



# Optimizing Placement and Scheduling for VNF by a Multi-objective Optimization Genetic Algorithm

Phan Duc Thien<sup>1,2</sup> · Fan Wu<sup>1</sup> · Mahmoud Bekhit<sup>3,4</sup> · Ahmed Fathalla<sup>5</sup> · Ahmad Salah<sup>6,7</sup>

Received: 4 September 2023 / Accepted: 3 February 2024  
© The Author(s) 2024

## Abstract

Virtual network functions (VNFs) have gradually replaced the implementation of traditional network functions. Through efficient placement, the VNF placement technology strives to operate VNFs consistently to the greatest extent possible within restricted resources. Thus, VNF mapping and scheduling tasks can be framed as an optimization problem. Existing research efforts focus only on optimizing the VNFs scheduling or mapping. Besides, the existing methods focus only on one or two objectives. In this work, we proposed addressing the problem of VNFs scheduling and mapping. This work proposed framing the problem of VNFs scheduling and mapping as a multi-objective optimization problem on three objectives, namely (1) minimizing line latency of network link, (2) reducing the processing capacity of each virtual machine, and (3) reducing the processing latency of virtual machines. Then, the proposed VNF-NSGA-III algorithm, an adapted variation of the NSGA-III algorithm, was used to solve this multi-objective problem. Our proposed algorithm has been thoroughly evaluated through a series of experiments on homogeneous and heterogeneous data center environments. The proposed method was compared to several heuristic and recent meta-heuristic methods. The results reveal that the VNF-NSGA-III outperformed the comparison methods.

**Keywords** Heuristic algorithms · Mapping and scheduling · Multi-objective optimization · NSGA-III · Virtual network functions · VNFs

---

✉ Fan Wu  
wufan@hnu.edu.cn

✉ Ahmed Fathalla  
fathalla\_sci@science.suez.edu.eg

Phan Duc Thien  
phanducthien82@hnu.edu.cn

Mahmoud Bekhit  
mahmoud.bekhit@acu.edu.au

Ahmad Salah  
ahmad@hnu.edu.cn

- <sup>1</sup> College of Information Science and Engineering, Hunan University, Changsha 410082, China
- <sup>2</sup> Nam Dinh University of Technology Education, Nam Dinh, Vietnam
- <sup>3</sup> University of Technology of Sydney, Sydney, Australia
- <sup>4</sup> Australian Catholic University, Sydney, Australia
- <sup>5</sup> Department of Mathematics, Faculty of Science, Suez Canal University, Ismailia, Egypt
- <sup>6</sup> College of Computing and Information Sciences, University of Technology and Applied Sciences, Ibri, Sultanate of Oman

## 1 Introduction

Network functions virtualization (NFV) technology decouples the existing network equipment, which is reliant on hardware, into distinct hardware and software components. Therefore, it utilizes software-based network functions on a high-performance universal server using virtualization technology [1]. NFV-related technologies include various elementary technologies, such as virtualization, Virtual Network Function (VNF) placement, Service Function Chaining (SFC), and resource management technologies. There a great demand to advance the research on how to map and schedule VNFs efficiently. The VNF placement technology aims to operate VNFs with stable performance within the limited resources through efficient placement. This is achieved through the use different approaches such as optimization algorithms, Integer Linear Programming (ILP), greedy algo-

- <sup>7</sup> Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt

rithm, and dynamic programming algorithm, which have been previously employed in the computing field.

Recent research has tackled the VNF placement [2, 3] and scheduling subproblems [4, 5]. However, most studies address these issues separately, potentially compromising strict service deadlines. Notably, works such as Refs. [6] and [7] focus on the broader VNF layout and scheduling problem. Gholipour et al. [6] emphasize placing new VNFs without considering the reusability of deployed ones, possibly incurring additional costs. In addition, Li et al. [7] overlook communication delays in planning, risking service deadline violations. In Ref. [8], the author considered the transmission and VNF chain processing delays using a GA-based algorithm to optimize the distribution coefficient and enhancing virtual link transmission speeds. Luizelli et al. reduced virtual switching costs by deploying sub-chains on physical servers [9]. However, these solutions address single factors in the VNF planning process which is lacking a comprehensive approach. Consequently, these methods may incur high-performance penalties due to exponentially increasing computational costs in each iteration and for each additional decision parameter.

Thus, although the planning and placement optimization of VNFs has been extensively researched, there are still significant gaps in the research field. Most existing research focuses on single and bi-objective optimization [10, 11], which may not fully capture the complexity and trade-offs inherent in network design. Furthermore, many of these studies do not fully consider the dynamic nature of network traffic and the need for flexible and adaptive solutions. There is a motivation to consider more objectives during the searching task for the optimal planning and placement optimization of VNFs due to the complex nature of this problem.

The motivation of this work is to formulate the two problems of mapping VNFs and then scheduling these VNFs as a multi-objective problem with three objectives, namely (1) minimizing line latency over network links, (2) reducing the processing capacity of each virtual machine (VM), and (3) reducing the processing latency of the VMs. Then, it was proposed to utilize the well-known NSGA-III method to solve this multi-objective optimization problem as a combinatorial problem, where the three objectives are to be minimized. In the proposed work, tasks were formulated as binary and discrete variables. Then, it was proposed to adapt the NSGA-III algorithm for the VNFs mapping and scheduling problem. Experiments on homogeneous and heterogeneous data center environments were used to assess the performance of the offered solutions. The significant contributions of the proposed work can be summarized as follows:

1. We proposed framing the problem of finding the optimal VNFs mapping and scheduling finding as a multi-

objective problem with three goals, namely minimizing line latency over network links, reducing the processing time of different VMs and the processing latency of all VMs. We utilized an adapted NSGA-III method to solve this multi-objective problem; it is called VNF-NSGA-III. To our knowledge, this is the first work to consider these three objectives during searching for the optimal VNFs mapping and scheduling.

2. Our proposed method was evaluated through a set of comprehensive experiments. The obtained results show the proposed VNF-NSGA-III algorithm outperformed the methods of comparison on the three objective values. The experiments include different environments such as homogeneous and heterogeneous data centers.

The remainder of this article is organized as follows: Sect. 2 discusses the literature foundations and overview of the VNF. Section 3 presents and discusses the problem definition. Section 4 proposes an adaptive VNF-NSGA-III technique for scheduling the VNFs problem as a multi-objective optimization problem. Section 5 discusses the simulation and experiment results. Section 6 concludes the paper.

## 2 Background and Related Work

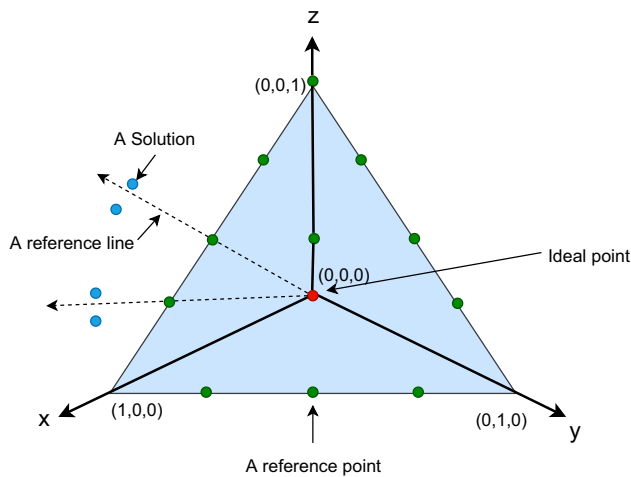
In this section, we will introduce a background of the NSGA-III algorithm which is the proposed method utilized to address the problem at hand. Then, the existing methods which addressed the problem of VNFs mapping and scheduling will be exposed.

### 2.1 Background

Optimization is vital field in artificial intelligence which successfully addressed various problems in several domains [12]. Multi-objective optimization problems exist widely in real-world applications [13–15] where the number of objectives in the optimization model is more than two objectives.

Deb and his colleagues recently announced the NSGA-III [16], the next and improved version of NSGA-II. It has been explicitly proposed for multi-objective optimization (i.e., with more than three objectives). In addition, NSGA-III presents an alternative technique using a point-based solution reference instead of the NSGA-II method, called crowded distance, to maintain a uniform distribution of multiple solutions. The following section will briefly present NSGA-III and its advantages in solving multi-objective optimization problems.

Denote the values of the first, second, and third objectives in a three-objective problem as  $x$ ,  $y$ , and  $z$ , respectively. The ideal point in minimization is  $(0,0,0)$ , the normalized values



**Fig. 1** Illustration of NSGA-III reference 3D points

of  $(x,y,z)$ , with values ranging from 0 to 1. The best reference points for each target are  $(0,y,z)$ ,  $(x,0,z)$ , and  $(x,y,0)$ , while the worst examples are  $(1,y,z)$ ,  $(x,1,z)$ , and  $(x,y,1)$ . The distance between the two worst situations is divided into four portions in Fig. 1, displaying the reference points of three-objective optimization. As a result, 15 reference points are generated in this scenario utilizing NBI. Furthermore, a reference line connects the ideal point and a reference point logically. Hence, the solution closest to the reference line will be allocated to the corresponding point along that line to assign a solution to each reference point.

Before the selection step in each generation, parents produce offspring with the same quantity,  $N$ , through crossover and mutation operations. The combination between parents and offsprings, usually of size  $2N$ , are sorted by means of a fast nondominated-sorting algorithm. In case of the size of nondominated solutions having  $N$  those solutions are selected for the next generation, otherwise, a The niche-preservation operation is executed through an iterative process, linking a solution with the shortest relative distance to reference points. This procedure is reiterated until a satisfactory population is obtained. In cases where multiple solutions share the same distance, random selection is applied.

Despite the fact that NSGA-III was designed for multi-objective issues with more than three objectives [16], it has generally been applied to three-objective problems in the literature [16–21]. It was also compared to NSGA-II, which is another popular evolutionary multi-objective optimization algorithm (EMO) [17]. In multi-objective optimization, the performance supremacy of both variants is still contested, as there are some circumstances where NSGA-II outperforms NSGA-III [17].

The NSGA-III algorithm employs a set of reference points to preserve solution diversity, diverging from the

NSGA-II algorithm, which utilizes the crowding distance operation. Figure 2 shows an example of the NSGA-III and NSGA-II algorithms' outcomes in selecting a population in a two-objective optimization problem. In Fig. 2, the brown circles denote selected solutions, while white circles indicate rejected options. The NSGA-II algorithm chooses solution 4 in Fig. 2a, whereas the NSGA-III algorithm rejects it. In contrast, the NSGA-III algorithm, not the NSGA-II algorithm, selects solution two in Fig. 2b. That shows that the NSGA-II algorithm is highly diversified while NSGA-III is highly convergent.

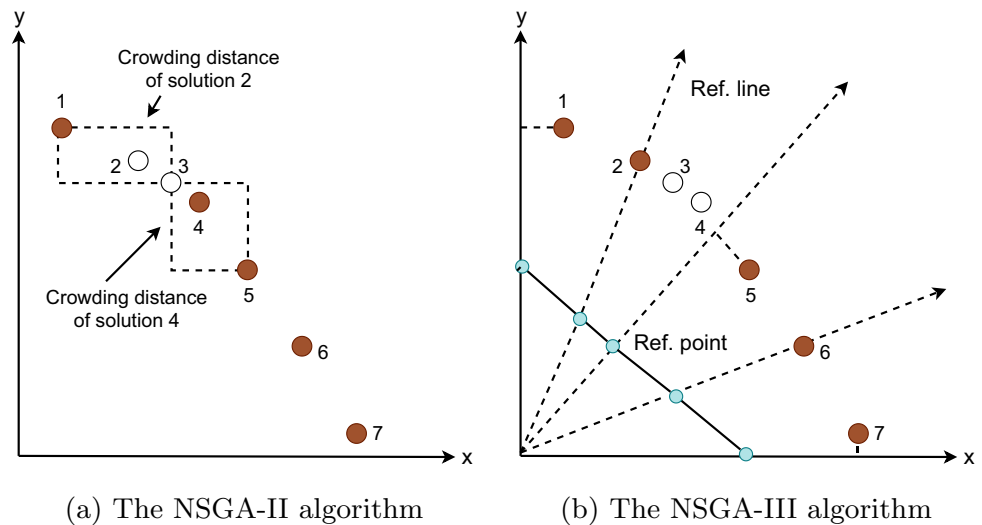
As previously noted, the NSGA-III algorithm performs well in both types of mathematical problems [16, 17]. In most cases, NSGA-III generated nondominated solutions linked to all reference points, even when the issues involved linear, convex, concave, and multimodal distributions. However, in scheduling problems, the quality of solutions of the NSGA-III algorithm, in terms of convergence and diversity, still has some drawbacks because the number of nondominated solutions is not always satisfied [18, 19].

## 2.2 Related Work

Effectively deploying network services via a VNF chain is a complex task involving challenges such as network function mapping, traffic routing, and service scheduling to fulfill user requirements [22]. While recent studies have concentrated on tackling these issues, most tend to address only a specific sub-problem among the mentioned challenges.

The VNF mapping sub-problem has been explored in various studies in the literature, such as Refs. [23], [24], [25], [26], [27], and [28]. For instance, Oljira et al. [27] addressed the virtual mobile core network function placement issue in NFV-based core networks, formulating it as a mixed integer linear programming (MILP) problem to minimize resource usage while considering virtualization overhead for specific latency requirements. Woldeyohannes et al. [28] formulated the VNF placement problem as a multi-objective integer linear programming problem, optimizing admitted flows, node and link utilization, and latency requirements, proposing a heuristic for efficient problem-solving. Ruiz et al. [23] introduced a genetic algorithm to address design problems related to VNF topology, VNF chain, and virtual topology in 5G networks, reducing active CPUs through node combination in multi-access edge computing (MEC) for improved energy efficiency. Liu et al. [29] studied the dynamic VNF placement problem, aiming to maximize network operator profits by deciding whether to migrate deployed VNFs or deploy new ones effectively, employing a column generation algorithm. In addition, some studies, such as Refs. [30], [31], and [32], consider VNF reuse or sharing in the placement problem to minimize resource costs. Despite these efforts, none of the mentioned studies addresses the comprehensive planning

**Fig. 2** The NSGA-II and NSGA-III algorithms' selection of the solution techniques



for VNFs. Numerous studies address the VNF scheduling problem, such as Refs. [33] and [34].

The authors in Ref. [33] introduced an energy-conscious method for mapping and scheduling services, which is formulated as an Integer Linear Programming (ILP) problem. Their contribution consists of a genetic-based heuristic algorithm designed to prioritize resolution tasks, with a particular focus on meeting deadlines and optimizing server consolidation. The authors of Ref. [34] provided a dynamic priority-based system for scheduling Virtual Network Functions (VNFs), taking into account the changing nature of the network. The scheduling takes place following the execution of correlated VNFs, based on revenue and delay budget. The problem is formulated as a MILP, incorporating constraints like SFC request VNF order, changing link availability in the STN topology, and VNF-related correlation handling. Some studies integrate machine learning models into VNF scheduling [35–37], enhancing flexibility as user requirements evolve. Li et al. [37] model the problem as an MDP, utilizing RL algorithms to find optimal schedules with a specific reward function for latency-guaranteed VNFs. In Ref. [36], RL covers discrete and continuous actions, minimizing VNF costs under end-to-end QoS constraints through a multi-agent solution, albeit requiring more iterations for cooperative learning.

Few authors concurrently address VNF planning and mapping, often focusing on resolving two goals simultaneously. In a recent study, Yi Xiang et al. [38] explored constrained multi-objective optimization to balance constraints and objectives, proposing a model with multiple conditions. They considered various problem types and applied suitable mechanisms to address relationships between rules and goals (e.g., constraint priority, goal priority, or switching between them). Jiugen Shi et al. [39] tackled minimizing VNF implementation costs and accounting cost rules by constructing a

hybrid ILP (MILP) and providing a two-way algorithm in Ref. [39]. The authors in Refs. [40, 41] addressed the VNF scheduling problem by trading off resource allocation across the network. Mahmoud Gamal et al. [42] utilized genetic algorithms to map and schedule heterogeneous incoming requests of virtual network functions for near-optimal convergence. Similarly, two superheuristic methods based on hill climbing and simulated annealing algorithms are proposed in Ref. [43]. These two methods are built to minimize the total energy used during the placement of virtual network functions, an energy-aware approach.

However, these studies only address one or two objectives of VNF mapping and planning and come at the cost of increased costs and processing time. To solve three or more goals simultaneously, the algorithms in the documents above still have limitations [44]. This requires choosing appropriate methods and algorithms to solve multiple plans for VNF scheduling and mapping simultaneously.

### 3 Mathematical Problem Formulation

In this section, the mathematical model for determining the appropriate mapping and scheduling for the received service requests is presented. The optimum VNFs scheduling onto physical nodes, on the other hand, is beyond the scope of this study. The proposed model is a development of the models provided in Refs. [8, 42, 45].

The proposed model uses the symbols in Table 1 to describe the variables and parameters in the mathematical formulas and algorithms. The proposed model has three objectives in order to optimize the scheduling problem and map the placement of the VNFs.

**Table 1** Math symbols

Symbol	Description
$\mathcal{V}$	Set of virtual machine nodes
$c_k$	Processing capacity of $k$ -th virtual machine
$\mathcal{E}$	Set of virtual links
$tc_{k,l}$	Transmission capacity which occurred at each VL connecting the VMs
$F$	Set of unique VNFs established in the cloud
$\mathcal{S}$	Sequence of the incoming service requests is indicated
$d_{i,j}$	Bandwidth demand to process the $j$ -th VNF of the $i$ -th incoming service request
$R_{i,j}^{k,l}$	Traffic flow of the $j$ -th VNF of the arriving service request $s_i$ passing through $(k, l)$ -th link
$\beta_{i,j}$	Transmission starting time of the $f_{i,j}$ and the $f_{i,j+1}$

### 3.1 System Model

In the proposed model, it is considered that the NFV network composes of a graph, undirected,  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . The  $\mathcal{V} = \{v_k | k = 1, 2, \dots, |\mathcal{V}|\}$  specifies a group of virtual machines. The  $\mathcal{E} = \{(k, l) | k, l = 1, 2, \dots, |\mathcal{V}|, k \neq l\}$  establishes a network of virtual links (VLs), linking the VMs. Moreover, a collection of one-of-a-kind VNFs created within the cloud system is defined by  $F = \{f_1, f_2, \dots, f_{|F|}\}$ . Every VM can run a group of different VNFs defined by  $V F_k$  for  $k = 1, \dots, |\mathcal{V}|$ .

The incoming service requests list is indicated by  $\mathcal{S} = [s_1, \dots, s_{|\mathcal{S}|}]$ , where every received request  $s_i \in \mathcal{S}$  for  $i = 1, 2, \dots, |\mathcal{S}|$  includes an ordered list of VNFs represented by  $f_{i,j}$  for  $i = 1, 2, \dots, |\mathcal{S}|$  and  $j = 1, 2, \dots, |s_i|$ , where  $f_{i,j} \in F$ .  $f_{i,j}$  is the  $j$ -th VNF of the  $i$ -th incoming service request ( $s_i$ ). Given that  $c_k$  for  $k = 1, 2, \dots, |\mathcal{V}|$  determines the processing capacity of the  $k$ -th VM, the physical machine (PM)'s memory and processing power, which hosts the VM, can be utilized to calculate processing capacity. Moreover, let  $tc_{k,l}$  for  $k = 1, 2, \dots, |\mathcal{V}|$  and  $l = 1, 2, \dots, |\mathcal{V}|$  defines the transmission capacity which occurred at each VL, where the VL link two different VMs. The  $d_{i,j}$  determines the bandwidth requirement for processing the  $j$ -th VNF of the  $i$ -th arriving service request for  $i = 1, 2, \dots, |\mathcal{S}|$  and  $j = 1, 2, \dots, |s_i|$ . Finally,  $R_{i,j}^{k,l}$  describes the flow of traffic of the  $i$ -th VNF of the coming request  $s_i$  traveling across the  $(k, l)$ -th link for  $i = 1, 2, \dots, |\mathcal{S}|$ ,  $j = 1, 2, \dots, |s_i|$ ,  $k = 1, 2, \dots, |\mathcal{V}|$ , and  $l = 1, 2, \dots, |\mathcal{V}|$ .

In this paper, the topology of the network is including the VLs which is connecting the VMs is assumed to be time-invariant. In addition, it is assumed that each service requests  $s_i$  for  $i = 1, 2, \dots, |\mathcal{S}|$  carries a varied a set of VNFs on it.

### 3.2 Objectives

#### 3.2.1 Minimizing the Transmission Delay

This objective aims to optimally reduce the transmission delay which is faced due to transferring service  $s_i$  between two different VMs on the corresponding link  $(k, l)$  (where  $k, l \in |\mathcal{V}|$  and  $k \neq l$ ). The transmission delay can be denoted as  $\frac{d_{i,j}}{tc_{k,l} - R_{i,j}^{k,l}}$  for the  $j$ -th VNF of the  $i$ -th service request on the  $(k, l)$  link. The  $\beta_{i,j}$  variable for  $i = 1, 2, \dots, |\mathcal{S}|$  and  $j = 1, 2, \dots, |s_i| - 1$  is utilized to identify the transmission time for  $f_{i,j}$  and the  $f_{i,j+1}$ , respectively, between two VMs, where the  $f_{i,j}$  is the  $i$ -th service request (i.e.,  $s_i$ ) and the  $j$ -th VNF of this  $s_i$ .

The first objective is expressed in Eq. 1 as follows:

$$MinZ_1(X) = \sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|s_i|-1} \left( \beta_{i,j} + \sum_{k=1}^{|\mathcal{V}|} x_{i,j}^k \cdot \sum_{l=1}^{|\mathcal{V}|} \frac{d_{i,j}}{tc_{k,l} - R_{i,j}^{k,l}} \right) \tag{1}$$

where  $tc_{k,l}$  for  $(k, l) | k, l = 1, 2, \dots, |\mathcal{V}|$  defines the transmission capacity which occurred at each VL connecting the VMs,  $d_{i,j}$  for  $i = 1, 2, \dots, |\mathcal{S}|$  and  $j = 1, 2, \dots, |s_i|$  defines the requested bandwidth to run the VNF number  $j$  which belongs to the service request number  $i$ ,  $R_{i,j}^{k,l}$  for  $i = 1, 2, \dots, |\mathcal{S}|$ ,  $j = 1, 2, \dots, |s_i|$ ,  $(k, l) | k, l = 1, 2, \dots, |\mathcal{V}|$  represents the traffic flow of the VNF number  $j$  which belongs to the service request number  $i$  transmitted through  $(k, l)$ -th link and  $x_{i,j}^k$  is a binary variable that determines whether the VNF number  $j$  which belongs to the service request number  $i$  can run on the  $k$ -th VM. Formally,  $x_{i,j}^k$  is calculated using Eq. 2:

$$x_{i,j}^k = \begin{cases} 1 & \text{If VNF number } J \text{ belongs to} \\ & s_i \text{ is run by a VM } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$



### 3.2.2 Minimizing the Processing Capacity

This objective aims to distribute the service requests  $s_i$  relatively to VMs according to the VMs computational capacity. As the bandwidth request  $d_{i,j}$  of running the VNF number  $j$  which belongs to the service request number  $i$  in different VMS can be different. Moreover, the shortage and oversupply of the VM processing power are not acceptable.

Finding the proper VM to handle the VNF number  $j$  which belongs to the service request number  $i$  in accordance with the bandwidth demand  $d_{i,j}$  of the arriving service requests and the computation capacity status of each VM is then an important task to complete. Then, this objective is formulated to solve this task and minimize the total processing capacity for each VM while ensuring that the total number of incoming service requests is implemented in order. The current model’s objective does not report the average percentage between the number of accepted service requests and the sum of VMs’s capacities. Finally, this objective is expressed with the help of Eq. 3 as follows:

$$MinZ_2(X) = \sum_{k=1}^{|\mathcal{V}|} \left( c_k - \sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|s_i|} d_{i,j} \cdot x_{i,j}^k \right) \quad (3)$$

where  $c_k$  for  $k = 1, 2, \dots, |\mathcal{V}|$  defines the processing capacity of the VM number  $k$ , the processing capacity can be calculated based on memory and computing capabilities of the PM hosting each VM,  $d_{i,j}$  for  $i = 1, 2, \dots, |\mathcal{S}|$  and  $j = 1, 2, \dots, |s_i|$  defines the bandwidth demand to run he VNF number  $j$  which belongs to the service request number  $i$ , and  $x_{i,j}^k$  is a variable to show whether the VNF number  $j$  which belongs to the service request number  $i$  can run on the  $k$ -th VM.

### 3.2.3 Minimizing the Processing Delay

While the service requests are arriving to be processed on separate VMs, it is possible that a processing delay will arise throughout the operation of handling the service requests. This objective aims to solve this problem and minimize the processing delay for the arrival service requests on VMs. As a result of this procedure, incoming service requests will be processed swiftly, and the system’s quality will be improved. The definition of the beginning time of running the VNF number  $j$  which belongs to the service request number  $i$  as  $\alpha_{ij}$ . This objective is formally expressed with the help of Eq. 4 as follows:

$$MinZ_3(X) = \sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|s_i|} \left( \alpha_{i,j} + \sum_{k=1}^{|\mathcal{V}|} x_{i,j}^k \cdot q_{i,j}^k \right) \quad (4)$$

where  $x_{i,j}^k$  is a variable that indicates whether the  $j$ -th VNF of service request  $s_i$  can run on the  $k$ -th VM,  $q_{i,j}^k$  represents a processing delay for the VNF number  $j$  which belongs to the service request number  $i$  while it is being running on the  $k$ -th VM,  $\alpha_{ij}$  is the beginning time of the processing for the VNF number  $j$  which belongs to the service request number  $i$ .

### 3.3 The List of the Proposed Models’ Constraints

Different constraints have been proposed in this model to ensure that VNF service requests are mapped and scheduled and are feasible and ready to apply.

1. Every  $j$ -th VNF for  $j = 1, 2, \dots, |s_i|$  of the received service request  $s_i$  for  $i = 1, 2, \dots, |\mathcal{S}|$  must run on at least one VM. Formally, this constraint is presented as follows:

$$\sum_k x_{ij}^k \geq \mathcal{V} \quad (5)$$

2. Every  $j$ -th VNF for  $j = 1, 2, \dots, |s_i|$  of the received service request  $s_i$  for  $i = 1, 2, \dots, |\mathcal{S}|$  must run at most  $N$  VMs. Formally, this constraint is presented as follows:

$$\sum_k x_{ij}^k \leq 1 \quad (6)$$

3. The total VM capacity  $c_k$  for  $k = 1, 2, \dots, |\mathcal{V}|$  must not exceed the data volume (in bits or packets)  $d_{i,j}$  for the VNF number  $j$  which belongs to the service request number  $i$  for  $i = 1, 2, \dots, |\mathcal{S}|$  run on the VM number  $k$ . Formally, this constraint is presented as follows:

$$\sum_k x_{ij}^k \cdot d_{i,j} \leq c_k \quad (7)$$

4. For  $i = 1, 2, \dots, |\mathcal{S}|$  traveling across the  $(k, l)$ -th link, the traffic flow  $R_{i,j}^{k,l}$  for  $k$  shall not be more the link capacity  $tc_{k,l}$ . Formally, this constraint is presented as follows:  $R_{i,j}^{k,l}(t)$

$$\sum_k \sum_l x_{i,j}^k \cdot R_{i,j}^{k,l} < tc_{k,l} \quad (8)$$

5. This constraint seeks to process all of the VNFs of the same service request in the same sequence which they were received. The VNF number  $j$  which belongs to the service request number  $i$  for  $i = 1, 2, \dots, |\mathcal{S}|$  must start running on the  $k$ -th VM after the previous  $(j - 1)^{th}$  VNF of the same  $s_i$  finalized its work:

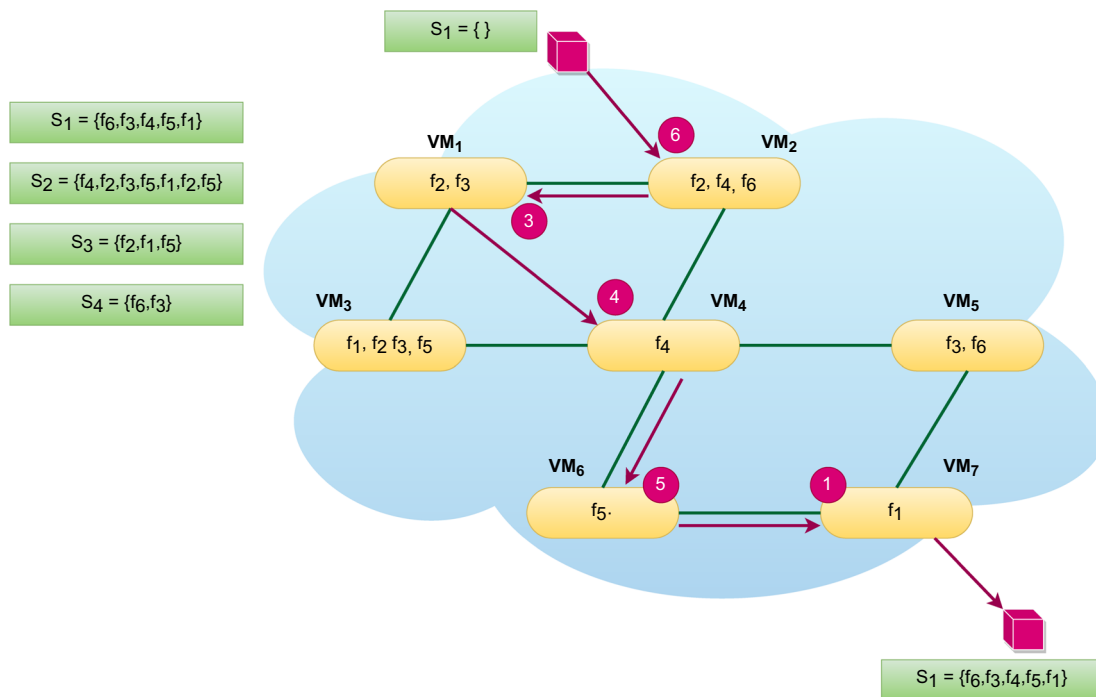


Fig. 3 An example of a potential mapping for  $s_1$  on an instance of network structure

$$\sum_{i=1}^{|S|} \sum_{j=2}^{|s_i|} \left( \beta_{i,j-1} + \sum_{k=1}^{|\mathcal{V}|} x_{i,j-1}^k \cdot \sum_{l=1}^{|\mathcal{V}|} \frac{d_{i,j}}{tc_{k,l} - R_{i,j}^{k,l}} \right) \leq \alpha_{ij} \tag{9}$$

6. After the VNFs of the exact service requests have completed their processing on the VMs, this restriction aims to send them in the correct order. After the  $j$ -th VNF has completed running on the VM number  $k$ , the traffic from the  $j$ -th VNF for the arrived service request  $s_i$  must transfer in order from this  $k$ -th VM to the next  $l$ -th VM. Put another way, the VNF number  $(j + 1)$  must not begin transmitting until the VNF number  $j$  which belongs to the service request number  $i$  finished its running on the VM number  $k$ . Formally, this constraint is presented as follows:

$$\sum_{i=1}^{|S|} \sum_{j=1}^{|s_i|-1} \left( \alpha_{i,j} + \sum_{k=1}^{|\mathcal{V}|} x_{i,j}^k \cdot q_{i,j}^k \leq \beta_{i,j} \right) \tag{10}$$

### 3.4 An Example of How to Map and Schedule a Set of Service Requests

Figures 3 and 4 provide an illustration of the problem effectively. There are seven VMs in the network, which support various VNF functionalities (e.g., 1. Load Balancing 2. Firewall, 3. Virtual Private Network (VPN) 4. Video Transcoder,

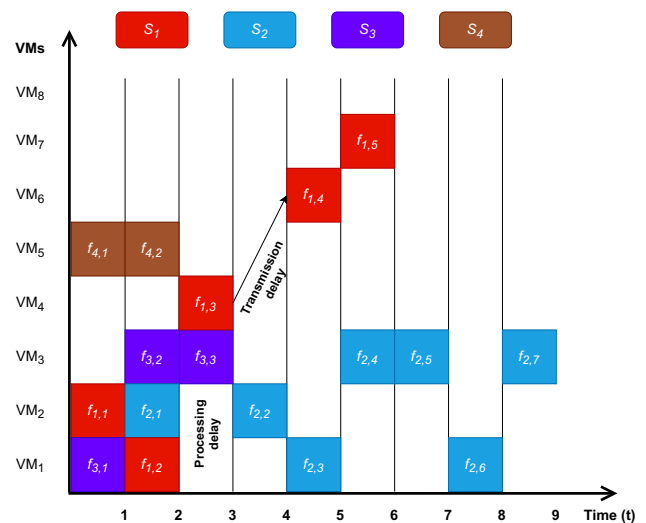


Fig. 4 A graphical illustration of mappings and scheduling for a sequence of the received service requests

5. Network Address Translation (NAT), and 6. Proxy). In Fig. 3, each item  $S_1, S_2, S_3,$  and  $S_4$  is an example of a sequence of VNF requests to be executed. In each sequence,  $S_1, S_2, S_3,$  and  $S_4$  contain a list of different virtual network functions that the user expects the system to execute in the correct order. Here we will describe in detail the processing of VNFs in  $S_1$  request chain as shown in Figs. 3. At the time  $t = 0$ , a set of inbound service requests  $s_1, \dots, s_4$  come requesting processing in these VMs, as illustrated in Figs. 3.

A distinct set of VNFs are arranged differently in each request for service.

Certain VMs will process these VNFs while others will not be since the VNFs are distributed to execute on VMs in a randomly distributed manner. According to Fig. 3, the first VNF of  $s_1$  (i.e.,  $f_6$ ) is handled in  $VM_2$ , and then the next of VNFs to the end VNFs of  $s_1$  (i.e.,  $f_3, f_4, f_5, f_1$ ) are each processed in  $VM_1$ , followed by  $VM_4, VM_6$  and  $VM_7$ , respectively. In this scenario, it is anticipated that each VM can execute a single VNF at once.

As illustrated in Fig. 4, a transmission delay for a set of VNFs may occur during their transfer between two different VMs. The first VNF of  $s_2$  (i.e.,  $f_4$ ) arrives at  $t = 0$  in the cloud simultaneously with the other VNFs of other requests, and this VNF can be running in either VM2 or VM4, depending on the configuration.

When this function  $f_4$  is assigned to  $VM_2$ , as illustrated in Fig. 4, it must wait for a one-time slot before it can be processed (since  $VM_2$  is still processing the first VNF of  $s_1$  at the time of assignment). However, it is possible to assign the function to  $VM_4$  in order to prevent this delay.

A processing delay for these VNFs may also arise during their processing on VMs, as seen in Fig. 4. Figure 4 shows the assignment schedule for the second VNF  $f_2$  of  $s_2$ , which is allocated to the  $VM_2$  needs to wait for a one-time window before it can begin processing. There are three objectives in this paper that may be used to optimize all of the problems listed in Fig. 4, which are defined in Eqs. 1, 3, and 4.

## 4 The Proposed Algorithm

In the previous section, we described the VNF mapping and VNF scheduling problem as a multi-objective optimization problem. Therefore, we choose the NSGA-III algorithm to solve the problem. We start by describing the main steps of the algorithm. Then we present some adjustments to solve the three optimization problems minimizing the transmission delay, the processing capacity, and the processing delay. In the following subsections, the proposed algorithm is explained in more details.

### 4.1 Main Steps

The proposed algorithm starts by initializing random initial solutions. To balance both convergence and diversity of Pareto solutions in accordance with the problem requirements, before the selection step in each generation, parents create their offspring with the same number of  $N$  by using crossover and mutation operations. The parent-child union is usually  $2 \times N$  in size and sorted using a fast non-sorting algorithm. In case the size of the solutions is not selected for  $N$  then those solutions are selected for the next generation;

otherwise, a Niche conservation operation is performed. This is an iterative process of associating a solution, having the shortest relative distance, with reference points. The procedure is repeated until a sufficient number of populations are obtained. If multiple solutions have the same distance, then a random selection will be adopted.

The detailed steps are as follows:

1. Step 1: Set random initial values for the algorithm's parameters including  $\mathcal{V}, c_k, \varepsilon, t_{c_{k,l}}, S, d_{i,j}, R_{i,j}^{k,l}, \beta_{i,j}$ , and the stopping condition.
2. Step 2: Minimize the transmission latency of service requests when moving from one VM to another, based on Eqs. 1 and 2.
3. Step 3: Finding the suitable virtual machine to execute the VNF number  $j$  of  $s_i$ , considering  $d_{i,j}$  of the received requests and each VM's current processing capacity status.
4. Step 4: While service request  $s_i$  is being processed on the  $k$ -th VM, minimizing the running time for the  $j$ -th VNF is a priority.
5. Step 5: Return the set of valid mapping and scheduling, which are represented as a set of solutions for every VNF of the input service request  $\{X^1, \dots, X^N\}$

Detailed descriptions of steps 2, 3, and 4 will be exposed in Sects. 4.2.1, 4.2.2, and 4.2.3, respectively. In the following subsection, we will discuss the proposed algorithm to solve specific goals, and then we will analyze and evaluate the time complexity of the proposed algorithm.

### 4.2 Algorithms for VNF Mapping and VNF Scheduling Problems

In this study, we adapted the NSGA-III algorithm to map and schedule VNFs. VNFs mapping and scheduling goals are handled by minimizing transmission latency, reducing processing time on each VM, and reducing the total capacity of the used VMs. Thereby effectively exploiting available resources, reducing the cost of creating and moving VMs, and solving the problem of high power consumption in data centers.

After initializing the proposed algorithm's parameters, the VMs data, the VNF list, the number of requests will be loaded into the algorithm. Checking input data and editing non-standard data will prevent the system from encountering unnecessary errors during execution. VNF scheduling and mapping process will go through different stages, namely (1) distributing VNF across VMs based on the VM's processing capacity, (2) arranging virtual network functions on each VM, and (3) scheduling VNF on VMs. At the VNF scheduling stage on the VM, the algorithm will initialize the variable that stores the number of scheduled requests  $gen = 0$ . The



algorithm will cycle through the VMs in turn and check if there is still virtual network functionality or not. The algorithm will check the current virtual machine’s ability to handle VNF, update the processing time if the VM responds, and it will update the transmission delay corresponding to each target. After each turn, the system will update the number of scheduled requests by increasing  $gen+ = 1$ . VNFs scheduling on the virtual server will be done until the list of virtual network functions to be scheduled is exhausted. The number of scheduling requests will correspond to the network size in our experiment. At the end of the VNF scheduling process, the fitness value is calculated by Eq. 11 corresponding to each network size.

In our study, based on the priority of the objectives, the importance of the first objective is 40%, the importance of the 2<sup>nd</sup> objective is 30%, and the importance of the 3rd objective is 30% that we choose the value of the weights in Eq. 11:

$$res = 0.4 * obj_1 + 0.3 * obj_2 + 0.3 * obj_3 \tag{11}$$

where  $obj_1$ ,  $obj_2$ , and  $obj_3$  are the values of the respective objectives.

We called the multi-objective optimization adapted NSGA-III algorithm for solving the VNF scheduling as VNF-NSGA-III. The flowchart of the proposed VNF-NSGA-III algorithm is depicted in Fig. 5. The system will check the processing capacity of VMs for network functions in the queue and update the transmission delay value and total machine capacity. VM needs to be used according to Eqs. 1, 3, and 4. This process is repeated until the stopping condition of the algorithm is reached.

#### 4.2.1 An Algorithm for Minimizing the Transmission Delay

The details of the algorithm to find minimum transmission delay when transferring from one VM to another are presented in Algorithm 1. We used  $x_{i,j}^k$  to define the  $j$ -th VNF of arriving request  $s_i$  can execute on the  $k$ -th VM node and  $\beta_{i,j}$  to identify the transmission starting time of the  $f_{i,j}$  and the  $f_{i,j+1}$ , respectively, from one VM to another.

The  $S$  represents the order of the received service requests,  $\mathcal{V}$  is the set of VMs nodes,  $d_{i,j}$  defines the required bandwidth to process the  $j$ -th VNF of the  $i$ -th coming request.

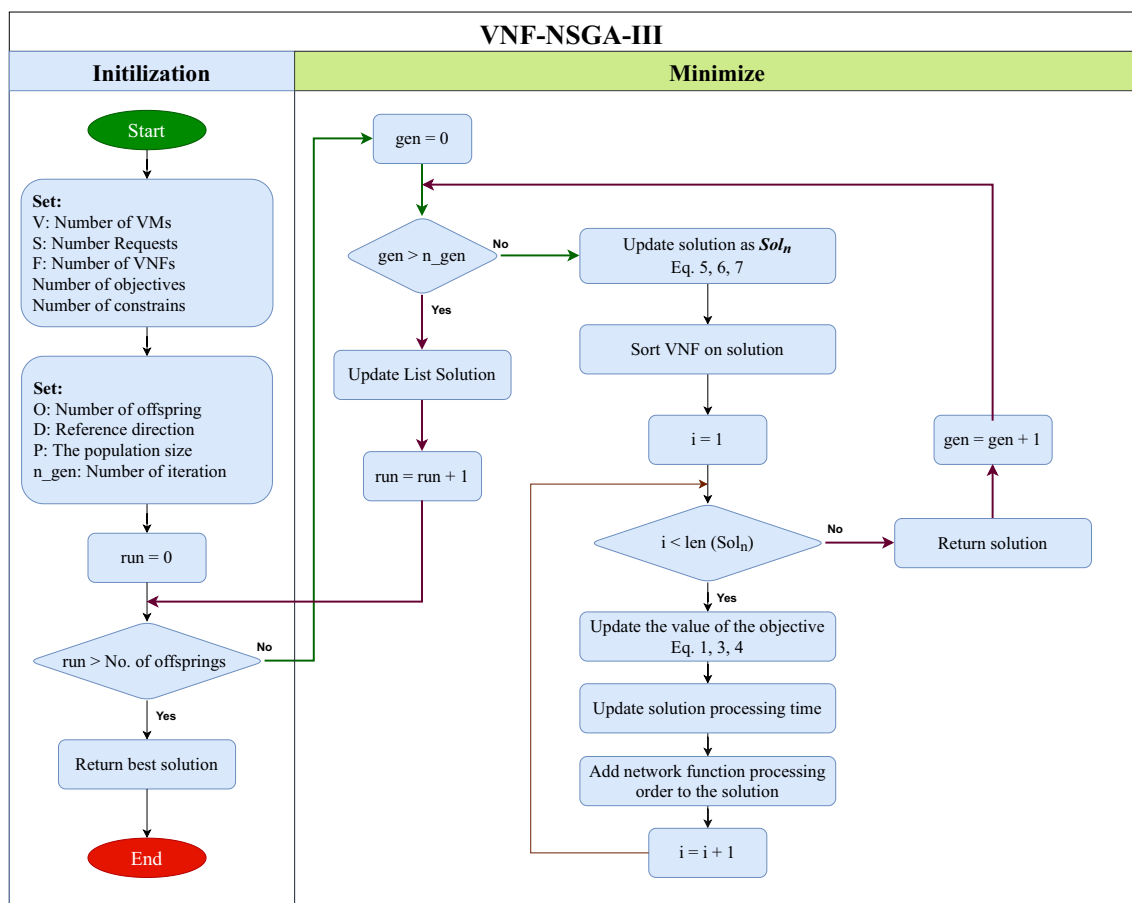


Fig. 5 A flowchart of the proposed VNF-NSGA-III algorithm

**Algorithm 1** Minimize the transmission delay algorithms

**Result:** Reduce the incoming service request's transmission delay for each VNF.

initialization

```

for  $i \leftarrow 1$  to  $|S|$  do
  for  $j \leftarrow 1$  to  $|s_i| - 1$  do
    for  $k \leftarrow 1$  to  $|\mathcal{V}|$  do
      if the  $j$ -th VNF of the  $s_i$  can run on the  $k$ -th VM then
         $x_{i,j}^k = 1$ 
      end
    else
       $x_{i,j}^k = 0$ 
    end
    Sum( $x_{i,j}^k$ )
  end
  for  $l \leftarrow 1$  to  $|\mathcal{V}|$  do
    Sum( $d_{i,j}/(tc_{k,l} - R_{i,j}^{k,l})$ )
  end
   $\beta_{i,j} + \text{Sum}(x_{i,j}^k) * \text{Sum}(d_{i,j}/(tc_{k,l} - R_{i,j}^{k,l}))$ 
end
end

```

*Time complexity analysis for Algorithm 1:* For a given system state, the task of finding the lowest transmission delay consists of two steps. The first step calculates the total number of VMs capable of handling VNF on demand, with a computational complexity of  $O(|\mathcal{V}|)$ . Then calculate the total transmission delay for the link when transferring VNF from one VM to another, the computational complexity is  $O(|\mathcal{V}|)$ . Therefore, the total computational complexity of being is  $O(|\mathcal{V}|^2)$ . Note that Algorithm 1 is executed only once for a given state during the entire scheduling process.

**4.2.2 An Algorithm for Minimize the Processing Capacity**

In this algorithm, we used  $c_k$  for  $k = 1, 2, \dots, |\mathcal{V}|$  to define the processing capacity of  $k$ -th VM, the memory and computational power of the PM that runs every VM can be used to calculate processing capacity.

**Algorithm 2** Minimizing the processing capacity algorithm.

**Result:** Minimizing the running power of VMs

initialization

```

for  $k \leftarrow 1$  to  $|\mathcal{V}|$  do
  Sum( $c_k$ )
end
for  $i \leftarrow 1$  to  $|S|$  do
  for  $j \leftarrow 1$  to  $|s_i| - 1$  do
    Sum( $x_{i,j}^k * d_{i,j}$ )
  end
end
Sum( $c_k$ ) - Sum( $x_{i,j}^k * d_{i,j}$ )

```

In Algorithm 2, the notation  $S$  represents the sequence of the received service requests,  $s_i$  represents the VNF number  $j$  of the service request number  $i$ ,  $\mathcal{V}$  represents the set of VMs nodes, and  $d_{i,j}$  defines the required bandwidth to execute VNF number  $j$  of the service request number  $i$  of the received request.

*Time complexity analysis for Algorithm 2:*

With the goal of minimizing processing power, Algorithm 2 performs two steps. First, it calculates the total processing power of VMs, the complexity is  $O(|\mathcal{V}|)$ . Second, the algorithm calculates in turn the power needed to process VNF on each VM; the time complexity of this step is  $O(|S| * |s_i|)$ . Thus, the overall time complexity of Algorithm 2 is  $O(|\mathcal{V}| + |S| * |s_i|)$ .

**4.2.3 An Algorithm for Minimizing the Processing Delay****Algorithm 3** Minimizing the processing delay algorithms.

**Result:** Minimizing the processing delay for the arrival service requests on VMs

initialization

```

for  $k \leftarrow 1$  to  $|\mathcal{V}|$  do
  Sum( $x_{i,j}^k * q_{i,j}^k$ )
end
for  $i \leftarrow 1$  to  $|S|$  do
  for  $j \leftarrow 1$  to  $|s_i| - 1$  do
    Sum( $\alpha_{i,j}$ )
  end
end
Sum( $\alpha_{i,j}$ ) + Sum( $x_{i,j}^k * q_{i,j}^k$ )

```

In Algorithm 3,  $q_{i,j}^k$  denotes a processing delay that occurred for the  $j$ -th VNF of the service request  $s_i$  while the service request was being processed on the  $k$ -th VM. Similar to Algorithm 2, the complexity of Algorithm 3 is  $O(|\mathcal{V}|) + O(|S| * |s_i|)$  as well.

*Overall time complexity analysis of the proposed algorithm:* With the requirement to optimize all three constraints: minimizing line latency of every link and minimizing processing capacity for all of the VM in addition to minimizing processing latency at all VM, all of the three algorithms must be implemented in the programming process to schedule the VNFs. Hence, the overall time complexity of the proposed algorithm is  $O(|\mathcal{V}|^2 + |\mathcal{V}| + |S| * |s_i|)$ .

**5 Simulation and Experiment Results**

This section illustrates the evaluation process of the proposed algorithm against various well-known algorithms on different evaluation parameters for different environments such as homogeneous and heterogeneous data center.

**Table 2** Table parameters of algorithms

Parameter	Small	Medium	Large
Number of VMs	10	20	30
Number of requests	100	200	300
Number of VNFs	5	5	5

## 5.1 Setup

To execute the optimization procedure depicted in Fig. 5, the proposed method leverages the Pymoo framework [46], which incorporates NSGA-III. Pymoo was specifically chosen for its ability to visualize lower- and upper dimensional data and assess solution quality from NSGA-III through performance metrics [16]. Furthermore, Pymoo offers multiple multi-criteria decision-making tools that can be applied post-convergence of NSGA-III to the Pareto front, facilitating the selection of the optimal solution in Fig. 5. In addition, it supports the design and development of custom problems and algorithms. For the proposed method, the formulated multi-objective optimization problem (equations in Sect. 3) has been written in Python using the Pymoo framework. The NSGA-III algorithm and compare the results with heuristic-based algorithms. The utilized dataset in this work was generated according to actual scenarios deploying virtual network functions on the network and is attached to the article. To ensure the fairness of the results, the parameters are set to be the same between algorithms and are shown in Table 2. Our simulations are performed on machines with a 3.3 GHz CPU (Intel Xeon W-1250) and 16 GB RAM.

## 5.2 Dataset

The evaluation of the proposed method considers three scenarios which are small, medium, and large networks. In the experiments, it is assumed that different network services need a set of VNFs for simplicity and generality (in our test, each network service would require 5 VNFs). Since solving the scheduling problem is achievable, then, the proposed method assumed that each VNF can be processed at least by one VM node and that each VM node is capable of serving 2 to 3 VNFs. VNFs are randomly integrated into NFV nodes at all three network sizes; thus, for each network service, the number of VNFs is a random integer between 2 and 5. The processing time for each VNF is a random integer between 1 and 5. In the proposed experiment, the population size was set to 50, and the number of repetitions was set to 300. Thus, it would take 15,000 iterations to get the best solution.

The capacity matrix of VNFs (small)

$$= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

## 5.3 Experimental Results and Discussion

The effectiveness of the proposed adaptation of the NSGA-III optimization algorithm for VNF placement is demonstrated through experimental evaluation, surpassing that of recent meta-heuristic methods and conventional heuristic approaches. The evaluation was performed on networks of diverse scales, encompassing small, medium, and large networks. Minimizing the line latency of the network link, diminishing the processing capacity of each VM, and decreasing the processing latency of all VMs comprised the evaluation criteria.

The proposed method is evaluated using several points. First, the proposed method VNF-NSGA-III was compared against a set of basic heuristics over three different problem sizes. The results of this comparison are listed in Table 3. Table 3 results demonstrate that the proposed optimization method outperforms all the other heuristics in terms of the objective values. The performance gap on the objective value increases as the problem size increases. In other words, the performance of the heuristic algorithms degrades as the problem size increases. On the other hand, the proposed method's performance is stable relative to the problem size. For instance, for the small-size network size, the achieved improvements of the proposed method were 103%, 59%, and 44% in comparison to the First-Fit, Next-Fit, and Random-Fit algorithms, respectively. For the medium-size network, the achieved improvements of the proposed method were 288%, 127%, and 152% relative to the First-Fit, Next-Fit, and Random-Fit algorithms, respectively. Finally, the performance gap for the large-size network was similar to the other small and medium-sized networks.

In scenarios involving small-sized networks, the proposed VNF-NSGA-III algorithm that was modified substantially enhanced all of the three objectives. For example, the proposed VNF-NSGA-III method's resulted in a reduction of line latency (Obj-1) by approximately 56.83%, from 966.00 in the First-Fit method to 417.00. In a similar vein, the reduction in processing capacity (Obj-2) and processing latency (Obj-3) signifies respective enhancements of 34.11% and

48.83%. Notably, the fitness value, which represents a synthesis of all objectives, exhibited a substantial increase of 50.84% in performance, rising from 902.17 to 443.77.

In scenarios involving medium-sized networks, the proposed VNF-NSGA-III method consistently outperformed the alternative methods. In comparison to the Next-Fit method, the algorithm reduced line latency (Obj-1) by more than 80%, demonstrating its scalability and efficacy in more complex network environments.

When applied to scenarios involving large-scale networks, the benefits of the proposed VNF-NSGA-III method became even more evident. In contrast to the First-Fit method, the line latency (Obj-1) was decreased by approximately 81.05 percent, and the processing latency (Obj-3) was enhanced by more than 768.33 percent. The significant decrease in the Fitness value from 13222.19 to 2855.74, representing an enhancement of 78.41%, highlights the algorithm's ability to withstand large-scale VNF deployments.

The execution durations for all network sizes suggest that although the proposed VNF-NSGA-III method necessitates a greater computational effort, the quality improvements in optimization more than compensate for the additional time commitment. The efficacy of the algorithm validates its viability for practical implementation in VNF placement assignments.

While Table 3 shows the three objectives combined in one value, the three objective values are depicted separately in Fig. 6 and the combined objective value as well. Figure 6 clearly emphasizes that the proposed VNF-NSGA-III method yielded the best results for each objective regardless of the network size. Of note, the best-performing heuristic was the First-Fit.

### 5.3.1 Analysis of Algorithm Performance with Varying VNF Requests

The second point of comparison is to understand the effect of the number of requests, the number of GA generations, and the population size on the objectives' values of the proposed methods. This effect of the number of requests on the three objective values is depicted in Fig. 7.

The proposed VNF-NSGA-III method's execution results in relation to the fluctuating number of VNF requests are depicted in Fig. 7. Each of the performance metrics illustrated in Fig. 7a through Fig. 7d was assessed during the experimental phase.

The fitness function value is illustrated in Fig. 7(a), demonstrating a non-linear rise with the augmentation of service requests. The curve indicates that the objective function cost of the algorithm increases in parallel with the network traffic, which signifies a corresponding rise in resource utilization or decline in performance efficiency.

Figure 7b illustrates the delay in transmission. The utilization of this metric is critical when evaluating the network's real-time capabilities. The exponential growth pattern depicted in the graph indicates that the network encounters escalating delays in response to a surge in service requests. This surge may be a result of increased traffic congestion or processing times.

Figure 7c illustrates the required total processing capacity of the VNFs. There is a direct correlation between the number of service requests and the required processing resources, as indicated by the upward trend. The graph illustrates a comparatively linear correlation until a specific point, at which point the capacity reaches a plateau. This plateau may suggest the algorithm has efficiently allocated resources or has reached a state of resource saturation.

Lastly, Fig. 7d depicts the Processing Delay Time. As the quantity of service requests increases, the curve displays a noticeable upward slope. The observed pattern suggests that the quantity of VNF requests substantially influences the processing latency. This can be attributed to various factors, including computational tension and queuing delays within the virtualized environment.

The aggregate of these outcomes serves as evidence of the algorithm's capacity to scale and adapt to fluctuating network requirements. The analysis of the graphs' development patterns offers valuable insights into the algorithm's performance across various load conditions. These patterns draw attention to potential optimization areas, specifically in the domains of transmission delay management and processing capacity under scenarios of high demand. Then, the effect of the number of generations and the population size on the proposed method performance is depicted in Figs. 8 and 9, respectively.

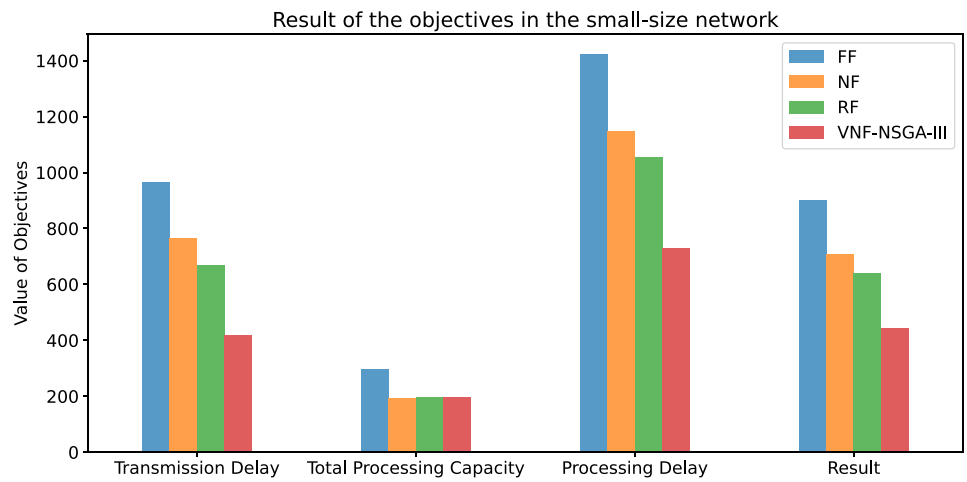
### 5.3.2 Performance Evaluation over Different number of Generations

The execution results of the proposed VNF-NSGA-III method as the number of generations is altered are illustrated in Fig. 8. Every subfigure ranging from Fig. 8a to Fig. 8d illustrates one particular aspect of the performance of the algorithm as the evolutionary process advances.

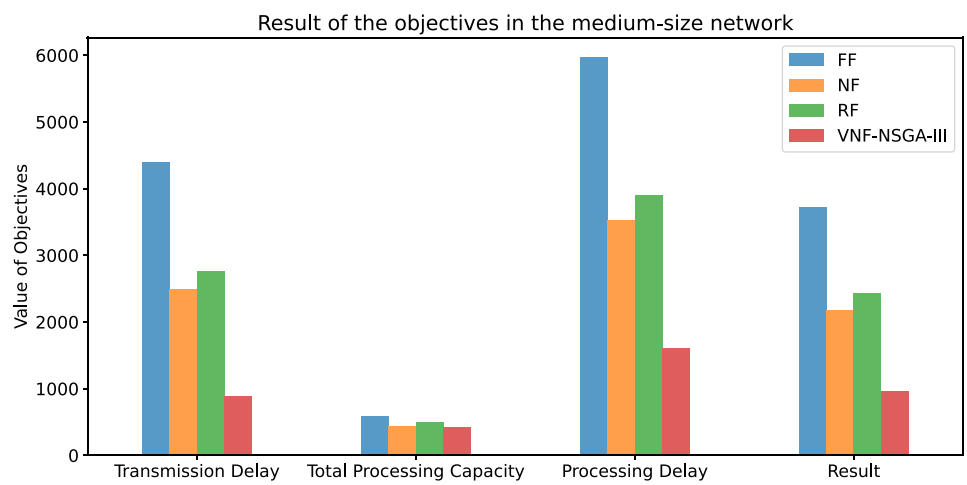
The fitness function value is illustrated in Fig. 8a across generations. The graph illustrates an initial significant decrease, followed by a gradual recovery of stability as the generational count rises. This implies that the algorithm rapidly locates a solution that is nearly optimal, and subsequent iterations make only minor adjustments to the solution.

For the same number of generations, the Transmission Delay Time is depicted in Fig. 8b. A substantial reduction in the metric during the initial phase signifies the expeditious optimization of transmission latencies. Nevertheless, as time passes, the rate of descent decreases, which suggests that the

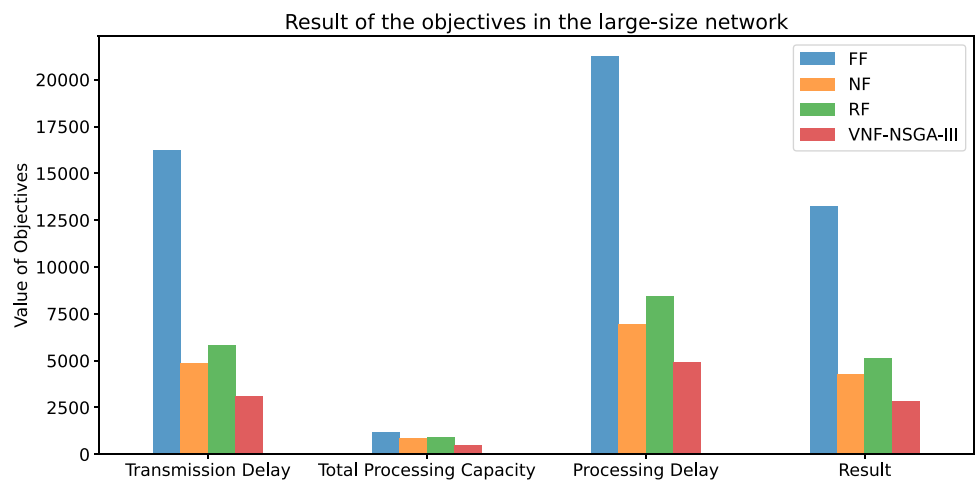
**Fig. 6** A comparison of the three objectives' values between the proposed algorithm and other algorithms



(a) Small-size network



(b) Medium-size network



(c) Large-size network



**Table 3** Simulation results

Network size	Result	First-Fit	Next-Fit	Random-Fit	VNF-NSGA-III
Small-size	Obj_1	966.00	766.00	668.00	<b>417.00</b>
	Obj_2	296.25	<b>191.25</b>	195.25	195.25
	Obj_3	1423.00	1146.00	1053.00	<b>728.00</b>
	Fitness	902.17	707.57	641.67	<b>443.77</b>
	Times (s)	0.02	0.02	<b>0.01</b>	198.55
Medium-size	Obj_1	4404.00	2487.00	2766.00	<b>882.00</b>
	Obj_2	592.13	438.63	505.13	<b>417.63</b>
	Obj_3	5968.00	3522.00	3899.00	<b>1605.00</b>
	Fitness	3729.64	2182.99	2427.64	<b>959.59</b>
	Times (s)	0.06	0.06	<b>0.04</b>	566.89
Large-size	Obj_1	16232.00	4857.00	5833.00	<b>3085.00</b>
	Obj_2	1185.31	865.81	926.31	<b>505.81</b>
	Obj_3	21246.00	6960.00	8422.00	<b>4900.00</b>
	Fitness	13222.19	4290.54	5137.69	<b>2855.74</b>
	Times (s)	0.22	0.22	<b>0.12</b>	2567.98

optimization benefits obtained from subsequent generations will diminish.

The Total Processing Capacity utilized by the algorithm is illustrated in Fig. 8c. Similar to the preceding metrics, an initial sharp decline is observed. As the algorithm approaches an optimal solution, this decrease gradually diminishes, resulting in negligible fluctuations in the processing capacity demands of subsequent generations.

Lastly, the processing delay time as a function of the number of generations is illustrated in Fig. 8d. Consistent with the other subfigures of Fig. 8, where substantial initial gains in delay reduction are succeeded by a plateau effect, the downward trend continues. As depicted in the graph, while the algorithm optimizes processing delays rapidly, little progress is made after numerous generations.

The trends identified in these plots serve as an indication of the algorithm's optimization efficacy. The exponential growth observed in the initial iterations serves as evidence of the algorithm's ability to identify viable solutions rapidly. Nevertheless, as the evolutionary process progresses, the incremental benefits decrease. This characteristic occurs when evolutionary algorithms approach the optimal solution space.

The results of this study highlight the significance of incorporating the number of generations into the algorithm's execution in order to strike a balance between solution quality and computational time. Furthermore, the diminishing improvements indicate the possibility of a threshold beyond which further generations fail to produce substantial enhancements in performance.

### 5.3.3 Impact of Population Size on the Proposed Algorithm's Performance

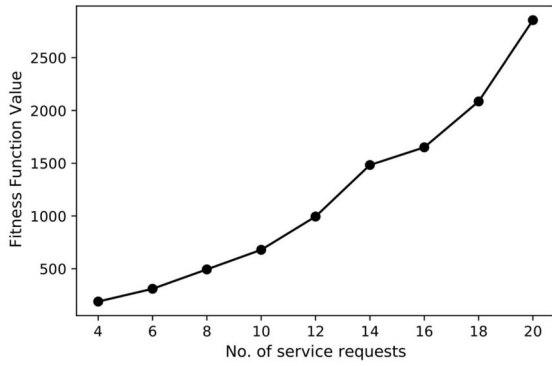
The impact of different population sizes on the performance of the algorithm is illustrated in Fig. 9. The performance metrics for these metrics are as follows: Processing Delay Time, Fitness Function Value, Transmission Delay Time, and Total Processing Capacity.

The fitness function value is illustrated in Fig. 9(a), demonstrating that it fluctuates in response to an increase in population size. The graph's varying characteristics indicate that although larger populations might provide a greater variety of solutions, they also introduce fitness variability as a result of a more expansive search space.

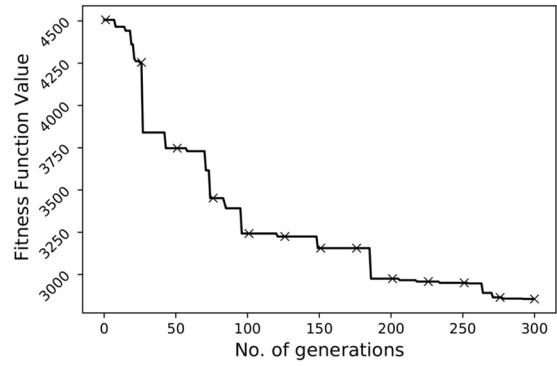
Figure 9b presents the transmission delay time, emphasizing a consistent pattern of greater delay as the population size increases. This may be the result of the algorithm dedicating more time to assessing an increased quantity of potential solutions, consequently resulting in extended transmission durations.

The proposed algorithm's total processing capacity is illustrated in Fig. 9c, which illustrates a non-linear correlation with the overall size of the population. The data from the graph shows that the processing capacity fails to demonstrate a consistent upward or downward trend as the population size changes. This indicates that the capacity requirement of the algorithm is significantly based on the particular population under evaluation.

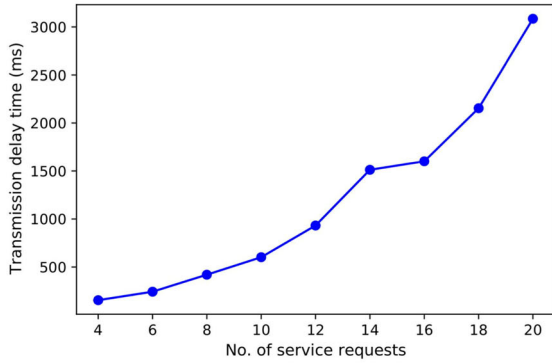
As the population size increases, the processing delay time is depicted in Fig. 9d, which concludes the illustration. The observed escalation in processing delays is consistent with the hypothesis that larger populations necessitate greater computational resources and time.



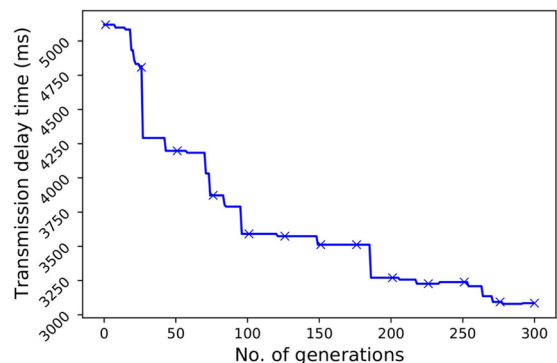
(a)



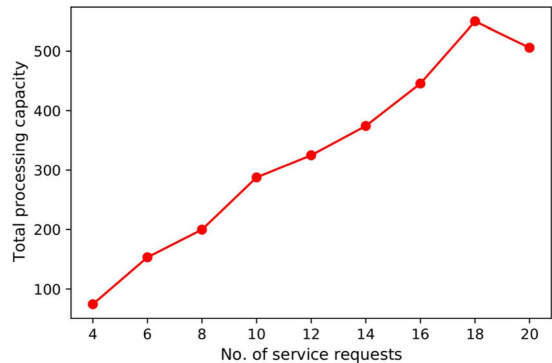
(a)



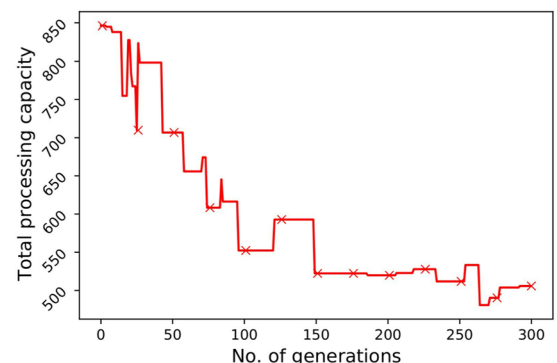
(b)



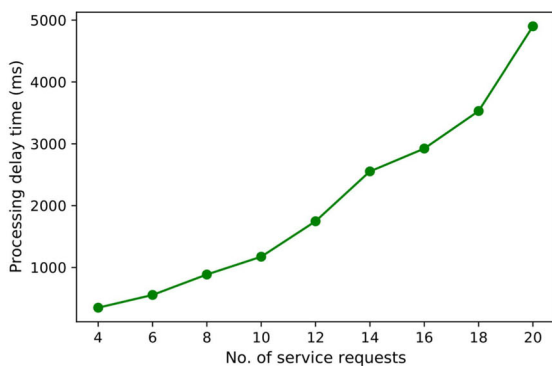
(b)



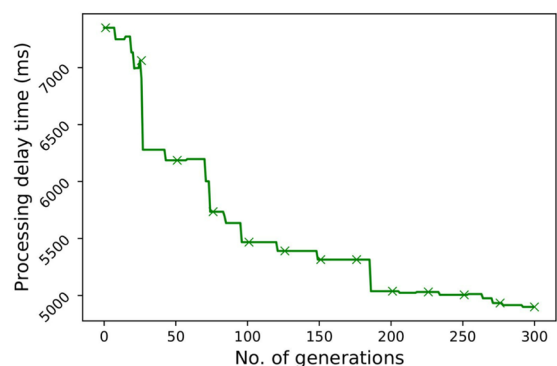
(c)



(c)



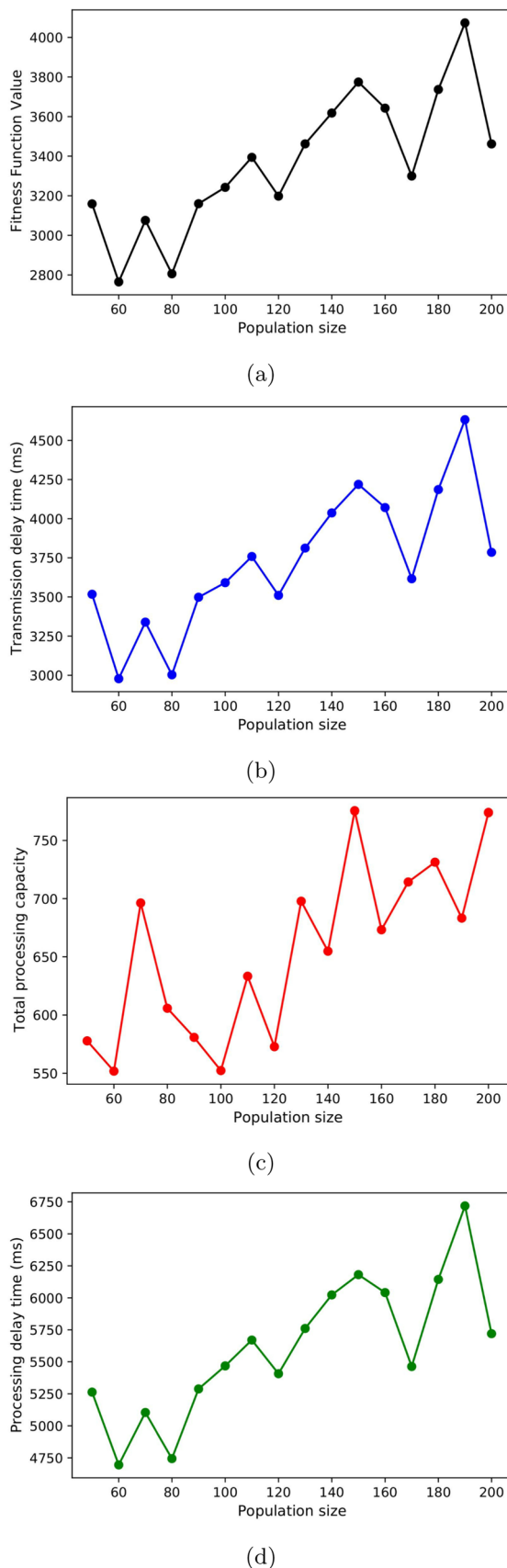
(d)



(d)

**Fig. 7** The algorithm execution results when changing the number of VNF requests

**Fig. 8** The proposed algorithm results on different number of generations



**Fig. 9** The results of the proposed algorithm on different population sizes

Collectively, these results illustrate the complicated impact that the size of the population has on the efficacy of algorithms. Although larger populations can facilitate a more comprehensive solution for space exploration, they may also result in longer processing periods and a greater demand for computational resources. It is essential to strike a balance between population size and computational efficiency in order to maximize search effectiveness.

The outcomes demonstrate that the proposed model is capable of effectively addressing the multi-objective optimization problem of VNF placement. The proposed VNF-NSGA-III algorithm, which has been utilized to include improved decision-making heuristics, presents a potentially effective strategy for handling the complex requirements of virtualized networks of the next generation. Further research could be devoted to optimizing the algorithm in an effort to further decrease execution times without compromising or enhancing optimization results.

## 6 Conclusions

In this work, the complex problem of virtual network function placement was addressed, which is crucial for the operation of network services in resource-constrained environments. The novelty of our approach lies in the formulation of VNF placement as a multi-objective optimization problem, aiming to minimize the processing capacity required by virtual machines, the latency across network links, and the processing latency of VMs. This formulation captures the essence of network efficiency and is a significant leap from traditional methods that target limited objectives. The VNF-NSGA-III algorithm, which is a modified version of the widely recognized NSGA-III algorithm, has been validated through a set of practical experiments. The performance evaluation demonstrates that VNF-NSGA-III outperforms various heuristic and contemporary meta-heuristic approaches. The results demonstrate the effectiveness of VNF-NSGA-III in achieving a well-balanced optimization across all of the three objectives, thus offering a comprehensive solution to the VNF placement challenge.

In the future, investigating the use of dynamic parameterization in the NSGA-III framework shows potential for further research. This has the potential to facilitate the creation of highly flexible algorithms that can effectively adapt to immediate changes in network conditions. Furthermore, there is potential to expand the parallelization of the NSGA-III algorithm. These advancements have the potential to greatly decrease the amount of time it takes to perform computations, thus increasing the practicality of the algorithm for large-scale network scenarios. Furthermore, the techniques suggested in this research can be extended to tackle additional multi-objective optimization issues that are inherent in

network function virtualization (NFV). Expanding the range of use, the principles of VNF-NSGA-III can be utilized to optimize different aspects of NFV infrastructure, thereby enhancing the resilience and flexibility of modern networks.

**Acknowledgements** We express our gratitude to the anonymous reviewers for their invaluable input on the quality of the work and their constructive suggestions for enhancing the presentation.

**Author Contributions** Methodology preparation and implementation were performed by PT, FW, MB, and AF. Data collection and analysis were performed by PT, FW, MB, and AF, and AS. The first draft of the manuscript was written by PT, AF, and AS. All the authors participated in writing and reviewing the final version of the manuscript. All the authors read and approved the final manuscript.

**Funding** This work is partly funded by the Program of the National Natural Science Foundation of China (Grant no. 62172151).

**Availability of data and materials** The data used in this work are available through a reasonable request from the corresponding author.

## Declarations

**Conflict of interest** The authors declare no competing interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. *Comput. Netw.* **54**(5), 862–876 (2010)
2. Promwongsa, N., Abu-Lebdeh, M., Kianpisheh, S., Belqasmi, F., Glietho, R.H., Elbiaze, H., Crespi, N., Alfandi, O.: Ensuring reliability and low cost when using a parallel vnf processing approach to embed delay-constrained slices. *IEEE Trans. Netw. Service Manag.* **17**(4), 2226–2241 (2020)
3. Miyamura, T., Misawa, A.: Joint optimization of optical path provisioning and vnf placement in vcdn. *Opt. Switching Netw.* **49**, 100740 (2023)
4. Fang, J., Zhao, G., Xu, H., Tu, H., Wang, H.: Reveal: Robustness-aware vnf placement and request scheduling in edge clouds. *Computer Networks*, 109882 (2023)
5. HARA, T., SASABE, M., SUGIHARA, K., KASAHARA, S.: Resource-efficient and availability-aware service chaining and vnf placement with vnf diversity and redundancy. *IEICE Transactions on Communications* (2023)
6. Gholipoor, N., Saeedi, H., Mokari, N., Jorswieck, E.A.: E2e qos guarantee for the tactile internet via joint nfv and radio resource allocation. *IEEE Trans. Netw. Service Manag.* **17**(3), 1788–1804 (2020)
7. Li, J., Shi, W., Yang, P., Shen, X.: On dynamic mapping and scheduling of service function chains in sdn/nfv-enabled networks. In: 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2019). IEEE
8. Qu, L., Assi, C., Shaban, K.: Delay-aware scheduling and resource optimization with network function virtualization. *IEEE Trans. Commun.* **64**(9), 3746–3758 (2016)
9. Luizelli, M.C., Raz, D., Sa'ar, Y.: Optimizing nfv chain deployment through minimizing the cost of virtual switching. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 2150–2158 (2018). IEEE
10. Khebbache, S., Hadji, M., Zeghlache, D.: A multi-objective non-dominated sorting genetic algorithm for vnf chains placement. In: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1–4 (2018). IEEE
11. Leivadreas, A., Kesidis, G., Ibnkahla, M., Lambadaris, I.: Vnf placement optimization at the edge and cloud. *Future Internet* **11**(3), 69 (2019)
12. Saber, S., Salem, S.: An improved light spectrum optimizer for parameter identification of triple-diode pv model (2023)
13. Fathalla, A., Li, K., Salah, A.: Best-kff: a multi-objective preemptive resource allocation policy for cloud computing systems. *Cluster Comput* **25**(1), 321–336 (2022)
14. Bekhit, M., Fathalla, A., Eldesouky, E., Salah, A.: Multi-objective vnf placement optimization with nsga-iii. In: International Conference on Advances in Computing Research, pp. 481–493 (2023). Springer
15. El-Ashmawi, W.H., Salah, A., Bekhit, M., Xiao, G., Al Ruqeishi, K., Fathalla, A.: An adaptive jellyfish search algorithm for packing items with conflict. *Mathematics* **11**(14), 3219 (2023)
16. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2013)
17. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3045–3052 (2016). IEEE
18. Ciro, G.C., Dugardin, F., Yalaoui, F., Kelly, R.: A nsga-ii and nsga-iii comparison for solving an open shop scheduling problem with resource constraints. *IFAC-PapersOnLine* **49**(12), 1272–1277 (2016)
19. Wangsom, P., Lavangananda, K., Bouvry, P.: The application of nondominated sorting genetic algorithm (nsga-iii) for scientific-workflow scheduling on cloud. In: The 8th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2017), pp. 269–287 (2017)
20. Tavana, M., Li, Z., Mobin, M., Komaki, M., Teymourian, E.: Multi-objective control chart design optimization using nsga-iii and mopso enhanced with dea and topsis. *Expert Syst. Appl.* **50**, 17–39 (2016)
21. Bi, X., Wang, C.: An improved nsga-iii algorithm based on objective space decomposition for many-objective optimization. *Soft Comput.* **21**(15), 4269–4296 (2017)
22. Nita, M.-C., Pop, F., Voicu, C., Dobre, C., Xhafa, F.: Momth: multi-objective scheduling algorithm of many tasks in hadoop. *Cluster Comput.* **18**(3), 1011–1024 (2015)
23. Ruiz, L., Barroso, R.J.D., De Miguel, I., Merayo, N., Aguado, J.C., De La Rosa, R., Fernández, P., Lorenzo, R.M., Abril, E.J.: Genetic algorithm for holistic vnf-mapping and virtual topology design. *IEEE Access* **8**, 55893–55904 (2020)
24. Martín-Pérez, J., Bernardos, C.J.: Multi-domain vnf mapping algorithms. In: 2018 IEEE International Symposium on Broadband

- Multimedia Systems and Broadcasting (BMSB), pp. 1–6 (2018). IEEE
25. Cao, J., Zhang, Y., An, W., Chen, X., Sun, J., Han, Y.: Vnf-fg design and vnf placement for 5g mobile networks. *Sci. China Inform. Sci.* **60**, 1–15 (2017)
  26. Yue, Y., Cheng, B., Wang, M., Li, B., Liu, X., Chen, J.: Throughput optimization and delay guarantee vnf placement for mapping sfc requests in nfv-enabled networks. *IEEE Trans. Netw. Service Manag.* **18**(4), 4247–4262 (2021)
  27. Oljira, D.B., Grinnemo, K.-J., Taheri, J., Brunstrom, A.: A model for qos-aware vnf placement and provisioning. In: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 1–7 (2017). IEEE
  28. Woldeyohannes, Y.T., Mohammadkhan, A., Ramakrishnan, K., Jiang, Y.: Cluspr: Balancing multiple objectives at scale for nfv resource allocation. *IEEE Trans. Netw. Service Manag.* **15**(4), 1307–1321 (2018)
  29. Liu, J., Lu, W., Zhou, F., Lu, P., Zhu, Z.: On dynamic service function chain deployment and readjustment. *IEEE Trans. Netw. Service Manag.* **14**(3), 543–553 (2017)
  30. Jin, P., Fei, X., Zhang, Q., Liu, F., Li, B.: Latency-aware vnf chain deployment with efficient resource reuse at network edge. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 267–276 (2020). IEEE
  31. Ren, H., Xu, Z., Liang, W., Xia, Q., Zhou, P., Rana, O.F., Galis, A., Wu, G.: Efficient algorithms for delay-aware nfv-enabled multicasting in mobile edge clouds with resource sharing. *IEEE Trans. Parallel Distributed Syst.* **31**(9), 2050–2066 (2020)
  32. Ebrahimzadeh, A., Promwongsa, N., Afrasiabi, S.N., Mouradian, C., Li, W., Recse, Á., Szabó, R., Glitho, R.H.: h-horizon sequential look-ahead greedy algorithm for vnf-fg embedding. In: 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 41–46 (2021). IEEE
  33. Assi, C., Ayoubi, S., El Khoury, N., Qu, L.: Energy-aware mapping and scheduling of network flows with deadlines on vnfs. *IEEE Trans. Green Commun. Netw.* **3**(1), 192–204 (2018)
  34. Maity, I., Vu, T.X., Chatzinotas, S.: D-schedule: Dependency-aware vnf scheduling in satellite-terrestrial networks. In: 2023 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1283–1288 (2023). IEEE
  35. Li, J., Shi, W., Zhang, N., Shen, X.: Delay-aware vnf scheduling: a reinforcement learning approach with variable action set. *IEEE Trans. Cognit. Commun. Netw.* **7**(1), 304–318 (2020)
  36. Akbari, M., Abedi, M.R., Joda, R., Pourghasemian, M., Mokari, N., Erol-Kantarci, M.: Age of information aware vnf scheduling in industrial iot using deep reinforcement learning. *IEEE J. Selected Areas Commun.* **39**(8), 2487–2500 (2021)
  37. Li, J., Shi, W., Zhang, N., Shen, X.S.: Reinforcement learning based vnf scheduling with end-to-end delay guarantee. In: 2019 IEEE/CIC International Conference on Communications in China (ICCC), pp. 572–577 (2019). IEEE
  38. Xiang, Y., Yang, X., Huang, H., Wang, J.: Balancing constraints and objectives by considering problem types in constrained multi-objective optimization. *IEEE Transactions on Cybernetics* (2021)
  39. Shi, J., Wang, J., Huang, H., Shen, L., Zhang, J., Xu, H.: Joint optimization of stateful vnf placement and routing scheduling in software-defined networks. In: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCLOUD/SocialCom/SustainCom), pp. 9–14 (2018). IEEE
  40. Zhang, Y., He, F., Sato, T., Oki, E.: Network service scheduling with resource sharing and preemption. *IEEE Trans. Netw. Service Manag.* **17**(2), 764–778 (2019)
  41. Schneider, S., Dräxler, S., Karl, H.: Trade-offs in dynamic resource allocation in network function virtualization. In: 2018 IEEE Globecom Workshops (GC Wkshps), pp. 1–3 (2018). IEEE
  42. Gamal, M., Abolhasan, M., Lipman, J., Ni, W.: Mapping and scheduling of virtual network functions using multi objective optimization algorithm. In: 2019 19th International Symposium on Communications and Information Technologies (ISCIT), pp. 328–333 (2019). IEEE
  43. Mosaiyebzadeh, F.: Energy-efficient virtual network function placement based on metaheuristic approaches. PhD thesis, Universidade de São Paulo
  44. Salem, S.: An improved binary quadratic interpolation optimization for 0-1 knapsack problems (2023)
  45. Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., Davy, S.: Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–9 (2015). IEEE
  46. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. *IEEE Access* **8**, 89497–89509 (2020)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.