Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A virtual network mapping algorithm based on integer programming*

Bo LU†, Jian-ya CHEN†‡, Hong-yan CUI, Tao HUANG, Yun-jie LIU

(*Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China*)

†E-mail: {billylu, jychen}@bupt.edu.cn

Received May 5, 2013; Revision accepted Aug. 30, 2013; Crosschecked Nov. 18, 2013

Abstract: The virtual network (VN) embedding/mapping problem is recognized as an essential question of network virtualization. The VN embedding problem is a major challenge in this field. Its target is to efficiently map the virtual nodes and virtual links onto the substrate network resources. Previous research focused on designing heuristic-based algorithms or attempting two-stage solutions by solving node mapping in the first stage and link mapping in the second stage. In this study, we propose a new VN embedding algorithm based on integer programming. We build a model of an augmented substrate graph, and formulate the VN embedding problem as an integer program with an objective function and some constraints. A factor of topology-awareness is added to the objective function. The VN embedding problem is solved in one stage. Simulation results show that our algorithm greatly enhances the acceptance ratio, and increases the revenue/cost (*R*/*C*) ratio and the revenue while decreasing the cost of the VN embedding problem.

Key words: Virtual network embedding, Integer programming, Topology-awareness, Network virtualization
doi:10.1631/jzus.C1300120 **Document code:** A **CLC number:** TP393

1 Introduction

Network virtualization has been identified as a promising technology to overcome the current ossification of the Internet by running multiple network services and experiments simultaneously on the same substrate network (Anderson *et al*., 2005; Chowdhury and Boutaba, 2010; Fischer *et al*., 2011; Wang *et al*., 2013). It could even become the basis of the future Internet (Anderson *et al*., 2005). Multiple service providers (SPs) will be able to provide the customized end-to-end services to the end users without a vast investment on physical infrastructure in the network virtualization environment (NVE). They will lease shared resources from different infrastructure providers (InPs) to build the heterogeneous virtual networks (VNs) (Feamster *et al*., 2007).

Virtual network embedding (VNE) solves the problem of mapping virtual resources to physical resources, which has led to applying virtualization in network resources. Its main object is to assign the virtual network requests reasonably to the substrate network while satisfying node and link constraints.

Most of the past algorithms for VNE problems are based on the heuristic method (Yu *et al*., 2008). However, these heuristic-based algorithms may result in poor performance because they restrict the solution space by preselecting node mappings without considering its relation to the link mapping stage. The algorithm presented in Chowdhury *et al*. (2009) introduced better correlation between node mapping and link mapping. An augmented graph is built by introducing meta-nodes for each virtual node and firstly connecting the meta-nodes to a subset of physical nodes. Then, the VNE problem is solved by

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (Nos. 2011CB302901 and 2012CB315801) and Fundamental Research Funds for the Central Universities, China (No. 2011RC0118)

© Zhejiang University and Springer-Verlag Berlin Heidelberg 2013

using a mixed integer programming (MIP) approach.

However, the algorithm presented in Chowdhury *et al*. (2009) has some issues which need to be improved: (1) It is not a real one-stage method; (2) It needs extra processing for the results of MIP before achieving the final result. To achieve the optimal result of MIP by using GLPK, they relaxed the integer constraints. However, the result of MIP includes some non-integer values, and it cannot represent the exact situation of VN embedding. In these formulations, '1' means the substrate node is to be used by a specific virtual node and '0' means that it is not to be used. Non-integer values mean a bad situation and additional processes are needed. Thus, they devised two extra methods, D-ViNE and R-ViNE, based on deterministic and randomized rounding techniques to deal with the result of MIP.

To overcome these problems, we propose a new algorithm based on integer programming. Our motivation for using integer programming is that the calculation will be easier and the problem will be solved in one stage if the results of the optimization problem are optimal and purely integral. To the best of our knowledge, this is the first attempt to apply integer programming to a VNE problem in one stage. The advantages of the proposed algorithm are: (1) It is a real one-stage method; (2) It can achieve the final optimal result without extra processes.

Differences from the method mentioned in Chowdhury *et al*. (2009) are: (1) No flow exists between virtual nodes, and each virtual path begins from the first virtual node and ends at the second virtual node without passing any other virtual node again in the augmented substrate graph; (2) The binary variable in the formulations denotes whether the connection exists between a given virtual node and a specific substrate node only, and is not considered between substrate nodes; (3) We use the formulae of the critical index which is modified to be quicker and easier to calculate in Eqs (2) and (3) as a factor of the objective function. These differences are the reason why our algorithm is better than the previous ones. The constraints of the flow variables and binary variables ensure that each virtual node is mapped on one and only one substrate node; one substrate node can be used only by one virtual node for a virtual request; a flow starts from a meta-node, passes the substrate nodes which satisfy the node and link constraints, and ends in another meta-node without passing any other virtual node. This reduces the complexity of the calculation and ensures that the optimal integer result can be achieved. The overuse of the bridge nodes and links is equivalent to partitioning the substrate, rather than increasing the possibility of failure to embed a virtual request. The use of a critical index ensures that the mapping tries to avoid using the bridge nodes and links, thereby improving the acceptance ratio of the embedding.

2 Related work

So far a number of algorithms have been proposed to solve the VNE problem. Some key features of these algorithms are: (1) They can process the online requests or only the offline requests; (2) They solve the VNE problem in one stage or two.

An offline method was presented in Zhu and Ammar (2006). When a group of incoming VNRs (decreasingly ordered by revenue) occurs during a time window, the online algorithm queues them and tries to efficiently allocate them. The objective of the online approach presented in Yu *et al.* (2008) was to achieve the maximization of long term average revenue. An algorithm based on the proximity principle which considers the distance factor in the node mapping step was presented in Liu *et al*. (2011). Some new online algorithms based on different principles were mentioned in Zhang (2012) and Zhang *et al*. (2013). An online algorithm proposing new virtual node and link mapping stages was presented in Chowdhury *et al*. (2009). All of the above methods are based on two stages, node mapping and link mapping. Butt *et al*. (2010) made two important contributions to improve the acceptance ratio of VNRs, i.e., topology-aware embedding and re-optimizing bottlenecked embeddings.

The online algorithm presented in Chowdhury *et al*. (2009) solves the VNE problem by using a mixed integer programming (MIP). To define an augmented graph over the substrate network by introducing a set of meta-nodes, one per virtual node connects to a cluster of candidate substrate nodes obeying location and capacity constraints. It is only the beginning of node mapping. Then, the integer constraints are relaxed to obtain a linear program. Finally, two extra methods D-ViNE and R-ViNE are devised based on deterministic or randomized rounding techniques to

obtain the final result of node mapping. The multi-commodity flow (MCF) algorithm and the shortest path algorithm were used to handle link mapping once node mapping had been achieved. These methods relax their integer constraints, and obtain non-integer values. After using MIP, they need extra processes to obtain the final result, by mapping nodes and links during two different stages.

Butt *et al*. (2010) put forward the concept of topology-awareness embedding to improve the acceptance ratio of VNRs. When a substrate node or link is susceptible to a bridge when its removal divides the substrate graph into two or more networks, it is treated as a critical substrate node or link. These critical resources will be considered in the objective function of our algorithm.

In this study, we propose a formal algorithm based on integer programming (Schrijver, 1986) for the virtual network embedding problem. Firstly, we will build a model of an augmented substrate graph extended from the physical network graph (Section 4.1). We regard each virtual node as a meta-node and connect the meta-nodes to the substrate network. Then, we handle each virtual link with constraints of bandwidth as a commodity constituting two meta-nodes. In the result, the mapping of a virtual link needs only to find the optimal flow for each commodity. Some binary constraints ensure that just one meta-edge can be selected for each meta-node. Thus, each virtual node links to exactly one substrate node effectively by the selected meta-edge. Its objective is the total revenue of the embedding. The constraints include capacity constraints, flow constraints, binary constraints, and domain constraints. Both virtual nodes and paths can be mapped without extra computing at the same time. It supports path splitting naturally for the situation that has been considered in the description of constraints. Our method also considers topology-awareness.

3 Definitions and formulations

In this section, we will introduce the basic definitions and formulations of the VNE problem.

3.1 Substrate network

The substrate network will be denoted by $G^{\mathrm{S}}=(N^{\mathrm{S}}, E^{\mathrm{S}})$, where N^{S} and E^{S} are the sets of substrate nodes and links, respectively. Each substrate node $n^{\mathrm{S}}\in N^{\mathrm{S}}$ is associated with the geographic location $\mathrm{loc}(n^{\mathrm{S}})$ and the CPU capacity $c(n^{\mathrm{S}})$. Each substrate link $e^{\mathrm{S}}(i, j)\in E^{\mathrm{S}}$ is associated with the bandwidth capacity $b(e^{\mathrm{S}})$ between nodes i and j. The set of all substrate paths is denoted by P^{S}.

3.2 VN request

Each virtual request will be denoted by $G^{\mathrm{V}}=(N^{\mathrm{V}}, E^{\mathrm{V}})$. Each virtual node $n^{\mathrm{V}}\in N^{\mathrm{V}}$ is associated with the geographic location $\mathrm{loc}(n^{\mathrm{V}})$ and the CPU capacity $c(n^{\mathrm{V}})$. Each virtual link is associated with the bandwidth capacity $b(e^{\mathrm{V}})$. The relevant non-negative value D^{V} denotes the distance between the virtual node $n^{\mathrm{V}}\in N^{\mathrm{V}}$ and the location indicated by $\mathrm{loc}(n^{\mathrm{V}})$.

3.3 Measurement of substrate network resources

When the physical resources are used partly by some virtual requests, it is necessary to recalculate the available capacities of the substrate resources. $R_N(n^{\mathrm{S}})$ denotes the available CPU capacity of the substrate node $n^{\mathrm{S}}\in N^{\mathrm{S}}$. $R_E(e^{\mathrm{S}})$ denotes the available bandwidth of the substrate link $e^{\mathrm{S}}\in E^{\mathrm{S}}$ similarly.

$$R_N(n^{\mathrm{S}}) = c(n^{\mathrm{S}}) - \sum_{n^{\mathrm{V}}\left|M_N(n^{\mathrm{V}})=n^{\mathrm{S}}\right.} c(n^{\mathrm{V}}),$$

$$R_E(e^{\mathrm{S}}) = b(e^{\mathrm{S}}) - \sum_{e^{\mathrm{V}}\left|M_N(e^{\mathrm{V}})=e^{\mathrm{S}}\right.} b(e^{\mathrm{V}}).$$

The available bandwidth capacity of a substrate path $p\in P^{\mathrm{S}}$ is given by

$$R_E(p) = \min_{e^{\mathrm{S}}\in p} R_E(e^{\mathrm{S}}).$$

3.4 Constraints of VN embedding

The embedding can be defined as a mapping action M from a virtual request G^{V} to a subset of the substrate network G^{S}. M_N: $N^{\mathrm{V}}\to N^{\mathrm{S}}$ expresses node mapping and M_E: $E^{\mathrm{V}}\to P^{\mathrm{S}}$ expresses link mapping. The mapping action can be expressed as follows:

$$M^N(n^{\mathrm{V}})\in N^{\mathrm{S}}$$
$$M^N(m^{\mathrm{V}})=M^N(n^{\mathrm{V}}), \text{ iff } m^{\mathrm{V}}=n^{\mathrm{V}} \quad \forall n^{\mathrm{V}}, m^{\mathrm{V}}\in N^{\mathrm{V}}$$
$$M^E(e^{\mathrm{V}})\subseteq P^{\mathrm{S}}(M^N(m^{\mathrm{V}}), M^N(n^{\mathrm{V}})) \quad \forall e^{\mathrm{V}}=(m^{\mathrm{V}}, n^{\mathrm{V}})\in E^{\mathrm{V}}$$

subject to

$$c(n^{\mathrm{V}}) \le R_N(M_N(n^{\mathrm{V}})), \tag{1a}$$

$$\mathrm{dis}(\mathrm{loc}(n^{\mathrm{V}}),\ \mathrm{loc}(M_N(n^{\mathrm{V}}))) \le D^{\mathrm{V}}, \tag{1b}$$

$$b(e^{\mathrm{V}}) \le R_E(p), \tag{1c}$$

where dis(i, j) denotes how far a substrate node i is placed from the substrate node j.

In Fig. 1, the VN request 1 obtains node mapping $\{a\to A,\ b\to C,\ c\to D\}$ and link mapping $\{(a,\ b)\to\{(A,\ B),\ (B,\ C)\},\ (a,\ c)\to\{(A,\ D)\},\ (b,\ c)\to\{(D,\ E),\ (E,\ C)\}\}$.

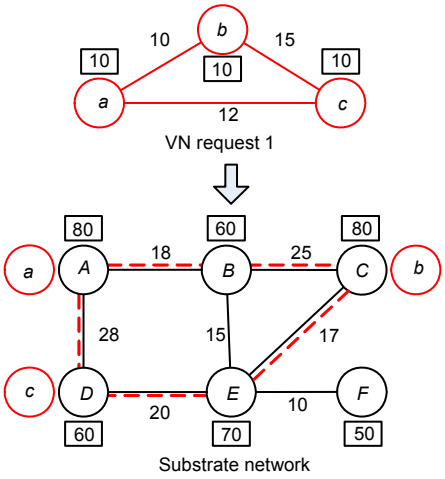

Fig. 1 Mapping a virtual request onto a shared substrate network

3.5 Critical index

The critical index of a resource measures the likelihood of the residual substrate network partition due to its unavailability (Butt *et al*., 2010). For example, the probability of the nodes of any virtual link to be embedded in two different partitions is 0.5 when a physical network is partitioned into two almost equal-sized partitions. In this case, we might have to reject almost 50% of the VN requests.

We denote CI by ξ (ξ: $x\to[0, 1]$), where x is either a substrate node or a link. The unavailability that x will partition the substrate graph has a higher likelihood when $\xi(x)$ gets a higher value. Eqs. (2) and (3) define the critical indexes of nodes and links, respectively:

$$\xi_N(n^{\mathrm{S}}) = \begin{cases} \sum_{C_i}\left(\frac{1}{|C|} - \left|\frac{1}{|C|} - p(C_i)\right|\right), & n^{\mathrm{S}} \in \text{cut-vertice}, \\ 1/2, & n^{\mathrm{S}} \notin \text{cut-vertice}, \end{cases} \tag{2}$$

$$\xi_E(e^{\mathrm{S}}) = \begin{cases} 1 - \frac{c_1 - c_2}{2}, & e^{\mathrm{S}} \in \text{cut-edge}, \\ \frac{1}{2}, & e^{\mathrm{S}} \notin \text{cut-edge}. \end{cases} \tag{3}$$

In Eq. (2), if n^{S} is a bridge node (cut-vertice) then removing it partitions the graph into more than two components. C is the set of components obtained when we remove n^{S}. $p\{C_i\}$ here is the fraction of the total nodes present in component C_i. If n^{S} is not a bridge node, then CI is 0.5. In Eq. (3), if e^{S} is a bridge link (cut-edge) then removing it partitions the graph into two components. c_1 and c_2 are the fractions of substrate nodes in each of these partitions. If e^{S} is not a bridge link, then CI is 0.5.

3.6 Objectives

The objectives in our study are to maximize the revenue and minimize the cost of the mapping in the long run, and to totally improve the R/C ratio. The acceptance ratio also deserves our attention.

Revenue is one of the main factors we use to judge the performance of an embedding algorithm because it represents the profit of the embedding action. Referring to Zhu and Ammar (2006) and Yu *et al*. (2008), the revenue of a VN request is defined as

$$R(G^{\mathrm{V}}) = \sum_{e^{\mathrm{V}}\in E^{\mathrm{V}}} b(e^{\mathrm{V}}) + \sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}} c(n^{\mathrm{V}}). \tag{4}$$

Cost is the price of the embedding. It is defined according to the nodes' CPU and links' bandwidth used for the virtual request in the substrate network:

$$C(G^{\mathrm{V}}) = \sum_{e^{\mathrm{V}}\in E^{\mathrm{V}}} \sum_{e^{\mathrm{S}}\in E^{\mathrm{S}}} f_{e^{\mathrm{S}}}^{e^{\mathrm{V}}} + \sum_{n^{\mathrm{V}}\in N^{\mathrm{V}}} c(n^{\mathrm{V}}), \tag{5}$$

where $f_{e^{\mathrm{S}}}^{e^{\mathrm{V}}}$ denotes the total amount of bandwidth allocated on the substrate link e^{S} for virtual link e^{V}. We create the objective function of the IP formulation based on Eq. (5).

The R/C ratio indicates the efficiency of a substrate network resource. It is an important factor to judge the performance of an embedding algorithm. Thus, the R/C ratio is a value in [0, 1] and the value '1' indicates the optimum embedding:

$$R/C=\frac{R(G^{\mathrm{V}})}{C(G^{\mathrm{V}})}=\frac{\sum\limits_{e^{\mathrm{V}}\in E^{\mathrm{V}}}b(e^{\mathrm{V}})+\sum\limits_{n^{\mathrm{V}}\in N^{\mathrm{V}}}c(n^{\mathrm{V}})}{\sum\limits_{e^{\mathrm{V}}\in E^{\mathrm{V}}}\sum\limits_{e^{\mathrm{S}}\in E^{\mathrm{S}}}f_{e^{\mathrm{S}}}^{e^{\mathrm{V}}}+\sum\limits_{n^{\mathrm{V}}\in N^{\mathrm{V}}}c(n^{\mathrm{V}})}. \quad (6)$$

The acceptance ratio indicates the proportion of accepted requests. It is defined by the following formulation:

$$\text{Acceptance ratio}=\frac{\text{Number of accepted requests}}{\text{Number of all requests}}.$$

4 Integer programming formulation for optimal VN embedding

In this section, we describe the integer programming algorithm in detail. Then, we briefly discuss the implementation of the new algorithm.

4.1 Construction of an augmented substrate graph

We create one cluster for each virtual node with radius D^{V} in the substrate network. Each node of the cluster satisfies the constraints in Eqs. (7a), (7b), and (1a).

$$\Omega(n^{\mathrm{V}})=\{n^{\mathrm{S}}\in N^{\mathrm{S}}\,|\,\mathrm{dis}(\mathrm{loc}(n^{\mathrm{V}}),\mathrm{loc}(n^{\mathrm{S}}))\le D^{\mathrm{V}}\}, \quad (7a)$$

$$\sum_{e^{\mathrm{V}}|n^{\mathrm{V}}\in e^{\mathrm{V}}}b(e^{\mathrm{V}})\le\sum_{e^{\mathrm{S}}|n^{\mathrm{S}}\in e^{\mathrm{S}}}R_E(e^{\mathrm{S}}). \quad (7b)$$

We use $\Omega(n^{\mathrm{V}})$ to express the cluster. In Fig. 2, the distance between the substrate nodes A or D and the virtual node a is within D^{V}. Hence, $\Omega(a)=\{A, D\}$. Similarly, $\Omega(b)=\{C, E, F\}$ and $\Omega(c)=\{A, B\}$.

$\mu(n^{\mathrm{V}})$ is a meta-node corresponding to n^{V}. We will connect $\mu(n^{\mathrm{V}})$ to each substrate node belonging to $\Omega(n^{\mathrm{V}})$, respectively. We define the created augmented substrate graph as $G^{\mathrm{S}'}=(N^{\mathrm{S}'}, E^{\mathrm{S}'})$.

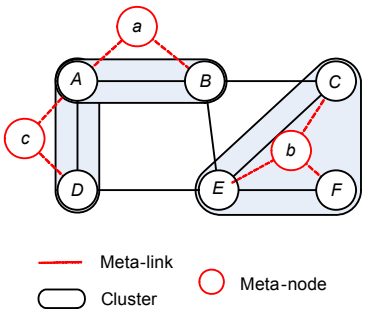

Fig. 2 Construction of an augmented substrate graph

4.2 Formulations of VNE_IP

Sets:

$M=\{\mu(n^{\mathrm{V}})|n^{\mathrm{V}}\in N^{\mathrm{V}}\}$,
$N^{\mathrm{S}'}=N^{\mathrm{S}}\cup M$,
$E^{\mathrm{S}'}=E^{\mathrm{S}}\cup\{(\mu(n^{\mathrm{V}}), n^{\mathrm{S}})|n^{\mathrm{V}}\in N^{\mathrm{V}}, n^{\mathrm{S}}\in\Omega(n^{\mathrm{V}})\}$.

Variables:

f_{uv}^{i}: A flow variable, which denotes the amount of one-way flow starting from u and ending at v on the substrate link (u, v) for the ith virtual link.

x_{uv}: A binary variable, which is set to '1' if the substrate node v is used by the virtual node u; otherwise, it has the value '0'.

Objective:

Minimize

$$\sum_{uv\in E^{\mathrm{S}}}\frac{\alpha_{uv}}{R_E(u,v)+\delta}\xi_E(u,v)\sum_i f_{uv}^{i}+\sum_{w\in N^{\mathrm{S}}}\frac{\beta_w}{R(w)+\delta}\xi_N(w)\sum_{m\in M}x_{mw}c(m). \quad (8)$$

Constraints:

Capacity constraints:

$$\sum_i(f_{uv}^{i}+f_{vu}^{i})\le R_E(u,v)\quad\forall u,v\in N^{\mathrm{S}'}. \quad (9)$$

Flow related constraints:

$$\sum_{w\in N^{\mathrm{S}'}}f_{uw}^{i}-\sum_{w\in N^{\mathrm{S}'}}f_{wu}^{i}=0\quad\forall i,\ \forall u\in N^{\mathrm{S}'}/\{s_i,t_i\}, \quad (10)$$

$$\sum_{w\in N^{\mathrm{S}}}f_{s_iw}^{i}-\sum_{w\in N^{\mathrm{S}}}f_{ws_i}^{i}=b(e_i^{\mathrm{V}})\quad\forall i, \quad (11)$$

$$\sum_{w\in N^{\mathrm{S}}}f_{t_iw}^{i}-\sum_{w\in N^{\mathrm{S}}}f_{wt_i}^{i}=-b(e_i^{\mathrm{V}})\quad\forall i, \quad (12)$$

$$f_{s_iu}^{i}=b(e_i^{\mathrm{V}})x_{s_iu}\quad\forall i,\ \forall u\in N^{\mathrm{S}}, \quad (13)$$

$$f_{ut_i}^{i}=b(e_i^{\mathrm{V}})x_{t_iu}\quad\forall i,\ \forall u\in N^{\mathrm{S}}. \quad (14)$$

Meta and binary constraints:

$$\sum_{w\in N}x_{mw}=1\quad\forall m\in M, \quad (15)$$

$$\sum_{m\in M}x_{mw}\le 1\quad\forall w\in N^{\mathrm{S}}, \quad (16)$$

$$f_{uv}^{i}=0\quad\forall u,v\in M. \quad (17)$$

Domain constraints:

$$f_{uv}^{i} \geq 0 \quad \forall u, v \in N^{S'}, \tag{18}$$

$$x_{uv} \in \{0, 1\} \quad \forall u \in M, \forall v \in N^{S}. \tag{19}$$

Parameters:

δ: A small positive constant to avoid dividing by zero in the calculation. We set δ to be 10^{-6}.

$R_E(u, v)$: A parameter which denotes the available bandwidth of the substrate link e^S.

$R_N(w)$: A parameter which denotes the available CPU capacity of the substrate node n^S.

α_{uv}: A parameter to load balancing while embedding a virtual request. α_{uv} is limited to the range from 1 to $R_E(u, v)$.

β_w: A parameter to load balancing while embedding a virtual request. β_w is limited to from 1 to $R_N(w)$.

$\xi_N(n^S)$: The critical index of the substrate node n^S which is defined in Eq. (2).

$\xi_E(e^S)$: The critical index of the substrate edge e^S which is defined in Eq. (3).

$c(m)$: A parameter which denotes the CPU capacity $c(m)$ when $m \in M$.

s_i: The source node for the ith link of a virtual request.

t_i: The sink node for the ith link of a virtual request.

Remarks:

The objective function (8) expresses the total cost of embedding when a virtual request arrives. Our target is to minimize the cost under all the constraints. We set α_{uv} and β_w to be $R_E(u, v)$ and $R_N(w)$ respectively in our method. $\xi_E(e^S)$ and $\xi_N(n^S)$ are factors of topology-awareness. These two variables are added to the objective function to avoid using the nodes and links with high unavailability to partition the substrate graph.

Constraint set (9) contains the edge capacity bounds. The summation of the two flows f_{uv}^{i} and f_{vu}^{i} on both directions of the physical link (u, v) is no more than its available bandwidth.

Constraint sets (10), (11), and (12) are related to the flow conditions, which means that the total amount of flow passing a node is 0, except for the source node s_i and the sink node t_i for the ith virtual link.

Constraint set (13) ensures that only one substrate link connects the source node s_i to its meta-node u where $x_{s_i u}$ is '1'. Constraint set (14) ensures that only one substrate link connects the sink node t_i to its meta-node u where $x_{t_i u}$ is '1'.

Constraint sets (15) and (16) refer to the conditions of the augmented portion. The set (15) ensures that each meta-node connects to just one physical node. The set (16) ensures that each physical node places on one meta-node at most. The constraint set (17) makes sure that no flow exists between virtual nodes.

Finally, constraint sets (18) and (19) denote the non-negative domain constraints on the variables f_{uv}^{i} and the binary constraint on x_{uv}, respectively.

4.3 Details of the implementation

The GNU Linear Programming Kit (GLPK) package is intended for solving large-scale linear programming (LP) and mixed integer programming (MIP). We use the GLPK command-line solver GLPSOL to solve VNE_IP. In Fig. 3, a simple flowchart about our algorithm depicts the primary procedure of IP mapping, without regard to the removal of virtual mapping.

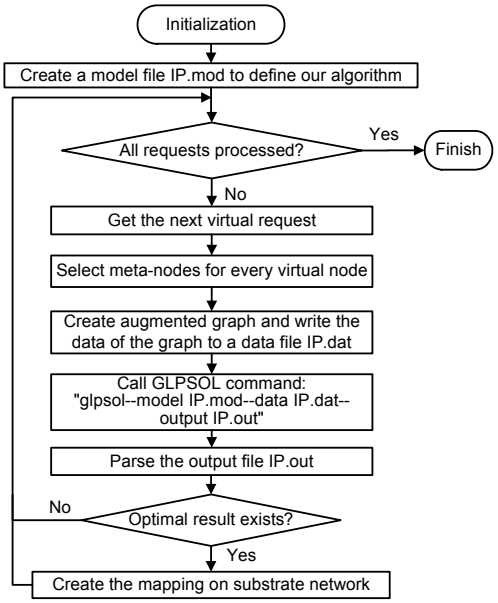

Fig. 3 Simple flowchart of VNE_IP

To start our simulation, we initialize the environment, create the substrate network, and input all

virtual requests into a queue which will pop up every request orderly. Then, we use the script language of GLPK to define our algorithm and write it to the file IP.mod. Part of the IP.mod is shown in Fig. 4.

```
s.t. capcon{u in NM, v in NM}:
sum{i in F} (f[i, u, v]+f[i, v, u])<=b[u, v];
/* capacity constraint */

s.t. demsat1{i in F}:
sum{w in N} f[i, fs[i], w]−sum{w in N} f[i, w, fs[i]]=fd[i];
s.t. demsat2{i in F}:
sum{w in N} f[i, fe[i], w]−sum{w in N} f[i, w, fe[i]]=−fd[i];
s.t. demsat3{i in F, u in M, v in M}: f[i, u, v]=0;
/* demand satisfaction */

s.t. flocon{i in F, u in NM diff{fs[i], fe[i]}}:
sum{w in NM} f[i, u, w]−sum{w in NM} f[i, w, u]=0;
/* flow conservation */

s.t. metcon1{m in M}: sum{w in N} x[m, w]=1;
s.t. metcon2{w in N}: sum{m in M} x[m, w]<=1;
s.t. metcon3{i in F, u in N}: f[i, fs[i], u]=fd[i]*x[fs[i], u];
s.t. metcon4{i in F, u in N, v in M diff{fs[i]}}: f[i, v, u]=0;
s.t. metcon5{i in F, u in N}: f[i, u, fe[i]]=fd[i]*x[fe[i], u];
s.t. metcon6{i in F, u in N, v in M diff{fe[i]}}: f[i, u, v]=0;
```

Fig. 4 Constraint part in the GLPK file

When a virtual request arrives, we create the augmented graph and write the data to the file IP.dat. The mod file IP.mod and the data file IP.data are ready, and we can obtain the output file IP.out by calling the command GLPSOL. We can acquire the final result by parsing the file IP.out. If the result is the optimal, the mapping will be built on the substrate network.

The mod format file IP.mod is the intuitive and accurate definition of our algorithm. In Fig. 4, it is the constraint part of the script. The constraint 'capcon' corresponds to Eq. (9), 'demsat1' corresponds to Eq. (11), while 'demsat2' corresponds to Eq. (12).

5 Simulation and analysis

In this section, we first introduce the simulation environment and parameter setting, and then analyze the performance of the new algorithm by comparing it with the baseline algorithm.

5.1 Simulation setting

We use a GT-ITM tool (Zegura *et al*., 1996) to generate the substrate network topology and VNRs.

The setting of the experimental environment is shown in Table 1. The substrate network has 100 nodes. Each pair of substrate nodes is randomly connected with a probability of 0.5. The CPU and bandwidth values of the substrate network are real numbers uniformly distributed between 50 and 100. The virtual network requests were generated following the previous work (Yu *et al*., 2008; Chowdhury *et al*., 2009). The arrival of virtual requests follows the Poisson process with the mean of four requests per 100 time units. In each VN request, the number of virtual nodes is randomly generated between 2 and 20. The average VN connectivity is 50%. The CPU values of the virtual nodes are real numbers uniformly distributed between 0 and 20, while the bandwidth values of the virtual links are uniformly distributed between 0 and 50.

Table 1 Setting of the experimental environment

Item	Value
Substrate network	
Number of substrate nodes	100
Probability of connectivity	0.5
CPU and bandwidth	50–100
Mean value of the Poisson process for the arrival of virtual requests	4 requests per 100 time units
VN request	
Number of virtual nodes	2–20
Average VN connectivity	0.5
CPU	0–20
Bandwidth	0–50

5.2 Comparison method

Our contribution is compared with four algorithms with different node mapping and link mapping strategies from previous research (Yu *et al*., 2008; Chowdhury *et al*., 2009) modified. Table 2 lists these algorithms.

Algorithm D-ViNE uses the deterministic node mapping mentioned in Chowdhury *et al*. (2009) and the splittable link mapping with multi-commodity flow (MCF); Algorithm D-ViNE-SP uses deterministic node mapping and link mapping based on the

shortest path method; Algorithm R-ViNE uses randomized node mapping and splittable link mapping with MCF; Algorithm G-SP uses the greedy node mapping mentioned in Yu *et al*. (2008) and link mapping based on the shortest path method.

Table 2 Algorithm comparison

Notation	Algorithm description
D-ViNE	Deterministic node mapping with splittable link mapping using MCF
D-ViNE-SP	Deterministic node mapping with shortest path based link mapping
R-ViNE	Randomized node mapping with splittable link mapping using MCF
G-SP	Greedy node mapping with shortest path based link mapping
IP	Algorithm based on integer programming, node & link mapping in one stage

5.3 Results and analysis

Six performance metrics are used for evaluation purposes in this experiment. The acceptance ratio, the average revenue (R), the average cost (C), the average node utilization, the average link utilization, and the R/C ratio are measured for VN requests over time. We plot the performance metrics actually performed in these algorithms (Table 1) in the long run. We summarize our key observations below:

1. IP mapping leads to a higher acceptance ratio. It is noticed that the proposed algorithm leads to a higher acceptance ratio than the existing algorithms through IP mapping (Fig. 5).

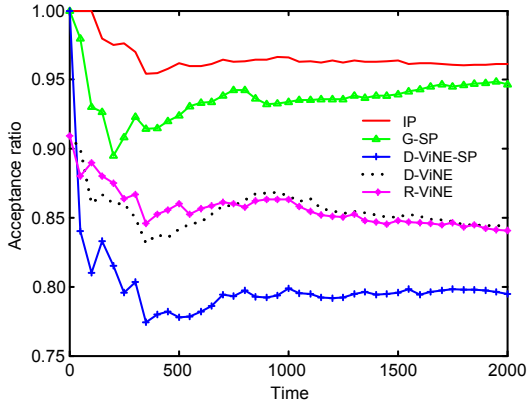

Fig. 5 Acceptance ratio of the VN request

2. IP mapping leads to lower cost and larger revenue. Figs. 6 and 7 show that the proposed algorithm leads to lower cost and larger revenue. Higher revenue and lower cost along with better acceptance ratios imply that our proposed algorithm is actually effective and efficient.

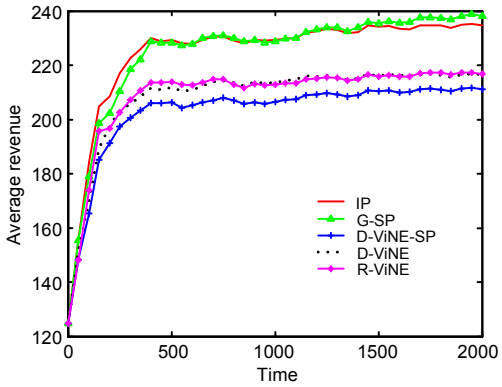

Fig. 6 Average revenue of the VN request over time

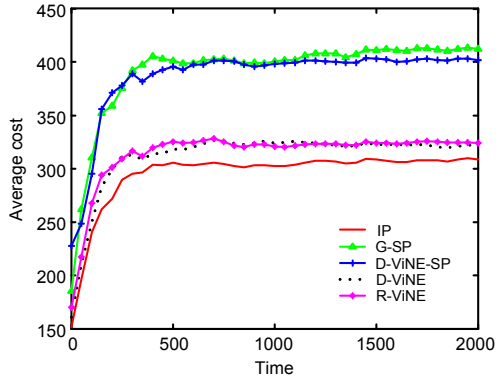

Fig. 7 Average cost of the VN request over time

3. IP mapping increases node utilization. Fig. 8 depicts the average utilization of substrate nodes for different VN embedding algorithms. Higher acceptance ratios of the proposed algorithm mean that more virtual requests are successfully mapped on the substrate network at the same time. Thus, more substrate node resources are used because any given virtual request uses the fixed substrate node resources. The substrate node resources used are increased, and the node utilization is correspondingly increased.

4. IP mapping leads to lower link utilization. Fig. 9 depicts the average utilization of substrate links for different VN embedding algorithms. IP mapping is efficient with the minimum cost to satisfy the largest demand. The proposed algorithm achieves a highly optimized path with the fewest hop counts. The fewest hop counts can minimize the objective

function, reduce the cost of the embedding, and decrease the link utilization.

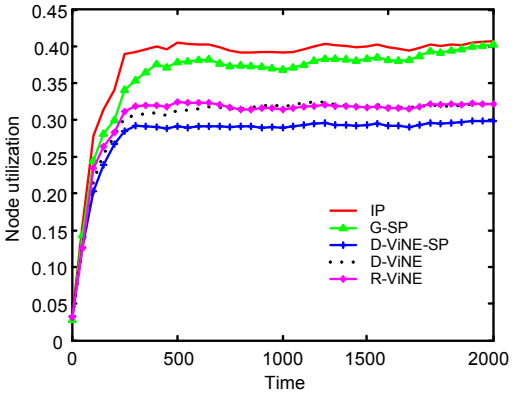

Fig. 8 Average node utilization of the VN request over time

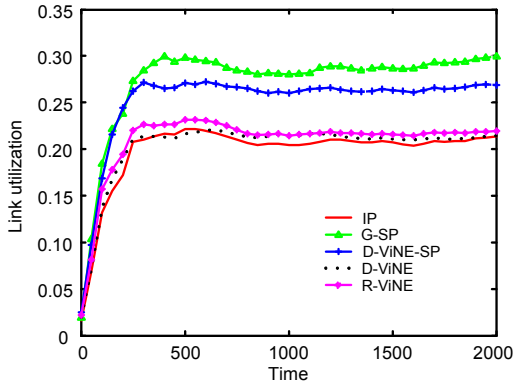

Fig. 9 Average link utilization of the VN request over time

5. IP mapping increases the *R*/*C* ratio. Fig. 10 depicts the *R*/*C* ratio for different VN embedding algorithms. The proposed algorithm leads to a much higher *R*/*C* ratio.

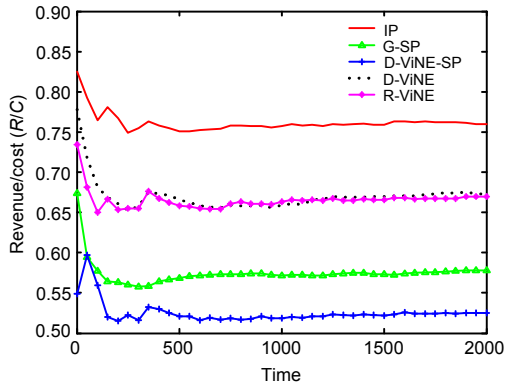

Fig. 10 *R*/*C* ratio of the VN request over time

6 Conclusions

The main concern of the virtual network embedding problem is how to develop an algorithm both effective and easy to implement. In this study, we propose an algorithm based on integer programming. We argue that this algorithm greatly increases the solution space and the quality of the linear programming algorithms. It is the first attempt to solve the embedding problem by using an integer program. We then add the factor of a topology-awareness objective function to obtain a higher acceptance ratio. Simulation showed that the new algorithm based on integer programming greatly enhances the performance of the embedding with a higher revenue/cost ratio and acceptance ratio. This method solves the embedding problem in just one stage.

However, a number of unresolved issues should be discussed for further research. First of all is the analysis of processing strategies for the proposed algorithm when the substrate network approaches its saturated state. Secondly, the influence of the physical network scale was barely mentioned. We believe it is an important issue. Thirdly, GLPK used in our algorithm requires a lot of files to be read and written, and is not efficient. A program developed specifically for this problem can be faster. Finally, the faults of physical network resources such as node or link failures are not considered in our algorithm. We believe that our model can be developed for this condition.

References

Anderson, T., Peterson, L., Shenker, S., Turner, J., 2005. Overcoming the Internet impasse through virtualization. *Computer*, **38**(4):34-41. [doi:10.1109/MC.2005.136]

Butt, N.F., Chowdhury, M., Boutaba, R., 2010. Topology-awareness and reoptimization mechanism for virtual network embedding. *Networking*, **6091**:27-39.

Chowdhury, N.M.M.K., Boutaba, R., 2010. A survey of network virtualization. *Comput. Netw.*, **54**(5):862-876. [doi:10.1016/j.comnet.2009.10.017]

Chowdhury, N.M.M.K., Rahman, M.R., Boutaba, R., 2009. Virtual Network Embedding with Coordinated Node and Link Mapping. IEEE INFOCOM, p.783-791. [doi:10.1109/INFCOM.2009.5061987]

Feamster, N., Gao, L.X., Rexford, J., 2007. How to lease the Internet in your spare time. *ACM SIGCOMM Comput. Commun. Rev.*, **37**(1):61-64. [doi:10.1145/1198255.1198265]

Fischer, A., Botero, J.F., Duelli, M., Schlosser, D., Hesselbach, X., de Meer, H., 2011. ALEVIN—a framework to develop, compare, and analyze virtual network embedding algorithms. *Electron. Commun. EASST*, **37**:1-12.

Liu, J., Huang, T., Chen, J.Y., Liu, Y.J., 2011. A new algorithm based on the proximity principle for the virtual network embedding problem. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **12**(11):910-918. [doi:10.1631/jzus.C1100003]

Schrijver, A., 1986. Theory of Integer and Linear Programming. John Wiley & Sons, NewYork, USA.

Wang, A.J., Iyer, M., Dutta, R., Rouskas, G.N., Baldine, I., 2013. Network virtualization: technologies, perspectives, and frontiers. *J. Lightwave Technol.*, **31**(4):523-537. [10.1109/JLT.2012. 2213796]

Yu, M.L., Yi, Y., Rexford, J., Chiang, M., 2008. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(2):17-29. [doi:10.1145/1355734.1355737]

Zegura, E.W., Calvert, K.L., Bhattacharjee, S., 1996. How to Model an Internetwork. IEEE INFOCOM, p.594-602.

Zhang, D., 2012. A Study on Virtual Network Decomposing Mapping Algorithm Based on Network Balance. 4th Int. Conf. on Computational and Information Sciences, p.880-883. [doi:10.1109/ICCIS.2012.55]

Zhang, M., Wu, C.M., Hang, Y., Yang, Q., Wang, B., Jiang, M., 2013. Robust dynamical virtual network provisioning. *Chin. J. Electron.*, **22**(1):151-154.

Zhu, Y., Ammar, M.H., 2006. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. IEEE INFOCOM, p.1-12. [doi:10.1109/INFOCOM.2006.322]

Recommended paper related to this topic

A new algorithm based on the proximity principle for the virtual network embedding problem

Authors: Jiang Liu, Tao Huang, Jian-ya Chen, Yun-jie Liu
doi:10.1631/jzus.C1100003
Journal of Zhejiang University-SCIENCE C (Computers & Electronics), 2011 Vol.12 No.11 P.910-918

Abstract: The virtual network embedding/mapping problem is a core issue of network virtualization. It is concerned mainly with how to map virtual network requests to the substrate network efficiently. There are two steps in this problem: node mapping and link mapping. Current studies mainly focus on developing heuristic algorithms, since both steps are computationally intractable. In this paper, we propose a new algorithm based on the proximity principle, which considers the distance factor besides the capacity factor in the node mapping step. Thus, the two steps of the embedding problem can be better integrated and the substrate network resource can be used more efficiently. Simulation results show that the new algorithm greatly enhances the performance of the revenue/cost (R/C) ratio, acceptance ratio, and runtime of the embedding problem.