A Divide-and-Conquer Evolutionary Algorithm for Large-Scale Virtual Network Embedding

An Song, *Student Member, IEEE*, Wei-Neng Chen, *Senior Member, IEEE*, Yue-Jiao Gong, *Member, IEEE*, Xiaonan Luo, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The subgraph isomorphism problems, which aim to map subgraphs to a given graph, are widely seen in many applications and are usually nondeterministic polynomial-time complete (NP-complete). As a representative extension of the subgraph isomorphism problem, virtual network embedding (VNE) is a key problem in datacenter scheduling and network virtualization. Existing metaheuristic approaches to VNE problems tend to schedule networks as a whole. But when the problem scale grows, the performance of these approaches may degenerate due to the curse of dimensionality. In this article, we intend to propose a divide-and-conquer evolutionary algorithm with overlapping decomposition (ODEA) to solve large-scale VNE problems. First, realizing the fact that the decision variables in graph-based optimization problems like VNE are usually nonseparable, an overlapping decomposition method is introduced by investigating the characteristic of the network structure. In this method, the critical elements which have tight connections to many other nodes can belong to multiple subcomponents. As a result, the decision variables with tight connections can always be evolved together in multiple subcomponents. Second, to combine the subsolutions into a complete feasible solution, a competitive strategy is devised. Through the competition among critical elements, the optimizing information is shared among subcomponents, which can further improve the effectiveness of ODEA. The proposed ODEA can adopt different metaheuristics as the optimizer, and we conduct experiments on both the scenarios with a single virtual network and with a series of online networks. The experimental results verify that ODEA can significantly improve the performance of different metaheuristics in large-scale VNE problems.**

***Index Terms*—Graph matching, large-scale optimization, metaheuristics, virtual network embedding (VNE).**

Manuscript received July 18, 2018; revised May 2, 2019; accepted September 7, 2019. Date of publication September 26, 2019; date of current version May 29, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61976093, and Grant 61873097, in part by the Science and Technology Plan Project of Guangdong Province under Grant 2018B050502006, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant NRF-2019H1D3A2A01101977. *(Corresponding author: Wei-Neng Chen.)*

A. Song, W.-N. Chen, and Y.-J. Gong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com).

X. Luo is with the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China.

J. Zhang is with the Division of Electrical Engineering, Hanyang University, Seoul 15588, South Korea.

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2019.2941824

I. Introduction

GRAPH matching is a fundamental problem in different fields that employ structural representations, such as biology, social networks, and network management [1]. The general purpose of graph matching is to find the similarity relationship among graphs. In particular, the matching from subgraphs to a given graph, i.e., subgraph isomorphism [2], has drawn increasing attention, because in many cases we should determine whether a substructure is included within the whole structure. The subgraph isomorphism detection problem is NP-complete and the computational complexity is factorial in the worst case [1]. The concept of subgraph isomorphism is widely applied in different fields, such as molecular structure analysis [3], recognition of 3-D objects [4], database searching [5], and so on.

One representative extension of subgraph isomorphism is the virtual network embedding (VNE) problem [6], which is commonly seen in network virtualization [7] and cloud datacenter scheduling [8]. The objective of VNE is to find the optimal mapping from virtual networks (VNs) to a substrate network (SN), such that VNs and SNs are homeomorphic under resource constraints. The deployment of VNs is composed of two parts: 1) the virtual node mapping (VNoM, i.e., mapping from virtual nodes to substrate nodes) and 2) the virtual link mapping (VLiM, i.e., mapping from virtual links to substrate links). Both of them should satisfy the constraints of networks, such as the CPU constraints of substrate nodes and the bandwidth constraints of substrate links. Due to these constraints, the VNE problem is intractable, which is known to be nondeterministic polynomial-time hard (NP-hard) [6]. Even if all VNs are known in advance, the VNE problem is still NP-hard since it can be transformed into the NP-hard multiway separator problem [9].

As VNE problems are important and challenging, many methods have been proposed and they can be roughly classified into three categories: 1) exact algorithms [10]–[13]; 2) heuristic algorithms [14]–[18]; and 3) metaheuristic algorithms [19]–[23]. Exact algorithms formulate VNE problems as integer linear programming problems [24], e.g., the VNE node-link formulation (VNE-NLF) [10]. Although the global optimum is guaranteed to be found, exact algorithms are time-consuming when the scale of the problem increases. Heuristic algorithms utilize problem-dependent information to search for near-optimal solutions, such as the algorithm with subgraph isomorphism detection (ASID) [18] and the random

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

walk with breadth-first search (RW-BFS) [15]. Heuristic-based methods are time-efficient but the approximated solutions found by heuristics might be far from the global optimum in complicated situations, leading to poor performance [6]. To further improve the quality of solutions, metaheuristic algorithms were proposed to solve VNE problems, e.g., the unified enhanced particle swarm optimization (UEPSO) [21] and the ant colony optimization based on topology decomposition (ACO-TD) [19]. Compared with exact algorithms, the execution time of metaheuristic approaches is controllable and acceptable. Compared with heuristics, metaheuristic methods have a stronger optimization capability, which has been shown in recent studies [19]–[21], [25].

However, as the scale of the problem further increases, the search space will increase exponentially and the structure of VNs also becomes more complicated, which makes the VNE problem even more difficult. The exponentially growing complexity is a great obstacle for existing metaheuristic approaches, since most of them optimize all decision variables as a whole, which may be badly influenced by the curse of dimensionality. To overcome this shortcoming, in this article, we propose the overlapping divide-and-conquer evolutionary algorithm (ODEA) to solve large-scale VNE problems. ODEA has the following two key features.

1) ODEA adopts the divide-and-conquer strategy to deal with large-scale networks. In general, one node usually has only a local influence on the whole network. In other words, a node may have a strong influence on its neighbors and weak influence on remote nodes. Therefore, it is natural to utilize the divide-and-conquer strategy to partition a large network into small subnetworks [26], [27]. Following this idea, a whole VN in ODEA is partitioned into several small sub-VNs and then the mappings of sub-VNs are optimized cooperatively. Compared to the original network, sub-VNs have lower dimensions and much simpler structure. Thus, the search space for embedding such sub-VNs can be reduced.
2) The overlapping decomposition is devised in ODEA to deal with the interconnection among subcomponents. The classical divide-and-conquer strategy in many evolutionary algorithms decomposes a large problem into mutually exclusive subcomponents and each subcomponent is optimized independently, such as differential grouping (DG) [28] and its extension DG2 [29] in continuous optimization. Such exclusive decomposition can work well on totally or partially separable problems. However, sometimes it is impossible to obtain independent exclusive subcomponents in practice, particularly in graph-based optimization problems including VNE. If these nonseparable problems are decomposed into exclusive subcomponents, the dependence among interconnected subcomponents will be ignored, leading to poor optimization performance. To alleviate such deficiency, we devise the overlapping decomposition for ODEA. The critical elements in a VN can be assigned to multiple subcomponents rather than a single exclusive subcomponent. Thus, the dependence among sub-VNs can be taken into account in ODEA.

All in all, there are three major procedures in ODEA: 1) graph partitioning; 2) subgraph mapping; and 3) graph integration. The graph partitioning procedure partitions a large VN into several overlapping sub-VNs. The overlapping mechanism can thus handle the dependence among sub-VNs. Then, the subgraph mapping and graph integration procedures are alternate in each iteration. The subgraph mapping procedure is responsible for the mapping of sub-VNs to the SN. Metaheuristic methods are adopted as the optimizer for mapping. The graph integration procedure integrates subsolutions found so far to construct the whole solution. During the graph integration, overlapping elements in sub-VNs may have multiple values in different subcomponents. An appropriate value for each overlapping element is selected by the competitive strategy in ODEA.

We conduct experiments to explore the optimizing behavior of ODEA. The performance of ODEA is tested on both the scenarios of offline and online VNs. Four metaheuristics, including the set-based PSO (SPSO) [30], UEPSO [21], the PSO with random walk (RWPSO) [22], and CB-GA [25] are implemented under the ODEA. The experimental results show that ODEA is general for different metaheuristics and is promising for solving large-scale VNE problems.

The reminder of this article is organized as follows. In Section II, we discuss related works about VNE problems and divide-and-conquer evolutionary algorithms. Then, the VNE problem is formally defined in Section III. Section IV describes the proposed ODEA in details. In Section V, we study the optimizing behavior of different decomposing methods and the influence of parameters. In Section VI, sufficient experiments are conducted to verify the generality and the effectiveness of ODEA. The conclusions are finally drawn in Section VII.

II. Related Works

The VNE problem has been proven NP-hard and metaheuristic algorithms have been widely applied. In this section, we first review the metaheuristic algorithms for VNE. Then, we also make a brief review on the existing divide-and-conquer techniques used by evolutionary algorithms for solving large-scale optimization problems.

A. Metaheuristic Algorithms for VNE Problems

Metaheuristic algorithms are a class of algorithms inspired by natural phenomenon, such as particle swarm optimization (PSO) [30], ant colony optimization (ACO) [31], genetic algorithm (GA) [32], etc. As globally optimal solutions to VNE problems are hard to find, researchers pay more attention to finding near-optimal solutions with metaheuristic algorithms

The most popular metaheuristic method for VNE is PSO. Zhang *et al.* [21] proposed a variant of PSO, named UEPSO, to solve VNE problems. Positions of particles in UEPSO represent the VNoM and velocities are vectors with binary values which represent the difference between current solutions and the best solution. Following the basic PSO procedure, UEPSO replaces the arithmetic operators defined on

the continuous space in PSO with the operators on the discrete space. Therefore, particles in UEPSO can search for optimal solutions to VNE on discrete space. Based on the arithmetic operators of UEPSO, Cheng *et al.* [22] further proposed the PSO with a Markov random walk model (RW-PSO). Incorporated a modified PageRank algorithm, RW-PSO evaluates the ranks for both virtual nodes and substrate nodes. During optimization, the virtual nodes with high ranks are more probable to be embedded on the substrate nodes with high ranks. In this way, not only the network resources but also the network topologies are taken into account.

Other metaheuristic methods for VNE include GA and ACO approaches. Mi *et al.* [33] applied GA to solve VNE problems and proposed two GA-based algorithms, called CB-GA and RW-GA. The chromosomes in CB-GA and RW-GA represent the VNoM only. In each generation, the chromosomes are randomly paired and execute the crossover and mutation process with predefined probabilities. This operation is repeated until the maximum number of iterations is reached. Compared to PSO-based approaches for VNE problems, GA-based methods provide better diversity of populations so that more potential solutions can be found. Chang *et al.* [25] proposed the ACO-based algorithm with the random walk model (RW-ACO) to solve VNE problems. In RW-ACO, the artificial ant colony is launched to search the optimal mapping of VNs. The solutions are constructed step by step according to the artificial pheromones which are associated with the usage of substrate bandwidth. Solutions with less usage of bandwidth can reinforce artificial pheromones and their components will be reused with higher probabilities in future generations. Fajjari *et al.* [23] applied max–min ACO to solve VNE (VNE-AC). The objective of VNE-AC is to minimize the mapping cost of VNs and maximize the minimum of residual bandwidth in the SN. VNE-AC can improve the acceptance rate and provider's revenue compared to heuristic approaches.

The experimental results in metaheuristics [19], [21], [25] have shown that metaheuristic algorithms for VNE can significantly outperform several classical heuristics, such as ASID [18], deterministic VNE with the shortest path (D-ViNE-SP) [34], and VNE-Greedy [16]. However, many metaheuristics adopt the nondecomposition mechanism and optimize all VNoM and VLiM as a whole. When the scale of VNs increases and the topologies of networks become complicated, the metaheuristics with the nondecomposition mechanism might lose their effectiveness. To alleviate such deficiency, we devise the divide-and-conquer metaheuristics to deal with large-scale VNs, which will be introduced in Section IV.

B. Evolutionary Algorithms Under the Divide-and-Conquer Concept

In order to solve large-scale optimization problems, the divide-and-conquer methodology is adopted by evolutionary algorithms and leads to a new evolutionary scheme, cooperative co-evolution (CC) [28], [35], [36]. Usually, CC first divides the optimization problem into several mutually exclusive subcomponents, and then each subcomponent is optimized by a specific evolutionary optimizer. So far, many grouping strategies have been proposed for continuous function optimization problems, such as DG [28], [35], [36], global DG (GDG) [37], and DG2 [29].

However, most practical applications are discrete combinatorial optimization problems [36], [38], [39]. For these problems, the decomposition strategy designed in continuous space cannot be applied directly. Hence, how to decompose application problems is still a challenge for CC-based approaches. Mei *et al.* [40] applied CC to solve capacitated arc routing problems. They divided close routes into the same group with the fuzzy k-medoids methods. As a result, the dependence among groups can be reduced. Gomes *et al.* [41] extended the CC architecture in solving heterogeneous multiagent systems. Through investigating the behavior of agents, the agents with similar behavior can be assigned to the same subcomponent. In this way, homogeneous subteams are formed inside the whole heterogeneous team. Gong *et al.* [42] proposed a multiobjective CC algorithm to solve the sparse unmixing of hyperspectral data. The decision vectors interact with each other in this problem and they adopted the random grouping to split nonseparable decision variables into low-dimensional decision vectors.

Generally, there are insufficient researches of CC in solving nonseparable practical problems. Moreover, most existing CC-based works cannot deal with the interconnection among subcomponents. Since CC decomposes problems into independent and exclusive subproblems, the coordination and dependence of subcomponents are difficult to take into account. In this article, we study the overlapping decomposition rather than exclusive decomposition, which can consider the dependence and interconnection among subproblems while CC-based approaches cannot.

III. Network Model and VNE Problem Definition

The VNE problem is an extension of the graph isomorphism problem. The problem can be abstracted as mappings from VNs to an SN subject to resource constraints. In this section, we first present the network model and then the mathematical formulation of VNE problems is provided.

A. Network Model

The SN and the VN are modeled as an undirected weighted graphs $G_s = (N_s, L_s)$ and $G_v = (N_v, L_v)$, respectively. Here, N_s and L_s represent the set of substrate nodes and substrate links where the subscript "s" stands for SNs. Similarly, N_v and L_v represent the set of virtual nodes and virtual links where the subscript "v" stands for VNs. Each substrate node $n \in N_s$ (or virtual node $n \in N_v$) has a node weight $nw_s(n)$ (or $nw_v(n)$). Similarly, each substrate link $l \in L_s$ (or virtual link $l \in L_v$) has a link weight $lw_s(l)$ (or $lw_v(l)$) [14], [43]. In VNE problems, the weight of nodes usually represents CPU resources and the weight of links represents bandwidth. An example of a typical SN and a VN are depicted in Fig. 1. Nodes A–F are substrate nodes which constitute the SN. Nodes a–c are virtual nodes

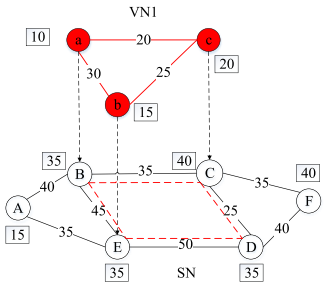


Fig. 1. Examples of VNE. The node mapping for VN1 is $\{a \rightarrow B, b \rightarrow E, c \rightarrow C\}$ and the link mapping is $\{(a, b) \rightarrow (B, E), (a, c) \rightarrow (B, C), (b, c) \rightarrow (E, D, C)\}$, which needs to satisfy the resource constraints from nodes and links.

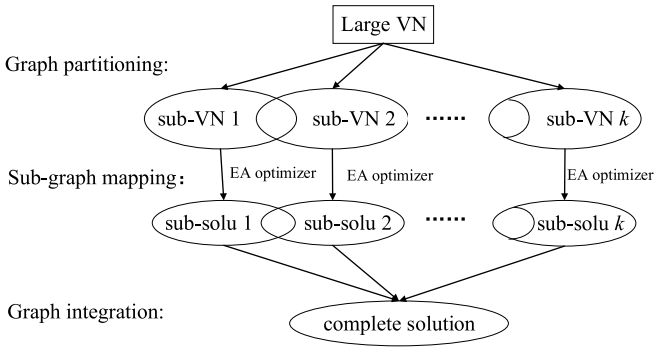


Fig. 2. Framework of ODEA. "Sub-solu" stands for subsolution.

which constitute the VN. All links and nodes are labeled with corresponding weights.

B. VNE Problem Definition

Given the SN $G_s = (N_s, L_s)$ and the VN $G_v = (N_v, L_v)$, generally, the VNE problem can be divided into two one-to-one mappings, i.e., the VNoM, Γ_N: $N_v \rightarrow N_s$, and the VLiM, Γ_L: $L_v \rightarrow P_s$ where P_s is the set of all loop-free substrate paths in the SN G_s. As shown in Fig. 1, the VNoM for VN1 is $\{a \rightarrow B, b \rightarrow E, c \rightarrow C\}$ and the VLiM for VN1 is $\{(a, b) \rightarrow (B, E), (a, c) \rightarrow (B, C), (b, c) \rightarrow (E, D, C)\}$. During the mapping, the following constraints should be satisfied [6], [15], [21].

In VNoM, first, each virtual node should be mapped to one substrate node

$$\forall x \in N_V, \Gamma_N(x) \in N_S \tag{1}$$

and two virtual nodes are not allowed to be mapped to the same substrate node

$$\forall x, y \in N_V, \Gamma_N(x) = \Gamma_N(y), \text{iff } x = y. \tag{2}$$

Besides, the weights of substrate nodes should be larger than or equal to those of virtual nodes

$$\forall x \in N_V, nw_V(x) \leq nw_S(\Gamma_N(x)). \tag{3}$$

As for VLiM, each virtual link l is mapped to a path on the SN

$$\forall l \in L_V, \Gamma_L(l) \in P_S. \tag{4}$$

The weight of a path P is defined as the lowest weight of the substrate links constituting P, denoted as $pw(P)$. During VLiM, the weights of substrate paths should be larger than or equal to the weights of virtual links

$$\forall l \in L_V, lw_V(l) \leq pw(P) \quad \text{where } P = \Gamma_L(l). \tag{5}$$

The objective of VNE problems is to find the optimal mapping from VNs to an SN, which can minimize the allocated resources (including node weights and link weights). When embedding a single VN, the allocated node weights equal the sum of all virtual node weights, which means they are identical in different solutions to VNoM. However, the allocated link weights of SNs are correlated to the solutions to VLiM. More precisely, if a virtual link is mapped to a long path on the SN, this virtual link will occupy more link weights of SNs than that is mapped to a short path. Therefore, from the perspective of optimization, we consider the objective function of embedding a single VN as minimizing the total allocated link weights for virtual links [21], [44], formulated as

$$\min : \sum_{l \in L_V} \sum_{i=1}^{|\Gamma_L(l)|} lw_V(l) \tag{6}$$

where $|.|$ is the length of a substrate path. For example in Fig. 1, the total allocated link weight is $30 \times 1 + 20 \times 1 + 25 \times 2 = 100$.

In practice, VN requests arrive at an SN continually and each VN request is associated with its duration. In the online situation, we denote the ith VN request as $\text{VNR}^i = (G_v, t_a, t_d)$ where t_a is the time when VNR^i arrives and t_d is the duration of VNR^i. When VNR^i comes to the SN at time t_a, the VNE algorithm should find the mappings for virtual nodes and virtual links. If any feasible solutions can be found, the corresponding resources (including node weights and link weights) are allocated to VNR^i for t_d time units. Otherwise, this VN request will be rejected. If a VN request is completed, the allocated resources of the VN will be released for future VN requests.

IV. ODEA

As aforementioned, we propose the ODEA to solve large-scale VNE problems. In this section, we first briefly introduce the basic framework of ODEA and then present the details of major procedures, i.e., graph partitioning, subgraph mapping, and graph integration.

A. Framework of ODEA

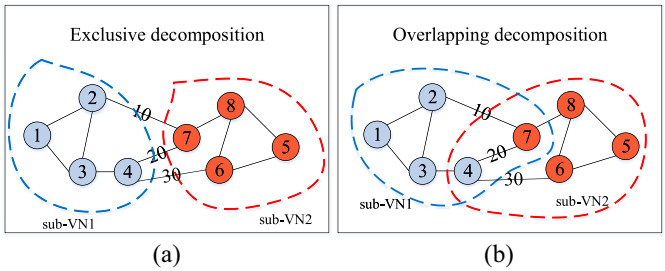
Fig. 2 presents the general framework of ODEA. Following the divide-and-conquer strategy, ODEA first partitions a large VN into several sub-VNs (i.e., graph partitioning procedure). Note that these sub-VNs are overlapped with each other due to the overlapping decomposition. Then each sub-VN is optimized by a specific evolutionary optimizer (i.e., subgraph mapping procedure). Naturally, k subsolutions will be generated for k subproblems, respectively. Since sub-VNs are

Algorithm 1 Graph Partitioning

input: the VN G_v, the number of groups *sub_num*, the maximum overlapping nodes *max_OL* in each group;

output: sub-VNs $\boldsymbol{subs}=\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{sub\_num}\}$

1: $\boldsymbol{subs}=MLkB(G_v, sub\_num)$;//exclusive decomposition
2: **for** i=1:|$\boldsymbol{subs}$|
3: $\boldsymbol{s}=\boldsymbol{subs}(i)$;
4: find the virtual nodes set ***conn*** whose elements connect to $\boldsymbol{s}$, $\boldsymbol{conn} = \{v \in N_v | \exists w \in \boldsymbol{s}, (v, w) \in L_v\}$;
5: **if** |***conn***| > *max_OL*
6: sort the virtual nodes in ***conn*** in decreasing order according to connection strength (7);
7: select top *max_OL* nodes in ***conn*** into ***OL_nodes***;
8: **else**
9: $\boldsymbol{OL\_nodes} = \boldsymbol{conn}$;
10: **end if**
11: $\boldsymbol{s} = \boldsymbol{s} \cup \boldsymbol{OL\_nodes}$;
12: $\boldsymbol{subs}(i) = \boldsymbol{s}$;
13: **end for**



(a) (b)

Fig. 3. Examples of exclusive decomposition and overlapping decomposition. Only the links connecting to two sub-VNs are labeled with weights for the sake of brevity. First, the exclusive decomposition is obtained by the MLkP scheme. Then the connection strength of connected nodes for each sub-VN is evaluated. For example in (a), the connection strength between the node 7 and sub-VN1 is $(10 + 20)\times 2 = 210$ while the connection strength between the node 6 and sub-VN1 is $30 \times 1 = 30$. Thus, the nodes with top connection strength (e.g., node 7) are more probable to be added to the overlapping decomposition.

overlapped, there exist inter-relations among subsolutions. Finally, subsolutions obtained by evolutionary optimizers are carefully integrated to construct a whole solution, especially for overlapping elements (i.e., graph integration).

B. Graph Partitioning

The basic idea of graph partitioning is to put interactional nodes into the same group to reduce the dependence among subproblems [28], [29]. However, as the topologies of VNs are connected graphs, all decision variables are indeed interacting with each other through virtual links so that the ideal decomposition does not exist. Therefore, we should take the influence of dependent sub-VNs into account.

Based on the above analysis, we develop the overlapping decomposition strategy for ODEA. The overlapping decomposition contains two steps. First, the whole large VN is partitioned into exclusive small sub-VNs with the exclusive decomposition. Second, some critical virtual nodes are selected and are put into several sub-VNs to construct the overlapping decomposition. The complete graph partitioning procedure is shown in Algorithm 1.

1) Exclusive Decomposition: The traditional the divide-and-conquer framework divides the whole dimension into several exclusive subproblems. Following this idea, VNs are partitioned into exclusive sub-VNs in the exclusive decomposition at first, which provides the input of the overlapping decomposition.

Most existing partitioning strategies divide VNs into several simple structures (e.g., star topologies [45]) or the combinations of simple structures (e.g., the combinations of ring topologies and tree topologies [19]). In these strategies, the size and the number of sub-VNs are correlated to the scale of the whole VNs, which means they are not controllable. As a consequence, these strategies have two shortcomings: 1) if the size of sub-VNs is too large, the embedding of large sub-VNs is still as difficult as embedding the whole VN and 2) if the size of sub-VNs is too small, the whole VN will be partitioned into too many sub-VNs and there will be many interconnections among sub-VNs. These connections increase the dependence among subproblems so that the performance of divide-and-conquer framework might be limited.

To find the appropriate decomposition for large-scale VNs, we adopt the multilevel k-way partitioning (MLkP) scheme [46]. Through this scheme, an arbitrary network can be partitioned into k subnetworks with a similar scale and the edge-cut among networks is small. Given the expected number of sub-VNs, *sub_num*, we partition the VN into *sub_num* exclusive sub-VNs (line 1 in Algorithm 1). Naturally, the appropriate value of parameter *sub_num* is related to the scale of VNs and will be studied later in Section V. Fig. 3(a) illustrates an example of exclusive decomposition. The VN is provided with eight nodes and it is divided into two sub-VNs (i.e., $sub\_num = 2$). Virtual nodes 1–4 compose sub-VN1 and the nodes 5–8 compose sub-VN2.

2) Overlapping decomposition: After the exclusive decomposition, the whole VN is partitioned into exclusive sub-VNs, and the interconnections among sub-VNs are ignored. However, the topologies of VNs are connected graphs and thus the dependence among sub-VNs should never be neglected. When the mappings of these sub-VNs are optimized independently, the embedding of one sub-VN might be disturbed by other interactional sub-VNs. In addition, some virtual nodes can strongly connect to different sub-VNs and it might be irrational to divide these virtual nodes into only one sub-VN. For example in Fig. 3(a), although virtual node 4 is divided into sub-VN1, it also has connections to sub-VN2. Therefore, it should be acceptable to include node 4 into sub-VN2, too.

To achieve the desired effect, we devise the overlapping decomposition strategy, which partitions a VN into overlapping sub-VNs. The virtual nodes that connect to multiple sub-VNs are overlapped with different sub-VNs so that the partitioning results are more reasonable. Through the overlapping elements, the optimization of sub-VNs is not independent and thus the disturbance from other sub-VNs is reduced.

The overlapping decomposition is based on the results of exclusive decomposition. After the exclusive sub-VNs are obtained, each sub-VN extends its range to overlap critical virtual nodes. The selection of critical virtual nodes to be overlapped is quite important to the performance of overlapping

decomposition. Herein, we devise two rules to find these critical nodes. First, if a virtual node does not connect to a sub-VN directly, the impact of the virtual node on the sub-VN can be ignored. In Fig. 3(a), node 7 has connections to sub-VN1 while node 5 does not have. Therefore, node 7 has more impacts on sub-VN1 than node 5 and the impact of node 5 is ignored in this situation. As a result, the first rule is that only the connected nodes should be overlapped by a sub-VN.

Second, if there are lots of nodes that connect to a sub-VN, we should select a part of the connected nodes to overlap. Here, the connected nodes have different connection strength to a sub-VN. On the one hand, the connection strength is influenced by the number of connections. If a virtual node has many connections to a sub-VN, this node should have a stronger connection strength than the virtual nodes with few connections. In Fig. 3(a), node 7 has two connections to sub-VN1 while node 6 has only one connection. Thus, node 7 may have stronger connection strength than node 6. On the other hand, as the topologies of VNs are weighted graphs, the connection strength is also influenced by the weights of connections (i.e., link weights). If the connections between the virtual node and the sub-VN are provided with high link weights, they can raise the connection strength as well.

To measure the connection strength quantitatively, we devise the *CS* metric to evaluate the connection strength between the sub-VN G_v' and the connected virtual node n, which is calculated as

$$CS(G'_v, n) = |L| \times \sum_{l \in L} lw_v(l)$$
$$L = \left\{ l_{mn} \in L_v | m \in N'_v \right\} \tag{7}$$

where the virtual link set $\boldsymbol{L}$ represents the connections between the sub-VN G_v' and the virtual node n. The *CS* metric comprehensively considers the impact from the number of connections and link weights. The more connections and link weights that exist between the virtual node and the sub-VN, the stronger the corresponding connection strength is. Virtual nodes with large *CS* values are more worthy of overlapping. For example in Fig. 3 (a), the *CS* between sub-VN1 and node 7 is evaluated as $(10 + 20) \times 2 = 60$ while the *CS* between sub-VN1 and node 6 is evaluated as $30 \times 1 = 30$. Since node 7 has a stronger connection strength than node 6, node 7 is more worthy of overlapping by sub-VN1 than node 6.

The entire overlapping decomposition strategy is shown in lines 2–13 of Algorithm 1. First, we find the connected nodes to sub-VNs (line 4 in Algorithm 1). If there are lots of connected nodes and they exceed the maximum number of overlapping nodes *max_OL*, the connected nodes are sorted in decreasing order according to the *CS* metric [i.e., (7)] (line 6 in Algorithm 1). Then, the top *max_OL* connected nodes and their links are added to the sub-VNs (line 7, and lines 11 and 12 in Algorithm 1). The parameter *max_OL* is specified in advance and we will study it later in Section V. Fig. 3(b) depicts an example of the overlapping decomposition. The VN with eight nodes is partitioned into two overlapping sub-VNs. The nodes 1–4, 7 compose sub-VN1 and nodes 4–8 compose sub-VN2. Nodes 4 and 7 are overlapped by these two sub-VNs.

Algorithm 2 Integration

input: the sub-VN ***sub***, the mapping ***ms*** of sub-VN, the best solution of VNE ***gbest***

output: the fitness value of the sub-solution ***ms***;

1: ***gbest'***=***gbest***;
2: **for** each virtual node *node* in ***sub***
3: prune the mapping results of *node* and its connected links from ***gbest'***;
4: **end for**
5: ***ms*** = ***ms*** ∪ ***gbest'***;
6: evaluate the fitness value of ***ms***;

C. Subgraph Mapping and Graph Integration

After the whole VN is divided into overlapping sub-VNs, different metaheuristics for VNE can be adopted to embed sub-VNs in the subgraph mapping procedure. The embedding of sub-VNs includes the mapping of virtual nodes and all connected virtual links, which means the links that connect two sub-VNs are also embedded [e.g., links (7, 8) and (4, 6) for embedding sub-VN1 in Fig. 3(b)].

1) Graph Integration: After the embedding of sub-VNs, the next is to integrate subsolutions to evaluate the fitness value of them. Actually, the embedding of sub-VNs is a partial solution since only a part of a VN is mapped. As the objective of VNE is to minimize the allocated resources for all virtual links [i.e., (6)], the fitness value for partial solutions cannot be evaluated directly. Partial solutions should be complemented as whole solutions, and then the fitness values of them can be evaluated. For fair comparisons, the complemented part should be identical, such that the only difference among complemented solutions is the embedding of sub-VNs. Hence, the fitness values of complemented solutions can reflect the quality of embedding sub-VNs.

To complement partial solutions, we integrate the embedding of sub-VNs with the best solution ***gbest***, found by the population so far in the graph integration procedure (shown in Algorithm 2). Since the embedding of sub-VNs includes the mapping of virtual nodes and their connected links, we first prune the mappings of these elements from the best solution (lines 2–4 in Algorithm 2). Then the embedding of the sub-VN is integrated with the pruned best solution (line 5 in Algorithm 2).

Note that a feasible solution to VNE problems should satisfy the constraints of both node mappings and link mappings, seeing (1)–(5). During the graph integration procedure, the best solution and the embedding of sub-VNs may have a conflict. For example, some substrate nodes can be selected in both the embedding of sub-VNs and the best solution. In this situation, the integration of ***gbest*** and the embedding of sub-VNs may violate the constraint represented by (2) (i.e., two virtual nodes are not allowed to be mapped on the same substrate node). To avoid the violation of constraints, we use ***gbest*** as a reference and the embedding of sub-VNs should avoid the conflict with the best solution during the subgraph mapping procedure. As a result, the complemented solutions can satisfy the constraints of VNE problems and they are guaranteed to be feasible.

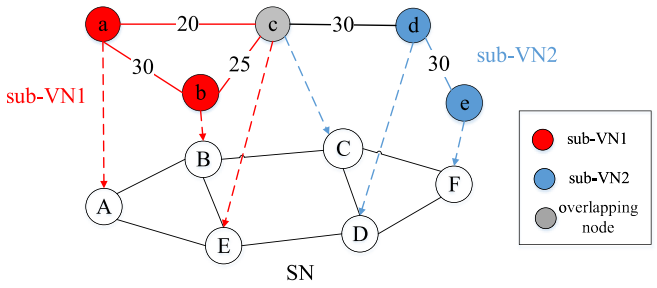


Fig. 4. Mapping of the VN with two overlapping sub-VNs. After the sub-VN1 and sub-VN2 are optimized, the overlapping node c has two different mappings $c \rightarrow E$ and $c \rightarrow D$ since the node c is mapped twice. With the competitive strategy, the better one will be chosen.

Algorithm 3 Competitive Strategy

input: globally best solution ***gbest***, the set of overlapping virtual nodes ***OV***;

output: updated ***gbest***;

1: **for** each virtual node v in ***OV***
2: candidate(v)={n∈***N***s|substrate node n hosts v};
3: connect(v)={***sub***∈***subs***|v is directly connected to ***sub***};//***subs*** is the set of all sub-VNs decomposed by Algorithm 1
4: **for** each sub-VN ***sub*** in connect(v)
5: evaluate the connection strength between v and ***sub*** using Eq.(7);
6: **end for**
7: select the strongest ***sub***∈connect(v) connected to v;
8: select the substrate node n from candidate(v) corresponding to ***sub*** and construct the virtual node mapping (v, n);
9: ***gbest'*** = ***gbest***\v;
10: ***cpt*** = ***gbest'***∪(v, n);// construct the competitor
11: **if** *func*(***cpt***) < *func*(***gbest***)// *func*() is the objective function, seeing Eq. (6)
12: ***gbest*** = ***cpt***;
13: **end if**
14: **end for**
15: replace the worst solution in the population with the updated ***gbest***;

2) Competitive Strategy: In overlapping decomposition, some critical virtual nodes are overlapped by multiple sub-VNs and thus an overlapping virtual node may be mapped to different substrate nodes in multiple sub-VNs. For example in Fig. 4, virtual node c is overlapped in sub-VN1 (composed of the nodes a–c) and sub-VN2 (composed of the nodes c–e). In sub-VN1, virtual node c is mapped to substrate node E while it is mapped to substrate node C in sub-VN2. As each virtual node should be mapped to one substrate node, there is a conflict among overlapping nodes when integrating subsolutions.

To deal with the conflict in graph integration, we devise the competitive strategy for overlapping virtual nodes. The basic idea is to construct competitors for the ***gbest*** solution in the use of overlapping virtual nodes. The pseudo code of the competitive strategy is presented in Algorithm 3. After the graph integration procedure, a rough ***gbest*** solution is found. For each overlapping node v, we first collect the candidate substrate nodes candidate(v) occupied by v [e.g., candidate(c) = {E, C} in Fig. 4] and collect the sub-VNs connect(v) that directly connect to v (e.g., connect(c) = {sub-VN1, sub-VN2} in Fig. 4). Note that each candidate node in candidate(v) corresponds to a sub-VN in connect(v) [e.g., E corresponds to sub-VN1 and C corresponds to sub-VN2 in Fig. 4]. Then, we evaluate the connection strength CS [i.e., (7)] between the overlapping node v and connected sub-VNs, and select the sub-VN with the strongest connection (lines 4–7 in Algorithm 3). After that, the candidate substrate node that corresponds to the strongest sub-VN will be used to construct a competitor for the overlapping virtual node v. For example in Fig. 4, the CS between c and sub-VN1 is evaluated as $(20 + 25) \times 2 = 90$ while CS between c and sub-VN2 is evaluated as $30 \times 1 = 30$. Since sub-VN1 has stronger connection strength, the corresponding substrate node E will be chosen to construct the competitor. The competitor construction is carried out in two steps. The first step is to delete the mapping of v from the ***gbest*** solution. Then the second step is to map v to the candidate substrate node and add the mapping to the competitor (lines 9 and 10 in Algorithm 3). Finally, if the competitor is better than the ***gbest*** solution, ***gbest*** will be replaced by the competitor. After all overlapping virtual nodes have been utilized to challenge ***gbest***, the competitive strategy finishes. With the competitive strategy, K overlapping virtual nodes will construct K competitors to challenge the ***gbest*** solution. By the competitive strategy, we can make full use of the information of overlapping nodes and thus the quality of ***gbest*** can be further improved.

Note that the challenged ***gbest*** synthesize the optimizing information of all sub-VNs, especially for overlapping virtual nodes. We replace the worst solution in the population with the enhanced ***gbest*** (line 15 in Algorithm 3). Through this, the optimizing information of overlapping nodes is injected into other sub-VNs and the competition result is shared with the whole population.

D. ODEA Algorithm

Now, that the graph partitioning, subgraph mapping, and graph integration procedures have been defined, we can give the full ODEA algorithm, as provided in Algorithm 4.

At first, the population is initialized randomly. Since ODEA can incorporate different metaheuristics for VNE, the population can be initialized by the corresponding initialization strategy according to combined metaheuristics. Then the VN is partitioned into several sub-VNs with the overlapping decomposition algorithm (line 3 in Algorithm 4).

After graph partitioning, the embedding of sub-VNs is optimized by the specific metaheuristic algorithm. Note that all sub-VNs are iterated over a random permutation in each generation (line 5 in Algorithm 4). This mechanism has two benefits. First, since the scale of all sub-VNs is similar, they should be treated equally. If sub-VNs are embedded with a fixed order, the sub-VNs that appear earlier in this order can have more impact on the following sub-VNs in the last. On the contrary, the random permutation can provide an equal probability of priority so that sub-VNs are embedded fairly. Second, in the overlapping decomposition, overlapping virtual nodes are embedded along with different sub-VNs and they might have different embedding results. The random permutation is able

Algorithm 4 ODEA

input: the virtual network G_v, the substrate network G_s and the metaheuristic algorithm A;

output: the best solution found by the population ***gbest***;

1: initialize the population ***pop*** randomly;
2: ***gbest*** ← the best solution from ***pop***;
3: decompose G_v into sub-VNs ***subs*** using Algorithm 1;
4: **for** *gen*=1:*max_gen*
5: shuffle the order of ***subs*** randomly;
6: **for** each ***sub*** in ***subs***
7: partial solutions ***sub_solus*** ← optimize the mapping of ***sub*** with *A* referring to ***gbest***;
8: complemented solutions ***com_solus*** ← unify ***sub_solus*** with ***gbest*** using Algorithm 2;
9: ***gbest'*** ← the best complemented solution in ***com_solus***;
10: **if** *func*(***gbest'***)<*func*(***gbest***)// *func*() is the objective function, seeing Eq. (6)
11: ***gbest*** = ***gbest'***;
12: **end if**
13: **end for**
14: competitive strategy using Algorithm 3;
15: **end for**

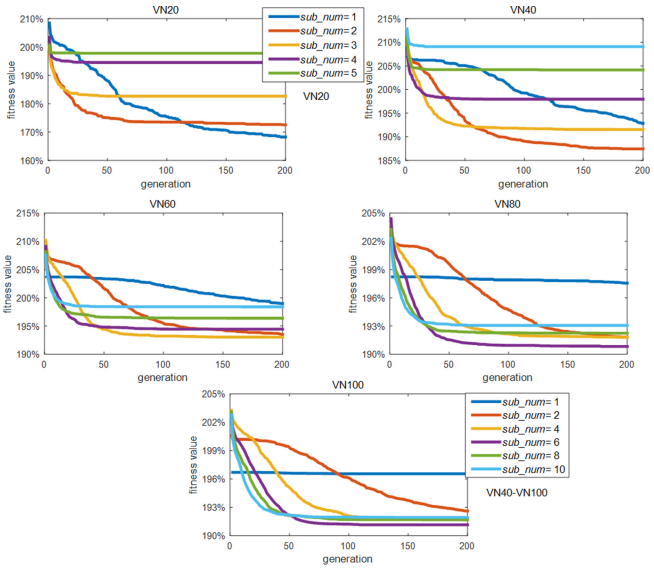


Fig. 5. Converging curves with varied *sub_num* for the exclusive decomposition.

to explore more various combinations of embedding results for overlapping nodes and it is helpful to find good combinations of values.

During the subgraph mapping procedure, different metaheuristics for VNE can be adopted and the best solution ***gbest*** found so far is taken as a reference to avoid the conflict between partial solutions and the best solution (line 7 in Algorithm 4). Next, the partial solutions are complemented with ***gbest*** in graph integration procedure (line 8 in Algorithm 4). In this way, the fitness value of embedding sub-VNs can be evaluated. The best embedding result of sub-VNs is compared with ***gbest*** and the better one will be preserved (lines 10–12 in Algorithm 4). After all sub-VNs are optimized, a rough ***gbest*** solution is obtained and the competitive strategy is used to enhance ***gbest*** (line 14 in Algorithm 4).

V. Parameter Study of ODEA

There are two parameters in ODEA, the number of sub-VNs *sub_num* (line 1 in Algorithm 1) and the maximum number of overlapping nodes *max_OL* (line 5 in Algorithm 1). In this section, we first introduce the experimental settings and then the study of these two parameters is presented.

A. Experimental Settings

To study the impact of parameters on optimization, we conduct experiments on embedding a single VN. For fair comparisons, the topologies of all networks are generated by the GT-ITM tool [47], which is used in many VNE studies [34], [48]. The connectivity rate of SNs is fixed at 10% [21]. The node weights and link weights of SNs are uniformly distributed between 50 and 100 [34]. The substrate nodes vary from 20 to 100 in increments of 20 to figure out the trend of parameters. As for VNs, we follow the replication idea proposed in [44]. The node weights and link weights of VNs are 10% of corresponding substrate nodes and links. This way of generating VNs can guarantee that the global optimum exists and is known in advance.

We adopt SPSO [30] as the embedding algorithm. The related parameters of SPSO, such as the inertia weight, are set according to [30]. The maximum generation in all experiments is set to 200 to ensure the convergence of compared algorithms. We use Java to implement all algorithms and perform the programs on a machine with Intel Core i5-4590 CPU at 3.30 GHz. The operating system is Linux and the JDK version is 1.8. Thirty independent runs are performed to study the average performance.

B. Study of the Parameter sub_num

sub_num is the number of sub-VNs that we expect to partition. The setting of *sub_num* should balance the scale of sub-VNs and the dependence among sub-VNs. A large *sub_num* means the scale of sub-VNs is small and the connections among *sub-VNs* will increase. Embedding small sub-VNs is much easier than large ones but the connections among sub-VNs might deteriorate the performance of ODEA. To investigate the parameter *sub_num*, we conduct experiments on different scale of networks with various *sub_num*. The parameter *max_OL* is set to 3 when analyzing parameter *sub_num*, which will be analyzed later.

Figs. 5 and 6 depict the converging curves of varied *sub_num* with the exclusive decomposition and the overlapping decomposition, respectively. The title "VN20" represents that the VN is provided with 20 nodes. *sub_num* is varied in {1, 2, 4, 6, 8, 10} for VNs with more than 40 nodes and is varied in {1, 2, 3, 4, 5} for VNs with 20 nodes. Note that *sub_num* = 1 means VNs are not partitioned in fact. The ordinates in Figs. 5 and 6 are fractions of the optimal fitness value, where 100% means the algorithm finds the optimal value and 200% means the double of the optimal value. From Figs. 5 and 6, we can first observe that *sub_num* influences the performance of ODEA as approximate unimodal

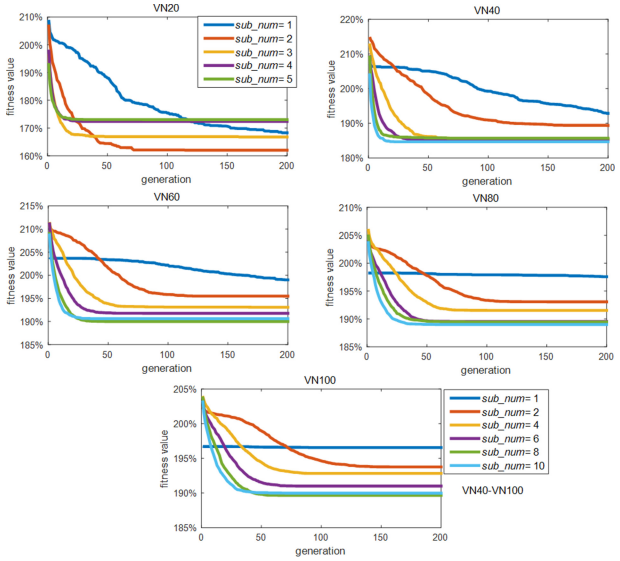


Fig. 6. Converging curves with varied *sub_num* for the overlapping decomposition.

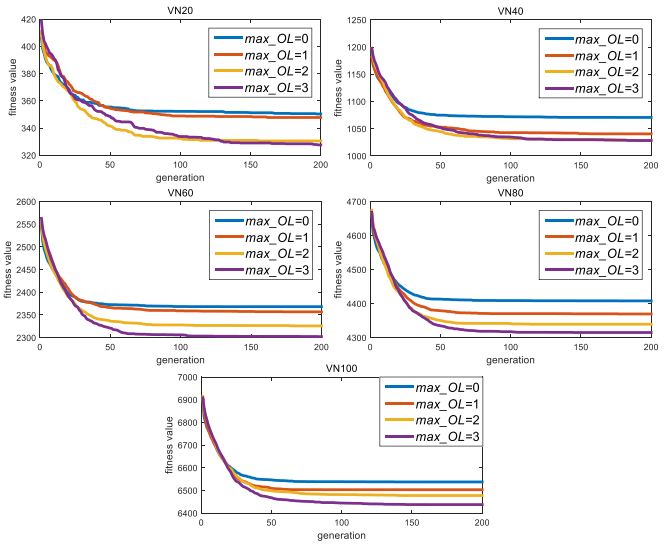


Fig. 7. Converging curves with varied *max_OL* for overlapping decomposition.

functions in each instance. For example, in the instance of "VN60" in Fig. 5, the performance of ODEA becomes better and better when *sub_num* increases from 1 to 4. However, the performance of ODEA becomes poorer and poorer when *sub_num* increases from 6 to 10. Second, the best value of *sub_num* is positively correlated to the scale of networks. In small-scale instances, ODEA with small *sub_num* can obtain good results while the algorithms with large *sub_num* perform better in large-scale VNs. Based on the above analysis, we use following formula to linearly fit *sub_num:*

$$sub\_num = \underset{s\in\{2,4,6,8,\ldots\}}{\arg\min} \{|\text{sizeof}(G_V)/\rho - s|\} \tag{8}$$

where G_v is a VN and ρ is the slope to describe the trend of *sub_num*. ρ is set to 15 for the exclusive decomposition and is set to 10 for the overlapping decomposition according to Figs. 5 and 6.

Another interesting phenomenon is that, as shown in Figs. 5 and 6, the performance of nondecomposition algorithms (i.e., $sub\_num = 1$) degrades when the scale of VNs increases. In small-scale VNs, the nondecomposition algorithms can optimize solutions well, such as in VN20 and "VN40." However, in large-scale VNs (e.g., "VN80" and "VN100"), the converging curves of nondecomposition algorithms are approximately horizontal lines, which means the convergence speed in large-scale instances is much slower than that in small networks, leading to poor performance.

C. Study of the Parameter max_OL

To reduce the influence of dependent sub-VNs, the sub-VNs are overlapped with each other during the optimization. The maximum number of overlapping nodes for sub-VNs is denoted as *max_OL*. To investigate parameter *max_OL*, we conduct experiments on different scale of networks with various *max_OL*. The parameter *sub_num* is set as (8) when analyzing *max_OL*.

Fig. 7 depicts the converging curves of varied *max_OL* with the overlapping decomposition. *max_OL* is varied in {0, 1, 2, 3} and $max\_OL = 0$ means sub-VNs are exclusive with each other (i.e., the exclusive decomposition). From Fig. 7, the better performance of algorithms can be observed on larger *max_OL* in each instance. This result verifies that the dependence among sub-VNs indeed degrades the performance of ODEA. That is, using more overlapping nodes means that more connections among sub-VNs are considered during optimization, which can reduce the influence of dependence from sub-VNs and improve the performance of ODEA.

Besides, we can see that the influence of *max_OL* is similar in all instances and $max\_OL = 3$ can always obtain the best results. This result implies that *max_OL* is independent of the scale of VNs. Based on the observation of Fig. 7, *max_OL* is set to 3 in all instances.

In the experiments, *max_OL* is tested up to 3. If *max_OL* is larger than 3, the performance might be better but the time complexity increases at the same time. In VNE problems, since each virtual link is mapped to the substrate path using the shortest path algorithm, the mapping of virtual links consumes more running time than the mapping of virtual nodes. The time complexity of the shortest path algorithm is usually $O(N^2)$ where N is the number of substrate nodes. For the overlapping decomposition, the overlapping nodes add extra virtual links to the sub-VNs and these extra virtual links will consume more running time. Given the size of VN n and number of sub-VNs *sub_num*, the size of each sub-VN is about (n/sub_num). With the specified connectivity rate r, there are about $(n/sub\_num)\times r$ connections between each overlapping node and the sub-VN. Since we have *max_OL* overlapping nodes, the total extra virtual links for each sub-VN are about $(n/sub\_num)\times r \times max\_OL$. Hence, the extra time complexity for all overlapping nodes is $O([n/sub\_num]\times r \times max\_OL\times N^2)$. The parameter *sub_num* increases with the scale of VNs and thus the ratio (n/sub_num) approaches a constant. The connectivity rate r and the number

of substrate nodes N are also constants. Therefore, only the value of *max_OL* influences the complexity and larger *max_OL* results in a higher time complexity. To balance the tradeoff of efficiency and effectiveness, *max_OL* is tested up to 3 in our experiments.

We also analyze the parameters *sub_num* and *max_OL* by varying both at the same time. The experimental results on VN20 and VN100 are presented in Fig. S5 in the supplementary materials. From Fig. S5 in the supplementary materials, it can be observed that the algorithm with large *max_OL* usually achieves better performance. The optimal parameters for VN20 are $sub\_num = 2$ and $max\_OL = 3$, and those for VN100 are $sub\_num = 10$ and $max\_OL = 3$. The analysis results are consistent with the experiments in Figs. 6 and 7.

VI. Comparative Experiments

To verify the promising performance of ODEA, we conduct comparative experiments in this section. First, as different metaheuristics for VNE can be incorporated, we combine different metaheuristics with ODEA to verify its generality. Second, we compare the performance of ODEA in online scenarios to make the experimental results more practical.

A. Generality of ODEA

The objective of VNE problems is to minimize the allocated resources for mapping VNs. To fairly compare the optimizing capability of different algorithms, we investigate the embedding of a single VN. The objective function is formulated in (6) and the network attributes are similar to those in Section V. To test algorithms in large-scale situations, experiments are conducted on two kinds of network scale, VNs with 80 nodes and VNs with 100 nodes (most existing works only test the VNs with no more than 40 nodes [21], [49]). For each kind of network scale, 12 different instances are generated and 30 independent runs for each instance are executed for statistics.

To verify the generality of ODEA, we combine ODEA with four representative metaheuristic approaches: 1) SPSO [30]; 2) UEPSO [21]; 3) RWPSO [22]; and 4) CB-GA [25], and the resultant algorithms are denoted as OD-SPSO, OD-UEPSO OD-RWPSO, and OD-CB-GA, respectively. Moreover, to investigate the effectiveness of overlapping decomposition, we compare ODEA-based approaches with the corresponding ones based on the exclusive decomposition, denoted as EX-SPSO, EX-UEPSO and EX-RWPSO, EX-CB-GA. We also adopt the variant of approximation algorithm VF2 [50] as a baseline, denoted as VNE-VF in the supplementary materials. All compared algorithms are concluded in Table I. Based on the experiments in Section V, the number of sub-VNs *sub_num* is set according to (8) for exclusive decomposition and overlapping decomposition. The maximum number of overlapping nodes *max_OL* is set to 3 for overlapping decomposition in all instances.

Table II presents the comparison results of fitness values in large-scale situations for different algorithms. Due to the space limitation, a part of results including CB-GA are put in the supplementary materials. The column "optima" represents the optimal fitness value in each instance [44]. A two-tailed t-test at significance level 0.05 is conducted to examine whether the results are significantly different or not. According to the t-test results, the best results are highlighted in bold.

TABLE I
Compared Algorithms

Notation	Algorithm description
SPSO	Set-based particle swarm optimization for discrete optimization.
EX-SPSO	the combination of SPSO and exclusive decomposition
OD-SPSO	the combination of SPSO and overlapping decomposition
UEPSO	unified enhanced particle swarm optimization for VNE
EX-UEPSO	the combination of UEPSO and exclusive decomposition
OD-UEPSO	the combination of UEPSO and overlapping decomposition
RWPSO	particle swarm optimization with Markov random walk model for VNE.
EX-RWPSO	the combination of RWPSO and exclusive decomposition
OD-RWPSO	the combination of RWPSO and overlapping decomposition
CB-GA	Genetic algorithm to solve VNE problem.
EX- CB-GA	the combination of CB-GA and exclusive decomposition
OD- CB-GA	the combination of CB-GA and overlapping decomposition
VNE-VF	A variant of approximation algorithm VF2 with backtracking and pruning.

From Table II, we can see that all decomposition-based algorithms (with the prefix "OD" or "EX") can significantly outperform the original metaheuristics in most instances. Such promising results verify that the divide-and-conquer strategy is effective in optimizing the embedding of large-scale VNs. The effectiveness of decomposition-based algorithms comes from two sides. On the one hand, ODEA and EX-VNE adopt the divide-and-conquer mechanism and a large VN is decomposed into small sub-VNs. Embedding sub-VNs is much easier than embedding large VNs. Thus, the better embedding results of sub-VNs can be found to improve the quality of whole solutions. On the other hand, the large VNs are decomposed by MLkP method, which can reduce the dependence among sub-VNs. Hence, the disturbance from other sub-VNs can be reduced, which is also beneficial for ODEA and EX-VNE.

In terms of the comparison between the exclusive decomposition and the overlapping decomposition, algorithms with overlapping decomposition can outperform those with exclusive decomposition, especially on networks with 100 nodes. For UEPSO and RWPSO on the networks with 80 nodes, there is no significant difference between exclusive decomposition and overlapping decomposition on some instances (e.g., "VN80-2" and "VN80-4"). However, exclusive decomposition cannot outperform overlapping decomposition on any instances. As for SPSO, OD-SPSO significantly outperforms EX-SPSO on all VNs with 80 nodes. On the networks with 100 nodes, all metaheuristics with overlapping decomposition can significantly outperform exclusive decomposition on all instances. The superiority of overlapping decomposition implies that the dependence among sub-VNs indeed degrades the capability of ODEA. In exclusive decomposition, all sub-VNs are optimized independently. Hence, when embedding a sub-VN, other connected sub-VNs might disturb the optimization. With the overlapping decomposition, sub-VNs are embedded with the portion of other connected

TABLE II
COMPARISON RESULTS OF LINK COSTS IN LARGE-SCALE SITUATIONS FOR DIFFERENT ALGORITHMS

instance	optimal		SPSO	EX-SPSO	OD-SPSO	UEPSO	EX-UEPSO	OD-UEPSO	RWPSO	EX-RWPSO	OD-RWPSO
		mean	4.108E+03	3.992E+03	**3.941E+03**	4.569E+03	**4.133E+03**	**4.113E+03**	4.534E+03	4.169E+03	**4.128E+03**
VN80-1	2029	std	5.195E+01	4.964E+01	5.886E+01	3.034E+01	5.472E+01	5.626E+01	4.299E+01	5.082E+01	7.247E+01
		t-test	–	8.859	12.026	–	36.950	43.541	–	27.808	27.764
		mean	4.489E+03	4.366E+03	**4.304E+03**	4.914E+03	**4.504E+03**	**4.485E+03**	4.916E+03	**4.490E+03**	**4.507E+03**
VN80-2	2296	std	4.664E+01	4.466E+01	5.754E+01	3.055E+01	5.955E+01	5.026E+01	4.129E+01	4.537E+01	4.314E+01
		t-test	–	10.933	14.365	–	33.962	41.105	–	38.019	39.133
		mean	4.457E+03	4.325E+03	**4.290E+03**	4.857E+03	4.466E+03	**4.425E+03**	4.840E+03	**4.480E+03**	**4.454E+03**
VN80-3	2270	std	4.623E+01	4.576E+01	5.038E+01	4.538E+01	6.083E+01	5.973E+01	5.388E+01	5.146E+01	6.034E+01
		t-test	–	11.250	12.814	–	34.042	31.177	–	28.995	30.344
		mean	4.295E+03	4.196E+03	**4.108E+03**	4.738E+03	**4.344E+03**	**4.336E+03**	4.732E+03	**4.364E+03**	**4.338E+03**
VN80-4	2172	std	4.400E+01	4.827E+01	4.822E+01	3.489E+01	5.446E+01	4.141E+01	3.660E+01	4.819E+01	5.320E+01
		t-test	–	7.862	13.872	–	33.148	36.429	–	32.240	36.657
		mean	4.531E+03	4.399E+03	**4.338E+03**	4.932E+03	**4.514E+03**	**4.504E+03**	4.919E+03	**4.521E+03**	**4.499E+03**
VN80-5	2320	std	4.250E+01	3.122E+01	5.029E+01	2.618E+01	6.188E+01	6.568E+01	4.038E+01	4.309E+01	4.076E+01
		t-test	–	15.505	16.569	–	32.671	33.233	–	41.415	41.997
		mean	4.311E+03	4.188E+03	**4.148E+03**	4.741E+03	4.343E+03	**4.299E+03**	4.721E+03	**4.335E+03**	**4.322E+03**
VN80-6	2191	std	3.617E+01	3.950E+01	4.453E+01	3.610E+01	5.220E+01	4.585E+01	4.027E+01	4.147E+01	4.865E+01
		t-test	–	14.883	16.190	–	34.041	49.978	–	41.136	34.100
		mean	6.636E+03	6.454E+03	**6.409E+03**	7.218E+03	6.756E+03	**6.595E+03**	7.209E+03	6.725E+03	**6.589E+03**
VN100-1	3355	std	4.825E+01	5.553E+01	5.269E+01	4.127E+01	7.539E+01	4.969E+01	3.154E+01	6.999E+01	5.940E+01
		t-test	–	13.627	15.915	–	31.152	57.197	–	38.188	52.120
		mean	7.094E+03	6.928E+03	**6.847E+03**	7.675E+03	7.217E+03	**7.071E+03**	7.645E+03	7.215E+03	**7.068E+03**
VN100-2	3701	std	4.911E+01	5.410E+01	6.303E+01	3.036E+01	6.058E+01	5.020E+01	4.113E+01	5.835E+01	5.971E+01
		t-test	–	11.032	18.845	–	36.059	62.014	–	34.040	45.130
		mean	6.925E+03	6.744E+03	**6.687E+03**	7.509E+03	7.049E+03	**6.901E+03**	7.468E+03	7.042E+03	**6.890E+03**
VN100-3	3601	std	5.408E+01	5.849E+01	6.065E+01	3.241E+01	5.795E+01	6.356E+01	5.855E+01	5.589E+01	4.898E+01
		t-test	–	13.909	15.249	–	36.196	45.515	–	30.347	39.160
		mean	6.610E+03	6.473E+03	**6.402E+03**	7.248E+03	6.764E+03	**6.619E+03**	7.244E+03	6.741E+03	**6.617E+03**
VN100-4	3326	std	5.272E+01	6.976E+01	7.105E+01	2.953E+01	6.112E+01	6.280E+01	3.503E+01	7.119E+01	7.409E+01
		t-test	–	7.251	11.660	–	39.835	55.785	–	37.836	39.875
		mean	6.501E+03	6.339E+03	**6.276E+03**	7.091E+03	6.613E+03	**6.420E+03**	7.086E+03	6.602E+03	**6.446E+03**
VN100-5	3265	std	5.428E+01	4.822E+01	4.430E+01	3.432E+01	6.503E+01	6.667E+01	2.830E+01	6.466E+01	6.202E+01
		t-test	–	12.888	15.951	–	35.880	56.075	–	39.574	52.967
		mean	6.522E+03	6.368E+03	**6.329E+03**	7.137E+03	6.679E+03	**6.522E+03**	7.113E+03	6.662E+03	**6.538E+03**
VN100-6	3317	std	5.398E+01	5.355E+01	5.486E+01	3.111E+01	5.677E+01	5.187E+01	3.467E+01	5.081E+01	7.784E+01
		t-test	–	11.635	12.199	–	49.413	62.353	–	35.641	35.667

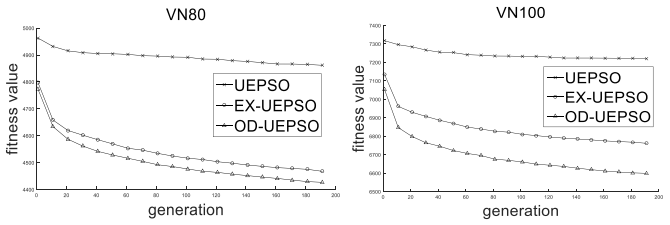


Fig. 8. Converging curves of different algorithms on the networks with 80 and 100 nodes.

sub-VNs (i.e., the overlapping nodes). Hence, each sub-VN is not optimized independently and thus the disturbance from connected sub-VNs can be reduced.

We can also observe that, although ODEA obtains the best performance, the fitness values yielded by ODEA are still almost double the ideal best fitness values. This phenomenon implies that the average length of substrate paths to host virtual links is about two. As a consequence, the performance of ODEA approaches can still be improved in the future. In addition, SPSO-based approaches can outperform other approaches in most cases. This is because SPSO uses probabilities to exactly record promising elements and introduces heuristic information frequently during the step-by-step solution construction.

Fig. 8 depicts the converging curves of compared algorithms on networks with 80 nodes and 100 nodes. In the figure, one metaheuristic approach, its ODEA and EX-VNE variants are compared. From Fig. 8, we can observe that the metaheuristics with overlapping decomposition are able to converge to the lowest fitness values. The original metaheuristics achieves the worst performance and the metaheuristics with exclusive decomposition obtain the middle results between above two kinds of algorithms. The converging curves of original metaheuristics are very gentle which means that the optimizing capability of original metaheuristics is too weak in large-scale VNE problems. As the dimensions of VNs increase, the structure of VNs becomes more complicated. It is difficult for original metaheuristics to compare different mappings of VNs from the perspective of whole solutions and find a better structure of embedding. As for ODEA-based approaches, the converging curves are much steeper than the original metaheuristics. The steep curves mean that ODEA-based approaches can continually find better solutions, even in the high dimensional problem space. In ODEA, the whole complicated VNs are decomposed into small sub-VNs with simple structure. Therefore, the better embedding of sub-VNs is easier to find and thus the better embedding of the whole VNs can also be found continually during the optimization.

TABLE III
COMPARISON RESULTS OF ALGORITHMS WITH AND WITHOUT THE COMPETITIVE STRATEGY

	OD-RWPSO*	OD-RWPSO	*t*-test	OD-UEPSO*	OD-UEPSO	*t*-test
VN80-1	4.140E+03	**4.094E+03**	3.572	4.125E+03	**4.091E+03**	2.919
VN80-2	3.907E+03	**3.867E+03**	3.848	3.882E+03	3.866E+03	1.115
VN80-3	3.925E+03	**3.878E+03**	3.692	3.905E+03	**3.868E+03**	2.881
VN80-4	4.014E+03	**3.956E+03**	4.303	3.994E+03	**3.964E+03**	2.428
VN80-5	4.762E+03	4.738E+03	1.637	4.752E+03	**4.719E+03**	2.116
VN80-6	4.496E+03	4.477E+03	1.673	4.471E+03	4.458E+03	0.916
VN100-1	6.689E+03	**6.655E+03**	2.055	6.698E+03	**6.643E+03**	3.310
VN100-2	6.450E+03	**6.395E+03**	3.324	6.450E+03	**6.389E+03**	4.116
VN100-3	6.488E+03	**6.432E+03**	3.839	6.474E+03	**6.414E+03**	4.357
VN100-4	6.791E+03	**6.741E+03**	3.122	6.795E+03	**6.749E+03**	2.370
VN100-5	7.184E+03	**7.114E+03**	3.675	7.190E+03	**7.129E+03**	4.547
VN100-6	7.227E+03	**7.153E+03**	4.611	7.220E+03	**7.153E+03**	4.624

The mark "*" represents the algorithm without the competitive strategy.

To investigate the effect of the competitive strategy, we compare ODEA without the competitive strategy, which is marked with "*" in Table III. From Table III, it can be observed that OD-RWPSO and OD-UEPSO can both significantly outperform its variant without the competitive strategy on 10 out of 12 instances. On the one hand, the competitive strategy makes full use of the optimizing results of overlapping nodes with the consideration of VN topology information. Hence, a better combination values of overlapping virtual nodes are explored. On the other hand, the updated ***gbest*** after competition is injected to other sub-VNs. As a result, the improvement of one sub-VN by the competitive strategy is shared within the whole population. Therefore, the competitive strategy can significantly improve final solutions for ODEA in general.

B. Evaluation for Online VNs

In reality, the VNE system needs to deal with many VN requests demanded by customers. As VN requests can arrive at the system at any time, only arrived VNs are known to optimizers while future VNs are unknown. When a VN request comes to the system, the specific VNE approach will search for the optimal embedding of VNs. If the VN cannot be embedded due to the limitation of resources, it will be rejected by the system. Different from embedding a single VN which is a static situation, online VNs are dynamic and more challenging than single VN, which is necessary to test VNE approaches from multiple views. In this section, we first introduce the comparative metrics for online scenarios and then the experiments for online VNs are presented.

1) Comparative Metrics: The objective of embedding a single VN is to minimize the allocated resources formulated by (6). However, as online scenarios are dynamic, different metrics are utilized to evaluate the performance of algorithms. Similar to the previous works [21], [22], [25], three kinds of metrics are utilized: 1) the average revenue (avgR); 2) the revenue to cost ratio (R/C); and 3) the acceptance ratio.

The revenue of infrastructure providers (InPs) mainly comes from the satisfaction of the CPU demand and the bandwidth demand of VNs. The more virtual resources customers require, the more revenue InPs will earn. Given a VN G_v, the revenue obtained by G_v at time t is formulated as

$$R(G_v, t) = \sum_{n \in N_v} nw_v(n) + \sum_{l \in L_v} lw_v(l) \tag{9}$$

which is the summation of node weights and link weights in the VN. In the online scenarios, VN requests arrive at the system continuously. To evaluate the revenue in a long term, the long-term average revenue is defined as

$$\text{avgR} = \lim_{T \to \infty} \sum_{t=0}^{T} R(G_v, t)/T. \tag{10}$$

From the perspective of InPs, an excellent VNE algorithm should make more revenue for them. Hence, the larger the average revenue, the better the VNE approaches.

For embedding the same VN, different embedding results might occupy various substrate resources. Using more substrate resource will pay more costs. Given a VN G_v and an SN G_s, the cost of embedding G_v at time t is evaluated as

$$C(G_v, t) = \sum_{n \in N_v} nw_V(n) \ + \sum_{l \in L_V} \sum_{i=1}^{|\Gamma_L(l)|} lw_V(l) \tag{11}$$

which is the total allocated node weights and link weights. From the perspective of long term, the long-term revenue to R/C is defined as

$$R/C = \lim_{T \to \infty} \left(\sum_{t=0}^{T} R(G_v, t) \Big/ \sum_{t=0}^{T} C(G_v, t) \right). \tag{12}$$

In R/C ratio, the cost is always larger or equal to the revenue and thus the range of R/C belongs to (0, 1]. R/C ratio can directly reflect the optimizing capability of algorithms. For embedding the same VN, the revenue is the same but the cost is different. A good VNE algorithm can pay lower costs and thus the R/C ratio becomes larger. A value of 1.0 for R/C means that the algorithm can find the optimal solutions.

In terms of the quality of service, a good VNE algorithm should accept more VN requests to satisfy users' requirements. The acceptance ratio of VN requests is defined as

$$\lim_{T \to \infty} \left(\sum_{t=0}^{T} \text{VNR}_a \Big/ \sum_{t=0}^{T} \text{VNR} \right) \tag{13}$$

where VNR_a is the number of accepted VNs and VNR is the total arrived VNs. The range of acceptance ratio is [0, 1]. A value of 0 means none of VNs can be successfully embedded on the SN and a value of 1.0 represents the opposite. Obviously, the larger the acceptance ratio, the better service algorithms can provide.

2) Comparison Results for Online VNs: To make the research more realistic, we simulate the online environments in our experiments. Similar to previous works [10], [51], we assume that VNs arrive in a Poisson process and the rate is about 5 VNs per 100 time units. Each VN request is associated with the duration which follows the exponential distribution with an average of 500 time units [21]. The scale of VNs is uniformly distributed between 80 nodes and 100 nodes, and

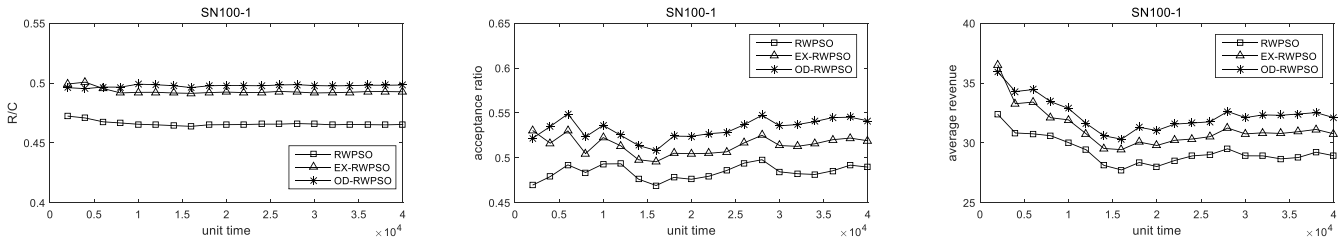


Fig. 9. Comparison results of online VNs in terms of the R/C ratio, acceptance ratio, and average revenue.

the scale of SNs is 100 nodes. The connectivity rate for VNs and SNs is fixed at 0.1. Note that the topologies of VNs and SNs are both generated independently and thus they are not identical. The node weights and link weights are randomly distributed between 50 and 100 for SNs and between 1 and 5 for VNs [34], [43]. Each simulation runs for 40 000 time units and the optimization does not affect the simulation time. According to Fig. 7, the maximum generation is set to 100 since it is enough for all algorithms to converge to satisfactory fitness values.

Fig. 9 presents the comparison results of online VNs obtained by different algorithms in terms of R/C ratio, acceptance ratio, and average revenue. In each sub figure, we compare the original metaheuristics and its variant with the exclusive decomposition and the overlapping decomposition. Due to the space limitation, only the results of RWPSO are presented.

In R/C ratio, ODEA-based algorithms can obtain higher R/C ratios than the corresponding original metaheuristics. Higher R/C ratios imply the allocated resources in ODEA are smaller than those in the original metaheuristics on average. Thus, the optimizing capability of ODEA is still better than original metaheuristics for dealing with online VNs. The R/C ratios obtained by the overlapping decomposition are competitive or slightly better than the exclusive decomposition. The superiority of overlapping decomposition verifies that the overlapping mechanism indeed reduces the disturbance from dependent sub-VNs and thus the optimizing capability of ODEA can be further improved.

As for acceptance ratios, the acceptance ratios obtained by ODEA are higher than original metaheuristics and exclusive-decomposition approaches. This is because less substrate resources are needed by ODEA to embed VNs and relatively more resources are remained for accepting future VNs. The last metric is the average revenue. To some degree, the average revenue is correlated to the acceptance ratio when the size of VNs is similar. Higher acceptance ratios represent more VN requests are accepted, which is more probable to bring high revenue naturally. Hence, ODEA can gain more average revenue than the original metaheuristics and exclusive-decomposition approaches. The trend of curves in average revenue is also similar to those in acceptance ratios.

In general, thanks to the decomposition mechanism, ODEA can still outperform traditional metaheuristics for solving online VNs, in terms of R/C ratio, acceptance ratio, and average revenue. At the same time, the overlapping mechanism can effectively reduce the influence of dependence from other sub-VNs and thus the optimizing capability is also further improved, which is reflected in higher R/C and acceptance ratios than the exclusive decomposition.

VII. Conclusion

Solving VNE problems in large-scale situations is important and challenging. In this article, we propose ODEA to solve large-scale VNE problems. In ODEA, large VNs are decomposed into overlapping sub-VNs, which can reduce the disturbance from interconnected subcomponents. We conduct comparative experiments on embedding a single VN and online VNs. The comparison results on large-scale scenarios verify that the proposed ODEA is promising.

In ODEA, all sub-VNs are optimized with the same optimizer. Actually, these sub-VNs might have different attributes and thus they can be optimized by various algorithms. How to select appropriate optimizers adaptively to embed sub-VNs will be studied in the future.

References

[1] D. Conte, P. Foggia, C. Sansong, and M. Vento, "Thirty years of graph matching in pattern recognition," *Int. J. Pattern Recognit.*, vol. 18, no. 3, pp. 265–298, 2004.

[2] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, 1976.

[3] P. J. Hansen and P. C. Jurs, "Chemical applications of graph theory. Part I. Fundamentals and topological indices," *J. Chem. Educ.*, vol. 65, no. 7, pp. 574–580, 1988.

[4] Y. Gao, M. Wang, D. Tao, R. Ji, and Q. Dai, "3-D object retrieval and recognition with hypergraph analysis," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4290–4303, Sep. 2012.

[5] L. Hong, L. Zou, X. Lian, and P. S. Yu, "Subgraph matching with set similarity in a large graph database," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2507–2521, Sep. 2015.

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

[7] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, no. 5, pp. 862–876, 2010.

[8] W. Lu, P. Lu, Q. Sun, S. Yu, and Z. Zhu, "Profit-aware distributed online scheduling for data-oriented tasks in cloud datacenters," *IEEE Access*, vol. 6, pp. 15629–15642, 2018.

[9] D. G. Andersen. (2002). *Theoretical Approaches to Node Assignment*. [Online]. Available: http://www.cs.cmu.edu/∼dga/papers/index.html

[10] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 4, pp. 356–368, Dec. 2013.

[11] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *IEEE Commun. Lett.*, vol. 16, no. 5, pp. 756–759, May 2012.

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

[13] J. Inführ and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints," in *Network Optimization*. Heidelberg, Germany: Springer, 2011, pp. 105–117.

[14] S. Haeri and L. Trajković, "Virtual network embedding via Monte Carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, Feb. 2018.

[15] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun.*, vol. 41, no. 2, pp. 38–47, 2011.

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

[17] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. INFOCOM*, 2009, pp. 783–791.

[18] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proc. 1st ACM Workshop VISA*, 2009, pp. 81–88.

[19] F. Zhu and H. Wang, "A modified ACO algorithm for virtual network embedding based on graph decomposition," *Comput. Commun.*, vol. 80, pp. 1–15, Apr. 2016.

[20] S. Su, Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, and X. Zhao, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.

[21] Z. B. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, and Y. Luo, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *Int. J. Commun. Syst.*, vol. 26, no. 8, pp. 1054–1073, 2013.

[22] X. Cheng *et al.*, "Virtual network embedding through topology awareness and optimization," *Comput. Netw.*, vol. 56, no. 6, pp. 1797–1813, 2012.

[23] I. Fajjari, N. A. Saadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *Proc. IEEE ICC*, 2011, pp. 1–6.

[24] A. Schrijver, *Theory of Linear and Integer Programming*. Amsterdam, The Netherlands: Wiley, 1998.

[25] X. L. Chang, X. M. Mi, and J. K. Muppala, "Performance evaluation of artificial intelligence algorithms for virtual network embedding," *Eng. Appl. Artif. Intel.*, vol. 26, no. 10, pp. 2540–2550, 2013.

[26] J. Shi, B. Sullivan, M. Mazzola, B. Saravi, U. Adhikari, and T. Haupt, "A relaxation-based network decomposition algorithm for parallel transient stability simulation with improved convergence," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 3, pp. 496–511, Mar. 2018.

[27] L. Dai and B. Bai, "Optimal decomposition for large-scale infrastructure-based wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4956–4969, Aug. 2017.

[28] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[29] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[30] W.-N. Chen, J. Zhang, H. S. Chung, W.-L. Zhong, W.-G. Wu, and Y.-H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.

[31] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[32] M. A. Rashid, F. Khatib, M. T. Hoque, and A. Sattar, "An enhanced genetic algorithm for Ab initio protein structure prediction," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 627–644, Aug. 2016.

[33] X. Mi, X. Chang, J. Liu, L. Sun, and B. Xing, "Embedding virtual infrastructure based on genetic algorithm," in *Proc. Parallel Distrib. Comput. Appl. Technol.*, 2012, pp. 239–244.

[34] M. Chowdhury, M. R. Rahman and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[35] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[36] W.-N. Chen, Y.-H. Jia, F. Zhao, X.-N. Luo, X.-D. Jia, and J. Zhang, "A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 842–857, Oct. 2019.

[37] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, p. 13, 2016.

[38] J. Liu, W. Zhong, H. A. Abbass, and D. G. Green, "Separated and overlapping community detection in complex networks using multiobjective evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–7.

[39] X. Zou and J. Liu, "A mutual information-based two-phase memetic algorithm for large-scale fuzzy cognitive map learning," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 2120–2134, Aug. 2018.

[40] Y. Mei, X. Li, and X. Yao, "Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 435–449, Jun. 2014.

[41] J. Gomes, P. Mariano, and A. L. Christensen, "Dynamic team heterogeneity in cooperative coevolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 934–948, Dec. 2018.

[42] M. Gong, H. Li, E. Luo, J. Liu, and J. Liu, "A multiobjective cooperative coevolutionary algorithm for hyperspectral sparse unmixing," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 234–248, Apr. 2017.

[43] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 3, pp. 334–348, Sep. 2015.

[44] A. Fischer and H. de Meer, "Generating virtual network embedding problems with guaranteed solutions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 504–517, Sep. 2016.

[45] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. INFOCOM*, 2006, pp. 1–12.

[46] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

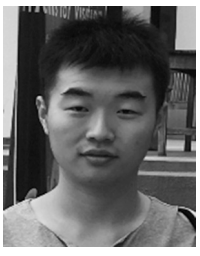
[47] E. W. Zegura, K. L. Calvert and S. Bhattacharjee, "How to model an internetwork," in *Proc. 15th Annu. IEEE INFOCOM*, 1996, pp. 594–602.

[48] 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.

[49] M. M. A. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "Multi-path link embedding for survivability in virtual networks," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 253–266, Jun. 2016.

[50] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[51] M. R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 2, pp. 105–118, Jun. 2013.



An Song (S'15) received the M.S. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2018. He is currently pursuing the Ph.D. degree in computer science with the South China University of Technology, Guangzhou.

His current research interests include evolutionary computation algorithms and their applications on large scale computing, virtual network embedding, and workflow scheduling.

Wei-Neng Chen (S'07–M'12–SM'17) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

Since 2016, he has been a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has coauthored over 100 international journals and conference papers, including over 40 papers published in the IEEE TRANSACTIONS journals. His current research interests include computational intelligence, swarm intelligence, network science, and their applications.

Prof. Chen was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation Award in 2016, and the National Science Fund for Excellent Young Scholars in 2016. He is currently the Vice-Chair of the IEEE Guangzhou Section. He is also a Committee Member of the IEEE CIS Emerging Topics Task Force. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and the *Complex and Intelligent Systems*.

Yue-Jiao Gong (S'10–M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She is currently a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 80 papers, including over 30 IEEE TRANSACTIONS papers, in the above area.

Xiaonan Luo received the B.S. degree from Jiangxi Normal University, Nanchang, China, in 1983, the M.S. degree from Xidian University, Xi'an, China, in 1985, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1991.

He is currently a Professor with the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China. He was the Director of the National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou, China. His current research interests include computer graphics, CAD, image processing, and mobile computing.

Prof. Luo was a recipient of the National Science Fund for Distinguished Young Scholars Granted by the National Nature Science Foundation of China.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Professor with the Division of Electrical Engineering, Hanyang University, Seoul, South Korea. His current research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in the above area.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.