underset{v,u \in V^S}{R}(v,u) = e^{-(\kappa * Length(v,u)+\lambda)} \quad (1)$$

The values of κ and λ show the decrease rate of the relationship strength between the root node and its ρ hops neighbor nodes. Therefore, the sufficient capacity describes the rich-

ness of CPU and bandwidth for each substrate node, which can be described as:

$$\underset{v\in V^S}{S}(v) = \sum_{\forall u\in Set\_of\_v's\_Neighboring\_Nodes} R(v,u).CPU(u).BW(v,u) \quad (2)$$

The formula (2) shows the sufficient capacity of certain substrate node, which indicates the larger value of this formula, the richer sufficient capacity that the substrate node may has.

B. An enhanced greedy node mapping algorithm

In this subsection, we proposed an enhanced greedy node mapping algorithm that exploits the sufficient capacity from SN. Our algorithm collects a group of virtual requests during a time and then tries to map the virtual requests onto the substrate network that satisfies some constrains, such as VR's CPU and bandwidth demand. ***The enhanced greedy node mapping algorithm*** is presented as follows.

For a given virtual request

Step1: Sort the virtual nodes according to their *sufficient capacity* and put them into the queue Q in descending order.

Step2: If Q$==\Phi$, stop here.

Step3: Else, take one virtual node which has the largest sufficient capacity.

Step4: Sort the substrate nodes according to their *sufficient capacity,* and put them into the queue X in descending order.

Step5: If X$==\Phi$, stop here.

Step6: Else, take next substrate node from X.

Step7: Find the substrate node with the largest sufficient capacity, and satisfying the restrictions (substrate node's available CPU capability>=virtual node CPU request). If true, map the virtual node and refresh X, Q, then, GOTO Step2.

Step8: Else, GOTO Step3.

The enhanced node mapping algorithm proposed exploits the sufficient capacity from SN, and tries to map the virtual nodes onto the substrate nodes which have the largest sufficient capacity. By this means, the algorithm tends to balance the substrate resource use, which helps to accept more future VRs.

IV. LINK MAPPING

After the node mapping stage, we need to map the virtual links onto the substrate network. The virtual link mapping stage of the VNE problem can be seen as multiple-constraint routing problems. The paths algebra framework provides a flexible framework to solve this problem. Additionally, contrary to shortest path based mono-constraint approaches, the paths algebra is based on multi-constraint routing algorithm, it provides the capability of finding all possible paths between each pair of substrate network, which indicates that the algorithm based on the paths algebra framework will have a large enough solution, where the algorithm based on this framework can gain an enhanced optimization performance.

A. The paths algebra framework

The paths algebra framework has been discussed in the previous paper [17], which is presented by means of the following examples.

1) Path characterization

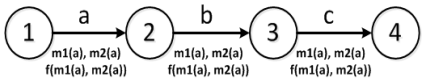

Fig. 3. (a) Example of a simple path [17].

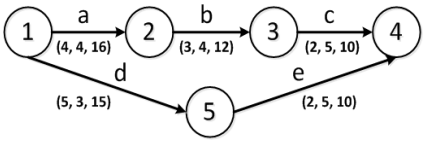

Fig. 3. (b) Example of two paths to be ordered [17].

Let $G' = (V', A')$ be a weighted and directed graph, where V' is the set of nodes, and A' the set of arcs. As presented in Fig.3.(a), the set of nodes is given by $V' = \{1,2,3,4\}$ and the set of arcs is given by $A' = \{a,b,c\}$. The source and destination nodes are $(s,d) = (1,4)$. This path can be represented either as a succession of vertices $p_{1,4}$ or as a succession of arcs $p_{a,c}$.

Each arc in this example is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where $m_1(x)$ and $m_2(x)$ are the values of metrics m_1 and m_2 on the arc $x \in A'$; $f[m_1(x), m_2(x)]$ is a function of combination of metrics applied to the metrics of $m_1(x)$, $m_2(x)$.

In general, the paths algebra uses M as the set of m adopted routing metrics and F as the set of k metrics combination.

The set of combined-metrics of all edges is given by

$$\overline{C}(p_{a,c}) = \begin{bmatrix} \overline{C}_a \\ \overline{C}_b \\ \overline{C}_c \end{bmatrix} = \begin{bmatrix} m_1(a) & m_2(a) & f[m_1(a), m_2(a)] \\ m_1(b) & m_2(b) & f[m_1(b), m_2(b)] \\ m_1(c) & m_2(c) & f[m_1(c), m_2(c)] \end{bmatrix}$$

A synthesis $\overline{S}[.]$ is a set of binary operations applied on the values of the links combined-metrics along a path to obtain a resulting value that characterizes this path as far as the constraint imposed by the combined-metric is concerned. So far, the syntheses are restricted to the following set: $\{add(), mult(), \max(), \min()\}$.

If the routing algorithm is mono-constraint, only one value is obtained as the synthesis result and it is called weight-word. If the routing algorithm is multi-constraint, with k constraints, then k values are obtained. In this example, $\overline{S}[.] = [S_1 S_2 S_3]^t$.

The weight-word has as many letters as the path's number of arcs. The first letter corresponds to resulting value of the synthesis applied to the whole path; the second letter corresponds to resulting value of the synthesis applied to the subpath obtained by dropping out the last arc; the last letter corresponds to the resulting value of the synthesis applied to the subpath made of only the first arc. Any number of letters can be retained as the synthesis result and this is called an abbreviation: $\overline{b}_j(\overline{S}[.])$ represents a j -letters abbreviation; $\overline{b}_\infty([S.])$ represents no abbreviation, i.e., all letters are taken into account.

2) Path ordering

Consider the network represented in Fig.3.(b), where two paths connect the source node 1 to the destination node 4. These paths are $\alpha = (1,2,3,4) = (a,b,c)$ and $\beta = (1,5,4) = (d,e)$. Each path's arc is characterized by a triple $(m_1(x), m_2(x), f[m_1(x), m_2(x)])$, where $f[m_1(x), m_2(x)] = m_1(x) \times m_2(x)$. The syntheses to be used in this example are given by $\overline{S}[.] = [\min() \max() add()]^t$.

Table I. Synthesis result of the network

Path	S_1 min	S_2 max	S_3 Add
α	2; 3; 4	5; 4; 4	38; 28; 16
β	2; 5	5; 3	25; 15

The result of the synthesis is shown in Table I. A path α is worse or less optimized than a path β, if $\overline{S}[\alpha] \preceq_{ML} \overline{S}[\beta]$, where $\preceq_{ML}$ stands for multidimensional lexical ordering. In the example $\preceq_{ML} = \{\geq, \leq, \geq\}$, that is translated by the following ordering relations:

$$S_1[\alpha] \preceq S_1[\beta] \Rightarrow S_1[\alpha] \geq S_1[\beta];$$
$$S_2[\alpha] \preceq S_2[\beta] \Rightarrow S_2[\alpha] \leq S_2[\beta];$$
$$S_3[\alpha] \preceq S_3[\beta] \Rightarrow S_3[\alpha] \leq S_3[\beta];$$

Different syntheses also have different priorities. In the example, S_1, S_2 and S_3 priorities go from the highest to the lowest.

Table II. Paths ordering of the network

Abbreviation $\overline{b}_j(\overline{S}[.])$	Result
$\overline{b}_1[S_1]\ \overline{b}_1[S_2]\ \overline{b}_1[S_3]$	$S_1 \Rightarrow \alpha \equiv \beta$ $S_2 \Rightarrow \alpha \equiv \beta$ $S_3 \Rightarrow \alpha \prec \beta$
$\overline{b}_\infty[S_1]\ \overline{b}_\infty[S_2]\ \overline{b}_\infty[S_3]$	$S_1 \Rightarrow$ 1st letters are equal $\Rightarrow \alpha \equiv \beta$
	$S_2 \Rightarrow$ 2nd letters $\Rightarrow$ $4 > 3 \Rightarrow \beta \prec \alpha$
$\overline{b}_1[S_1]\ \overline{b}_\infty[S_2]\ \overline{b}_1[S_3]$	$S_1 \Rightarrow \alpha \equiv \beta$ $S_2 \Rightarrow$ 1st letters are equal $\Rightarrow \alpha \equiv \beta$ $S_2 \Rightarrow$ 2nd letters $\Rightarrow$ $4 > 3 \Rightarrow \beta \prec \alpha$

Table II summarizes the results obtained for three different ordering criteria. It is important to realize that the syntheses letters are examined from the highest priority to the lowest priority synthesis. When the paths are considered equivalent, then we will examine either the next letter of the same synthesis or will move to the next synthesis. This is determined by the adopted abbreviation.

B. Paths algebra application in the VNE problem

The paths algebra is a useful tool to find a suitable assignment of the SN for various virtual requests. It is very powerful to calculate all possible paths, which involves different metrics as spare the resource use of bandwidth and CPU, and ranks them from best to worst according to the evaluation function. The solution space is large enough, which guarantees an optimization performance for our algorithm.

Table.III. Metrics for virtual link mapping

Metric	**Definition**
CPU	Available CPU resource in a substrate node.
BW	Available bandwidth resource in a substrate link.
Stress	Number of VR running on a specific substrate node or link.
Length	The path length when virtual link mapped.

Table III shows some metrics in the paths algebra that we will use in link mapping stage, which are associated with some substrate link weight and some node character. The paths algebra has been developed associating weights to the substrate links that represents the network, so we did the following transformation. Given a link (v,u), as one possible objective could be to maximize the spare CPU capacity and stress for substrate node, the *CPU* or stress (for node) of link (v,u) will be described as $\min(CPU(v), CPU(u))$ or $\min(Stress(v), Stress(u))$.

Consider the objectives for VNE problem is to minimize the use of substrate resource while having a high VR acceptance ratio. For a virtual request, the revenue of all mappings is the same. But the cost of a virtual request depends on the different mappings onto the SN, the longer path length of virtual link mapped, the higher bandwidth resource cost the

virtual link would have. So, the way to minimize the use of substrate resource while having a high VR acceptance ratio is to minimize the path length that VR mapped while having a good load balance. One of the feasible optimal objective is to minimize the sum of the substrate path length that virtual request mapped and the substrate node stress. Therefore, for a virtual request $G^R = (V^R, E^R)$, the optimal objective for a virtual request is defined as:

$$Minimize: \alpha \sum_{\substack{\forall v' \in V^R, \exists v \in V^S \\ st: v' \to v}} Stress(v) + \beta \sum_{\substack{\forall (u',v') \in E^R, \exists (u,v) \in E^S \\ st: (u',v') \to (u,v)}} Length(u,v) \quad (3)$$

Where α and β reflect the relative importance of the substrate node stress and the substrate path length that virtual request mapped.

Correspondingly, the metrics, syntheses and ordering relations to be used to satisfy the optimal objective are $M = \{\{Length, Num, CPU, BW, \{Length \quad Stress\}\}$; $S = \{add(), add(), \min(), \min(), \min()\}$; $\preceq_{ML}$ is taken as $\{\geq, \geq, \leq, \leq, \leq\}$, where α , β is given the value 1.

Therefore, ***the link mapping algorithm based on paths algebra framework*** is presented as follows.

For a given virtual request

Step1: After node mapping stage,

$(u', v') \to (u, v)((u', v') \in E^R, (u, v) \in E^S)$

Step2: Enumerate all paths between the SN peer nodes (u, v) via *the LADN algorithm* [16].

Step3: Calculate the synthesis result based on paths algebra framework for each sub-path of link (u, v).

Step4: If the sub-path that can satisfy the virtual demand of CPU and bandwidth and have better performance under the constraint of $\min(\{Hops + Stress\})$. Map the VR link, refresh value of the chosen path under the paths algebra framework and GOTO Step1, until mapping all the virtual links for a given VR.

Step5: Else, return false.

C. Example

After node mapping stage given in Fig.1, virtual link (a, c) is mapped onto substrate path (C, B). So here we took $(a, c) \to (C, B)$ as an example.

First, enumerating all paths between the SN peer nodes (C, B):{C,A,B};{C,D,E,B};{C,D,G,H,F,E,B} via *the LADN algorithm* [16];

Second, calculating the synthesis results of each possible substrate path for virtual link (a, c). The results are presented as follows.

All path of link (*C, B*)	S1 *Hops*	S2 *Stress*	S3 *CPU*	S4 *BW*	*Min({Hops+ Stress})*
(C,A,B)	2;1	0;0	55;60	15;22	2;1
(C,D,E,B)	3;2;1	0;0;0	55;60;60	12;20;20	3;2;1
(C,D,G,H, F,E,B)	6;5;4; 3;2;1	0;0;0; 0;0;0	50;50;50; 60;60;60	10;10;20; 20;20;20	6;5;4; 3;2;1

The results show the path (C, D, G, H, F, E, B) cannot fulfill the bandwidth demand of VR link (a, c) , both path (C, A, B) and path (C, D, E, B) can fulfill the CPU and bandwidth demand of CR link (a, c). However, choosing the path (C, A, B) can get a better performance under the constraint of optimization objective. Therefore, the virtual link (a, c) is mapped onto the substrate path (C, A, B).

At last, refresh value of the chosen substrate path, and map next virtual link for a given request.

All path of link (*C, B*)	S1 *Length*	S2 *Stress*	S3 *CPU*	S4 *BW*	*Min({Length+ Stress})*
(C,A,B)	2;1	2;1	10;10	3;10	2;1

V. Simulation Results and Analysis

The simulation environment, the performance metrics, and the simulation results will be discussed in this section.

A. Simulation environment

The SN topologies used in our algorithm are generated by the Georgia Tech Internetwork Topology Models (GT-ITM) Tool [15]. 100 nodes were randomly selected from the SN to construct an adjacency matrix of 100 × 100. Similarly, the virtual request used in this paper was randomly selected from the GT-ITM to construct the adjacency matrix, whose size is depended on the number of virtual nodes. Then, both the substrate nodes and virtual nodes were sorted in descending order according to their sufficient resource. The values of metric κ , λ , α , β are given 1, 0, 1, 1.

Our proposed algorithm is called an enhanced algorithm based on paths algebra (GBPA). And the baseline algorithm (also called GARSP) [2] was used to evaluate the performance of proposed algorithm.

B. Performance metrics

a) **Average path length for each virtual request mapped:** Measures the average path length for each virtual request that has been accepted by SN, which reflects the substrate bandwidth resource usage.

b) **Load balance:** Measure the balance of the total substrate resource that has been used, which was represented as the variance of the number of VR that has been accepted for each substrate node.

c) **Average node (link) stress:** Measures the CPU (bandwidth) usage percentage of an SN node (link) when an algorithm embeds some VN requests in the SN, which was represented as the average number of the virtual request that has been accepted for each substrate node.

d) **Acceptance ratio:** Measures the percentage of total VN requests accepted by an algorithm over a given period.

C. Simulation results

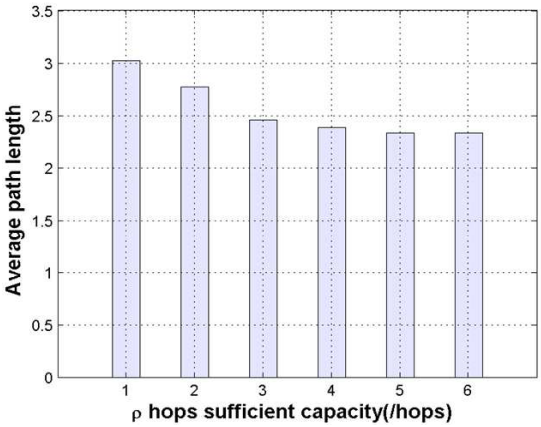

Fig. 4. Influence of different hops on mapped path length.

Fig.4 shows the influence of different hops on the average path length for each mapped virtual request. With the increase of the value of ρ, the average path length for each mapped virtual request was shortened gradually. However, when the value of ρ is bigger than 4 hops, then the average path length for each mapped virtual request is almost stable at a certain value.

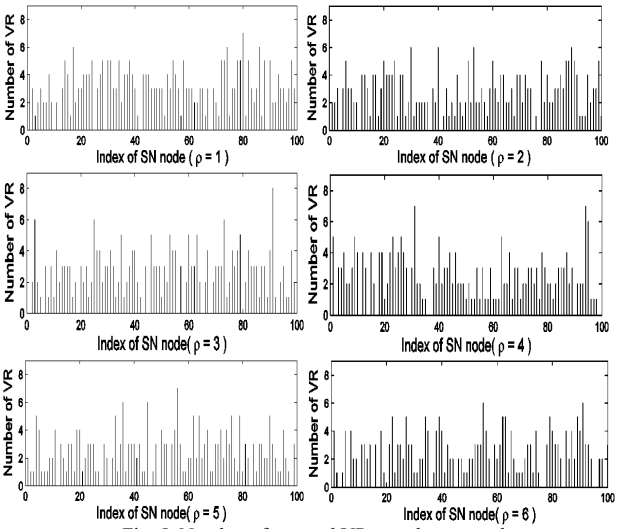

Fig. 5. Number of mapped VR on substrate node.

	ρ =1	ρ =2	ρ =3	ρ =4	ρ =5	ρ =6
Mean	3.02	2.77	2.45	2.38	2.33	2.33
Variance	2.60	2.49	2.62	2.90	2.85	3.01

Table IV. Average path length and variance of paths length of mapped VR given in Fig.5.

Fig.5 and Table IV show that average node (link) stress and the load balance, which was correspondingly represented as the mean of the number of VR that has been accepted for each substrate node and the variance of the number of VR that has been accepted for each substrate node. When the value of ρ is bigger than 4 hops, the variance of the path length for each virtual request would grows gradually, which means an increasing tendency of average node (link) stress.

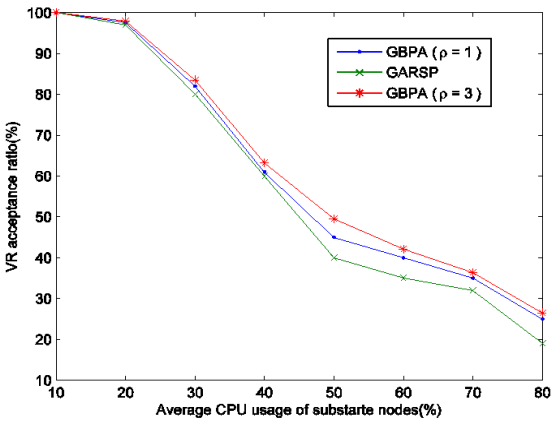

Fig. 6. Influence of different hops on the VR acceptance ratio.

Fig.6 shows the virtual acceptance ratio for different average CPU usage of substrate nodes when ρ is given the value 1 and 3, also comparing with the GARSP algorithm on VR acceptance ratio. The result shows that the proposed algorithm based on paths algebra has an enhanced capability of receiving virtual request when giving the same value of the average substrate node's CPU usage, especially, ρ is given the value 3. For the reason that the node mapping stage can affect the efficiency of the link mapping stage, when ρ is given the value 3, as Table IV shows that the SN has an high load balance, which helps a lot to the VR acceptance ratio. Additionally, the paths algebra provides a large solution to get the optimal result under the restrictions of optimal objective, which ensuring the link stage could have an enhance performance.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we define the sufficient capacity metric, it is very important in influencing the node mapping and link mapping algorithm's performance. Additionally, the algorithm based on the paths algebra framework, which ensuring the routing convergence for multi-constraint routing problem, it

improves the link mapping stage algorithm performance, while its applications on the node mapping stage still need further research.

ACKNOWLEDGMENT

This study is supported by the Natural Science Foundation of Shandong Province (Grant No. ZR2013FM029) and the University Innovation Project of Jinan (Grant No. 201303010).

REFERENCES

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," Computer, vol. 38, pp. 34-41, 2005.

[2] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 17-29, 2008.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. Meer, X. Hesselbach, "Virtual network embedding: a survey," *IEEE Communications Surveys & Tutorials,* vol. 15(4), pp. 1888-1906, 2013.

[4] X. Cheng and S. Su, "Virtual network embedding through topology awareness and optimization," *Computer Networks,* vol. 56, pp. 1797-1813, 2012.

[5] H. Y. Cui, S. H. Tang, J. Y. Chen, and Y. J. Liu, "A novel method of virtual network embedding based on topology convergence-degree," in *IEEE ICC*, 2013, pp. 246-250.

[6] T. Huang, J. Liu, J. Y. Chen, and Y. J. Liu, "A topology-cognitive algorithm framework for virtual network embedding problem," *China Communications,* vol. 11(4), pp. 73-84, 2014.

[7] X. Cheng and S. Su, "Virtual network embedding through topology awareness and optimization," *Computer Networks,* vol. 56, pp. 1797-1813, 2012.

[8] X. L. Li, H. M. Wang, B. Ding, X. Y. Li, and D. W. Feng, "Resource allocation with multi-factor node ranking in data center networks," *Future Generation Computer Systems,* vol. 32, pp. 1-12, 2014.

[9] J. F. Botero, M. Molina, X. H. Serra, and J. R. Amazonas, "A novel paths algebra-based strategy to flexibly solve the link mapping stage of VNE problems," *Journal of Network and Computer Applications,* vol. 36, pp. 1735-1752, 2013.

[10] I. Houidi,W. Louati,W. Bean-Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Comput. Netw.*, vol. 55, no. 4, pp. 1011–1023, Mar. 2011.

[11] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1077–1086, Jun. 2013.

[12] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking,* vol. 20, pp. 206-219, 2012.

[13] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer,* vol. 38, pp. 34-41, 2005.

[14] X. Masip-Bruin, M. Yannuzzi, J. Domingo-Pascual, A. Fonte, M. Curado, E. Monteiro, F. Kuipers, P. Van Mieghem, S. Avallone, G. Ventre, P. Aranda-Gutiérrez, M. Hollick, R. Steinmetz, L. Iannone, K. Salamatian, Research challenges in QoS routing, Computer Communications, Volume 29, Issue 5, 6 March 2006.

[15] G. Sun, H. F. Yu, V. A, And L. M. Li, "A cost efficient framework and algorithm for embedding dynamic virtual network requests," Future Generation Computer Systems, Vol. 29, pp. 1265-1277, 2013.

[16] Herman WP, Amazonas JR. Hop-by-hop routing convergence analysis based on paths algebra. In: Proceedings of the 2007 electronics, robotics and automotive mechanics conference-CERMA 2007, Mexico, 2007.

[17] José Roberto de Almeida Amazonas_y, Germán Santos-Boada and Josep Solé-Pareta, A paths algebra framework for routing and resources assignment in EONs. Transparent Optical Networks (ICTON), 2015 17th International Conference-Budapest, Hungary, 2015.