

# On Virtual Network Embedding: Paths and Cycles

Haitao Wu, Fen Zhou, *Senior Member, IEEE*, Yaojun Chen, Ran Zhang

## Abstract

Network virtualization provides a promising solution to overcome the ossification of current networks, allowing multiple Virtual Network Requests (VNRs) embedded on a common infrastructure. The major challenge in network virtualization is the Virtual Network Embedding (VNE) problem, which is to embed VNRs onto a shared substrate network and known to be  $\mathcal{NP}$ -hard. The topological heterogeneity of VNRs is one important factor hampering the performance of the VNE. However, in many specialized applications and infrastructures, VNRs are of some common structural features *e.g.*, paths and cycles. To achieve better outcomes, it is thus critical to design dedicated algorithms for these applications and infrastructures by taking into accounting topological characteristics. Besides, paths and cycles are two of the most fundamental topologies that all network structures consist of. Exploiting the characteristics of path and cycle embeddings is vital to tackle the general VNE problem. In this paper, we investigated the path and cycle embedding problems. For path embedding, we proved its  $\mathcal{NP}$ -hardness and inapproximability. Then, by utilizing Multiple Knapsack Problem (MKP) and Multi-Dimensional Knapsack Problem (MDKP), we proposed an efficient and effective MKP-MDKP-based algorithm. For cycle embedding, we proposed a Weighted Directed Auxiliary Graph (WDAG) to develop a polynomial-time algorithm to determine the least-resource-consuming embedding. Numerical results

H. Wu is with the Department of mathematics, Nanjing University, China. He is also with the CERI-LIA at the University of Avignon. (email: whtmath2011@163.com).

F. Zhou is with the LISITE lab of the Institut Supérieur d'Electronique de Paris. He is also with the CERI-LIA at the University of Avignon. (emails: fen.zhou@isep, univ-avignon).fr).

Y. Chen is with Department of Mathematics, Nanjing University, Nanjing 210093, China. (email: yaojunc@nju.edu.cn).

R. Zhang is with School of Mathematics, Shanghai University of Finance and Economics, Guoding Road 777, 200433, Shanghai, P.R.China. (email: zhang.ran@mail.shufe.edu.cn).

showed our customized algorithms can boost the acceptance ratio and revenue compared to generic embedding algorithms in the literature.

### Index Terms

Virtual Network Embedding (VNE), Path and Cycles Embeddings, Algorithm Design.

## I. INTRODUCTION

**N**OWADAYS, the trend of Internet, especially driven by Big data applications [1, 2], marches towards involving more network elements and end-users, larger volume of traffic, and more diversified applications. However, current Internet infrastructures, consisting of a variety of technologies to run distributed protocols, become barrier to improving Internet service. This diversification is often referred to as the Internet ossification problem [3]. Network virtualization has been regarded as a compelling tool to overcome the Internet ossification and attracting a lot of researches [4]–[9]. It supports various networks of diverse natures (*e.g.*, network architectures, protocols, and user interactions [4]) to coexist in a same substrate network and share substrate resources (*e.g.*, CPUs and bandwidths). In the paradigm of network virtualization, the role of traditional Internet Service Providers (ISPs) is separated into two new entities: Infrastructure Provider (InP) and Service Provider (SP). The InP owns and manages the substrate network while the SP focuses on offering customized services to clients. In this business model as shown in Fig. 1, the InP sets up and maintains the physical equipments and substrate resources such as optical fibers, CPUs and bandwidths as well as network protocols. Herein, these physical equipments and resources compose the attributes of the InP, which serve to discover resources for SPs [6]. The SP, pursuant to service demands of clients, creates Virtual Network Requests (VNR) (A VNR is a combination of Virtual Nodes (VNs) and Virtual Links (VLs) [5]). It then discovers resources available in substrate networks by the attributes of InPs and selects appropriate ones for the deployment of VNRs [6].

How to effectively allocate resources of the substrate network to VNRs is a vital problem in network virtualization, which is often referred to as the Virtual Network Embedding (VNE) problem [5]. Explicitly, the VNE needs to (a) find a Substrate Node (SN) to meet the computing

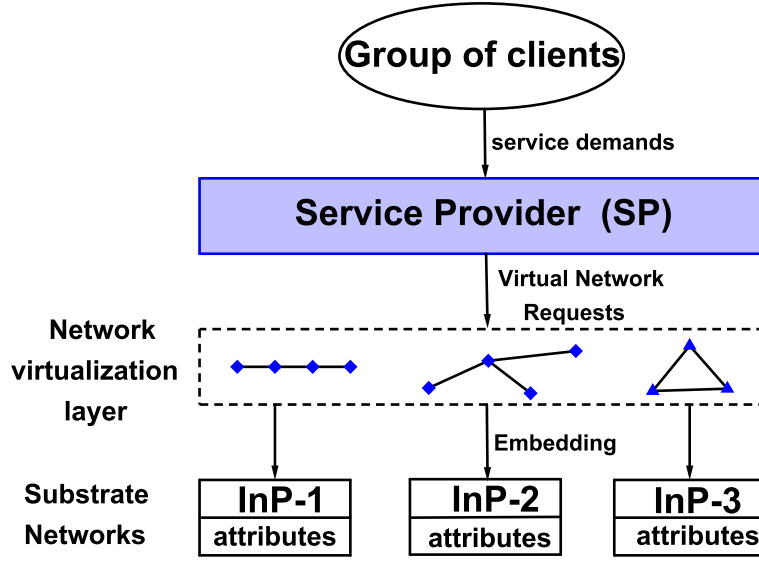


Fig. 1. The paradigm of network virtualization.

requirement of each VN, and (b) find a substrate path to satisfy the bandwidth requirement of each VL in a VNR. The former is also called *Node Mapping* and the latter is named *Link Mapping*. The VNE has been proven  $\mathcal{NP}$ -hard [10] and studied intensively [4, 7]–[9, 11]–[15]. These works introduce different methods like heuristic algorithms and Integer Linear Programming (ILP) models, *etc.*, and cover many aspects, such as distributed computing of the VNE and embedding across multiple substrate networks.

One of the key impediments in the general VNE problem is the topological heterogeneity of both VNRs and substrate networks [6]. However, this is not always true in many specific applications and substrate networks. For instance, the topologies of network service chains are paths [16], and there are many substrate optical rings (*i.e.*, cycles) [17]. For these applications and infrastructures, specialized cloud service providers outperforming the general SPs are desired, where dedicated algorithms, taking into account the topological characteristics of the VNRs and substrate networks, can be afforded. Besides, paths and cycles are two of the most fundamental topologies in network structures. Exploiting the characteristics of path and cycle embeddings is vital to tackle general topology embedding.

For example, if path and cycle embeddings can be effectively solved, we can decompose a

general VNR into paths and cycles and then embed them on the specialized platforms, as shown in Fig. 2, to boost the performance of the VNE. (The feasibility of embedding a VNR across multiple substrate networks has been verified in [9], which makes the idea of decomposition practicable.)

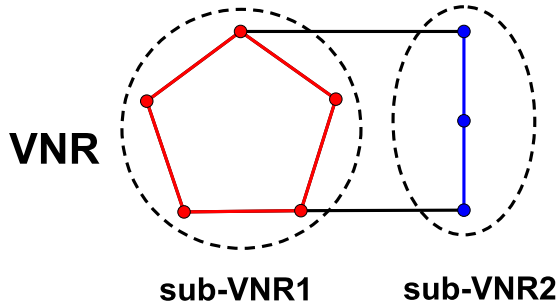


Fig. 2. A general VNR decomposed into two sub-VNRs, one cycle and one path.

In the wake of the idea of the two special embeddings (paths and cycles), there are some important questions not answered yet: How hard are they? Still  $\mathcal{NP}$ -hard or there exist polynomial-time algorithms to solve them exactly or approximately? In this paper, we comprehensively investigate path and cycle embeddings from theoretical hardness analysis to practical algorithm design. The main contributions of this work are summarized as follows.

- From the theoretical perspective, we formally define the decision problem  $Emb(G^s, G^r)$  of a single VNR  $G^r$  embedded onto a substrate network  $G^s$ . For path embedding, we provided a polynomial-time algorithm to optimally solve the path-to-path embedding. For general topology, we proved  $Emb(G^s, P^r)$  is  $\mathcal{NP}$ -hard and further analyzed its inapproximability. For cycle Embedding, we proved that  $Emb(C^s, C^r)$  can be polynomial-timely solved while the problems of maximizing acceptance ratio and revenue are still strongly  $\mathcal{NP}$ -hard.
- From the perspective of practical algorithm design, customized algorithms have been proposed by taking into account the nature of each embedding type. Following the idea of decomposing a substrate network into 'paths', we transformed this problem into a Multiple Knapsack Problem (MKP) and Multi-Dimensional Knapsack Problem (MDKP), and developed an efficient and effective MKP-MDKP-based algorithm to solve it. For cycle embedding, we proposed a Weighted Directed Auxiliary Graph (WDAG) and succeeded to

establish an one-to-one relation between each directed cycle in WDAG and each feasible embedding. Based on that, a polynomial-time algorithm is herein devised to achieve the least-resource-consuming embedding. To the best of our knowledge, it is the first time that the idea of embedding decomposition is proposed.

The rest of this paper is organized as follows. Section II briefly introduces the related work and our motivation. We present the network models and the formal description of the VNE problem in Section III. Then for the theoretical perspective of path embedding, we provide the proof of  $\mathcal{NP}$ -hardness and inapproximability in Section IV. For the practical algorithm, we present our design framework and devise the MKP-MDKP-based algorithms in Section V. For cycle embedding, Section VI elaborates the construction of WDAG, characterizes the one-to-one relation between directed cycles and feasible embedding ways, and further devises the specialized cycle-embedding algorithm. We conduct simulations under different scenarios in Section VII to demonstrate the superiority of our proposed algorithms over the existing general algorithms. Finally, Section VIII summarizes this paper.

## II. RELATED WORK AND MOTIVATION

The VNE, as the main challenging problem in network virtualization, drew a lot of attentions of researchers. In [5], the authors expanded the roles of the SP and InP in the paradigm of network virtualization and proposed a novel classification scheme for current VNE algorithms. Another comprehensive survey [6] elaborated and emphasized the importance of resource discovery and allocation of the VNE. Many solutions to the VNE problem have been proposed in the literature [4, 7]–[9, 11]–[15] including heuristic-based, ILP, *etc.* Later in [4, 7], researchers found that the topology information of VNRs and substrate networks can be utilized to improve the performance of the VNE. The authors of [7] applied a Markov random walk model, analogous to the idea of PageRank [18], to rank network nodes based on its resource and topological attributes. In [4], customized embedding algorithms for some special classes of topologies have been investigated and proven more effective than the general algorithms. Although VNRs may have arbitrary topologies, the network structures of some key applications and infrastructures are of common topologies *e.g.*, paths and cycles [16, 17]. However, few relevant works intentionally pay attention

to the two special but relatively common topologies in the VNE problem. Besides, paths and cycles are two of the most fundamental topologies in network structures. Since the general VNE problem is computationally hard, it is a pragmatic way to decompose VNRs into several specific substructures of paths and cycles and then effectively embed them separately. Embedding across multiple substrate networks and distributed embeddings of the VNE have been shown to be feasible [5, 9]. This makes the idea of decomposing VNRs and embedding separately practicable. Therefore, a devoted study to explore the characteristics of path and cycle embeddings is desired, which has not yet been researched.

In this work, we shall systematically, from theoretical hardness analysis to practical algorithm design, investigate the VNE problem for the two special topologies.

### III. NETWORK MODELS AND PROBLEM DESCRIPTION

In this section, we first present the network models considered in this paper and then give the formulation of the VNE. Some necessary notations are summarized in Table I.

#### A. Network Models

1) *Substrate Network*: In this paper, the substrate network  $G^s(V^s, E^s)$  is an undirected connected graph. Usually, there are two main resources on the substrate network managed by the InP, the computing capabilities of the SNs and the bandwidths of the SLs. Here, we denote by  $CPU(v^s)$  the computing capability of each SN  $v^s$ , and denote  $BW(e^s)$  as the bandwidth of SL  $e^s$ . Figure 3 gives an example of a 4-node substrate network, where each number in the square indicates the  $CPU$  capability of the SN and each number beside the SL indicates its  $BW$ .

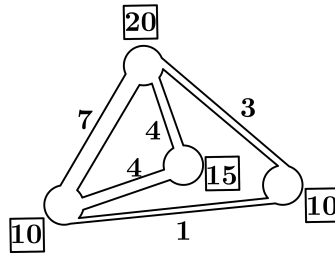


Fig. 3. A 4-node substrate network.

TABLE I  
NOTATIONS

Notation	Description
$G^s(V^s, E^s)$	A substrate network, where $V^s$ is the set of SNs, and $E^s$ is the set of Substrate Links (SLs)
$v^s$	$v^s \in V^s$ , a Substrate Node (SN)
$e^s$	$e^s \in E^s$ , a Substrate Link (SL)
$v_i^s v_j^s$	$v_i^s v_j^s \in E^s$ , the SL connecting $v_i^s \in V^s$ and $v_j^s \in V^s$
$\mathcal{P}_{v_i^s v_j^s}$	The set of all substrate paths from $v_i^s$ to $v_j^s$ in $G^s$
$P_{v_i^s v_j^s}$	$P_{v_i^s v_j^s} \in \mathcal{P}_{v_i^s v_j^s}$ , a substrate path from $v_i^s$ to $v_j^s$
$P^s$	A substrate network which is a path with $ P^s $ SLs on it
$C^s(v_1^s, \dots, v_m^s v_1^s)$	A substrate cycle with clockwise order of SNs <i>i.e.</i> , starting from $v_1^s$ clockwise to $v_m^s$ , where $m$ is the number of SNs on $C^s$
$G^r(V^r, E^r)$	A Virtual Network Request (VNR), where $V^r$ is the set of VNs and $E^r$ is the set of VLS
$v^r$	$v^r \in V^r$ , a Virtual Node (VN)
$e^r$	$e^r \in E^r$ , a Virtual Link (VL)
$v_i^r v_j^r$	$v_i^r v_j^r \in E^r$ , the VL connecting $v_i^r \in V^r$ and $v_j^r \in V^r$
$P^r(v_1^r v_2^r \dots v_n^r)$	A path VNR with $n$ VNs, <i>i.e.</i> , from $v_1^r$ , through $v_j^r, 2 \leq j \leq n-1$ , to $v_n^r$ .
$ P^r $	The length of $P^r$ , <i>i.e.</i> , the number of VLS in $P^r$
$C^r(v_1^r, \dots, v_n^r v_1^r)$	A cycle VNR with clockwise order of VNs, where $n$ is the number of VNs on $C^r$
$CPU(v^s)$	The CPU capacity of SN $v^s$
$BW(e^s)$ or $BW(v_i^s v_j^s)$	The bandwidth capacity of SL $e^s$ or $v_i^s v_j^s$
$CPU(v^r)$	The CPU demand of VN $v^r$
$BW(e^r)$ or $BW(v_i^r v_j^r)$	The bandwidth demand of SL $e^r$ or $v_i^r v_j^r$
$\mathcal{F}_{v^r}$	$\{v^s \in V^s   CPU(v^s) \geq CPU(v^r)\}$ , <i>i.e.</i> , the set of feasible SNs whose CPUs are not smaller than VN $v^r$ 's
$\mathcal{F}_{e^r}$ or $\mathcal{F}_{v_i^r v_j^r}$	$\{e^s \in E^s   BW(e^s) \geq BW(e^r) \text{ or } BW(v_i^r v_j^r)\}$ , <i>i.e.</i> , the set of feasible SLs whose bandwidths are not smaller than $e^r$ or $v_i^r v_j^r$
$v^r \rightarrow v^s$	$v^r$ is embedded on $v^s$
$e^r \rightarrow e^s$	The embedded substrate path of $e^r$ which passes through $e^s$

2) *Virtual Network Request*: A VNR  $G^r(V^r, E^r)$  is still modeled by an undirected connected graph, which is constructed by the SP according to the service demands of clients. The demanded computing capability of each  $v^r \in V^r$  is  $CPU(v^r)$ . The demanded bandwidth of each  $e^r \in E^r$  is  $BW(e^r)$ . In this work, we focus on two special topologies of VNRs *i.e.*, paths and cycles. Figure 4 illustrates a path VNR and a cycle VNR, where, similarly, the numbers in squares (beside VNs respectively) indicate the corresponding  $CPU$ s of VNs ( $BW$ s of VLs respectively).

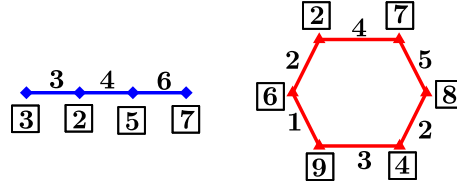


Fig. 4. An example of path and cycle VNRs.

## B. Problem Formulation

1) *Constraints*: As mentioned above, the VNE problem contains two constraints, *Node Mapping* and *Link Mapping* constraints.

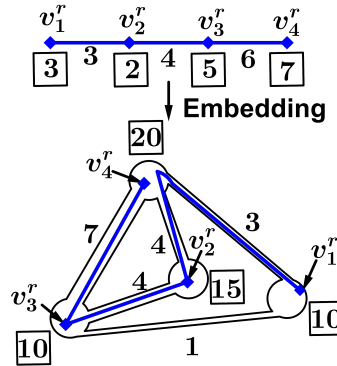


Fig. 5. Node and link mapping.

a) *Node Mapping Constraint*: For the node mapping, given the  $G^s(V^s, E^s)$  and a VNR  $G^r(V^r, E^r)$ , each VN  $v^r \in V^r$  must be embedded onto such an SN  $v^s \in V^s$ , *i.e.*,  $v^r \rightarrow v^s$ , that  $CPU(v^s) \geq CPU(v^r)$ . Meanwhile, for each SN  $v^s$  on which some VNs  $v^r \in V^r$  are embedded,



$CPU(v^s) \geq \sum_{v^r \rightarrow v^s} CPU(v^r)$ . Figure 5 demonstrates the node mapping where the numbers in squares indicate the  $CPU$  requirements or capabilities of VNs or SNs respectively. Besides, any two VNs  $v_i^r$  and  $v_j^r$  of a same VNR can not be embedded on a same SN  $v^s$ , *i.e.*, if  $v_1^r \rightarrow v^s$  and  $v_2^r \rightarrow v^s$  then  $v_1^r = v_2^r$  [8].

*b) Link Mapping Constraint:* For each VL  $v_i^r v_j^r$ , assuming  $v_i^r \rightarrow v_i^s$  and  $v_j^r \rightarrow v_j^s$ ,  $v_i^r v_j^r$  should be embedded on a substrate path  $P_{v_i^s v_j^s} \in \mathcal{P}_{v_i^s v_j^s}$ , and for each SL  $e^s$  on  $P_{v_i^s v_j^s}$ ,  $BW(e^s) \geq BW(v_i^r v_j^r)$ . Meanwhile, for each SL  $e^s$ , through which the embedded substrate paths of VLs  $e^r$  pass, *i.e.*,  $e^r \rightarrow e^s$ ,  $BW(e^s) \geq \sum_{e^r \rightarrow e^s} BW(e^r)$  as shown in Fig 5.

Here, we give the definition of the decision problem  $Emb(\cdot, \cdot)$  as follows.

**Definition 1:**  $Emb(G^s, G^r)$  is such a decision problem that its answer is **Yes** iff the VNR  $G^r$  can be embedded on the substrate network  $G^s$  satisfying the node and link mapping constraints, and **No** otherwise.

The definition can be extended to a set of  $n$  VNRs:  $Emb(G^s, \{G_1^r, G_2^r, \dots, G_n^r\})$  whose answer is **Yes** iff all of  $G_{i,1 \leq i \leq n}^r(V_i^r, E_i^r)$  can be embedded on the  $G^s$  while simultaneously satisfying the node and link mapping constraints, and **No** otherwise. Here, the node mapping constraint of a set of VNRs embedded on a substrate network is that for each SN  $v^s$ ,  $CPU(v^s) \geq \sum_{v^r \rightarrow v^s} CPU(v^r)$ ,  $\forall v^r \in V_1^r \cup V_2^r \cup \dots \cup V_n^r$ , and the link constraint means for each SL  $e^s$ ,  $BW(e^s) \geq \sum_{e^r \rightarrow e^s} BW(e^r)$ ,  $\forall e^r \in E_1^r \cup E_2^r \cup \dots \cup E_n^r$ .

*2) Objective Functions:* There are two main objective functions in the study of the VNE problem: The acceptance ratio and the revenue.

*a) The Acceptance Ratio (AcR):* Given a substrate network  $G^s$  and a set  $\{G_1^r, G_2^r, \dots, G_n^r\}$  of VNRs, the objective is to maximize the number of VNRs that can be embedded on the  $G^s$ . We denote by **AcR** the acceptance ratio problem as formally defined below.

$$\text{Maximize } |S| \quad (\mathbf{AcR}),$$

$$\text{s.t. } Emb(G^s, S) \text{ is } \mathbf{Yes},$$

$$S \subseteq \{G_1^r, G_2^r, \dots, G_n^r\}.$$

*b) The Revenue (Rev):* Each VNR  $G_i^r$  is associated with a revenue  $w_i$ . Its objective is to maximize the total revenue of VNRs that can be embedded on  $G^s$ . We denote by **Rev** the

revenue problem defined below.

$$\text{Maximize } \sum_{G_i^r \in S} w_i \quad (\mathbf{Rev}),$$

$$\mathbf{s.t.} \quad Emb(G^s, S) \text{ is Yes,}$$

$$S \subseteq \{G_1^r, G_2^r, \dots, G_n^r\}.$$

One may notice that AcR actually is a special case of Rev by setting each revenue  $w_i$  to be one. In this paper, the set of VNRs  $\{G_1^r, G_2^r, \dots, G_n^r\}$  is particularly a path set or a cycle set, *i.e.*,  $\{P_1^r, P_2^r, \dots, P_n^r\}$  or  $\{C_1^r, C_2^r, \dots, C_n^r\}$ .

#### IV. THEORETICAL HARDNESS ANALYSIS OF PATH EMBEDDING

In this section, we explore the theoretical harness of the path embedding problem. Given a substrate network  $G^s$  and a path VNR  $P^r$ , an elementary and essential question is how hard  $Emb(G^s, P^r)$  is. Before answering it, we give some terminologies in graph theory as follows.

- *Trail*: In a graph  $G(V, E)$ , a *trail* is such a subgraph that can be expressed as a sequence of vertices " $v_1, v_2, \dots, v_n$ ", where for any two adjacent vertices  $v_i$  and  $v_{i+1}$ ,  $1 \leq i \leq n - 1$ ,  $v_i v_{i+1}$  is an edge, *i.e.*,  $v_i v_{i+1} \in E$  and no repeated edge occurs in the trail, *i.e.*, the pair  $(v_i, v_{i+1})$  only occurs once in the trail. A *closed trail* is such a trail " $v_1, v_2, \dots, v_n$ ", with  $v_1 = v_n$ . Figure 6 illustrates a trail  $T_1$  (left in blue), a closed trail  $T_2$  (right in red), and the corresponding vertex sequences.

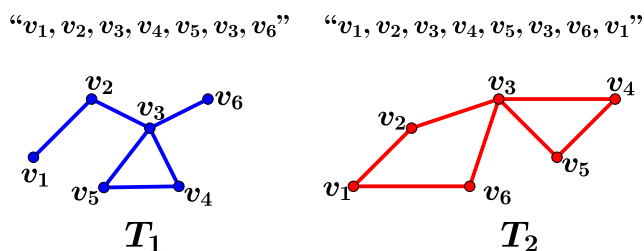


Fig. 6. An example of trail and closed trail.

- *Eulerian trail*: In a graph  $G$ , an Eulerian trail is a trail traversing all edges. Notice that not every graph has an Eulerian trail. An *Eulerian circuit* is a closed Eulerian trail. Therefore,

in Fig. 6, if we treat the two trails as two graphs (not subgraphs), then they can also be regarded as an Eulerian trail and an Eulerian circuit respectively.

- *Eulerian graph*: A graph  $G$  is called Eulerian graph iff it has an Eulerian circuit. In Fig. 6, if we treat  $T_2$  as a graph, then  $T_2$  is an Eulerian graph.
- *Supereulerian graph*: A graph  $G$  is called Supereulerian graph iff it contains a spanning subgraph<sup>1</sup> which is an Eulerian graph. Similarly, if a graph  $G$  is formed by adding one edge  $v_2v_6$  to  $T_2$  in Fig. 6, *i.e.*,  $G = T_2 + v_2v_6$ , then  $G$  is a Supereulerian graph.

#### A. The Hardness of $Emb(G^s, P^r)$

Let  $G^s(V^s, E^s)$  and  $P^r(V^r, E^r)$  be the substrate network and the path VNR respectively. To characterize the hardness of  $Emb(G^s, P^r)$ , we adopt the *uniform setting* of the *CPUs* and *BWs*.

**Definition 2: uniform setting:** ①  $\forall v^s \in V^s, CPU(v^s) = 2$  and  $\forall e^s \in E^s, BW(e^s) = 1$ ; ②  $\forall v^r \in V^r, CPU(v^r) = 1$  and  $\forall e^r \in E^r, BW(e^r) = 1$ .

The number of VNs that can be embedded is at most  $|V^s|$ . Thus, a special case of  $Emb(G^s, P^r)$  is to answer whether a path VNR  $P^r(v_1^r v_2^r, \dots, v_{|V^s|}^r)$  can be embedded onto  $G^s(V^s, E^s)$ . The following lemma gives the necessary and sufficient condition that whether a  $P^r$  of  $|V^s|$  VNs can be embedded onto  $G^s$ .

**Lemma 1:** In the uniform setting, a  $P^r$  of  $|V^s|$  VNs can be embedded on the  $G^s$  **if and only if**  $G^s$  has a trail traversing all SNs. In other words,  $G^s$  contains a spanning subgraph having an Eulerian trail.

**Proof.** If  $P^r(v_1^r v_2^r, \dots, v_{|V^s|}^r)$  can be embedded on  $G^s$  by, *w.l.o.g.*,  $v_i^r \rightarrow v_i^s, \forall i$ , then we can express this embedding way as a sequence " $v_1^s P_{v_1^s v_2^s} v_2^s P_{v_2^s v_3^s} \dots v_{|V^s|}^s$ ". This sequence, if expressing each  $P_{v_i^s v_{i+1}^s}$  by its vertex sequence, in fact is a trail which traverses all SNs in  $G^s$  (but not necessary all SLs).

Conversely, if  $G^s$  has a trail, say  $T$ , traversing all SNs, let, *w.l.o.g.*, " $v_1^s, \dots, v_2^s, \dots, v_i^s, \dots, v_{|V^s|}^s$ " be  $T$ 's corresponding vertex sequence. By the definition of trail, there must be a path connecting  $v_i^s$  and  $v_{i+1}^s$ , say  $P_{v_i^s v_{i+1}^s}, \forall i$ , and  $P_{v_i^s v_{i+1}^s} \cap P_{v_j^s v_{j+1}^s} = \emptyset, \forall i \neq j$ . Therefore we can embed  $P^r$  by " $v_1^s P_{v_1^s v_2^s} v_2^s P_{v_2^s v_3^s} \dots v_{|V^s|}^s$ ". ■

<sup>1</sup>A spanning subgraph of  $G$  is such a subgraph that contains all vertices of  $G$ .

**Lemma 2:** The hardness of determining whether a graph contains a spanning subgraph having an Eulerian trail (or *SSET* in short) is equivalent<sup>2</sup> to that of determining whether a graph is a Supereulerian graph (or *SG*).

Consequently, by Lemmas 1 and 2, we have Theorem 1.

**Theorem 1:** In the uniform setting, the hardness of  $Emb(G^s, P^r)$  problem is not less than that of determining whether  $G^s$  is a Supereulerian graph.

Unfortunately, given a graph  $G$ , it is  $\mathcal{NP}$ -hard to determine whether  $G$  is a Supereulerian graph [19]. Thus,  $Emb(G^s, P^r)$  is also  $\mathcal{NP}$ -hard.

### B. The inapproximability of path embedding

After obtaining the  $\mathcal{NP}$ -hardness of  $(G^s, P^r)$ , immediately, we can get the inapproximability of the AcR and Rev problems of path embedding in the uniform setting as follows.

**Theorem 2:** For path embedding in the uniform setting, both the AcR and Rev problems have an  $\mathcal{NP}$ -hard gap  $[\epsilon, 1]^3$ ,  $\forall 0 < \epsilon < 1$ , i.e., unless  $\mathcal{NP} = \mathcal{P}$ , there is no polynomial-time algorithm of an approximation ratio  $\frac{1}{\epsilon}$  for both problems.

**Proof.** Given an instance of the AcR problem consisting of a  $G^s$  and a  $P^r$ , if the answer of  $Emb(G^s, P^r)$  is **Yes**, then  $OPT_{AcR} \geq 1$ , otherwise,  $OPT_{AcR} < \epsilon$ , where  $OPT_{AcR}$  is the optimal solution of the AcR. Since  $Emb(G^s, P^r)$  is  $\mathcal{NP}$ -hard, the AcR problem in the uniform setting has an  $\mathcal{NP}$ -hard gap<sup>4</sup>  $[\epsilon, 1]$ . For the Rev, as the AcR is its special case, the proof also follows. ■

In reality that *CPU* and *BW* are arbitrary, the path embedding should be more difficult than in the uniform setting. Here, we present the following theorem to explicitly show the inapproximability of both the AcR and Rev for path embedding in the realistic settings.

<sup>2</sup>The equivalence is under the polynomial-time Turing reduction.

<sup>3</sup>If an optimization problem has an  $\mathcal{NP}$ -hard gap  $[\alpha, \beta]$ , then it has no polynomial-time algorithm of an approximation ratio  $\frac{\beta}{\alpha}$  unless  $\mathcal{NP} = \mathcal{P}$  [20].

<sup>4</sup>For a maximization problem  $\Pi$  and an  $\mathcal{NP}$ -hard decision problem  $\Lambda$ , if  $\Lambda$  is **Yes**  $\Rightarrow OPT_{\Pi} \geq \beta$  and  $\Lambda$  is **No**  $\Rightarrow OPT_{\Pi} < \alpha$ , where  $OPT_{\Pi}$  is the optimal solution of  $\Pi$ , then  $\Pi$  has an  $\mathcal{NP}$ -hard gap  $[\alpha, \beta]$  [20].

**Theorem 3:** For path embedding in reality, unless  $\mathcal{NP} \subseteq ZPTIME(n^{\text{polylog}(n)})^5$ , there is no  $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |E^s|)$  approximation for both the AcR and Rev problems, where  $|E^s|$  is the number of SLs in the substrate network.

**Proof.** First, we introduce the Edge-Disjoint Paths (EDP) problem: Given a connected graph  $G(V, E)$  and a set of pairs of vertices  $\{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$  on it, the objective of the EDP is to connect as many pairs as possible via edge-disjoint paths. Unless  $\mathcal{NP} \subseteq ZPTIME(n^{\text{polylog}(n)})$ , there is no  $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |E|)$  approximation for the EDP problem [21].

Now given an instance of the EDP, *i.e.*, a graph  $G(V, E)$  where  $V = \{v_1, \dots, v_i, \dots, v_n\}$  and a set of pairs of vertices  $\{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$ , we translate it into an instance of the AcR in path embedding as follows. To complete the translation, for each  $v_i \in V$ , we first denote by  $N_{v_i}$  the total number of  $v_i$  appearing in the sets  $\{s_i\}_{i=1}^k$  and  $\{t_i\}_{i=1}^k$ .

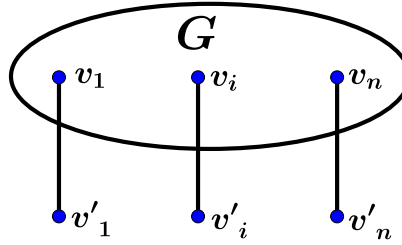


Fig. 7. The constructed substrate network.

The construction of the substrate network  $G^s$  is that: The set of SNs  $V^s$  consists of two parts,  $V = \{v_1, \dots, v_i, \dots, v_n\}$  and  $V' = \{v'_1, \dots, v'_i, \dots, v'_n\}$  where  $V'$  is a copy of  $V$ . The set of SLs  $E^s$  also consists of two parts,  $E$  and  $\{v_i v'_i\}_{i=1}^n$  as shown in Fig. 7.

The setting of *CPU*s and *BW*s is as follows.

- For each  $v_i \in V$ ,  $CPU(v_i) = 1$  and for each  $e \in E$ ,  $BW(e) = 1$ .
- For  $V'$ ,  $CPU(v'_1) = C_{v_1} \times N_{v_1}$  where  $C_{v_1} = 2$  and for  $2 \leq i \leq n$ ,  $CPU(v'_i) = C_{v_i} \times N_{v_i}$  where  $C_{v_i} = CPU(v'_{i-1}) + 1$ .
- For  $\{v_i v'_i\}_{i=1}^n$ ,  $BW(v_n v'_n) = B_{v_n} \times N_{v_n}$  where  $B_{v_n} = 2$  and for  $n-1 \geq i \geq 1$ ,  $BW(v_i v'_i) = B_{v_i} \times N_{v_i}$  where  $B_{v_i} = BW(v_{i+1} v'_{i+1}) + 1$ .

<sup>5</sup> $ZPTIME(n^{\text{polylog}(n)})$  is the set of languages that have randomized algorithms that always give the correct answer and have expected running time  $n^{\text{polylog}(n)}$ .

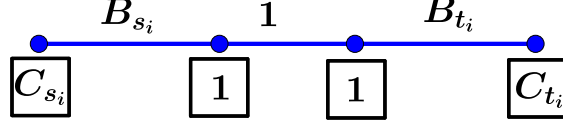


Fig. 8. The  $i$ -th path VNR  $P_i^r$ .

The path VNRs set is  $\{P_1^r, \dots, P_i^r, \dots, P_k^r\}$ , where the  $i$ -th  $P_i^r$ , which corresponds to  $(s_i, t_i)$ , consists of 4 VNs and its  $CPU$ s and  $BW$ s requirements are shown as in Fig. 8

From the setting of  $CPU$ s and  $BW$ s, we can see that  $C_{v_i}$  is increasing with the  $i$  growing while  $B_{v_i}$  is decreasing. Thus, following the constraints of node and link mappings, it is easy to see that the two ends of the  $P_i^r$  must be embedded on the copies of  $s_i$  and  $t_i$  respectively, if the  $P_i^r$  is embedded.

Hence, for any pairs we can connect by edge-disjoint paths in the instance of the EDP, we can embed their corresponding path VNRs in the substrate network, vice versa.

Besides, since the number of SLs  $|E^s| = |E| + |V| \leq 3|E|$ , we get the inapproximability. ■

## V. PRACTICAL ALGORITHM DESIGN FOR PATH EMBEDDING

Theorem 3 implies that for path embedding, it is implausible to find a polynomial-time algorithms of a proper approximation ratio. Thus, we should turn our attention from developing approximation algorithms to designing better heuristic algorithms, which is able to capture the "nature" of the path VNE problem. The following lemma captures the "nature" to some extent.

**Lemma 3:** In the uniform setting, if the substrate network is a path denoted by  $P^s$ , given a set of  $\{P_1^r, P_2^r, \dots, P_n^r\}$ , then the AcR and Rev problems can be solved in polynomial time.

**Proof.** We use Fig. 9 to demonstrate the substrate path  $P^s$  and the set of path VNRs  $\{P_1^r, P_2^r, \dots, P_n^r\}$ . In this special case, we can regard  $P^s$  as a "knapsack" with a capacity of  $|P^s|$  (edge number), and the  $n$  path VNRs  $P_{i,1 \leq i \leq n}^r$  as  $n$  items with a size of  $|P_{i,1 \leq i \leq n}^r|$  respectively.

The AcR can be easily solved in this way: First arrange the  $n$  path VNRs in the increasing order of their sizes, then sequentially pack them into the "knapsack" until cannot (This embedding is feasible since each middle SN  $v^s$  of  $CPU(v^s) = deg(v^s) = 2$ , and at most two VNs will be embedded on  $v^s$ ).

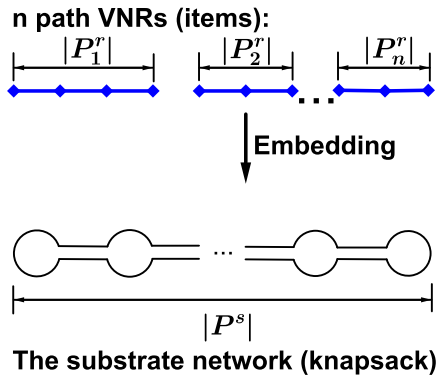


Fig. 9. An example of the substrate network being a path.

The Rev problem in this special case is equivalent to the classical 0-1 Knapsack Problem (KP) and it thus can be solved by a dynamic programming algorithm of time complexity  $\mathcal{O}(n|P^s|)$  [22]<sup>6</sup>. ■

Thus, in the uniform setting, if  $G^s$  is a path, then the AcR and Rev problems can be easily solved by leveraging KP. This result relatively reflects some essentials of the path VNE problem that can be regarded as "packing" (embedding) a set of "items" (VNRs) into a special "knapsack" (the substrate network). Inspired by this, given a substrate network  $G^s$  and a set of  $\{P_1^r, P_2^r, \dots, P_n^r\}$ , we propose a framework of algorithm design for the realistic settings. The main idea is described as follows. First, we decompose  $G^s$  into several substrate paths. This phase is thus called path decomposition. By regarding each substrate path as a knapsack and each  $P_j^r$  as an item with size  $|P_j^r|$  and profit  $w_j$ , and we then pack these items into multiple knapsacks, which can be formulated as a Multiple Knapsack problem (MKP). Finally, we assign the  $CPU$  and  $BW$  resources to those packed path VNRs, and it corresponds to the Multi-Dimensional Knapsack Problem (MDKP). To this end, we review the two well-studied MKP and MDKP.

### Multiple Knapsack Problem (MKP)

MKP [22] is a classical variation of KP. In MKP, there are a set of knapsacks  $M := \{1, \dots, i, \dots, m\}$  each with positive capacities  $b_i$ , and a set of items  $N := \{1, \dots, j, \dots, n\}$  each

<sup>6</sup>Although KP is  $\mathcal{NP}$ -hard, for the substrate network, whose space complexity is  $|P^s|$  (not  $\log(|P^s|)$ ), the dynamic programming algorithm runs in polynomial time (rather than pseudo-polynomial time).

with size  $s_j \geq 0$  and profit  $w_j \geq 0$ . The goal is to find a subset of the  $n$  items of maximum profit which can be packed into the  $m$  knapsacks.

In this paper, we measure the time complexity of solving an instance of MKP by its numbers of knapsacks and items,  $m$  and  $n$  respectively, denoted by  $T_{MKP}(m, n)$ .

### Multi-Dimensional Knapsack Problem (MDKP)

MDKP [22] is another well-known variation of KP. In  $d$  dimensional MDKP denoted by  $d$ -DKP, there are a knapsack of  $d$ -dimensional positive capacity attributes  $(b_1, \dots, b_i, \dots, b_d)$  and a set of items  $N := \{1, \dots, j, \dots, n\}$  each with profit  $w_j$  and  $d$ -dimensional size attributes  $(s_{j1}, \dots, s_{ji}, \dots, s_{jd})$ , where all of  $b_i$  and  $w_j$  and  $s_{ji}$  are non-negative. The goal is to find a subset of the  $n$  items of maximum profit which can be packed into the knapsack while not exceeding each of  $d$ -dimensional capacity attributes.

In this paper, we measure the time complexity of solving an instance of MDKP by its numbers of dimensions  $d$  and items  $n$  respectively, denoted by  $T_{MDKP}(d, n)$ . Next, we assume that the substrate network  $G^s(V^s, E^s)$  and a set of path VNRs  $\{P_1^r, P_2^r, \dots, P_n^r\}$  are given as the input of our algorithm.

#### A. Path Decomposition Phase

In this phase, we decompose the substrate network into a set of substrate paths. These decomposed paths are treated as the multiple knapsacks for the optimization of the next phase. Intuitively, if these "knapsacks" are of bigger capacities, *i.e.*, much longer, the embedding optimization in the next phase by the MKP will be better. We extract a substrate path  $P^s$  by finding the longest path in a Depth-first Search Tree (DST) of  $G^s$ . We repeat the process by keeping extracting substrate paths until  $G^s$  is completely decomposed into a set of  $P_i^s$ ,  $1 \leq i \leq m$ , where  $m$  is the number of substrate paths obtained.

#### B. Embedding by MKP

After the path decomposition phase, we regard each path VNR  $P_j^r$  as an item with size  $|P_j^r|$  and profit  $w_j$ , and treat each substrate path  $P_i^s$  as a knapsack with capacity  $|P_i^s|$  as shown in Fig. 10. Obviously, it is an  $m$ -knapsacks- $n$ -items MKP, where  $m \leq |E^s|$ .



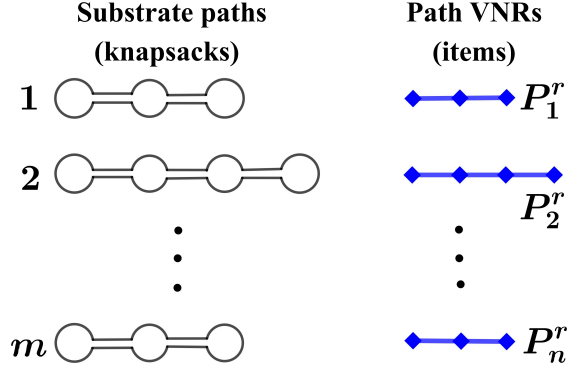


Fig. 10. MKP embedding.

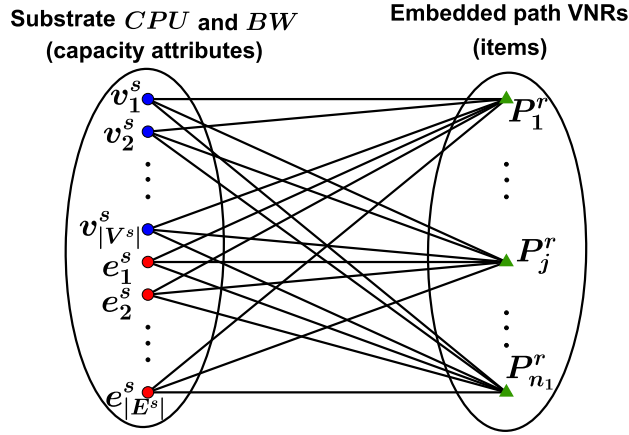


Fig. 11. MDKP assignment.

### C. Resource Assignment by MDKP

After embedding some path VNRs by MKP without considering *CPUs* and *BWs*, we need to assign the corresponding demanded *CPU* and *BW* resources to as many embedded path VNRs as possible. Here, we treat each embedded  $P_j^r$  as an item with  $(|V^s| + |E^s|)$ -dimensional size attributes:  $(s_{jv_1^s}, s_{jv_2^s}, \dots, s_{jv_{|V^s|}^s}, s_{je_1^s}, s_{je_2^s}, \dots, s_{je_{|E^s|}^s})$ . For the first  $|V^s|$  attributes, if some VNs  $v^r$  of the  $P_j^r$  are embedded on an SN, say  $v_k^s$ , then the attribute  $s_{jv_k^s} = \sum_{v^r \rightarrow v_k^s} CPU(v^r)$ , otherwise 0. For the last  $|E^s|$  attributes, if a VL  $e^r$  of the  $P_j^r$  is embedded on an SL, say  $e_l^s$ , then the attribute  $s_{je_l^s} = BW(e^r)$ , otherwise 0. Finally, the array of capacity attributes of the knapsack is that:  $(CPU(v_1^s), \dots, CPU(v_{|V^s|}^s), BW(e_1^s), \dots, BW(e_{|E^s|}^s))$ . How to assign resources to these

embedded path VNRs to maximize revenue is obviously a  $(|V^s| + |E^s|)$ -DKP with  $n_1$  items as shown in Fig. 11, where  $n_1 \leq n$  is the number of embedded path VNRs by MKP.

#### D. Final Assembled Algorithm and Time Complexity

After the resource assignment, we can update the *CPU* and *BW* of each SN and SL, resulting in a remained substrate network. We then continue the whole process, from path decomposition to resource assignment, to embed the rest path VNRs until no more paths can be embedded. The final assembled algorithm is shown in Algorithm 1.

---

#### Algorithm 1: Procedure of Path-Embedding (PE)

---

**Input** : A substrate network  $G^s(V^s, E^s)$  and a set of path VNRs  $\{P_1^r, P_2^r, \dots, P_n^r\}$ .

**Output**: Final revenue.

```

1 set Flag  $\leftarrow$  true;
2 while Flag do
3   run Path Decomposition Phase on  $G^s$ ;
4   run Embedding by MKP;
5   run Resource Assignment by MDKP;
6   if no path VNR can be embedded then
7      $\left|$  Flag  $\leftarrow$  false;
8   update the substrate network  $G^s$ ;

```

---

The time complexity of extracting one substrate path by constructing a *DST* is  $\mathcal{O}(|V^s| + |E^s|)$  and there are at most  $|E^s|$  paths extracted. Hence, the total time complexity is  $\mathcal{O}(|E^s| \times (|V^s| + |E^s|))$ . The time complexity of embedding by MKP and resource assignment by MDKP, depending on the algorithms for solving MKP and MDKP, are bounded by  $T_{MKP}(|E^s|, n)$  and  $T_{MDKP}(|V^s| + |E^s|, n)$  respectively [22], where  $n$  is the number of path VNRs. Network operators can select algorithms according to their computing capability. ([22] elaborates most of the current algorithms for MKP and MDKP.) The time complexity for updating the substrate network is  $\mathcal{O}(|V^s| + |E^s|)$ , and we repeat the **while**-loop at most  $n$  times. Thus, the time complexity of Procedure PE is  $\mathcal{O}(n \times (|E^s|^2 + T_{MKP}(|E^s|, n) + T_{MDKP}(|V^s| + |E^s|, n)))$ .

## VI. CYCLE EMBEDDING

For cycle embedding, a substrate network of cycle-like topology is naturally more suitable just like path embedding. Meanwhile, there are many substrate optical rings [17]. In this section, we investigate the cycle-to-cycle embedding problem, *i.e.*,  $Emb(C^s, \{C_1^r, C_2^r, \dots, C_n^r\})$  where  $C^s$  is the substrate cycle and  $\{C_1^r, C_2^r, \dots, C_n^r\}$  is the set of cycle VNRs (*CPUs* and *BWs* are arbitrary). In this paper, we focus on a natural embedding way called *Simplex Cycle Embedding*, which evenly embeds VNs and VLs on the substrate cycle.

**Definition 3: Simplex Cycle Embedding:** We assume the substrate cycle is  $C^s(v_1^s, \dots, v_m^s, v_1^s)$ . Given a starting SN  $v_i^s \in C^s$  with one direction *dir* (clockwise or anticlockwise denoted by "+" or "-" respectively), we arrange all SNs of  $C^s$  in such a sequence, denoted by  $Seq(v_i^s, dir)$ , that for *dir* = "+",  $[v_i^s, v_{i+1}^s, v_{i+2}^s, \dots, v_{i-1}^s]$ , and for *dir* = "-",  $[v_i^s, v_{i-1}^s, v_{i-2}^s, \dots, v_{i+1}^s]$ , where all arithmetical operations of subscripts of SNs are modulo  $m$ .

Given a cycle VNR  $C^r(v_1^r, \dots, v_n^r, v_1^r)$ , *Simplex Cycle Embedding* is that:  $v_j^r \rightarrow v_{i_j}^s, 1 \leq j \leq n$ , which can be expressed as  $\begin{pmatrix} v_1^r & \dots & v_j^r & v_{j+1}^r & \dots & v_n^r \\ v_{i_1}^s & \dots & v_{i_j}^s & v_{i_{j+1}}^s & \dots & v_{i_n}^s \end{pmatrix}$ , where the node mapping follows one direction *dir*. More specifically,  $\forall j, v_{i_j}^s$  is ahead of  $v_{i_{j+1}}^s$  in  $Seq(v_{i_1}^s, dir)$ . Figure 12 illustrates a triangle VNR embedded on a substrate cycle in clockwise and anticlockwise simplex cycle embeddings, where the numbers beside VLs indicate the demanded *BWs*.

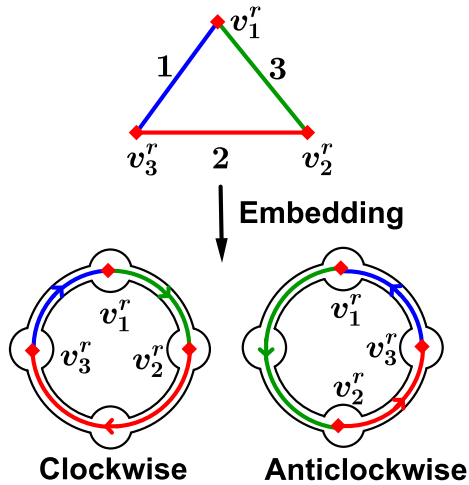


Fig. 12. Two simplex cycle embeddings: clockwise and anticlockwise.

Similarly, given a  $C^s$  and a  $C^r$ ,  $Emb(C^s, C^r)$  is the elementary and primary problem in simplex cycle embedding. The similar problem  $Emb(G^s, P^r)$  of path embedding is  $\mathcal{NP}$ -hard even in the uniform setting. Is  $Emb(C^s, C^r)$  also  $\mathcal{NP}$ -hard? Moreover, even the answer of  $Emb(C^s, C^r)$  is **Yes**, different embedding ways could result in different resource consumptions: In Fig. 12, the  $BW$  consumption of the clockwise is  $1 + 2 \times 2 + 3 = 8$  while that for the anticlockwise is  $2 \times 3 + 2 + 1 = 9$ . Can we efficiently find the least-resource-consuming embedding way? How about the AcR and Rev problems in cycle-to-cycle embedding?

Following the three important questions, we unfold this section as follows.

- First, we construct a Weighted Directed Auxiliary Graph (WDAG) in polynomial-time and prove that each of its directed cycles corresponds to a feasible simplex cycle embedding.
- Then, the minimum weighted directed cycle corresponds to the least-resource-consuming embedding, which can be obtained by dynamic programming in polynomial time.
- Finally, we prove that both the AcR and Rev problems are strongly  $\mathcal{NP}$ -hard, and thus devise effective heuristic algorithms to solve them.

Given a substrate cycle  $C^s(v_1^s, \dots, v_m^s v_1^s)$  and a cycle VNR  $C^r(v_1^r, \dots, v_n^r v_1^r)$ , for all  $v_j^r$ , let  $\mathcal{F}_{v_j^r} = \{v_i^s \in V^s | CPU(v_i^s) \geq CPU(v_j^r)\}$ , *i.e.*, the set of feasible SNs on which  $v_j^r$  can be embedded, and for all  $v_j^r v_{j+1}^r$ ,  $\mathcal{F}_{v_j^r v_{j+1}^r} = \{e^s \in E^s | BW(e^s) \geq BW(v_j^r v_{j+1}^r)\}$ , *i.e.*, the set of feasible SLs whose  $BW$  is not smaller than  $v_j^r v_{j+1}^r$ 's. If  $Emb(C^s, C^r)$  is **Yes** in simplex cycle embedding, there must exist an embedding way that  $v_1^r \rightarrow v_{i_1}^s \in \mathcal{F}_{v_1^r}$ , following one direction  $dir$ . With respect to the condition that  $v_1^r \rightarrow v_{i_1}^s \in \mathcal{F}_{v_1^r}$  and the embedding direction  $dir$ , we construct a WDAG denoted by  $\hat{G}_{dir}^{v_1^s}(\hat{V}, \hat{A})$ , where  $\hat{V}$  is the vertex set and  $\hat{A}$  is the arc set.

1) The vertex set  $\hat{V}$  comprises of  $n$  parts  $\{\hat{\mathcal{F}}_{v_j^r} |_{j=1}^n\}$ , where the  $j$ -th part  $\hat{\mathcal{F}}_{v_j^r}$  corresponds to the set  $\mathcal{F}_{v_j^r}$ . Except  $\hat{\mathcal{F}}_{v_1^r}$ , there is a one-to-one mapping, denoted by  $MP$ , between vertices in  $\hat{\mathcal{F}}_{v_j^r}$  and SNs in  $\mathcal{F}_{v_j^r}$ . In other words,  $\forall \hat{v} \in \hat{\mathcal{F}}_{v_j^r}$  one-to-one corresponds to  $MP(\hat{v}) \in \mathcal{F}_{v_j^r}$ . In  $\hat{\mathcal{F}}_{v_1^r}$ , there is only one vertex,  $\hat{v}_1$ , which corresponds to  $v_{i_1}^s$ , *i.e.*,  $MP(\hat{v}_1) = v_{i_1}^s$  as shown in Fig. 13.

2) The arc set  $\hat{A}$  is iteratively constructed as below: First starting at  $\hat{v}_1$ , for each such vertex in  $\hat{\mathcal{F}}_{v_2^r}$ , say  $\hat{v}_2$ , that satisfies two criteria with  $\hat{v}_1$ , an arc is constructed with  $\hat{v}_1$  as tail and  $\hat{v}_2$  as head. The two criteria are as follows. **Criterion 1:**  $MP(\hat{v}_1)$  is ahead of  $MP(\hat{v}_2)$  in  $Seq(v_{i_1}^s, dir)$ . **Criterion 2:** each SL  $e^s$ , lying in the segment from  $MP(\hat{v}_1)$  to  $MP(\hat{v}_2)$  following  $dir$ , belongs to

$\mathcal{F}_{v_1^r v_2^r}$ . Besides, a weight is assigned to this arc which equals  $|MP(\hat{v}_1) - MP(\hat{v}_2)| \times BW(v_1^r v_2^r)$ , where  $|MP(\hat{v}_1) - MP(\hat{v}_2)|$  is the number of SLs in the segment from  $MP(\hat{v}_1)$  to  $MP(\hat{v}_2)$  following  $dir$ . Next, for each such vertex in  $\hat{\mathcal{F}}_{v_2^r}$  with incoming edges, say  $\hat{v}_2$  and  $d^-(\hat{v}_2) > 0$ , we repeat the same procedure on it as we did for  $\hat{v}_1$ : Searching those vertices in  $\hat{\mathcal{F}}_{v_3^r}$  which satisfy the two criteria with  $\hat{v}_2$ ; and arcs are constructed with  $\hat{v}_2$  as tail; and weights are computed and assigned to these arcs. After some iterations, at certain vertex part, say  $\hat{\mathcal{F}}_{v_j^r}$ ,  $j < n$ , if there is no vertex in  $\hat{\mathcal{F}}_{v_j^r}$  whose indegree is greater than 0, then the whole process is terminated. Otherwise, we reach the vertex part  $\hat{\mathcal{F}}_{v_n^r}$ . For each such vertex in  $\hat{\mathcal{F}}_{v_n^r}$  with non-zero indegree, say  $\hat{v}_n$  and  $d^-(\hat{v}_n) > 0$ , an arc with the corresponding weight is constructed with  $\hat{v}_n$  as tail and  $\hat{v}_1$  as head, if  $\hat{v}_n$  satisfies **Criterion 2** with  $\hat{v}_1$ , *i.e.*, each SL  $e^s$  lying in the segment from  $MP(\hat{v}_n)$  to  $MP(\hat{v}_1)$  following  $dir$ , belongs to  $\mathcal{F}_{v_n^r v_1^r}$ . Figure 13 shows a complete WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$ .

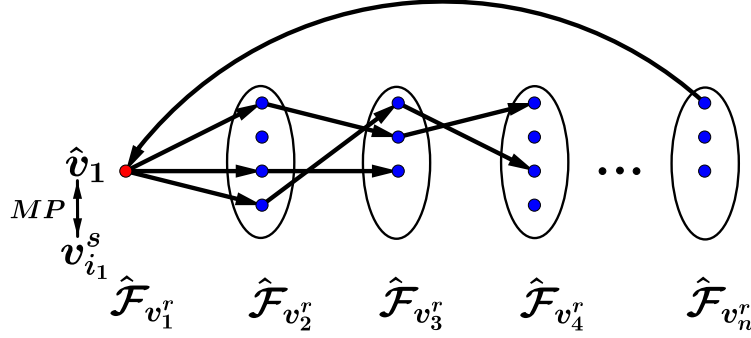


Fig. 13. The WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$  with respect to  $v_{i_1}^s$  and  $dir$ .

**Theorem 4:** The time complexity of constructing a WDAG is  $\mathcal{O}(m^3n)$ , where  $m$  and  $n$  are the SN and the VN numbers of the substrate cycle  $C^s$  and the cycle VNR  $C^r$  respectively.

**Proof.** The time complexity of constructing a WDAG consists of two parts: 1) establishing the vertex set  $\hat{V}$ ; 2) constructing the weight arc set  $\hat{A}$ . For the first part, we need to set up the  $n$  vertex parts  $\hat{\mathcal{F}}_{v_j^r}, \forall 1 \leq j \leq n$ , and build a one-to-one mapping  $MP$  between vertices in  $\hat{\mathcal{F}}_{v_j^r}$  and  $\mathcal{F}_{v_j^r}$ . Since each  $|\hat{\mathcal{F}}_{v_j^r}| \leq m$ , time consumption of this part is up to  $\mathcal{O}(mn)$ . For the second part, in the worst case, for each vertex pair  $(\hat{v}_j, \hat{v}_{j+1})$  between  $\hat{\mathcal{F}}_{v_j^r}$  and  $\hat{\mathcal{F}}_{v_{j+1}^r}$ , we need to check **Criterion 1** and **Criterion 2** to decide whether to construct an arc. The time consumption of checking **Criterion 1** and **Criterion 2** is  $\mathcal{O}(m)$ . There are at most  $m^2n$  arcs, and thus the time

consumption of constructing the arc set  $\hat{A}$  is  $\mathcal{O}(m^3n)$ . Combing  $\mathcal{O}(mn)$  and  $\mathcal{O}(m^3n)$ , the total time complexity is  $\mathcal{O}(m^3n)$ . ■

The WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$  has an important property described in Theorem 5.

**Theorem 5:** Given a substrate cycle  $C^s(v_1^s, \dots, v_m^s v_1^s)$  and a cycle VNR  $C^r(v_1^r, \dots, v_n^r v_1^r)$ , under the condition that  $v_1^r \rightarrow v_{i_1}^s \in \mathcal{F}_{v_1^r}$  and following  $dir$ , there is a one-to-one relation between each directed cycle in the WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$  and each feasible simplex cycle embedding way.

**Proof.** For each feasible simplex cycle embedding way under the condition that  $v_1^r \rightarrow v_{i_1}^s$  and following  $dir$ , *w.l.o.g.*, we assume it is  $\left( \begin{array}{cccccc} v_1^r & v_2^r & \dots & v_j^r & v_{j+1}^r & \dots & v_n^r \\ v_{i_1}^s & v_{i_2}^s & \dots & v_{i_j}^s & v_{i_{j+1}}^s & \dots & v_{i_n}^s \end{array} \right)$ , where each  $v_j^r \rightarrow v_{i_j}^s$  and  $v_{i_j}^s \in \mathcal{F}_{v_j^r}$ . Since  $MP$  is a one-to-one mapping from  $\hat{\mathcal{F}}_{v_j^r}$  to  $\mathcal{F}_{v_j^r}$ , we use  $MP^{-1}$  to represent the inverse, *i.e.*,  $MP^{-1}(v_{i_j}^s) \in \hat{\mathcal{F}}_{v_j^r}$ ,  $MP(MP^{-1}(v_{i_j}^s)) = v_{i_j}^s$ . As it is feasible,  $MP^{-1}(v_{i_{j+1}}^s)$  must satisfy **Criterion 1** and **Criterion 2** with  $MP^{-1}(v_{i_j}^s)$ . Therefore, according to the construction process, there is an arc with  $MP^{-1}(v_{i_j}^s)$  as tail and  $MP^{-1}(v_{i_{j+1}}^s)$  as head, and  $(MP^{-1}(v_{i_1}^s), \dots, MP^{-1}(v_{i_j}^s), \dots, MP^{-1}(v_{i_n}^s)MP^{-1}(v_{i_1}^s))$  forms a directed cycle in  $\hat{G}_{dir}^{v_{i_1}^s}$ .

For each directed cycle in  $\hat{G}_{dir}^{v_{i_1}^s}$ , say  $\hat{C}$ , according to the construction process,  $\hat{C}$  must pass through exactly one vertex in each vertex part  $\hat{F}_{v_j^r}, \forall 1 \leq j \leq n$ , say  $(\hat{v}_1, \dots, \hat{v}_j, \dots, \hat{v}_n)$ . Since  $\hat{v}_{j+1}$  satisfies **Criterion 1** and **Criterion 2** with  $\hat{v}_j$ , it is obvious that

$MP(\hat{C}) = \left( \begin{array}{cccccc} v_1^r & \dots & v_j^r & \dots & v_n^r \\ MP(\hat{v}_1) \dots MP(\hat{v}_j) \dots MP(\hat{v}_n) \end{array} \right)$  must be a feasible simplex cycle embedding way. ■

Subsequently, combining with the weights assigned to arcs of the WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$ , the sum of arc weights of a directed cycle in  $\hat{G}_{dir}^{v_{i_1}^s}$  is equal to the total *BW* consumption of the embedding way which corresponds to the directed cycle. Since the *CPU* consumption is fixed to the sum of all demanded *CPUs* of VNs, to obtain the least-resource-consuming embedding way, we just need to search the minimum weighted directed cycle in the WDAG  $\hat{G}_{dir}^{v_{i_1}^s}$  which can be solved by dynamic programming in polynomial time.

To obtain the optimal least-resource-consuming simplex cycle embedding, we just need to construct  $2 \times |\mathcal{F}_{v_1^r}|$  WDAGs (two directions), search the directed cycle with the minimum weight in each WDAG, and output the smallest one among them. We formally give Algorithm 2.

---

**Algorithm 2:** Procedure of Cycle-to-Cycle Embedding (C2CE)

---

**Input** :  $C^s(v_1^s, \dots, v_m^s, v_1^s)$  and  $C^r(v_1^r, \dots, v_n^r, v_1^r)$

**Output:** The least-resource-consuming embedding way.

```

1 set  $\mathcal{F}_{v_j^r} \leftarrow \{v_i^s | CPU(v_i^s) \geq CPU(v_j^r)\}, \forall j;$ 
2 set  $\mathcal{F}_{v_j^r v_{j+1}^r} \leftarrow \{v_i^s v_{i+1}^s | BW(v_i^s v_{i+1}^s) \geq BW(v_j^r v_{j+1}^r)\};$ 
3 set  $EMB \leftarrow \emptyset, Cost \leftarrow \infty;$ 
4 for  $v_{i_1}^r \in \mathcal{F}_{v_1^r}$  do
5   for  $dir$  ("+" or "-") do
6     construct the WDAG;
7     search the minimum weighted directed cycle  $\hat{C}$  in the WDAG;
8     if  $w(\hat{C}) < Cost$  then
9        $EMB \leftarrow MP(\hat{C}), Cost \leftarrow w(\hat{C});$ 
10 return  $EMB;$ 

```

---

In *Lines 1-2*, for each VN  $v_j^r$  and VL  $v_j^r v_{j+1}^r$ , we set up the feasible SN sets and SL sets. The time complexity of *Lines 1-2* is  $\mathcal{O}(mn)$ . At *Lines 3*, we set a variable  $EMB$  to record the optimal embedding way whose initial value is  $\emptyset$  and another variable  $Cost$  to record the  $BW$  consumption of the  $EMB$  whose initial value is large enough denoted by  $\infty$ . In *Lines 4-9*, for each  $v_{i_1}^r \in \mathcal{F}_{v_1^r}$  and each direction  $dir$  ("**+**" or "**-**"), we construct the corresponding WDAG at *Line 6* whose time complexity is  $\mathcal{O}(m^3n)$  by Theorem 4. At *Line 7*, we search the minimum weighted directed cycle  $\hat{C}$ , and its time complexity is  $\mathcal{O}(m^2n)$  by dynamic programming. At *Lines 8-9*, if the weight of  $\hat{C}$  denoted by  $w(\hat{C})$  is less than current  $Cost$ , we replace  $Cost$  by  $w(\hat{C})$  and  $EMB$  by the embedding way denoted by  $MP(\hat{C})$  which corresponds to  $\hat{C}$ . Finally, at *Line 10*, we return the  $EMB$  (if  $EMB = \emptyset$  then  $C^r$  can not be embedded on  $C^s$  by simplex cycle embedding). The total time complexity of Algorithm 2 is  $\mathcal{O}(mn) + 2m \times (\mathcal{O}(m^3n) + \mathcal{O}(m^2n)) + \mathcal{O}(1) = \mathcal{O}(m^4n)$ , where  $m$  and  $n$  are the SN and the VN numbers of the substrate cycle  $C^s$  and the cycle VNR  $C^r$  respectively.

For the  $Emb(C^s, C^r)$  problem in simplex cycle embedding, we can solve it in polynomial

time. How about the AcR and Rev problem? Unfortunately, both of them are still  $\mathcal{NP}$ -hard.

**Theorem 6:** In cycle-to-cycle embedding, both the AcR and Rev problems are  $\mathcal{NP}$ -hard. Moreover, the hardnesses of each problem is no less than any  $d$ -DKP, where  $d$  is any constant integer.

**Proof.** Since the AcR problem is a special case of the Rev problem, we just need to prove the AcR problem is  $\mathcal{NP}$ -hard. To this end, we polynomial-timely reduce the  $\mathcal{NP}$ -hard problem "Cardinality  $d$ -DKP" [22] to the AcR problem. Cardinality  $d$ -DKP is a special  $d$  dimensional MDKP, *i.e.*, the knapsack is with a  $d$ -dimensional capacity attributes  $(b_1, \dots, b_i, \dots, b_d)$  and each  $j$ -th item is with a  $d$ -dimensional size attributes  $(s_{j1}, \dots, s_{ji}, \dots, s_{jd})$ . The objective is to maximize the number of packed items.

Given an instance in Cardinality  $d$ -DKP, we construct the substrate cycle in such way: There are  $d$  SNs in the substrate cycle  $C^s(v_1^s, \dots, v_i^s, \dots, v_d^s)$ , and  $CPU(v_1^s) = b_1$  and  $CPU(v_i^s) = B_i \times b_i, \forall 1 < i \leq d$  where  $B_i$  are relatively big numbers explained later. Each SL is with  $BW = n$ , *i.e.*, the number of items. We construct  $n$  cycle VNRs in such way: For the  $j$ -th cycle VNR  $C^r(v_{j1}^r, \dots, v_{ji}^r, \dots, v_{jd}^r)$ , there are  $d$  VNs, and  $CPU(v_{j1}^r) = s_{j1}$  and  $CPU(v_{ji}^r) = B_i \times s_{ji}, \forall 1 < i \leq d$  such that  $B_i \times s_{ji} > \max_{1 < k < i} (B_k \times b_k, b_1)$  (by setting  $CPU$ s like this the  $v_{ji}^r$  can only be embedded on  $v_i^s$ ). Each VL is with  $BW = 1$ . Thus, the solution of the instance of Cardinality  $d$ -DKP is equivalent to that of the AcR problem. ■

As shown in [22], even the 2-DKP is strongly  $\mathcal{NP}$ -hard and the hardness of solving  $d$ -DKP keeps entrenched with the increase of  $d$ . To effectively solve the strongly  $\mathcal{NP}$ -hard problem, we herein develop a heuristic algorithm based on the optimization for single cycle embedding as follows. Intuitively, for a cycle  $C^r$ , if its ratio of revenue to resource consumption is higher than the others, it tends to be embedded so as to achieve a more efficient income for the InP. This consists of the main motivation of our greedy strategy in Algorithm 3: Given a substrate cycle  $C^s$  and a set of cycle VNRs  $\{C_1^r, C_2^r, \dots, C_n^r\}$ , for each  $C_j^r$ , we first estimate the ratio of revenue to resource consumption, *i.e.*,  $\frac{w_j}{\sum_{v^r \in C_j^r} CPU(v^r) + \sum_{e^r \in C_j^r} BW(e^r)}$ . We then arrange them in the descending order of the estimated ratios, and sequentially embed them on the  $C^s$  by procedure C2CE until no more cycle VNR can be embedded by simplex cycle embedding. However, one thing should be noted that the simplex cycle embedding has its own shortage, *i.e.*, it limits the



solution space. Therefore, if no cycle VNR can be embedded by simplex cycle embedding, we continue the embedding by running general algorithms. Via this combination, both merits of simplex cycle embedding and general algorithms can be conflated.

---

**Algorithm 3:** Procedure of Greedy Revenue (GR)

---

**Input** :  $C^s$  and  $\{C_1^r, C_2^r, \dots, C_n^r\}$ .

**Output:** The final revenue.

- 1 **set** cycle VNRs in descending order by estimated ratios;
  - 2 **run** C2CE to sequentially embed cycle VNRs until can't;
  - 3 **run** general algorithms;
  - 4 **return** final revenue;
- 

## VII. NUMERICAL RESULTS

In this section, we compare our proposed algorithms PE for path embedding and GR for cycle embedding to the existing general algorithms. Two general embedding algorithms from [7] and [4], denoted by RW and BA respectively, are used as our benchmarks. Here, we use GRRW and GRBA to denote the procedures GR which invoke RW and BA respectively. We repeat each simulation 50 times under the same circumstance to ensure sufficient statistical accuracy, and a 95% confidence interval is given to each numerical result. All the simulations have been run by MATLAB 2015a on a computer with 3.2 GHz Intel(R) Core(TM) i5-4690S CPU and 8 GBytes RAM.

### A. Evaluation Environments

#### 1) Path Embedding:

a) *Substrate Networks:* We use the GT-ITM tool [23], prevailing in the generation of practical network topologies, to randomly generate two substrate networks denoted by  $G_1^s, G_2^s$  respectively. Both of these substrate networks have 100 SNs and 1000 SLs, corresponding to a medium-sized ISP. Besides, we also use a complete graph of 100 SNs as the substrate network denoted by  $CG^s$ . The *CPU* and *BW* of each SN and SL are set as 100 units.

b) *Virtual Network Requests*: The length of each path VNR is randomly generated in the range of  $[5, 10]$ . The *CPU* and *BW* of each VN and VL are randomly generated in the range of  $[1, 5]$  units. The number of path VNRs is set as 1000 in each simulation.

### 2) *Cycle Embedding*:

a) *Substrate Networks*: We set up three substrate cycles denoted by  $C_{20}^s$ ,  $C_{25}^s$ ,  $C_{30}^s$  respectively, whose number of SNs are 20, 25 and 30 respectively, corresponding to the sizes of existing substrate optical rings. The *CPU* and *BW* of each SN and SL are set as 100 units.

b) *Virtual Network Requests*: The number of VNs of each cycle VNR is randomly generated in the range of  $[5, 10]$ . The *CPU* and *BW* of each VN and VL are randomly generated in the range of  $[1, 5]$  units. The number of cycle VNRs is set as 100 in each simulation.

### 3) *Performance Metrics*:

The comparisons are performed for both the AcR and Rev problems.

- The AcR problem: The revenue of each VNR is set to be one. Besides, we tweak the objective function of the AcR problem as  $\frac{|S|}{n}$ , where  $S$  is the subset of embedded VNRs and  $n$  is the number of total VNRs.
- The Rev problem: The revenue of each VNR is proportional to its VN number in the range of  $[5, 10]$ .

## B. *Simulation Results*

### 1) *Path Embedding*:

Figures 14 and 15 respectively demonstrate the numerical results of the AcR and Rev problems in path embedding. The average acceptance ratio of all the three substrate networks, as shown in Fig 14(a), is 41.06% for PE, compared to 30.70% and 29.75% for RW and BA respectively. The average revenue, as shown in Fig. 15(a), is 3052.12 for PE, compared to 2308.05 and 2251.01 for RW and BA respectively. For time complexity as shown in Figs. 14(b) and 15(b), PE is of an average run time of 1.32s, obviously smaller than that of RW and BA, 20.44s and 8.28s respectively. In summary, PE is much more efficient and effective than the two general algorithms RW and BA in path embedding.

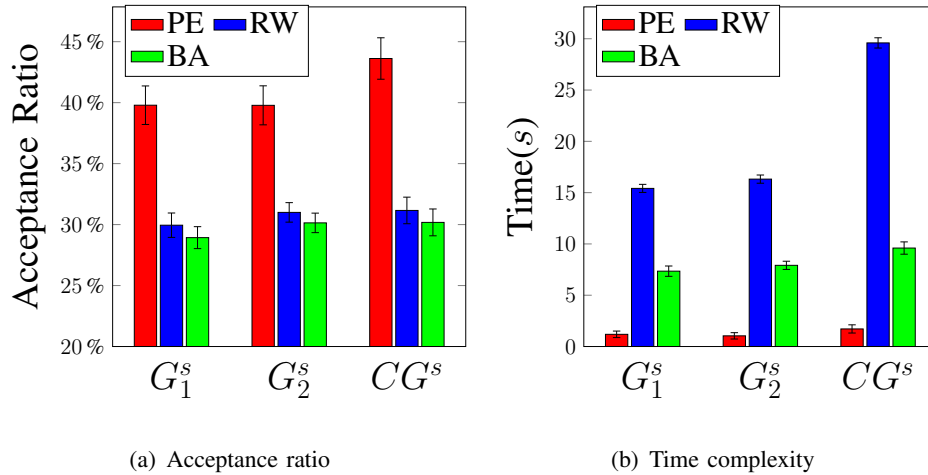


Fig. 14. Numerical results of the AcR problem in path embedding.

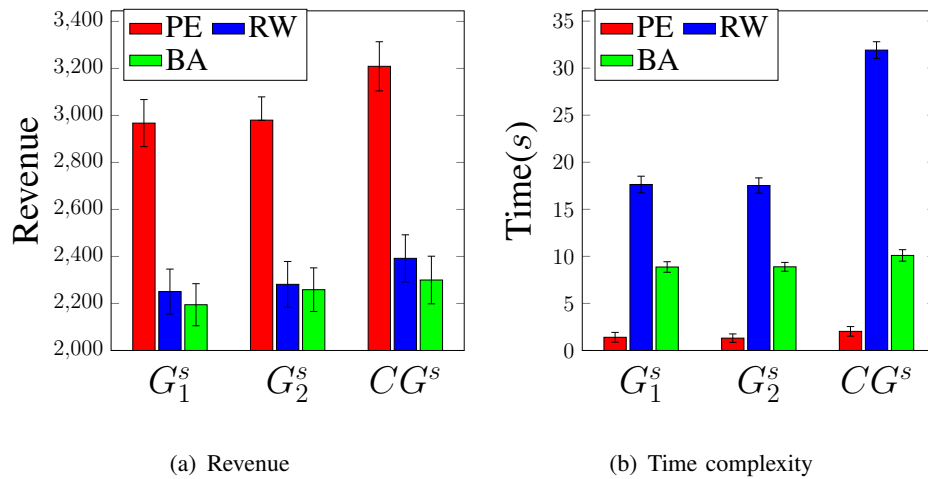


Fig. 15. Numerical results of the Rev problem in path embedding.

## 2) Cycle Embedding:

Figures 16 and 17 respectively showcase the numerical results of the AcR and Rev problems in cycle embedding. The average acceptance ratio, as shown in Fig. 16(a), of GRRW and GRBA are 31.13% and 30.31% respectively compared to 25.63% and 24.84% of RW and BA respectively. The average revenue, as shown in Fig. 17(a), of GRRW and GRBA are 239.68 and 234.96 respectively while 189.80 and 185.62 of RW and BA respectively. From the aspect of final results of acceptance ratios and revenues, GRRW and GRBA take advantage over RW and BA. For the time complexity as shown in Figs. 16(b) and 17(b), while the run times of RW and BA are

relatively stable and smaller than  $2.5s$ , that of GRRW and GRBA are quickly climbing because the time complexity of construction of the WDAG is fourth-order about the number of SNs. But, the corresponding acceptance ratios and revenues do not improve much with the increase of run times of GRRW and GRBA. Thus, in general substrate networks, in the future more work is needed to balance the size of decomposed substrate cycles and develop decomposition strategies so as to constitute cycle embedding algorithms as competitive as PE for path embedding.

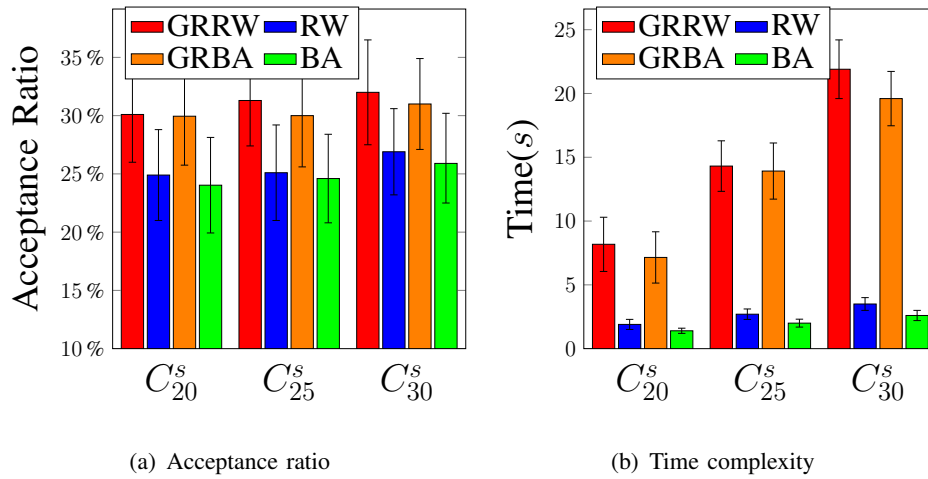


Fig. 16. Numerical results of the AcR problem in cycle embedding.

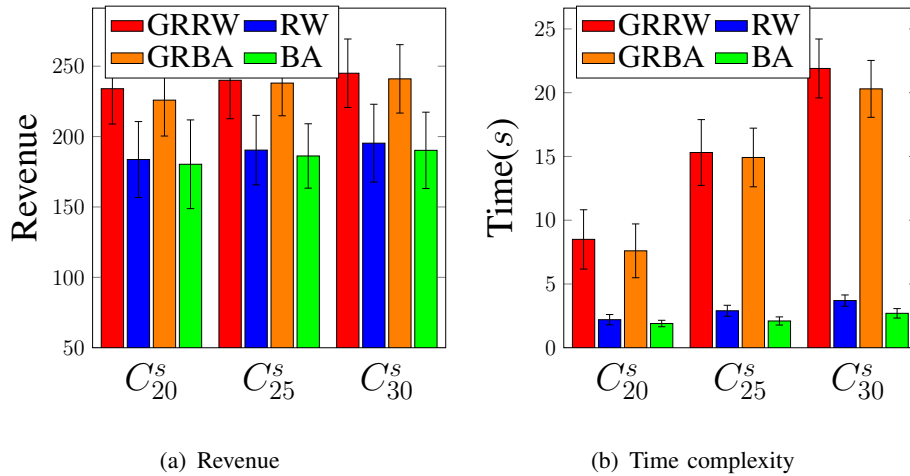


Fig. 17. Numerical results of the Rev problem in cycle embedding.

## VIII. CONCLUSIONS

In this work, we systematically investigated the VNE problems in path and cycle topologies. For path embedding, we proved its  $\mathcal{NP}$ -hardness and inapproximability. Following the idea of expanding substrate networks into "paths", we further developed the MKP-MDKP-based algorithms for the path embedding, which turn out to be more efficient and effective than its counterparts. Regarding cycle embedding, we proposed an auxiliary graph WDAG, based on which we are able to characterize the one-to-one relation between a directed cycle in WDAG and a feasible simplex cycle embedding. Herein is devised a polynomial-time algorithm C2CE to obtain the optimal least-resource-consumption embedding solution.

As the future work, to realize the idea of decomposing general VNRs into special topologies to efficiently tackle the general VNE problem, there is still a need for dealing with communications between multiple InPs.

## APPENDIX

In this appendix, we present the proof of Lemma 2. To this end, we need to prove that one of the two problems can be reduced to the other in polynomial time.

First, we prove  $SSET \leq_T^P SG$ : Given a graph  $G(V, E)$ , for each vertex pair  $v, u \in V$ , we construct a graph  $G_{vu}$  by adding a new vertex  $v^*$  and connecting  $v^*v$  and  $v^*u$ . Afterwards, we obtain a set  $\mathcal{G} = \{G_{vu} | \forall v, u \in V\}$  and  $|\mathcal{G}| = \binom{|V|}{2}$ . It is easy to see that  $G$  contains a spanning subgraph which has an Eulerian trail iff there exists one  $G_{vu} \in \mathcal{G}$  which is a Supereulerian graph.

Then for  $SG \leq_T^P SSET$ : Given a graph  $G(V, E)$ , arbitrarily selecting one vertex  $v \in V$ , we construct a graph  $G^*$  by adding two new vertices  $u_1$  and  $u_2$  and connecting them to  $v$ . It is easy to see that  $G$  is a Supereulerian graph iff  $G^*$  contains a spanning subgraph which has an Eulerian trail.

## REFERENCES

- [1] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Netw.*, vol. 29, no. 5, pp. 36–42, Sept. 2015.

- [2] R. Mao, H. Xu, W. Wu, J. Li, Y. Li, and M. Lu, "Overcoming the challenge of variety: big data abstraction, the next evolution of data management for AAL communication systems," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 42–47, Jan. 2015.
- [3] J. Turner and D. Taylor, "Diversifying the internet," in *Proc. IEEE GLOBECOM*, Dec. 2005, pp. 755–760.
- [4] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, Apr. 2008.
- [5] A. Fischer, J. Botero, M. Beck, H. Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.
- [6] A. Belbekkouche, M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, Fourth 2012.
- [7] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.
- [8] 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.
- [9] I. Houidi, W. Louati, W. Ameer, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011–1023, Mar. 2011.
- [10] D. Andersen, "Theoretical approaches to node assignment," pp. 1–13, Dec. 2002, unpublished manuscript.
- [11] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 816–827, Mar. 2014.
- [12] M. Chowdhury, M. 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.
- [13] Q. Hu, Y. Wang, and X. Cao, "Resolve the virtual network embedding problem: A column generation approach," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 410–414.
- [14] A. Leivadreas, 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.
- [15] M. Rahman and R. Boutaba, "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, Jun. 2013.
- [16] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *Proc. IEEE SDN4FNS*, Nov. 2013, pp. 1–7.
- [17] S. Shirazipourazad, C. Zhou, Z. Derakhshandeh, and A. Sen, "On routing and spectrum allocation in spectrum-sliced optical networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 385–389.
- [18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proc. WWW*, Apr. 1998, pp. 107–117.
- [19] P. Catlin, "Supereulerian graphs: A survey," *Journal of Graph Theory*, vol. 16, pp. 177–196, Jun. 1992.
- [20] D. Du, K. Ko, and X. Hu, *Design and Analysis of Approximation Algorithms*. Springer-Verlag New York, 2012.
- [21] M. Andrews, J. Chuzhoy, S. Khanna, and L. Zhang, "Hardness of the undirected edge-disjoint paths problem with congestion," in *46th Annual IEEE FOCS*, Oct. 2005, pp. 226–241.

- [22] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 2004.
- [23] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM*, Mar. 1996, pp. 594–602.