Distributed Virtual Network Embedding System With Historical Archives and Set-Based Particle Swarm Optimization

An Song, *Student Member, IEEE*, Wei-Neng Chen, *Senior Member, IEEE*, Tianlong Gu, Huaqiang Yuan, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Virtual network embedding (VNE) is an important problem in network virtualization for the flexible sharing of network resources. While most existing studies focus on centralized embedding for VNE, distributed embedding is considered more scalable and suitable for large-scale scenarios, but how virtual resources can be mapped to substrate resources effectively and efficiently remains a challenging issue. In this paper, we devise a distributed VNE system with historical archives (HAs) and metaheuristic approaches. First, we introduce metaheuristic approaches to each delegation of the distributed embedding system as the optimizer for VNE. Compared to the heuristic-based greedy algorithms used in existing distributed embedding approaches, which are prone to be trapped in local optima, metaheuristic approaches can provide better embedding performance for these distributed delegations. Second, an archive-based strategy is also introduced in the distributed embedding system to assist the metaheuristic algorithms. The archives are used to record the up-to-date information of frequently repeated tasks. By utilizing such archives as historical memory, metaheuristic algorithms can further improve embedding performance for frequently repeated tasks. Following this idea, we incorporate the set-based particle swarm optimization (PSO) as the optimizer and propose the distributed VNE system with HAs and set-based PSO (HA-VNE-PSO) system to solve the VNE problem in a distributed way. HA-VNE-PSO is empirically validated in scenarios of different scales. The experimental results verify that HA-VNE-PSO can scale well with respect to substrate networks, and the HA strategy is indeed effective in different scenarios.**

***Index Terms*—Distributed systems, metaheuristic, particle swarm optimization (PSO), virtual network embedding (VNE).**

Manuscript received August 24, 2018; accepted November 17, 2018. Date of publication January 3, 2019; date of current version January 19, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61332002, and Grant 61876111, in part by the Natural Science Foundation of Guangdong under Grant 2015A030306024, in part by the Science and Technology Plan Project of Guangdong Province under Grant 2018B050502006, in part by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing, and in part by the Fundamental Research Funds for the Central Universities. This paper was recommended by Associate Editor S. Mostaghim. *(Corresponding authors: Wei-Neng Chen; Jun Zhang.)*

A. Song, W.-N. Chen, and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

T. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

H. Yuan is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

This paper has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the author. This includes a PDF file that contains additional tables and figures not included within the paper itself. The total size of the file is 5.59 MB.

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

Digital Object Identifier 10.1109/TSMC.2018.2884523

I. Introduction

NETWORK virtualization is a promising technology for the future development of the Internet that allows multiple users to rent the infrastructure simultaneously [1]–[3]. In network virtualization environments, the instantiations of virtual infrastructures are heterogeneous and they can coexist on a shared substrate network (SN). One main challenge in network virtualization is the deployment of virtual networks (VNs) on an SN, called VN embedding (VNE). Due to the capacity limitation of SNs, the number of VNs that can be embedded on a specific SN is restricted. Optimal embedding results can make an SN accept more VN requests. Besides, from the perspective of infrastructure providers (InPs), the optimal deployments of VNs will reduce the costs and energy consumption, which can bring more profits and improve the quality of services. Therefore, searching for optimal solutions to VNE problems is of great significance in network virtualization.

VNE problems have been proven nondeterministic polynomial hard [4], [5] and many approaches with different methodologies were proposed, including exact approaches [6], [7], heuristic approaches [8]–[12], and metaheuristic approaches [13]–[17]. Traditional exact algorithms, such as the sequential request processing [6] and the VNE with delay, routing, and location constraints [7], ensure that globally optimal solutions can be found, but may incur exponentially increasing running time when the problem scale increases. To overcome such deficiency, many researchers have turned to finding near optimal solutions. Some heuristics for VNE problems, e.g., the algorithm based on subgraph isomorphism detection (ASID) [18] and VNE with Monte Carlo search [11], have been proposed. Although heuristic approaches have shown good performance in some scenarios, most of them are easily trapped into local optima which might be far away from the global optima [1], [13]. To further optimize the deployments of VNs, metaheuristic approaches were applied in solving VNE problems, such as the unified enhanced particle swarm optimization (UEPSO) [13] and the particle swarm optimization (PSO) variant with random walk [17]. Metaheuristic approaches utilize iteratively searching techniques to find near optimal solutions, and thus they may consume more running time than heuristic approaches but they can find much better solutions.

Until now, most VNE algorithms are centralized [1] which means all tasks are performed on one centralized node. Centralized approaches may cause problems in two ways. First, centralized approaches rely on a single node to compute all embedding tasks so that the scalability of algorithms is limited, even for metaheuristic and heuristic approaches. In reality, the size of SN topologies might be more than one million nodes (e.g., Amazon EC2 [19]), thus the scalability and the efficiency of approaches in large-scale network topologies need to be considered. Second, in centralized algorithms, multiple users are not allowed to operate the resources of SNs simultaneously. Therefore, all VN requests have to be carried out by service providers (SPs) one by one, which is infeasible for the systems that serve millions of users [19]. On the contrary, in distributed VNE approaches [20]–[23], the computational tasks are assigned to several nodes instead of a single node and multiple users are allowed to operate the resources of SNs concurrently. In this way, the scalability and the concurrency of VNE systems can be guaranteed. Hence, considering the scalability and the concurrency, distributed VNE approaches are more appropriate than centralized ones in large scale scenarios, such as the agent-based distributed VNE (ADVNE) [21] and the distributed and parallel VNE (DPVNE) [22].

In the literature, existing distributed approaches are combined with heuristic VNE algorithms to embed VNs. Due to the greedy search behavior in heuristics, solutions obtained by heuristic VNE algorithms are usually local optima especially when the size of SNs increases as aforementioned. As a result, poor optimizing capability of heuristics degrades the performance of distributed approaches. Moreover, as distributed approaches are only aware of the local states of SNs, they merely search for solutions in the local area of SNs instead of the whole problem space. Therefore, this characteristic makes heuristic VNE algorithms become even worse in the distributed environment. Based on the above analysis, the combination of heuristic algorithms limits the development of distributed VNE approaches in practice.

In this paper, we propose a distributed system for solving VNE problems. The proposed system has the following two features.

1) We combine metaheuristic approaches and the distributed VNE framework together. On the one hand, metaheuristic approaches of VNE have shown promising performance compared to exact approaches and heuristic approaches [13], [16]. Therefore, the combination of metaheuristics can improve the optimizing capability of distributed VNE approaches. On the other hand, metaheuristic approaches might be time-consuming since they search for solutions iteratively. The combination of metaheuristics and distributed approaches is able to improve the efficiency of metaheuristics in return. In this paper, we devise a set-based PSO for VNE (SPSO-VNE) and incorporate SPSO-VNE as the optimizer for embedder nodes. To the best of our knowledge, we are the first to apply metaheuristics in solving distributed VNE problems.
2) We introduce archives to record historical embedding information and utilize these archives to embed coming VNs. Under many circumstances, the phenomenon of users or enterprises submitting their VN requests repetitively, is extremely common especially in the molecular dynamics [24], [25] and the computational biology [26], [27]. While embedding these repeated VNs, historical embedding results can be utilized to improve the performance of VNE systems. Therefore, we introduce archives to record historical embedding results and devise the historical-archive (HA) strategy, which uses the historical information to embed coming VNs. In this way, the quality of embedding can be further improved.

Synthesizing above two features, we propose the distributed VNE system with HAs and set-based PSO (HA-VNE-PSO) to solve VNE problems in a distributed way.

The rest of this paper is organized as follows. In Section II, the network model and the VNE problem formulation are presented. In Section III, we introduce the background of this paper. In Section IV, the proposed HA-VNE-PSO system is described. In Section V, we evaluate the effectiveness and scalability of our system on the scenarios with different scale. Finally, Section VI concludes this paper.

II. Network Model and Problem Statement

The objective of the VNE problem is to find the optimal deployment of VNs on a specified SN to fully utilize substrate resources. In this section, we present the network model in this paper and describe the VNE problem in detail.

A. Network Modeling

The SN is represented by an undirected weighted graph, $G_s = (N_s, L_s, A_s^n, A_s^l)$, where N_s is the set of substrate nodes and L_s is the set of substrate links. The A_s^n and A_s^l are the attributes of substrate nodes $n \in N_s$ and substrate links $l \in L_s$, respectively. Here, the subscript "s" stands for SNs. Without loss of generality, we consider the CPU capacity for substrate nodes and consider the bandwidth capacity for substrate links [9], [13], [18]. The residual CPU of the substrate node n is represented by c_s^n and the residual bandwidth of the substrate link l is represented by b_s^l. All loops-free paths of the SN are denoted by P_s. As shown in Fig. 1, vertices A–F are substrate nodes, which compose the SN. The weights on the edges represent the available bandwidth and the numbers on nodes represent the available CPU resources.

Similarly, an undirected weighted graph $G_v = (N_v, L_v, R_v^n, R_v^l)$ is used to represent a VN where N_v is the set of virtual nodes and L_v is the set of virtual links. To be consistent with SNs, R_v^n, and R_v^l are CPU requirements for virtual nodes $n \in N_v$ and bandwidth requirements for virtual links $l \in L_v$, respectively. Here, the subscript "v" stands for VNs. The CPU requirements of the virtual node n is represented by c_v^n and the bandwidth requirements of the virtual link l is represented by b_v^l. As shown in Fig. 1, vertices a–c are virtual nodes, which compose the VN1.

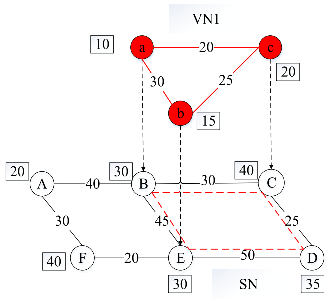


Fig. 1. Examples of VNE.

The weights on edges represent the bandwidth demands and the numbers on nodes represent the CPU demands. In online VNE systems, the ith VN request $\text{VNR}^i = (G_v, t_a, t_d)$ submitted by a customer at time t_a has its duration time t_d. Suppose the ith VN request VNR^i arrives at time t_a, if there are enough available resources in the SN, the InP allocates corresponding resources to VNR^i, including residual CPU and bandwidth. VNR^i will stay in the SN for t_d time units until the task finishes. After t_d time units, VNR^i will depart from the SN and the allocated resources will be released as well.

B. VNE Problem Statement

Given the SN $G_s = (N_s, L_s, A_s^n, A_s^l)$ and the VN $G_v = (N_v, L_v, R_v^n, R_v^l)$, the VNE problem can be abstracted as two mappings: 1) the virtual node mapping (VNoM) $M_n : N_v \to N_s'$ and 2) the virtual link mapping (VLiM) $M_l : L_v \to P_s'$, where $N_s' \subset N_s$ and $P_s' \subset P_s$. As shown in Fig. 1, the VNoM is $\{a{\to}B,\ b{\to}E,\ c{\to}C\}$ and the VLiM is $\{(a, b) \to (B, E), (a, c) \to (B, C), (b, c) \to (E, D, C)\}$. A binary variable x_i^u is used to represent the VNoM. If the virtual node $u{\in}N_v$ is mapped to the substrate node $i \in N_s$, x_i^u is equal to 1, else x_i^u is equal to 0. The VLiM is represented by a binary variable f_{ij}^{uv} as well. If the virtual link $l_{uv}{\in} L_v$ is mapped to substrate link $l_{ij} \in L_s$, then f_{ij}^{uv} is equal to 1, else f_{ij}^{uv} is equal to 0. Due to the limitation of substrate resources, following constrained rules should be obeyed. First, each node in a VN must be mapped to the only one substrate node, formally,

$$\forall u \in N_v, \sum_{i \in N_s} x_i^u = 1 \tag{1}$$

and two virtual nodes in the same VN cannot be assigned to the same substrate node, formally,

$$\forall i \in N_s, \sum_{u \in N_v} x_i^u \leq 1. \tag{2}$$

Then, the resource requirements of VNs need to be satisfied. The CPU resources provided by substrate nodes should be larger or equal to those demanded by virtual nodes, such that

$$\forall u \in N_v \quad \forall i \in N_s, x_i^u \times c_v^u \leq c_s^i. \tag{3}$$

Obviously, the bandwidth resources provided by substrate links should also be larger than or equal to those demanded by virtual links, such that

$$\forall l_{ij} \in L_s \quad \forall l_{uv} \in L_v, f_{ij}^{uv} \times b_v^{l_{uv}} \leq b_s^{l_{ij}}. \tag{4}$$

When embedding a single VN, there might be more than one feasible solution and of course, the assigned substrate resources are different. In our network model, the CPU resources allocated for virtual nodes are identical among different solutions. However, the assigned bandwidth resources for the virtual links depend on the substrate path length, which means mapping virtual links to longer substrate paths will consume more bandwidth resources. In this paper, like previous cost-optimizing VNE algorithms [13], [15], the cost function of metaheuristics when embedding a single VN is formulated as

$$\min \sum_{l_{uv} \in L_v} \sum_{l_{ij} \in L_s} f_{ij}^{uv} \times b_v^{l_{uv}}. \tag{5}$$

In the above formula, the costs of bandwidth are evaluated by the sum of all occupied bandwidth in the SN. Lower costs of bandwidth represent less allocated resources and better deployments of VNs.

In an online VNE system, VNs arrive at and depart from the system frequently. As future VNs are unknown to the system, VNs are handled one by one. When a VN comes to the system, it will be embedded on the SN with the specified VNE algorithm. If there are enough resources, the VN will stay in the system until it finishes. Otherwise, the VN will be rejected.

III. Background

In this paper, we combine the distributed VNE approach with metaheuristics to improve its optimizing capability. In this section, we first briefly introduce metaheuristics and then the related work about VNE approaches is reviewed.

A. Particle Swarm Optimization

PSO is a population-based stochastic optimization algorithm inspired by birds flocking and fish schooling, which is first proposed by Kenndy and Eberhart [28]. PSO is composed of a population of NP ($\text{NP} = 1, 2, 3\ldots$) particles and each particle of PSO represents a potential solution to the problem. The ith particle in PSO maintains two vectors, a position vector $\boldsymbol{X}_i = (x_i^1, x_i^2, \ldots, x_i^n)$ and a velocity vector $\boldsymbol{V}_i = (v_i^1, v_i^2, \ldots, v_i^n)$ where n is the dimension of the problem. First, PSO randomly initializes the population including positions and velocities. Then, each particle iteratively updates its position and velocity according to the best solutions found by itself and the swarm so far. The updating rules for the ith particle on dimension j are given by

$$v_i^j \leftarrow wv_i^j + c_1 r_1^j \left(\text{pbest}_i^j - x_i^j\right) + c_2 r_2^j \left(\text{gbest}^j - x_i^j\right) \tag{6}$$

$$x_i^j \leftarrow x_i^j + v_i^j \tag{7}$$

where $\textbf{pbest}_i$ and **gbest** are the best solution historically found by the ith particle and the entire swarm, respectively. r_1 and r_2 are random numbers uniformly distributed in [0, 1]. w is the inertial weight. c_1 and c_2 represent the relative importance between cognition and social influence. After updating, particles can fly to better positions and the best position of the swarm is the near optimal solution found by the algorithm.

PSO was designed for continuous optimization problems, but most practical optimization problems are defined on discrete space. To utilize PSO in solving discrete optimization problems, the updating rules of velocities and positions need to be modified. Chen *et al.* proposed set-based PSO (S-PSO) [29] which uses crisp sets to represent the discrete space. Given the universal set $\boldsymbol{E}$ of a specific problem, the position of the ith particle $\boldsymbol{X}_i$ is defined as the subset of $\boldsymbol{E}$, $\boldsymbol{X}_i \subset \boldsymbol{E}$. The velocity of the ith particle $\boldsymbol{V}_i$ is defined as a set with possibilities, given by

$$\boldsymbol{V}_i = \{e \backslash p(e) | e \in \boldsymbol{E}\} \tag{8}$$

where $p(e) \in [0,1]$ is the possibility of selecting e.

The velocity updating rule and position updating rule in S-PSO are redefined on crisp sets and sets with possibilities. In velocity updating, the promising elements of the best solutions can be found through the subtraction operator. The promising elements are assigned with high possibilities in updated velocities. In this way, these promising elements are more probable to be reused in future generations. After velocity updating, particles in S-PSO update their positions to find optimal solutions. The position updating in S-PSO is step-by-step and problem-related heuristic information can be combined easily in position updating. The experimental results in [29] verify that S-PSO performs well in solving discrete combinatorial optimization problems.

B. Metaheuristic Approaches for VNE

Metaheuristic approaches are a class of algorithms inspired by natural phenomena, such as the genetic algorithm (GA) [30], ant colony optimization (ACO) [31] and PSO [32]. Metaheuristic approaches utilize iterative optimization to find near optimal solutions, which are widely applied in solving intractable optimization problems [33], such as multiobjective optimization [34] and bearing fault detection [35].

Recently, researchers have proposed several metaheuristics to solve VNE problems [13], [14], [16] which can achieve promising performance. Zhang *et al.* [13] applied PSO to solve VNE and proposed UEPSO. In UEPSO, the operators of position updating and velocity updating are redefined on the discrete space and thus particles in UEPSO can fly to better positions guided by the best solution found so far. Based on the operators of UEPSO, Cheng *et al.* [17] proposed the PSO with Markov RW (RWPSO). RWPSO devises a heuristic indicator, node rank, which can reflect the topology attributes of networks to evaluate the importance of substrate and virtual nodes. Although the updating operators in RWPSO and UEPSO are similar, RWPSO can achieve better performance since network structures are considered in the optimization. Su *et al.* [14] devised a new energy-aware model for VNE (EA-VNE) to study the energy consumption when maintaining a VNE system. To solve the EA-VNE problem, they proposed an efficient heuristic approach and a PSO-based approach (EA-VNE-PSO). The experimental results in [14] show that EA-VNE-PSO can save more energy than heuristic approaches.

Fajjari *et al.* [15] applied ant colony metaheuristics to solve VNE (VNE-AC). VNE-AC first divides a VN into a set of solution components in which virtual access nodes are removed. Then the artificial ant colony is released to find the mapping of solution components step by step. At the end of each iteration, the pheromone trail is evaporated for all solution components and is enhanced by the best solution. Zhu and Wang [16] proposed a modified ACO for VNE based on graph decomposition (ACO-TD). The topology of a VN is decomposed into a combination of ring and tree structures. The ring structures are mapped first before the tree structures. Different solution constructing rules and heuristic information are devised for ring and tree structures. Due to the decomposition mechanism, ACO-TD can handle more complicated VNs than previous ACO approaches. Chang *et al.* [36] used GAs to solve VNE and proposed two GA-based approaches, CB-GA and RW-GA. Both approaches use chromosomes to represent candidate solutions to VNE. The one-point crossover and mutation are applied to each chromosome with a predefined probability. The difference of CB-GA from RW-GA is that CB-GA only uses network resources to evaluate substrate nodes while RW-GA considers the topology similarity of substrate nodes.

In general, compared to heuristic VNE algorithms, metaheuristic approaches are more probable to escape from local optima due to the learning mechanism [1]. Hence, the quality of solutions obtained by metaheuristics can be improved.

C. Distributed Approaches for VNE

Contrary to centralized approaches that are studied intensively, there is a lack of distributed solutions to VNE problems currently [1].

ADVNE [20], [21], proposed by Houidi *et al.* in 2008, is the first fully distributed VNE approach. In ADVNE, each VN is subdivided into several hub-and-spoke clusters and each cluster is assigned to a substrate node in the SN to execute the embedding task. Based on multi agents, these clusters are embedded on SNs in a distributed approach. However, since each substrate node in ADVNE is autonomic, the message overhead among substrate nodes is unavoidable. The message overhead will increase rapidly when the size of SN increases, which might decrease the efficiency and scalability of ADVNE in large scale situations. In addition, the quality of embedding (e.g., embedding costs), and the constraints of SNs (e.g., CPU and bandwidth constraints) are neglected in ADVNE. As a result, the practicality of ADVNE is limited in real life applications. Similarly, co-operative VNE [37] utilizes autonomic and independent substrate nodes to embed VNs in a distributed way. They coordinate centralized and distributed algorithms to achieve successful VNE process.

In order to reduce the message overhead and improve the embedding quality, DPVNE was proposed by Beck *et al.* [22], [23] in 2013. First, the SN is partitioned into sub-SNs hierarchically and these sub SNs are stored as a binary tree. Then, some substrate nodes are selected to be delegation nodes, which determine the entrance of external VN requests from customers. Delegation nodes receive VN requests and assign

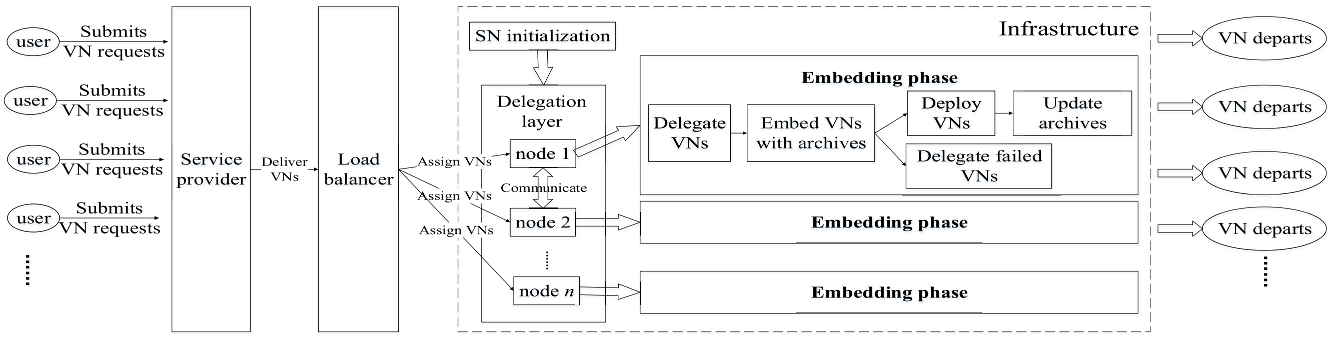


Fig. 2. HA-VNE-PSO system.

them to sub-SNs based on heuristic information. By integrating with a heuristic algorithm, DPVNE attempts to embed VNs within the assigned sub-SNs. The embedding procedure is executed by the embedder node in each sub-SN. In this way, multiple VN requests are allowed to demand substrate resources simultaneously since several delegation nodes are selected. Furthermore, Hahn *et al.* [38] studied DPVNE in the real multicore environment and each host application runs on a CPU core, which makes the simulation more practical.

Compared to ADVNE [20], DPVNE reduces the message overhead and improves the embedding quality. Consequently, DPVNE is more promising in real distributed systems [39], [40]. Nevertheless, DPVNE uses heuristics, such as ASID [18] and RW-MaxMatch [17], as the embedding algorithm. Heuristic VNE approaches are only suitable in small scale applications since it might easily get stuck in local optima when the SN scale increases. Besides, the hierarchical partition of the SN makes DPVNE search for solutions in the local area and hence heuristic algorithms are more inappropriate in the distributed VNE approaches.

Up to now, most studies of distributed VNE problems only consider the combination of heuristics. Since metaheuristic approaches of VNE have shown promising performance, it is reasonable to combine metaheuristics in distributed VNE problems. In this paper, we combine the distributed VNE system with the proposed SPSO-VNE to enhance the performance of distributed approaches, which will be elaborated in the next section.

IV. Proposed HA-VNE-PSO System

In this paper, we propose the HA-VNE-PSO system to solve distributed VNE problems. Compared to existing works, HA-VNE-PSO has the following two features.

1) HA-VNE-PSO combines metaheuristics rather than heuristics as the optimizer. The metaheuristics used in HA-VNE-PSO is the SPSO-VNE, which is specially devised for discrete VNE problems.
2) HA-VNE-PSO introduces archives to store historical information and uses archives to embed repeated VNs, which can further improve the performance of distributed VNE systems. In this section, we first present the basic procedure of HA-VNE-PSO and then each functional module of the system is introduced.

A. Basic Procedure of HA-VNE-PSO System

The basic procedure of HA-VNE-PSO system is depicted in Fig. 2. In the beginning, the SN is partitioned into several sub-SNs and some substrate nodes are assigned as delegation nodes forming the delegation layer. Each delegation node manages the resources of its sub-SNs and it is responsible for delegating VN requests to sub-SNs. The SN partition and the selection of delegation nodes are carried out in the initialization phase.

At first, customers ask for network services from the SP and submit VN requests to it. The collected VNs in the SP will be delivered to delegation nodes through the load balancer. The load balancer checks the state of all delegation nodes to identify whether they are free and available, and assigns VNs to the free delegation nodes. If a delegation node is busy for handling a VN, it cannot handle another one at the same time. Then, delegation nodes delegate VNs to the sub-SNs and these VNs will be embedded on the sub-SNs. In HA-VNE-PSO, we incorporate metaheuristics and introduce the archive-recording mechanism to optimize the embedding. If there are enough resources within the sub-SN, the VN will be deployed on it and corresponding resources will be allocated to the VN. Meanwhile, archives will be updated to record the up-to-date embedding results. Otherwise, the VN will be delegated to other sub-SNs which might be managed by other delegation nodes. Therefore, delegation nodes need to communicate with each other to deliver VNs. Finally, embedded VNs will depart from the system and release the allocated resources when they are complete.

As multiple delegation nodes are selected from the SN, multiple VNs are allowed to be delegated to sub-SNs simultaneously. In this way, VNs can be embedded on different sub-SNs at the same time so that HA-VNE-PSO is able to handle VN requests in a distributed approach.

B. Initialization Phase

Before embedding the first VN, the HA-VNE-PSO system needs to be initialized and this only happens once. The initialization includes the partition of the SN and the assignment of delegation nodes. We adopt the initialization procedure for distributed VNE systems proposed in [22].

The SN partition is a recursive process. First, one SN is divided into two parts. The size of each part is about half of the initial SN size and the substrate nodes in each part

Algorithm 1 Delegating

```
input: The VN request $v$ to be delegated, delegation node $n$, list subs
       of all sub SNs managed by $n$.
1:  If this type of $v$ has been recorded in archive arc
2:     Delegate $v$ to the same sub SN sn in arc;
3:     If vn can be embedded on sn
4:        Return;
5:     End If
6:  Else
7:     For each sn in subs
8:        If $\pi(sn, v) > 1.0$ (Eq.(9)) && size of sn >= size of $v$
9:           Add sn to candidate list candi;
10:       End If
11:    End For
12:    Sort candi ordered by $\pi$ (ascending);
13:    For each sn in candi
14:       Delegate $v$ to sn;
15:       If $v$ can be embedded on sn
16:          Return;
17:       End if
18:    End For
18: End If
```

are highly interconnected, which is similar to the network clustering process. Then, in each part, the partition process is executed again and two smaller sub parts are obtained from each part. The partition process is repeated until the size of sub-SNs is small enough and thus the hierarchical partition is completed. Fig. 3 depicts an example of hierarchical partition. There are eight substrate nodes in the SN and the SN is recursively partitioned twice. The partition algorithm used in hierarchical partition is the multilevel recursive bisection partitioning algorithm, proposed by Karypis and Kumar in 1998 [41], which can divide a well-connected network into nonoverlapping partitions.

The assignment of delegation nodes is based on the hierarchical partition. In each sub-SN, a substrate node is marked as an embedder node that is able to run the specified embedding algorithm. Some embedder nodes that are located on the same layer are assigned as delegation nodes. For example in Fig. 3, node 1 and node 6 can be assigned as delegation nodes at the same time while node 1 and node 2 cannot. Each delegation node manages sub-SNs within its range. For example in Fig. 3, if node 1 is assigned as delegation node, it only manages sub-SNs (b), (d), and (e). In this way, different delegation nodes can delegate VNs to mutually exclusive sub-SNs simultaneously so that the consistency and the concurrency of the SN are maintained.

C. Embedding Phase

The embedding phase is carried out after initialization. As repeated VN requests are identical in topologies and resource requirements, in the embedding phase, we mark repeated VNs with the same type and introduce archives to record the embedding information of them. Each archive records one type of VNs. The content of archives might be problem-dependent. In the context of our network model, the information stored in each archive contains the type of the VN, the occurrence frequency of the VN and the results of VNoM

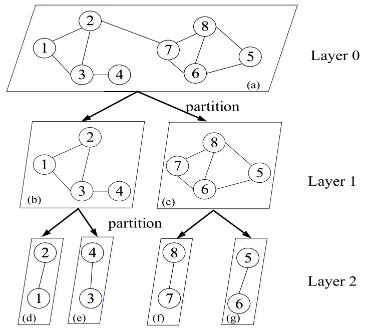


Fig. 3. Hierarchical partition of an SN with eight nodes.

Algorithm 2 HA Strategy

```
input: the archive arc, the VN $G_v$, the parameter $p$.
1:  For each individual ind in population pop
2:     Generate a random number $r_1$ in [0,1];
3:     If $r_1 > 0.5$ // about half of the population use archives
4:        Initial ind randomly;
5:     Else // use archives
6:        Construct virtual node mapping node_map = {};
7:        Sort the virtual nodes according to NR values defined
          in (10)
8:        For each virtual node $v$ in $G_v$
9:           Generate a random number $r_2$ in [0,1];
10:          Select substrate node $s$ = arc_map($v$) from arc;
11:          If $r_2 < p$ && is_available(v, s)
12:             Add mapping ($v$, $s$) to node_ map;
13:          Else
14:             Select the substrate node $s$ randomly;
15:             Add mapping ($v$, $s$) to node_map;
16:          End If
17:       End For
18:       Construct virtual link mapping link_map from node_map;
19:       Add link_map and node_map to ind;
20:    End If
21: End For
output: initialized population pop
```

and VLiM. The information in archives will be utilized in following procedures.

1) Delegating VNs: Delegation nodes delegate VNs to the appropriate sub-SNs. There are two issues that should be considered in delegating. First, as a VN is mapped on the sub-SN, the embedding results of the VN (including VNoM and VLiM) are related to that sub-SN. In other words, the embedding results of a VN only make sense under a specified sub-SN. For example in Fig. 3, the embedding results of a VN on sub-SN (b) cannot be applied in sub-SN (c) for they have different topologies. Therefore, to utilize the historical embedding information in archives, VNs need to be delegated to the sub-SNs that are the same as those in the archives. Second, as sub-SNs have different residual resources (e.g., the number of substrate nodes and residual bandwidth), VNs should be delegated to the sub-SNs that have enough resources. For example, a VN with four virtual nodes should be delegated to the sub-SNs with no less than four substrate nodes.

Based on the above analysis, we devise the delegating procedure for HA-VNE-PSO presented in Algorithm 1. When a coming VN v needs to be delegated to a sub-SN, we first

Algorithm 3 Updating Archives

input: The archive list: *arc_list*; the new embedded VN *vn*; the maximum size of archive list: *max_arc*; the minimum recording size of the VN: *min_VN*.

```
1:  For each archive arc in arc_list
2:    If the type of arc == the type of vn
3:      Replace arc with vn;
4:      Return arc_list;
5:    End If
6:  End For
7:  If the size of vn >= min_VN && the size of arc_list < max_arc
8:      Add vn to arc_list;
9:      Return arc_list;
10: End If
11: If the size of arc_list == max_arc
12:   u = the VN in arc_list with the lowest frequency;
13:   If the frequency of u < the frequency of vn
14:     Delete u from arc_list;
15:     Add vn to arc_list;
16:     Return arc_list;
17:   End If
18: End If
```

output: the updated *arc_list*

check whether the VN type of v has been recorded by archives or not (the archive-recording mechanism will be introduced later). The VN type is an attribute stored in archives by the system and we use the VN type as ID to justify whether two VNs are identical. If v has been recorded by archives before, it will be delegated to the sub-SN recorded in archives to utilize historical embedding information (lines 1 to 2 in Algorithm 1). Nevertheless, if the VN v is not recorded before or the historical sub-SN fails to host v, v will be delegated to other sub-SNs with enough resources managed by the delegation node. Inspired by the heuristic information proposed in [22], we evaluate the resource potential of each sub-SN G_s' relative to the VN G_v, formulated as

$$\pi\left(G_s', G_v\right) = \left(\sum_{n\in N_s'} c_s^n \Big/ \sum_{n\in N_v} c_v^n\right) \times \left(\sum_{l\in L_s'} b_s^l \Big/ w \times \sum_{l\in L_v} b_v^l\right) \tag{9}$$

where w is a parameter to control the amount of expected substrate bandwidth and is set to 10 according to [22]. A sub-SN G_s' with a larger potential π means it has more substrate resources relative to the VN G_v. If the potential of a sub-SN is larger than 1.0, then this sub-SN will be added to a candidate list. The candidate list is sorted in ascending order by the potential and the VN v will be delegated to sub-SNs in the list one by one until it is successfully embedded (lines 12–17 in Algorithm 1). In the delegating procedure, the archives are also sent to sub-SNs to utilize the historical embedding results, which will be introduced in the next section.

Based on the delegation mechanism, metaheuristics can be combined with the distributed VNE approach. As shown in Fig. 3, the SN is hierarchically partitioned into several sub-SNs and two delegation nodes are assigned (i.e., node 1 and node 6). As these two delegation nodes can embed VNs with metaheuristics simultaneously, the total time for embedding VNs is reduced and thus the efficiency of the metaheuristic VNE system can be improved overall. In addition, since the scale of sub-SNs is smaller than the whole SN, the searching space of embedding VNs is reduced, which can further improve the efficiency of VNE systems.

2) Embedding VNs With Archives: After delegating VNs, the embedder node in the sub-SN will run the embedding algorithm to embed the VN on the sub-SN. In HA-VNE-PSO, we combine the distributed VNE system with the SPSO-VNE and devise the HA strategy to improve the embedding quality of distributed approaches.

a) HA strategy: HA strategy is an initialization strategy for metaheuristics to use historical information stored in archives, which is presented in Algorithm 2. The initialization for VNE is to find the initial solutions including VNoM and VLiM. Most metaheuristics for VNE initialize the population randomly [13], [14], [17]. As repeated VNs have identical topologies and resource requirements, the historical embedding results can be reused in initialization to enhance the optimization.

There are two issues that need to be considered in initialization. The first one is the diversity of the population. If all individuals use the same archive to initial themselves, individuals might be similar to each other. Thus, the diversity of the population might be lost, which makes the population trap in local optima easily and converge prematurely. To avoid this, in the HA strategy, only about half of the population is initialized with archives and the rest uses random initialization (lines 3 and 4 in Algorithm 2). Besides, we introduce the parameter $p\in[0, 1]$ to control the use of embedding results in archives. $p = 1$ means all node mappings in the archive will be reused and $p = 0$ means the opposite. In this way, even though the individuals are initialized with the same archive, the initialization for each individual is different. The second issue is the change of SNs. As old VN requests depart from SNs and new VN requests arrive, the resources of SNs change frequently. Due to this fact, the embedding results in archives cannot be utilized directly. Therefore, every time the history results are reused, the availability of the reused elements should be checked.

In HA strategy, for the individuals that are initialized with archives, we first find the VNoM for each virtual node. The resources demanded by virtual nodes are various and the virtual nodes that demand more resources are usually more critical to the embedding quality. As a result, we embed these virtual nodes in priority. The resources demanded (or offered) by a virtual (or substrate) node are evaluated by the NR metric [13], defined as

$$\mathrm{NR}(n) = \begin{cases} c_v^n \times \sum_{l\in L(n)} b_v^l, & \text{if } n \in N_v \\ c_s^n \times \sum_{l\in L(n)} b_s^l, & \text{if } n \in N_s \end{cases} \tag{10}$$

where $L(n)$ is the set of adjacent links of a node n. The virtual nodes are sorted in descending order according to NR values so that the virtual nodes with large NR values, which means they demand more substrate resources, will be mapped first. We use function *arc_map*(.): $N_v \rightarrow N_s$ to represent the VNoM results stored in archives and the function *is_available*(.): $N_v, N_s \rightarrow$ *Boolean* is used to represent the availability of the

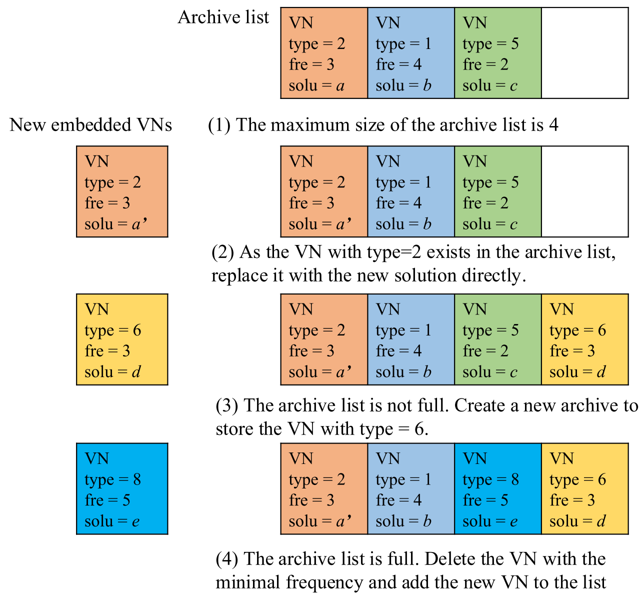

Archive list
VN type = 2 fre = 3 solu = a
VN type = 1 fre = 4 solu = b
VN type = 5 fre = 2 solu = c
New embedded VNs
VN type = 2 fre = 3 solu = a'
(1) The maximum size of the archive list is 4
VN type = 2 fre = 3 solu = a'
VN type = 1 fre = 4 solu = b
VN type = 5 fre = 2 solu = c
(2) As the VN with type=2 exists in the archive list, replace it with the new solution directly.
VN type = 6 fre = 3 solu = d
VN type = 2 fre = 3 solu = a'
VN type = 1 fre = 4 solu = b
VN type = 5 fre = 2 solu = c
VN type = 6 fre = 3 solu = d
(3) The archive list is not full. Create a new archive to store the VN with type = 6.
VN type = 8 fre = 5 solu = e
VN type = 2 fre = 3 solu = a'
VN type = 1 fre = 4 solu = b
VN type = 8 fre = 5 solu = e
VN type = 6 fre = 3 solu = d
(4) The archive list is full. Delete the VN with the minimal frequency and add the new VN to the list

Fig. 4. Examples of updating archive lists.

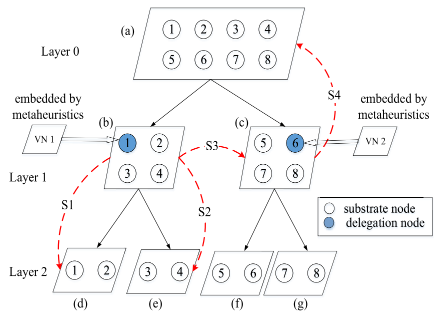


Fig. 5. Delegation procedure of VNs. Specially, the red curves S1–S4 represent the delegation of unsuccessfully embedded VN1. VN1 is embedded on sub-SN (d), (e), (b), (c), and (a) successively.

substrate nodes. If the substrate node s is already used by other virtual nodes in the same VN [constraint (2)] or the residual CPU of substrate node s is not enough to host virtual node v [constraint (3)], the function *is_available*(v, s) will return false. Otherwise, it will return true. Then, for embedding each virtual node v, if the substrate node s is available for v and the generated random number is smaller than parameter p, v will be mapped to s (lines 10–12 in Algorithm 2).

If individuals are not initialized with archives (to keep the diversity of the population) or the historical embedding results in archives are unavailable, the random initialization is utilized (lines 4 and 14 in Algorithm 2). In the random initialization, the substrate node to host the virtual node v is selected from the candidate set $\boldsymbol{S}$ whose elements are the substrate nodes which satisfy the constraints in (2) and (3). The selection probability of each substrate node i in $\boldsymbol{S}$ is defined as

$$\text{probability}_i = \text{NR}(i) \Big/ \sum_{j\in \boldsymbol{S}} \text{NR}(j). \tag{11}$$

Finally, we select a substrate node in the candidate set $\boldsymbol{S}$ by roulette wheel selection.

A complete solution to the VNE problem includes the node mappings and link mappings. After all virtual nodes in the VN request are mapped based on HA strategy, the VLiM can be constructed by several methods, such as K-shortest path [8]. Actually, some metaheuristic VNE approaches only utilize the VNoM to represent their population [13], [17], so that the operation of adding VLiM to individuals can be omitted (lines 18 and 19 in Algorithm 2). In this way, the HA strategy can also be used in other population-based metaheuristics for VNE.

b) SPSO-based optimizer for VNE: With the HA strategy, the initial population (solutions) is obtained. After initialization, the integrated metaheuristics will iteratively search for optimal solutions within the sub-SN. In this paper, we propose SPSO-VNE as the optimizer, which is based on the set-based PSO framework.

In SPSO-VNE, we use set notation to represent the VNoM and VLiM. For the kth particle, the VNoM is denoted as a crisp set $\boldsymbol{X}_k = \{(u,\ v)|u \in N_v,\ v \in N_s\}$ and the VLiM is denoted as $\boldsymbol{Y}_k = \{(l,\ p)|l \in L_v,\ p \in \boldsymbol{P}_s\}$ where $\boldsymbol{P}_s$ is the set of loops-free paths in the SN. Since $\boldsymbol{Y}_k$ can be obtained from $\boldsymbol{X}_k$ with existing algorithms [8], [13], $\boldsymbol{X}_k$ is used to represent the position of the kth particle. The velocity for the kth particle $\boldsymbol{V}_k$ is defined as a set with possibilities, $\boldsymbol{V}_k = \{(u, v)\backslash\gamma|u \in N_v,\ v \in N_s, \gamma \in [0, 1]\}$. γ implies the potential of VNoM elements. A large γ means the VNoM element is promising and is more probable to be selected in the new solutions.

In velocity updating, we update velocities to find the flying direction of particles. For the kth particle, we first find the difference between the current position $\boldsymbol{X}_k$ and the best solution found by the population **gbest** (or itself **pbest**$_k$)

$$\boldsymbol{E} = \boldsymbol{gbest} - \boldsymbol{X}_k = \{(u, v)|(u, v) \in \boldsymbol{gbest} \text{ and } (u, v) \notin \boldsymbol{X}_k\}. \tag{12}$$

Then the elements in $\boldsymbol{E}$ are associated with the probability γ

$$c \times \boldsymbol{E} = \{(u, v)\backslash\gamma|(u, v) \in \boldsymbol{E}, \gamma \in (0, 1)\} \tag{13}$$

where the parameter c plays the similar role of parameter $c_1 r_1$ in (6). The inertia of velocities is also maintained. Given the inertia weight $w \in (0, 1)$, we have

$$w \times \boldsymbol{V}_k = \{(u, v)\backslash\gamma \cdot w|(u, v)\backslash\gamma \in \boldsymbol{V}_k\}. \tag{14}$$

Finally, we compare the probability for the same VNoM elements and larger ones will be kept

$$w\boldsymbol{V}_k + c\boldsymbol{E} = \{(u, v)\backslash \max(\gamma_1, \gamma_2)| \\ (u, v)\backslash\gamma_1 \in w\boldsymbol{V}_k, (u, v)\backslash\gamma_2 \in c\boldsymbol{E}\}. \tag{15}$$

In position updating, the promising elements in updated velocities will be selected to construct new solutions. For each virtual node u, the substrate node v is selected in three steps. First, u is selected from the candidate set $\boldsymbol{S}$, where the probability of the elements in $\boldsymbol{S}$ are larger than the randomly generated threshold $\alpha \in (0, 1)$

$$\boldsymbol{S} = \{v \in \boldsymbol{N}_s|\gamma \geq \alpha, (u, v)\backslash\gamma \in \boldsymbol{V}_k\}. \tag{16}$$

This is because the VNoM elements with small values are not worthy of learning. Then, if no substrate nodes can satisfy the above requirement, the substrate node in the previous position is reused. Finally, if the reused substrate node is still not available, u is randomly selected from the SN.

The optimization of SPSO-VNE is based on the population initialized with archives. Hence, historical information is utilized during the optimization. If the best solution found by metaheuristics is valid, VNs will be deployed on the sub-SN. Otherwise, VNs will be delegated to other sub-SNs.

3) Updating Archives: Every time a VN is embedded successfully on a sub-SN, the archives will be updated to record up-to-date embedding results. Each archive records the embedding results of one type of VNs and these archives are organized as lists. There are several archive lists in the system and they are maintained by two kinds of nodes, the delegation nodes and the embedder nodes above the layer of delegation nodes. The embedder nodes below the delegation nodes are not necessary to maintain archive lists for they can get archives from delegation nodes.

In reality, the type of VNs is numerous, so that recording all kinds of VNs in the archives is not practical. Accordingly, we only record a portion of VNs, which means the size of archive lists is limited. Two rules are devised to record valuable VNs in archive lists. First, the frequency of different VN types might be various. Some VN requests are demanded frequently and some of them are not. As a result, recording the VNs that are demanded more frequently is reasonable. Second, the difficulty of embedding is related to the size of VNs. It is usually difficult to embed VNs with large size. Hence, recording the VNs with the large size is also reasonable.

According to the above analysis, we introduce two parameters to control the archive-recording process, the maximum size of an archives list, *max_arc*, and the minimum size of recorded VNs, *min_VN*. Actually, these two parameters are independent of the algorithm and they are only related to problems. For example, in real VNE systems, the information of all embedded VNs might be stored in a database or a cloud server. We can refer to them to obtain previous information of services, such as the scale, frequency and the type of VNs. Then the parameter *max_arc* and *min_VN* can be evaluated manually.

We only record the VNs that are embedded successfully. To facilitate understanding, we use examples to illustrate the archive updating procedure, as shown in Fig. 4 where each box is an archive entry related to a VN, "type" indicates the VN's type, "fre" represents the VN's occurrence frequency, and "solu" represents the solution to the embedding of VNs. During updating archives, if this type of the VN has been already recorded in the archive list before, the information stored in the archive list is replaced by the up-to-date results directly [e.g., Fig. 4(2)]. Otherwise, the size of the VN and archive list needs to be checked. If the size of the VN is not smaller than *min_VN* and the current size of the archive list is smaller than *max_arc*, the new embedded VN will be added to the archives list [Fig. 4(3)]. If the current size of the archive list is equal to *max_arc*, then we compare the lowest frequency of the VN from the archive list and the frequency of the new VN, and select the larger one into the archive list [Fig. 4(4)],

TABLE I
COMPARED APPROACHES IN THE EXPERIMENTAL STUDIES

Name	Description
DPVNE-RWMM	DPVNE is combined with heuristics RW-MaxMatch.
Centralized PSO	The centralized set-based PSO for VNE (SPSO-VNE), which is proposed in this paper.
DPVNE-PSO	DPVNE is combined with SPSO-VNE.
HA-VNE-PSO	The distributed VNE system with historical archives and SPSO-VNE.
DPVNE-ACO	DPVNE is combined with CB-ACO.
HA-VNE-ACO	The distributed VNE system with historical archives and CB-ACO.
DPVNE-GA	DPVNE is combined with CB-GA.
HA-VNE-GA	The distributed VNE system with historical archives and CB-GA.

TABLE II
EXPERIMENTS DESIGN ON MODERATE SCALE SNs

scenario	type of VN	type of SN	arriving rate	avg No. VN
VN40SN200	VN2-40	SN200	0.05	2000
VN20SN200	VN2-20	SN200	0.1	4000

The selection strategy can be formulated as

$$\max\{vn.\text{frequency}, u.\text{frequency}\},\ u = \underset{i \in \text{arc\_list}}{\arg\min}\{i.\text{frequency}\} \tag{17}$$

where vn.frequency is the frequency of the new embedded VN vn and *arc_list* is the archive list. The complete updating procedure is presented in Algorithm 3.

4) Delegating Unsuccessfully Embedded VNs: The successfully embedded VNs will be recorded in the archives while the unsuccessfully embedded VNs will be delegated to other sub-SNs. At first, the failed VN is delegated to the sub-SNs that satisfy the potential requirement (9) within the delegation node (line 13 in Algorithm 1, curves S1 and S2 in Fig. 5). If all of these sub-SNs cannot host the failed VN, the VN will be delivered to other delegation nodes (curve S3 in Fig. 5) in the same layer, namely delegation layer. If the VN still cannot be embedded within other delegation nodes, it will be repeatedly delegated to the sub-SNs in the upper layer(s) of the delegation layer until the VN is successfully embedded (curve S4 in Fig. 5). Finally, if the VN cannot be embedded on the largest sub-SN (i.e., the whole SN), it will be discarded by the system.

During the delegating, if a failed VN from the delegation node A is delivered to the delegation node B, the delegation node A will be locked and cannot handle coming VNs until the failed VN is successfully embedded or rejected. This mechanism can avoid inconsistencies of the system [22]. In the limiting case, all delegation nodes cannot embed the failed VN on sub-SNs and the failed VN will be handled by the whole SN. In this case, all delegation nodes are locked and thus HA-VNE-PSO behaves like centralized approaches. Therefore, delegating failed VNs influences the efficiency of HA-VNE-PSO, which will be studied in experiments later.

V. Experimental Studies

In this section, to verify the effectiveness of the proposed approach, we test the HA-VNE-PSO system in various scenarios and compare it with other approaches. First, we present the experimental environment and parameter settings of all compared approaches. Second, we introduce several metrics to measure the embedding quality. Then, to investigate the performance between distributed approaches and the centralized algorithm, we conduct experiments on moderate scale SNs. As one of the advantages of distributed approaches is the scalability, we analyze the scalability of distributed VNE approaches with metaheuristics in large scale SN topologies. In addition, we compare the message overhead and the load balance of distributed approaches. Besides, to be more realistic, we conduct experiments under real transit-stub topologies and consider the location constraints of substrate and virtual nodes. Finally, we study the sensitivity of the parameter p in HA-VNE-PSO.

A. Evaluation Settings

Two kinds of SN scale, moderate scale (SNs with 200 nodes) and large scale (SNs with 300 to 800 nodes), are investigated [22]. The average connectivity rate is fixed at 10% for SNs and 50% for VNs [13]. For fair comparisons, the network topologies of SNs and VNs are generated by the GT-ITM tool [42] similar to previous work [9]. The range of bandwidth and CPU is distributed uniformly from 1 to 50 for VNs and from 50 to 100 for SNs [17]. To be more realistic, the lifetime of each VN request follows the exponential distribution with an average of 500 time units [17]. Two kinds of VN series, the number of virtual nodes ranging from 2 to 20 (denoted as VN2-20) [13] and ranging from 2 to 40 (denoted as VN2-40), are studied. The number of virtual nodes in each VN follows a uniform distribution within its range as well. The maximum number of iterations is set to 100 for VN2-20 and is set to 150 for VN2-40 to ensure the convergence of metaheuristics.

There are three parameters in HA-VNE-PSO, parameter *min_VN* and *max_arc* in Algorithm 3 and parameter p in Algorithms 2. The parameters *min_VN* and *max_arc* are dependent on problems. *min_VN* represents the minimum size of VNs that can be recorded in archives. Since recording large VNs is more meaningful, *min_VN* is set to the average level of VN series. Hence, *min_VN* is set to 10 for VN2-20 and is set to 20 for VN2-40. *max_arc* represents the size of archive lists maintained by the system. It might be related to the substrate equipment and service scale. For easy comparisons, *max_arc* is set to 10 which means at most ten kinds of VNs are recorded in archives. The parameter p is set to 0.8 and we will discuss its sensitivity later. We select the layer 2 as delegation level, which means HA-VNE-PSO and DPVNE can embed at most 4 VNs at the same time [23].

HA-VNE-PSO is compared with DPVNE-RWMM [22] (DPVNE is combined with the heuristic VNE algorithm RW-MaxMatch), DPVNE-PSO (DPVNE is combined with the metaheuristic SPSO-VNE devised in this paper) and the centralized algorithm with the S-PSO based optimizer. In addition, we combine our distributed system with CB-GA and CB-ACO [36] to investigate different metaheuristics in solving distributed VNE problems. All compared approaches are concluded in Table I. All algorithms are implemented in Java and the programs are performed on a machine with Intel Core i3-4310 CPU at 3.40 GHz. The operating system is Linux and the JDK version is 1.8.

B. Metrics for Embedding Quality

The objective of embedding a single VN is to reduce the costs of embedding (5). However, the objective of embedding multiple VNs is more complicated, especially in online VNE systems.

First, an excellent VNE system needs to earn more revenue for SPs and InPs. The revenue mainly comes from the demanded resources (CPU and bandwidth) in VNs. Given a VN $G_v = (N_v, L_v, R_v^n, R_v^l)$, similar to previous works [10], [13], the revenue obtained from G_v at time t is evaluated as the sum of demanded CPU and bandwidth resources, formulated as

$$R(G_v, t) = \sum_{n \in N_v} c_v^n + \sum_{l \in L_v} b_v^l. \tag{18}$$

From the long-term view, the long-term average revenue is defined as

$$avgR = \lim_{T \to \infty} \sum_{t=0}^{T} R(G_v, t)/T. \tag{19}$$

avgR reflects the average revenues obtained by all accepted VNs and the larger the *avgR*, the better the VNE system.

The costs for embedding same VNs are various and they also influence the quality of embedding. The costs mainly come from the use of substrate resources of SNs. Given the VN $G_v = (N_v, L_v, R_v^n, R_v^l)$ and the SN $G_s = (N_s, L_s, A_s^n, A_s^l)$, the cost of embedding the VN G_v at time t is defined as the sum of assigned CPU and bandwidth resources in the SN, formulated as

$$C(G_v, t) = \sum_{n \in N_v} c_v^n + \sum_{l_{uv} \in L_v} \sum_{l_{ij} \in L_s} f_{ij}^{uv} \times b_v^{l_{uv}}. \tag{20}$$

Similar to the long-term average revenue, the long-term R/C ratio is defined as the ratio of the revenue to cost, formulated as

$$R/C = \lim_{T \to \infty} \left(\sum_{t=0}^{T} R(G_v, t) \Big/ \sum_{t=0}^{T} C(G_v, t) \right). \tag{21}$$

The range of R/C is (0, 1] and generally, a higher R/C ratio means that less substrate resources are allocated to VNs. Theoretically, a value of 1.0 would be optimal, indicating that the assigned resources in SNs equal the demanded resources by VNs.

The third metric is the acceptance ratio of VNs, defined as the ratio of accepted VNs to total VNs, formulated as

$$\lim_{T \to \infty} \left(\sum_{t=0}^{T} \mathrm{VNR}_a \Big/ \sum_{t=0}^{T} \mathrm{VNR} \right) \tag{22}$$

where VNR is the number of total VNs from users and VNR_a is the number of accepted VNs. The range of acceptance ratio

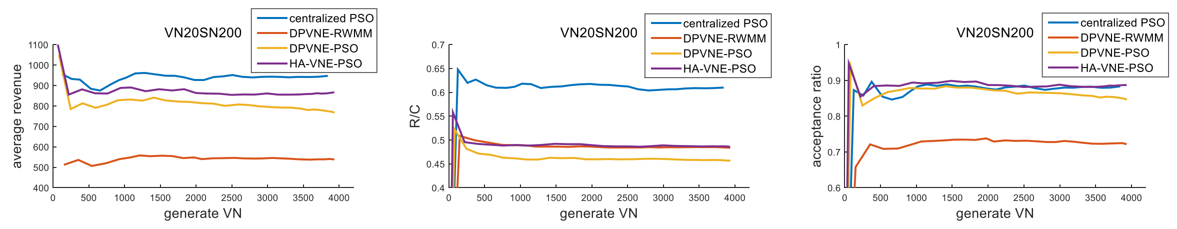


Fig. 6. Comparison results of DPVNE-RWMM, DPVNE-PSO, HA-VNE-PSO, and the centralized SPSO-VNE on moderate SNs.

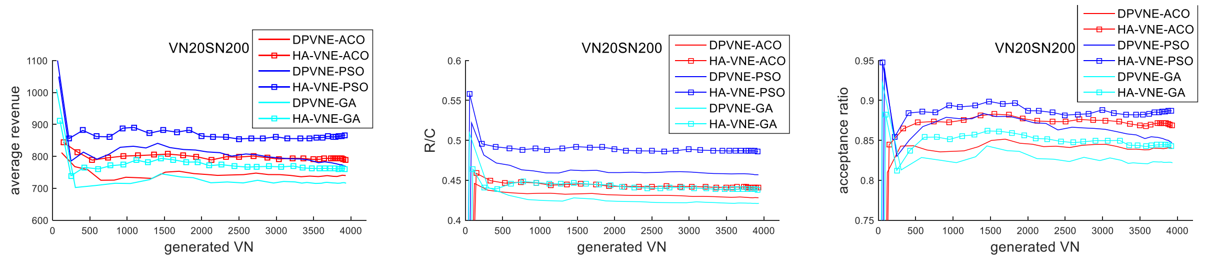


Fig. 7. Combination of different metaheuristics in distributed VNE problems.

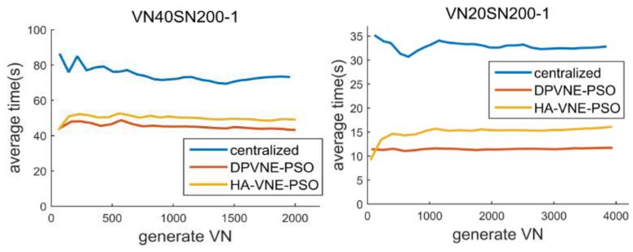


Fig. 8. Average running time for embedding single VN by DPVNE-PSO, HA-VNE-PSO, and centralized algorithms on moderate SNs.

is [0, 1]. Theoretically, a value of 1.0 would be optimal, indicating that all VNs can be accepted by the VNE system. Obviously, a larger acceptance ratio represents better quality of service.

C. Experiments on Moderate Scale SNs

We design two kinds of scenarios in this section concluded in Table II, where the column "scenario" is the combination of VN series and SNs. For example, "VN40SN200" represents the experiments on the VN series varying from 2 to 40 nodes and the size of SN is 200 nodes. In our experiments, it is assumed that there are total 40 different types of VNs uniformly distributed in each VN series. The column "arriving rate" is the probability of generating VNs. VN requests from customers arrive in a Poisson process with a specified arriving rate. The simulation runs for 40 000 time units and the average number of generated VN requests (the column "avg No. VN") can be evaluated. For example, with an arriving rate of 0.05, the average number of generated VN requests is about $40\,000 \times 0.05 = 2000$. Here, the time unit used in the simulation is one second.

Fig. 6 presents partial comparison results for online VN requests with regard to average revenue, R/C ratios and acceptance ratios. More experimental results are presented in the supplementary material. The abscissa of all figures is the number of generated VNs which are the same in all compared approaches. First, the centralized algorithm outperforms the distributed approaches (DPVNE-RWMM, DPVNE-PSO, and HA-VNE-PSO) in terms of average revenue and R/C ratio in all scenarios. The results are reasonable because the centralized algorithm is able to utilize the global information of SNs to optimize the embedding whereas the distributed approaches only utilize the local information of SNs. Hence, the centralized algorithm is more probable to get global optima. Fortunately, as long as the size of SNs is large enough, the quality of local optima in distributed methods will be improved. Hence, the gap between the distributed approaches and the centralized algorithm will narrow.

Compared to DPVNE-PSO, HA-VNE-PSO outperforms it in terms of the average revenue, R/C ratios and acceptance ratios in all scenarios. Owing to the proposed archive mechanism and HA strategy, the historical embedding results are utilized in future optimizations and thus the quality of embedding is improved, which is reflected in the higher R/C ratio of HA-VNE-PSO. As for acceptance ratios, HA-VNE-PSO is competitive to the centralized algorithm and is better than DPVNE-PSO in the scenarios VN20SN200. The combination of heuristics (i.e., DPVNE-RWMM) is the worst although its R/C ratios are not bad. The results of heuristics indicate that the combination of metaheuristics can improve the performance of distributed VNE systems.

We also compare other metaheuristics including ACO-based and GA-based approaches in distributed VNE problems, and the comparison results are shown in Fig. 7. First, the approaches with HAs can achieve better results in most instances. The results verify that the HA mechanism can improve the performance of different metaheuristics in distributed VNE problems. Second, the approaches with SPSO-VNE can outperform other GA-based and ACO-based approaches, especially in R/C ratios. The superiority of SPSO-VNE indicates that the redefined operators on crisp sets are

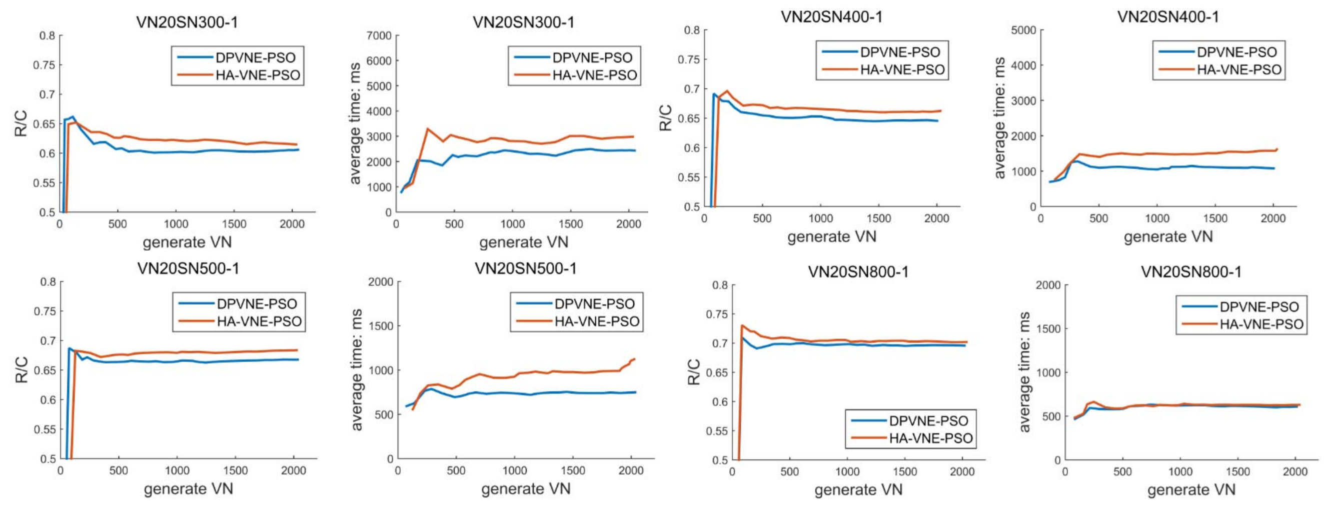


Fig. 9. Comparison results of DPVNE and HA-VNE-PSO on large scale SNs.

helpful for finding better solutions, even in the distributed environment.

To investigate the efficiency of the distributed approaches with metaheuristics, the running time for embedding single VN by the centralized S-PSO, DPVNE-PSO, and HA-VNE-PSO is recorded in Fig. 8. The average running time in distributed approaches is much faster than that in the centralized algorithm in all scenarios, which is the primary advantage of distributed approaches. The efficiency of distributed approaches comes from two sides. On the one hand, the hierarchical partitions for SNs reduce the searching space for embedding small VNs and thus reduce the computational complexity at the same time; on the other hand, the delegation mechanism allows multiple VNs to be embedded in the shared SN simultaneously so that the embedding tasks are executed parallel and concurrently. In average time, HA-VNE-PSO spends a little more running time than DPVNE-PSO. The extra running time might come from maintaining archives and utilizing historical results. Although HA-VNE-PSO consumes more running time, it is still much more efficient than the centralized algorithm. In addition, the gap between HA-VNE-PSO and DPVNE-PSO in efficiency will narrow when the SN scale increases, which will be shown in the next section.

Generally speaking, in moderate scale SNs, distributed approaches cannot show their superiority in terms of optimizing capability compared with the centralized algorithm. However, distributed approaches can show more efficiency than the centralized one. It is hard to say which approach is better in moderate scale SNs. If the price of infrastructure is really expensive, the centralized approach is a better choice due to the less use of resources. If customers emphasize the concurrency and efficiency of services, obviously, distributed approaches are preferred.

D. Experiments on Large Scale SNs

In order to investigate the scalability of distributed approaches with metaheuristics, we conduct experiments on large scale SNs. As the execution time of the centralized algorithm in large scale scenarios would easily exceed several days or even weeks [22], we only compare the distributed approaches, DPVNE-PSO and HA-VNE-PSO, in this part. The size of SNs varies from 300 nodes to 800 nodes and the VN series ranges from 2 to 20 nodes with the fixed arriving rate 0.05. Other settings of networks are the same as those in the former experiments.

Parts of the experimental results are depicted in Fig. 9. As the size of SNs increases, all generated VNs are accepted by large scale SNs, which indicates that the acceptance ratio is equal to 1.0 and the average revenue is same for DPVNE-PSO and HA-VNE-PSO. So we only compare the metric R/C ratio and the metric average time. From Fig. 9, it is important to note that HA-VNE-PSO extends well for large scale SNs as well as DPVNE-PSO and the optimizing capability of HA-VNE-PSO is still better than DPVNE-PSO, which is reflected in R/C ratio. These results validate that our proposed archive mechanism and HA strategy are still effective in large scale SNs. The improvement of R/C ratios is less apparent in large scale experiments. This is because that, as the size of SNs increases, more VNs can be embedded on small sub-SNs, which makes the embedding process easier. In this way, the room for improving optimization is decreased as well. For example, in the SN with 100 nodes, there are 5 partitions with 20 nodes while there are 40 such partitions in the SN with 800 nodes. Therefore, as long as the resources of sub-SNs are enough for VNs, more VNs will be embedded on these small sub-SNs instead of large ones. The average time for embedding a VN in HA-VNE-PSO is a little longer than DPVNE-PSO. Meanwhile, when the SN size increases, the difference of average time between two approaches narrows. For the SN size with more than 500 nodes, the average time of both approaches is similar or comparable.

Fig. 10 and Table III depict the trend of the final R/C ratios and running time for both approaches over the increase of SN size. We can observe that the R/C ratios of both are improved when the SN size increases. This phenomenon indicates that the influence of local optima in distributed approaches is reduced as long as the size of SNs is large

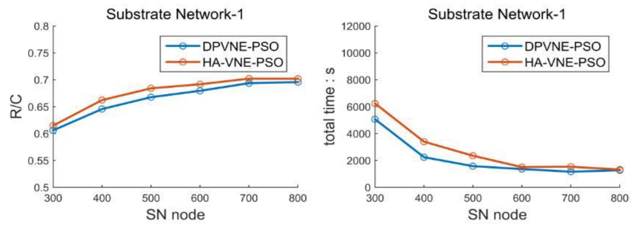


Fig. 10. Trend of final R/C ratio and total time.

enough. An interesting phenomenon shown in Fig. 10 and Table III is that, the running time of both approaches is decreased when the SN size increases. This result comes from two sides. First, larger SNs imply that more VNs can be embedded on small sub SNs which reduces the overall computational complexity. Second, each delegation node can manage more resources in large scale SNs and thus more VNs can be embedded within the scope of delegation nodes. Hence, the message overhead among delegation nodes is decreased so that the concurrency of the system is improved. Consequently, the efficiency of the distributed approaches is improved when the SN size increases.

E. Message Overhead and Load Balance

The embedder nodes in DPVNE-PSO and HA-VNE-PSO need to send messages to communicate with each other, including delegating VNs, deploying VNs, and so on. We compare the passing messages among delegation nodes. In large scale SNs, all VNs can be embedded within the scope of delegation nodes. Therefore, the message overhead among delegation nodes is zero. As a result, we only compare the message overhead for both distributed approaches in moderate scale SNs, presented in Table IV.

From Table IV, it can be seen that the message overhead of two approaches is similar. This is because they use the similar mechanism of delegation nodes. Actually, the HA strategy might reuse the same sub-SNs frequently. The overuse of same sub-SNs might increase the probability of unsuccessfully embedding, which causes a little more message overhead than DPVNE-PSO in some scenarios. This shortcoming will also be weakened when the SN size increases.

Another metric, load balance, is studied in our experiments. The load balance plays an important role in distributed systems. We devise the node load and link load to measure the load of the SN G_s. The node load at time t, $\mathrm{NL}(t)$, is formulated as

$$\mathrm{NL}(t) = \left(\sum_{n\in N_s} \mathrm{UN}(n)/|N_s|\right) / \max_{n\in N_s} \mathrm{UN}(n) \tag{23}$$

where the function $\mathrm{UN}(n)$ represents the utilization ratio of the substrate node n and $\|.\|$ represents the number of substrate nodes in G_s. The node load is defined as the ratio of the average utilization ratio of substrate nodes to the maximum utilization ratio among all substrate nodes. The range of $\mathrm{NL}(t)$ is (0, 1]. Theoretically, a value of 1.0 would be optimal which implies that the utilization ratio of all substrate nodes is the same and the load of the distributed system is completely balanced in nodes. From the long-term view, the average node load is defined as

$$\mathrm{avgNL} = \lim_{T\to\infty} \sum_{t=0}^{T} \mathrm{NL}(t)/T. \tag{24}$$

Similarly, the link load at time t, $\mathrm{LL}(t)$, is defined as

$$\mathrm{LL}(t) = \left(\sum_{l\in L_s} \mathrm{UL}(l)/|L_s|\right) / \max_{l\in L_s} \mathrm{UL}(l) \tag{25}$$

where the function $\mathrm{UL}(l)$ represents the utilization ratio of substrate link l and $\|.\|$ represents the number of substrate links in G_s. The range of $\mathrm{LL}(t)$ is also (0, 1] and theoretically, a value of 1.0 would be optimal as well. From the perspective of long term, the average link load is defined as

$$\mathrm{avgLL} = \lim_{T\to\infty} \sum_{t=0}^{T} \mathrm{LL}(t)/T. \tag{26}$$

The node load and link load in moderate scale SNs are presented in Figs. S13 and S14 in the supplementary materials. From the figures, we can see that the node load in HA-VNE-PSO is obviously better than that in DPVNE-PSO in moderate scenarios. The link load obtained by HA-VNE-PSO is competitive or slightly better than that obtained by DPVNE-PSO. In general, the load in HA-VNE-PSO is more balanced than that in DPVNE-PSO in moderate SNs which indicates HA-VNE-PSO is more reliable than DPVNE-PSO.

F. Simulations Under Location Constraints and Trans-Stub Topologies

To be more realistic, we consider the location constraints for virtual and substrate nodes. The network model is modified to include the location property. The location of the substrate node $n_s \in N_s$ and the virtual node $n_v \in N_v$ is represented by $\mathrm{loc}(n_s)$ and $\mathrm{loc}(n_v)$, respectively. During the node assignment, the location constraints should be satisfied, such that [43]

$$\mathrm{dis}(\mathrm{loc}(n_v), \mathrm{loc}(n_s)) \le D \tag{27}$$

where $\mathrm{dis}(i, j)$ measures the distance between the locations of two nodes i and j, and D is the maximum distance between two nodes. All virtual and substrate nodes are randomly located in the 100×100 grid [44]. Moreover, the topologies of SNs are generated by the real transit-stub networks [36], which are provided with 200 nodes and around 1000 links. The transit domains and nodes per transit are set to 4 and 2, respectively. The stubs per transit node and nodes per stub are set to 4 and 6, respectively. These topology parameters of SNs are similar to the work [36]. The number of virtual nodes ranges from 2 to 20 and the arriving rate is 0.1. Other attributes, such as CPU and bandwidth resources, are identical to the previous experiments. The experimental results are presented Fig. S17 in the supplementary materials.

The experimental results confirm that the combination with metaheuristics can indeed improve the performance of distributed VNE approaches (compare DPVNE-PSO and DPVNE-RWMM). Moreover, using historical information in

TABLE III
AVERAGE TIME FOR EMBEDDING ONE VN AND TOTAL R/C RATIOS ON LARGE SCALE SNs

	DPVNE-PSO	HA-VNE-PSO	DPVNE-PSO	HA-VNE-PSO
size of SNs	avg. time (ms)	avg. time (ms)	R/C	R/C
300	2453.75	3021.77	0.6081	0.6147
400	1085.25	1644.29	0.6460	0.6622
500	762.51	1137.68	0.6733	0.6840
600	657.48	728.04	0.6804	0.6916
700	562.89	741.25	0.6940	0.7020
800	613.99	639.59	0.6951	0.7019

TABLE IV
MESSAGE OVERHEAD AMONG DELEGATION NODES ON MODERATE SCALE SNs

scenario	DPVNE -PSO	HA-VNE -PSO	scenario	DPVNE -PSO	HA-VNE -PSO
VN40SN200_1	24681	25228	VN20SN200_1	38739	40990
VN40SN200_2	25235	25271	VN20SN200_2	40719	40916
VN40SN200_3	23105	23268	VN20SN200_3	40661	42346
VN40SN200_4	25021	26159	VN20SN200_4	40291	41255
VN40SN200_5	25734	26675	VN20SN200_5	42568	42875
VN40SN200_6	24867	25430	VN20SN200_6	40972	41943

archives can further improve the average revenue and acceptance ratios in transit-stub topologies (compare DPVNE-PSO and HA-VNE-PSO). There is an interesting result that the R/C ratios obtained by the centralized algorithm are lower than distributed approaches. This is because transit-stub topologies have small-world characteristics. Since distributed VNE approaches partition the whole SN into small sub-SNs, the dependency among sub-SNs becomes less. Hence, embedding VNs on sub-SNs is more effective.

At the same time, we record the usage ratio UR of historical mappings on these scenarios, which is computed as follows:

$$\mathrm{UR} = \frac{\text{used\_map}}{\text{all\_map}} \tag{28}$$

where *used_map* represents the number of using historical mappings to initialize particles, and *all_map* represents the total number of node mappings stored in archives. The experimental results are presented in Fig. S5 and Table S1 in the supplementary materials. It can be seen that more than 40% historical mappings in archives are reused when embedding repeated VNs although the status of SNs changes frequently. The results verify that the HA mechanism is available in dynamic situations.

G. Impact of Parameter p

There are three parameters in HA-VNE-PSO, *max_arc*, *min_VN*, and p (in Algorithm 2). The parameter *max_arc* and *min_VN* are problem dependent so p is the only parameter we need to investigate in HA-VNE-PSO. We test the parameter p varying in {0.1, 0.3, 0.5, 0.7, 0.8, 0.9} on moderate scale scenarios VN20SN200 to find the appropriate value of p. We also compare the algorithm without HA strategies as baselines.

The results for different p values are presented in Fig. S15 in the supplementary materials. According to the depicted results, we can observe that the behaviors for different values of p are unstable and it is difficult to assert which value of p is the best of all scenarios. This is reasonable because the topology structures of SNs and VNs are varied and the resources of SNs change frequently over time. It can also be observed that the performance with different values of p is similar. Therefore, for fair and easy comparison, we set p with the value which can make HA-VNE-PSO obtain relatively moderate performance. Thus, we set $p = 0.8$ in all our experiments which implies that about 80% VNoMs in the archives might be reused with the HA strategy.

VI. CONCLUSION

In this paper, we develop a new distributed system, HA-VNE-PSO, to handle the distributed VNE problem. In HA-VNE-PSO, we combine the distributed system with the devised SPSO-VNE to improve its optimizing capability. Furthermore, we propose the HA strategy to utilize the historical information in the archives. HA-VNE-PSO is compared with other approaches on moderate and large scale scenarios. The experimental results verify that HA-VNE-PSO is promising. In this paper, HA-VNE-PSO cannot survive physical failures. If some nodes or links fail in HA-VNE-PSO, the system cannot work normally. Therefore, technology to improve the reliability and robustness of distributed systems should be further studied in the future. Besides, extending the proposed distributed system to continuous space will be studied [45], [46].

REFERENCES

[1] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.
[2] Y. Jin and Y. Wen, "When cloud media meet network function virtualization: Challenges and applications," *IEEE MultiMedia*, vol. 24, no. 3, pp. 72–82, Aug. 2017.
[3] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 655–685, 1st Quart., 2016.
[4] D. G. Andersen. (2002). *Theoretical Approaches to Node Assignment*. [Online]. Available: http://www.cs.cmu.edu/~dga/papers/index.html
[5] A. Schrijver, *Theory of Linear and Integer Programming*. Amsterdam, The Netherlands: Wiley, 1998.
[6] 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.
[7] 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.
[8] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.
[9] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011.
[10] 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.
[11] 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.
[12] R. Lin *et al.*, "Virtual network embedding with adaptive modulation in flexi-grid networks," *J. Lightw. Technol.*, vol. 36, no. 17, pp. 3551–3563, Sep. 1, 2018.
[13] Z. Zhang *et al.*, "A unified enhanced particle swarm optimization-based virtual network embedding algorithm," *Int. J. Commun. Syst.*, vol. 26, no. 8, pp. 1054–1073, 2013.

[14] S. Su *et al.*, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1607–1620, Oct. 2014.

[15] 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.

[16] 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.

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

[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] (2017). *Amazing EC2*. [Online]. Available: https://aws.amazon.com/cn/

[20] I. Houidi, W. Louati, and D. Zeghlache, "A distributed and autonomic virtual network mapping framework," in *Proc. 4th Int. Conf. Auton. Auton. Syst. (ICAS)*, 2008, pp. 241–247.

[21] I. Houidi, W. Louat, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 5634–5640.

[22] M. T. Beck, A. Fischer, J. F. Botero, C. Linnhoff-Popien, and H. de Meer, "Distributed and scalable embedding of virtual networks," *J. Netw. Comput. Appl.*, vol. 56, pp. 124–136, Oct. 2015.

[23] M. T. Beck, A. Fischer, H. de Meer, J. F. Botero, and X. Hesselbach, "A distributed, parallel, and generic virtual network embedding framework," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2013, pp. 3471–3475.

[24] W. Humphrey, A. Dalke, and K. Schulten, "VMD: Visual molecular dynamics," *J. Mole. Graph.*, vol. 14, no. 1, pp. 33–38, 1996.

[25] H. J. Berendsen, D. van der Spoel, and R. van Drunen, "GROMACS: A message-passing parallel molecular dynamics implementation," *Comput. Phys. Commun.*, vol. 91, nos. 1–3, pp. 43–56, 1995.

[26] P. J. Cock *et al.*, "Biopython: Freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[27] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *J. Mole. Biol.*, vol. 235, no. 5, pp. 1501–1531, 1994.

[28] J. Kenndy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[29] W.-N. Chen *et al.*, "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.

[30] K. Choi *et al.*, "Hybrid algorithm combing genetic algorithm with evolution strategy for antenna design," *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, Mar. 2016.

[31] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, Jul. 2017.

[32] Q. Lin *et al.*, "Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 32–46, Feb. 2018.

[33] Z. Wang *et al.*, "A modified ant colony optimization algorithm for network coding resource minimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 325–342, Jun. 2016.

[34] W. Yuan, Y. Liu, H. Wang, and Y. Cao, "A geometric structure-based particle swarm optimization algorithm for multiobjective problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 9, pp. 2516–2537, Sep. 2017.

[35] A. Abid, M. T. Khan, and M. S. Khan, "Multidomain features-based GA optimized artificial immune system for bearing fault detection," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

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

[37] M. Feng, J. Liao, S. Qing, T. Li, and J. Wang, "COVE: Co-operative virtual network embedding for network virtualization," *J. Netw. Syst. Manag.*, vol. 26, no. 1, pp. 79–107, 2018.

[38] C. Hahn, S. Holzner, L. Belzner, and M. T. Beck, "Empirical evaluation of a distributed deployment strategy for virtual networks," in *Proc. Int. Conf. Mobile Secure Program. Netw.*, 2017, pp. 88–98.

[39] D. Zhang, S. K. Nguang, and L. Yu, "Distributed control of large-scale networked control systems with communication constraints and topology switching," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1746–1757, Jul. 2017.

[40] S. Yang, Q. Liu, and J. Wang, "Distributed optimization based on a multiagent system in the presence of communication delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 5, pp. 717–728, May 2017.

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

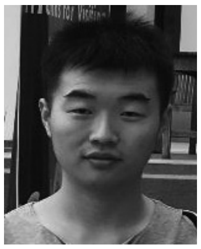
[42] E. W. Zegura, K. L. Calvert, and S. B. Acharjee, "How to model an internetwork," in *Proc. 15th Annu. IEEE INFOCOM*, 1996, pp. 594–602.

[43] 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.

[44] 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.

[45] Y.-H. Jia *et al.*, "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Trans. Evol. Comput.*, to be published.

[46] Q. Yang *et al.*, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.



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 (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, where he was also appointed as the Associate Dean in 2018. He has co-authored over 100 international journal and conference papers, including over 30 papers published in IEEE TRANSACTIONS journals. His current research interests include computational intelligence, swarm intelligence, evolutionary computation, planning and scheduling, network science, and their applications.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation Award in 2016, the National Science Fund for Excellent Young Scholars in 2016, and the National Natural Science Foundation of China—Royal Society Newton Fund Scholars in 2015. 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 of *Complex and Intelligent Systems*.

Tianlong Gu received the M.Eng. degree in computer science from Xidian University, Xi'an, China, in 1987 and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Huaqiang Yuan received the Ph.D. degree in computer software from Shanghai Jiao Tong University, Shanghai, China, in 1996.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include computational intelligence and cyberspace security.

Sam Kwong (M'93–SM'04–F'14) received the B.Sc. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree in electrical engineering from FernUniversität Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada, where he designed the diagnostic software to detect the manufacture faults of the very large scale integration chips in the Cyber 430 machine. He is currently the Head and the Professor of the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong is an Associate Editor of the IEEE Transactions on Evolutionary Computation, the IEEE Transactions on Industrial Informatics, IEEE Transactions on Industrial Electronics, and *Journal of Information Science*. He was elevated to IEEE Fellow for his contributions on optimization techniques for cybernetics and video coding in 2014. He is also appointed as IEEE Distinguished Lecturer for IEEE SMC Society in 2017. He is currently the Vice President of Conferences and Meetings with IEEE Systems, Man, and Cybernetics.

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.

From 2004 to 2016, he was a Professor with SUN Yat-sen University, Guangzhou, China. Since 2016, he has been with the South China University of Technology, Guangzhou, where he is currently a Cheung Kong Chair Professor. His current research interests include computational intelligence, cloud computing, big data, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 100 technical papers in the above areas.

Prof. 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, IEEE Transactions on Industrial Electronics, and IEEE Transactions on Cybernetics. He is the Founding and Current Chair of the IEEE Guangzhou Section, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and the ACM Guangzhou Chapter.