Virtual network function-forwarding graph embedding: A genetic algorithm approach

Quang Tran Anh Pham, Jean-Michel Sanner, Cédric Morin, Yassine Hadjadj-Aoul

Virtual Network Function Forwarding Graph Embedding: A genetic algorithm approach

Pham Tran Anh Quang*, Jean-Michel Sanner*, Cédric Morin*, Yassine Hadjadj-Aoul*
*B-COM, Rennes, France
†Orange Labs R&D, Rennes, France
‡Univ Rennes, Inria, CNRS, IRISA, France

Abstract

Network Function Virtualization (NFV) provides a simple and effective mean to deploy and manage network and telecommunications' services. A typical service can be expressed in the form of a Virtual Network Function - Forwarding Graph (VNF-FG). Allocating a VNF-FG is equivalent to place VNFs and virtual links onto a given substrate network considering resources and quality of service (QoS) constraints. The deployment of VNF-FGs in large-scale networks, such that QoS measures and deployment cost are optimized, is an emerging challenge. Single-objective VNF-FGs allocation has been addressed in existing literature; however, there is still a lack of studies considering multi-objective VNF-FGs allocation. In addition, it is not trivial to obtain optimal VNF-FGs allocation due to its high computational complexity even in case of single-objective VNF-FGs allocation. Genetic algorithms (GAs) have been proved its ability in coping with multi-objective optimization problems, thus we propose a GA-based scheme to solve multi-objective VNF-FGs allocation problem in this paper. The numerical results confirm that the proposed scheme can provide near Pareto-optimal solutions within a short execution time.

Index Terms

Network Function Virtualization, Service Function chain, Genetic algorithms, Routing.

I. Introduction

Network Function Virtualization (NFV) is able to cut down the expenses of deploying and operating large network infrastructures by providing the capabilities to migrate network functions from dedicated hardware appliances to general purpose computing, storage, and networking solutions. It also promises remarkable benefits to the 5G network architecture through enabling flexible and scalable provisioning of novel applications and network services. In addition to the possibilities of enabling multiple services to be deployed on the same physical infrastructure, it unveils new business models and/or economies of scale. Particularly, the Network–as–a–Service (NaaS) business model is anticipated to play a decisive role in 5G mobile networks by allowing Mobile Network Operators (MNOs) to exploit new revenue streams. Indeed, virtualization enables MNOs to abstract their physical network infrastructures into specific service slices operated by one or multiple mobile virtual network operators (MVNOs) or over the top (OTT) providers. The expected vertical applications range from high–definition video delivery to machine–to–machine applications. Thanks to virtualization and adaptive network service orchestration, Infrastructure Providers (InPs) are able to cope with various requirements that characterize future applications and services.

European Telecommunications Standards Institute (ETSI) is de-facto the reference Standards Developing Organizations (SDOs) for the NFV high level functional architecture specification. ETSI specifies three components for the NFV architecture: Virtual Network Functions (VNFs); NFV Infrastructure (NFVI), including the elements needed to run VNFs, such as the hypervisor node and the virtualization clusters; and MANagement and Orchestration (MANO), executing necessary operations to run, migrate, optimize VNF nodes and chains, possibly in coordination with transport network orchestrator.

In recent years, there are a vast number of studies on VNF placement, virtual network embedding (VNE), and component placement. Most of them deal with solving the problem of mapping input virtual network requests (VNRs), which are often converted to a VNF-Forwarding Graph (VNF-FG), onto a substrate networks. To successfully embed a VNF-FG, the placement of VNFs and virtual link have to satisfy multiple QoS metrics, e.g. latency, loss rate, etc, as well as minimize the deployment cost. This can be described as a multi-objective optimization problem (MOP). Artificial Intelligent (AI) techniques are able to deal with high complexity of MOPs via evolutionary algorithms. AI-optimization techniques, e.g. Genetic Algorithms (GAs) and Swarm Intelligence, can achieve out-standing performance and have been exploited in coping with MOPs [1]. However, existing algorithms based on GAs and on artificial bee colony optimization techniques have a rather high execution time, therefore not adapted with an on-line VNF-FGs allocation. In this work, we propose a GA-based approach to cope with the multi-objective VNF-FGs embedding. We encode the problem into chromosomes, which include bits representing the mapping between the VNF-FG and the substrate network. Then, we break the original problem into VNF placement problem and routing problem. For each VNF placement configuration, a set of non-dominated paths connecting the hosts of VNFs is identified by solving routing problem. Crossover and mutation operators on individuals help to explore possibilities of VNF placement.

The rest of this paper is structured as follows. A brief discussion of existing work is provided in Section II. Section III presents an overview of the system. A formulation of multi-objective VNF-FGs allocation can be found in Section IV. An

overview of the proposed genetic algorithm framework is presented in Section V. It is followed by detailed descriptions of proposed scheme in Section VI and Section VII. Section VIII presents the performance evaluation of the proposed scheme. Finally, the conclusion and future work can be found in Section IX

II. Related work

A. Virtualization

Virtualization unveils challenges, which need further attention from both academia and industry. The first challenge is VNF placement problem. The goal of this problem is to identify the physical hosts of VNFs in order to optimize a given set of metrics. VNFs can be placed in remote data centers or in single servers or clusters of servers. To reduce the cost of deployment, VNFs are usually placed in remote data center. It may nevertheless typically increase the latency and cause congestions because of the fix and often remote location of data centers. Deploying new micro data centers inside the network can alleviate the aforementioned problem [2]. Recent studies in VNF placement focused on efficient virtual function deployment and scalability. A number of virtual machine placement schemes have been proposed in order to efficiently deploy VNFs under cost and QoS constraints [3].

Virtual Network Embedding (VNE) is more complicated than VNF placement problem. In VNE, the objective is to efficiently map VNRs (or VNF-FGs) onto a substrate network. Each VNF-FG comprises of multiple VNFs connected in a given order. Due to its complexity [4], recent researches [5]–[11] focus on designing efficient algorithms for optimal VNF-FGs deployment. While the authors in [5]–[10] proposed heuristic algorithms for placing chains of VNFs (VNF-FGs), the authors of [11] adopts genetic algorithms to solve the problem. However, the common and major drawback of existing algorithms is the high computational complexity, thus limiting deployment in practice. For instance, the calculation time presented in [11] are from 100 s to 700 s even in moderate complex networks (20 to 100 nodes).

B. Genetic algorithms

The versatility of evolutionary algorithms has been at the origin of their use in different types of problems, including the resolution of multi-objective optimization problems [12]. Despite their proven ability to find multiple non-dominated solutions to several classical problems, it is essential to bring in scalable algorithms that have a faster rate of convergence.

Through elitism, genetic algorithms make it possible to have guarantees of non-degradation of the solutions while allowing a better convergence [13]. Indeed, elitism makes it possible to preserve the best individuals by replicating them in the next generation. The best-known elitist multiobjective evolutionary algorithm (MOEAs) is Non-dominated Sorting Genetic Algorithm II (NSGA-II) [14] which outperforms other elitist MOEAs. However, when it comes to high number of objectives (more than 3 objectives), a number of MOEAs is unable to find well-converged and diversified non-dominated solutions because of the loss of selection pressure in fitness evaluation [15]. Non-dominated Sorting Genetic Algorithm III (NSGA-III) [16] was introduced to address this challenge. Except for selection mechanism, other fundamental components of NSGA-III are similar to NSGA-II. The new population in NSGA-II is selected by a combination of non-dominated sorting and crowding-distance sorting, while NSGA-III selects the new population based on supplied reference points. These reference points can be identified by using the proposed procedure in [17]. NSGA-III have been demonstrated to work well from three to 15-objective of the problem presented in [18]. A variant of NSGA-III, named unified-NSGA-III [19], has been introduced to provide an evolutionary algorithm that is able to cope with mono-, multi- and many objective optimization problems. Another variant of NSGA-III, EliteNSGA-III [20], stores elite population and modifies the parent selection mechanism to enhance diversity. However, the authors [21] showed that NSGA-III does not always outperform NSGA-II. Due to the diversity of MOEAs, it is necessary to design a scheme that can be adopted by any MOEA. The proposed mechanism is able to work with any MOEA.

III. System Overview

We consider ETSI NFV MANO [22] model as a reference for the practical implementation of the proposed algorithms. MANO is composed of three types of elements: the Virtualized Infrastructure Manager (VIM), the VNF Manager (VNFM), and the NFV Orchestrator (NFVO). The architecture may have multiple VIMs or VNFMs, but only one NFVO.

A VIM is responsible of the management of the resources of an NFV infrastructure (NFVI). Typically, an NFVI is composed of a data center, or a set of data centers strongly interconnected. In order to interconnect different NFVIs managed by the VIMs, it is necessary to introduce another manager: the Wide Area Network (WAN) Infrastructure Manager (WIM). A WIM can be seen as a VIM that only manages network resources. VIMs and WIMs abstract the resources they are responsible for, and expose this abstraction to their northbound interfaces, accessed by the VNFMs and the NFVO. The management of the resources includes the discovery of these resources, their allocation, their release, the monitoring activities and the fault management.

The VNFMs are responsible of the VNFs' life cycle managements. VNFMs do not request resources directly: they entirely depends on the NFVO. As a consequence, they are not responsible for the placement of the VNFs in the network, and do not need to implement our algorithm.

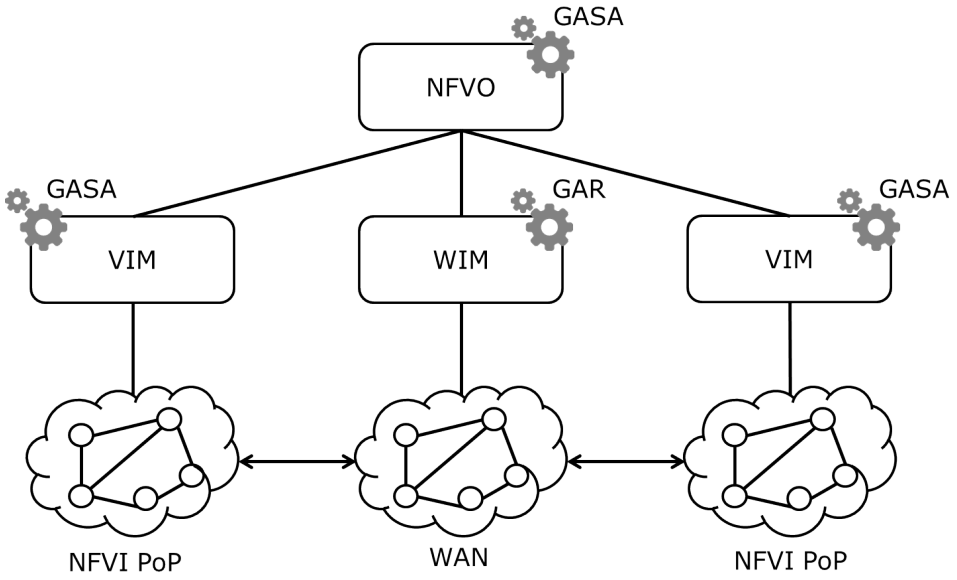


Fig. 1: Localization of GAVA and GAR in NFV MANO framework

Notation	Description
w_i^e	i^{th} measure of link e
$\Gamma_i^{e'}$	Requirement of i^{th} measure of virtual link e'
$\Psi_i^{n'}$	Requirement of i^{th} resource of VNF n'
v_i^n	Cost of resource i^{th} of substrate node n
z_i^e	Cost of resource i^{th} of substrate link e
r_k^n	Resource of k^{th} measure of substrate node n

TABLE I: Notations of QoS measures, resources, and deployment cost

The NFVO is the entry point of MANO for the users. It receives Network Services (NS) requests, and convert each of them to a set of interconnected VNFs called Service Function Chain (VNF-FGs). Based on the abstracted topologies provided by the VIMs and WIMs, it then selects where those VNFs should be placed, and how they should be interconnected. Once this placement is decided, the NFVO reserves the corresponding resources of the VIMs, and provides them to the VNFMs. The actual deployment of the VNFs is performed by the VNFMs. The NFVO is then responsible of the life cycle management of the NS, and offers monitoring information at its northbound interface.

NFVO is responsible for the deployment of VNF graph over the different VIMs, which is the typical action performed by our Genetic Algorithm VNF-FGs Allocation (GAVA). In addition, it can also be used by the VIMs. Indeed, VIMs receive resource requests that have to be mapped to the real NFVI, which is technically very similar to the VNF-FGs allocation problem. Regarding to the WIMs, they can use the Genetic Algorithm Routing (GAR), the subprocess called by GAVA, which is responsible of the paths' computation. The final suggested system is represented Fig. 1.

IV. Problem formulation

A substrate network is modelled as a directed graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where $\mathcal{V}_s$ is the set of nodes and $\mathcal{E}_s$ is the set of directed links. A VNF-FG can be presented in the shape of a directed graph $\mathcal{G}_v = (\mathcal{V}_v, \mathcal{E}_v)$, where $\mathcal{V}_v$ is the set of VNFs and $\mathcal{E}_v$ is the set of directed virtual links connecting VNFs in the VNF-FGs. We introduce the binary variables $\Phi_e^{e'}$ and $\Omega_n^{n'}$ to indicate if a directed link e is used by virtual link e' and VNF n' is hosted by substrate node n, respectively. Resources of the substrate network could be classified into two types: network resources and compute and storage resources. Network resources comprise characteristics of links connecting the substrate nodes, while compute and storage resources are the computing power (Central processing unit (CPU), Random-access memory (RAM)) and the amount of storage that are available at substrate nodes. All notations used in this problem formulation is presented in Tab. I. First, we consider the constraints of network resources. Measures of link e are presented by a vector $\mathbf{w}^e = \{w_i^e, i = 1, ..., M\}$, where M is the number of measures. Note that the index e and (n, m) (where n and m are the source and destination of link e, respectively) are used interchangeably in this paper. It is able to classify these measures into following groups: additive, multiplicative, and min-max QoS. As shown in [23], it is able to convert multiplicative measures to additive measures by taking the logarithm. For min-max measures, a filter can be used to prune the network by removing the unqualified links [23]. Meanwhile, path computation with additive routing was proved to be a NP-complete problem [24], thus more difficult to cope with. Consequently, we focus on the scenarios where

the length of a path comprises additive measures in this paper. These measures could be QoS metrics (e.g. latency) or costs of link usage. In this paper, we consider latency and loss-rate. Each virtual link has specific requests of measures, denoted as $\Gamma_i^{e'}$. The QoS constraints of each virtual link can be formulated as

$$\sum_e \Phi_e^{e'} w_i^e \leq \Gamma_i^{e'}, \forall i, e'. \tag{1}$$

Next, the compute and storage resources are considered. Each substrate node n has K types of resources denoted as $1, ..., K$. The amount of resource k at substrate node n is denoted as r_k^n. In this paper, three type of resources are considered, i.e. CPU, RAM, and storage memory (hard disk). The amount of resource k requested by VNF n' is denoted as $\Psi_k^{n'}$. The constraints of compute and storage resources are

$$\sum_{n'} \Omega_n^{n'} \Psi_k^{n'} \leq r_k^n, \forall n, k \tag{2}$$

In addition, each VNF is hosted by only one substrate node. Consequently, we have the following constraint

$$\sum_n \Omega_n^{n'} \leq 1, \forall n' \tag{3}$$

Let us denote the set of incoming links at node n as $\mathbb{I}_n$ and the set of outgoing links at node n as $\mathbb{O}_n$. A connection between two adjacent VNFs can be considered as a commodity. Consequently, it is able to formulate the continuous path constraints using the multicommodity flow model

$$\sum_{e \in \mathbb{O}_n} \Phi_e^{e'} - \sum_{\bar{e} \in \mathbb{I}_n} \Phi_{\bar{e}}^{e'} = \Omega_n^{n'} - \Omega_n^{m'}, \forall n \tag{4}$$

where n' and m' are the source and destination of e'. The objectives are to minimize all measures

$$\min \sum_{e'} \sum_e \Phi_e^{e'} w_i^e, i = 1 \dots k \tag{5}$$

and deployment cost

$$\min \sum_{k=1}^{K} \sum_{n'} \sum_n \Omega_n^{n'} v_k^n + \sum_{i=1}^{M} \sum_{e'} \sum_e \Phi_e^{e'} z_i^e, \tag{6}$$

where $c_{k,n}$ is the cost of usage a unit of resource k of node n. Summarily, the VNF-FG embedding problem can be expressed as follows

$$\begin{array}{ll} \textbf{Objectives} & (5), (6) \\ \textbf{s.t.} & (1), (2), (3), (4) \end{array} \tag{7}$$

Single objective optimization with linear constraints and multicommodity flow model has been showed as a NP-hard problem, which is unable to identify the optimal solution in polynomial time [25]. The problem is even more difficult when it comes to multi-objective optimization. GAs have been confirmed its capabilities in coping with multi-objective optimization problem, although the execution time could be long. In the next section, we are going to discuss how to solve this problem using GAs and propose algorithms to shorten the execution time.

V. Genetic algorithm overview

In this paper, we revise the existing framework for genetic algorithms to cope with VNF-FGs allocation problem. The objective is to provide a near Pareto solution within a short calculation time. All proposed algorithms presented in Section VI and Section VII are based on this framework. Fig. 2 presents the proposed genetic algorithm framework. A repairer module is added into the framework to narrow the search within the set of feasible solutions. Additionally, a repairer can detect a feasible yet dominated solution and exclude it from the set to enhance the performance of searching the Pareto solution.

In genetic algorithms, a population is a set of candidate solutions (individuals) which are sequences of bits representing the values of $\Omega_n^{n'}$ and $\Phi_e^{e'}$. The algorithm selects parents among population based on their quality. Then, the parent creates offspring by executing crossover operator. The offspring may be mutated by a mutation operator which flips bits arbitrarily. Thanks to crossover and mutation, the quality of population could be improved gradually. The following subsections details the key components of genetic algorithms.

A. Initialization

To generate the initial population, bits are set arbitrarily. As a result, they may not be feasible due to the routing and resource constraints. A repairer process, presented in the following sections, is able to transform unfeasible solutions to feasible solutions while keeping randomness. It also reduces the search space and enhances the convergence speed since the algorithm explores only feasible solutions.

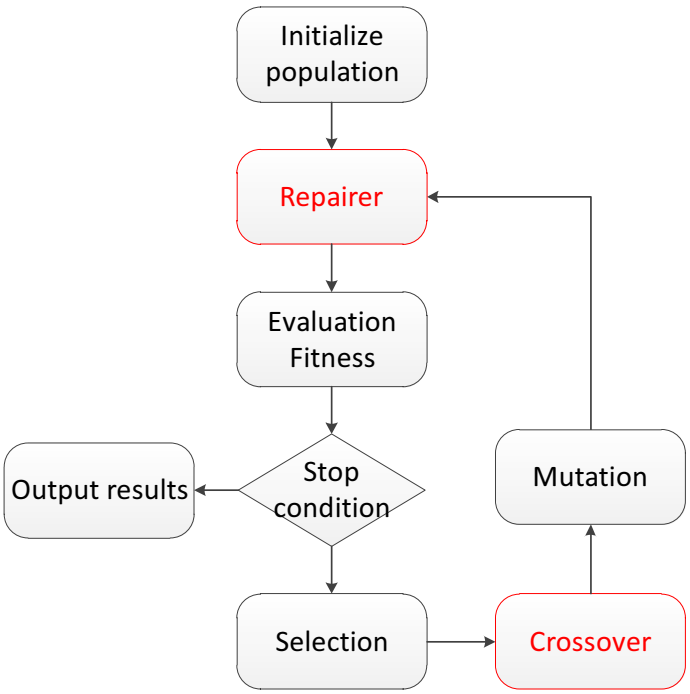


Fig. 2: Proposed Genetic Algorithm Flowchart

B. Repairer

This module plays a key role in the proposed framework. A repairer is to convert an unfeasible individual to a feasible individual. In other words, the output of the repairer is a solution that satisfies all constraints $(1), (2), (3), (4)$. The bits are modified by taking routing and resource constraints into consideration, while maintaining randomness in the population. In addition, the repairer is able to detect dominated solution earlier, thus narrowing the search space.

C. Selection

Genetic algorithms use a selection process to select individuals among the population to form a mating pool and generate new offspring, which are the basis of the next generation. Since the genes of individuals in the mating pool are inherited by the next generation, it is desirable that the mating pool consists of good individuals. A selection mechanism in GAs is a process to select better individuals in the population for the mating pool. The selection pressure is the degree of selecting better individuals. The higher the selection pressure, the more the selection favors the better individuals. The convergence rate of a GA is mainly proportioned to the selection pressure. GAs are able to identify optimal or near-optimal solutions under a wide range of selection pressure [26]. Nevertheless, the convergence rate will be slow if the selection pressure is too low, and it takes longer to find the optimal solution. Conversely, if the selection pressure is too high, there is an increased chance that the GA prematurely converges to a local optima and an suboptimal solution.

Tournament selection is one of the most important selection mechanisms in GAs. Its advantages are simplicity, robustness in the presence of noise, and adjustable selection pressure. Selection pressure is provided by holding a tournament among S competitors, where S is the tournament size. S individuals are randomly selected and the winner of the tournament is the individual with the best fitness among competitors.

As the mating pool is composed of tournament champions, its average fitness is higher than the population average fitness. It is this property that maintains the pressure in the selection process, leading to a successive improvement from generation to generation. The bigger is the tournament size, the more selection pressure is applied, since the winner of a large tournament, on average, have a higher fitness. Similarly, the smaller the size of the tournament, the lower the selection pressure, since the winner of a small tournament, on average, have a smaller fitness. In this paper, the binary tournament ($S = 2$) selection was selected to avoid the convergence toward a local optima that could happen with high selection pressure.

D. Crossover and mutation

A process of combining the bits of two parents to form an offspring which inherits characteristics of both parents, is a crossover. In some cases, the offspring could be better than its parents.

Mutation is an operator which flips the value of one or several bits of the offspring. It helps to maintain diversity in the population.

VI. Routing problem

A routing problem is to determine the path between two given nodes so that it satisfies given requirements, e.g. latency and/or loss rate. The VNF-FGs allocation problem turns into a routing problem when the VNF-FGs has 2 VNFs connecting by a virtual link and the hosts of VNFs are given. Consequently, we study the routing problem in this section and exploit it when solving VNF-FGs allocation problem. Alg. 1 presents the proposed framework to solve routing problem, named Genetic algorithm routing (GAR). The sub-module `Repairer` is presented in details in Section VI-B.

Input: Population size N, the number of generations G, source S, destination D, and QoS requests $\mathbf{q}$
Output: Set of paths
Initialize $n \leftarrow 0$, $g \leftarrow 0$;
Set of population $\mathbb{P} \leftarrow \emptyset$;
while $n < N$ **do**
  Initialize an individual p by assigning value for ϕ_e randomly;
  $p_0 \leftarrow$ `Repairer` $(S, D, p, \mathbf{q})$;
  $\mathbb{P} \leftarrow \mathbb{P} \cup \{p_0\}$
  Increase n;
end
while $g < G$ **do**
  if *Crossover* $\neq$ *null* **then**
    $\mathbb{P} \leftarrow$ `Crossover` $(S, D, \mathbb{P}, \mathbf{q})$;
  end
end
return $\mathbb{P}$

Algorithm 1: Genetic algorithm Routing (GAR)

A. Initialization

First, links are selected arbitrarily among available links to generate the initial population. Due to randomness, some individuals may be unfeasible. We propose an algorithm to correct unfeasible solution, thus narrowing the search space. A repairer process presented in the next subsection discusses in details the proposed approach.

B. Repairer

The proposed repairer are going to be discussed in this section. The main objective of the proposed repairer is to convert an unfeasible individual to a feasible non-dominated individual.

We introduce the cost of a path as a vector composed of elements corresponding to measures. The shortest path from n to d regarding QoS measure i (determined by Dijkstra algorithm) is denoted as $p_i^{n,d}$ and its measures can be expressed as a vector $\tilde{\mathbf{c}}_i^{n,d} = \left[\tilde{c}_{i,1}^{n,d}, ..., \tilde{c}_{i,M}^{n,d}\right]$, where M is the number of QoS measures and $\tilde{c}_{i,j}^{n,d}$ is the value of QoS measure j. Based on the definition of $\tilde{\mathbf{c}}_i^{n,d}$, we have $\tilde{c}_{i,i}^{n,d} \leq \tilde{c}_{j,i}^{n,d}, \forall j \neq i$. The upper-bound of QoS measure i of the paths from n to d, denoted as $u_i^{n,d}$, is defined as $u_i^{n,d} = \max_{j \neq i} \tilde{c}_{j,i}^{n,d}$.

Theorem 1. If $\tilde{c}_{i,i}^{m,d} + w_i^{(n,m)} > u_i^{n,d}$, there does not exist non-dominated path comprising link nm.

Proof. Assume that there is a non-dominated path $\mathcal{P}^*$ comprising link nm and $\tilde{c}_{i,i}^{m,d} + w_i^{(n,m)} > u_i^{n,d}, \forall i$. Intuitively, it exists a path from n to d, $\mathcal{P}_0$, such that $\tilde{c}_i^{\mathcal{P}_0} \leq u_i^{n,d}, \forall i$ (e.g. any shortest path from n to d).

All paths to d consisting of link nm have the measures i greater than or equal to $\tilde{c}_{i,i}^{m,d} + w_i^{(n,m)}$ since $\tilde{c}_{i,i}^{m,d}$ is the shortest length in measures i. Consequently, all paths traversing through nm are dominated by $\mathcal{P}_0$. Therefore, $\mathcal{P}^*$ could not be a non-dominated path. □

Theorem 2. In case of additive measures, Pareto-optimal paths are simple paths

Proof. In its definition, a simple path does not have repeating vertices.

Assuming that a Pareto-optimal path $\mathcal{P}^*$ that is not a simple path with a vertex M_0 visited twice is existed. The source and the destination of the path are s and d respectively. We denote the links originated from M_0 in first and second visits as (M_0, M_1) and (M_0, M_2) respectively. Consequently, the cost of $\mathcal{P}^*$, $\mathbf{c}^{P^*}$, can be determined as the sum of sub-paths due to additive metrics (all are simple paths)

$$\mathbf{c}^{\mathcal{P}^*} = \mathbf{c}^{\mathcal{P}_{s \to M_0}} + \mathbf{c}^{\mathcal{P}_{M_0 \to M_0}} + \mathbf{c}^{\mathcal{P}_{M_0 \to d}} \tag{8}$$

where $\mathcal{P}_{s\to M_0}$ and $\mathcal{P}_{M_0\to d}$ are the sub-paths from s to M_0 and from M_0 to d (traversing through M_2). $\mathcal{P}_{M_0\to M_0}$ is the cost of the loop that traversing through $M_0, M_1, ..., M_0$. $\mathcal{P}^*$ is dominated by the simple path $\mathcal{P}^{'}$ that has the same sub-paths from s to M_0 and from M_0 to d since the total cost of $\mathcal{P}^{'}$ is $\mathbf{c}^{\mathcal{P}_{s\to M_0}} + \mathbf{c}^{\mathcal{P}_{M_0\to d}}$ which is less than $\mathbf{c}^{\mathcal{P}^*}$. Thus, $\mathcal{P}^{'}$ dominates $\mathcal{P}^*$. It contradicts the initial assumption, thus all Pareto-optimal paths are simple paths. □

Due to randomness in initialization and mutation process, the feasibility of the solutions are not guaranteed. The responsibility of the repairer is to convert these unfeasible solutions to feasible solutions. Active entries of the inputs are stored in a set $\tilde{\mathcal{E}}$ (line 3). To maximize similarity of feasible solution and the input, these active entries are assigned higher priority to be selected through repairing procedure. The repairer identifies the shortest path using Dijkstra algorithm for each QoS measure (line 4) in order to predict the cost of paths and eliminate dominated paths in later steps. The main loop of repairing process (line 6 to line 22) is to determine all relaying nodes from the source to the destination. For each outgoing link from node n, the remaining bandwidth of the link, b_e, and the QoS measure, $w_i^e + c_i \leq q_i$, are checked (line 9). Then, the repairer checks if the destination m has been visited or belongs to blacklist $\mathbb{B}$ (line 11) since the Pareto-optimal path are simple paths as shown in Lemma 2 and there is no feasible path from nodes in the blacklist. Line 12 to line 15 are to verify whether the current link belongs to a non-dominated path following Lemma 1. If the set of candidate links $\mathcal{E}^*$ is empty, the algorithm reverts to the last selected vertex (line 16 and 17). Otherwise, the repairing procedure selects a link arbitrarily among common entries of $\tilde{\mathcal{E}}$ and $\mathcal{E}^*$ to enhance randomness in searching solutions (line 20 to line 21).

Input: An unfeasible path from S to D, QoS requests $\mathbf{q}$
Output: A feasible path satisfies QoS requests **foreach** *active bit of the input* **do** $\tilde{\mathcal{E}} \leftarrow e$;
foreach *QoS measure* i **do** Run Dijkstra for QoS measure $i \rightarrow \tilde{c}_i^{n,d}$;
$\mathcal{V}^* \leftarrow s$, $n \leftarrow s$, $\mathbf{c} \leftarrow \mathbf{0}$, and $\mathbb{B} \leftarrow \emptyset$;
while $n \neq d$ **do**
  $\mathcal{E}^* \leftarrow \emptyset$;
  foreach e *originated from* n **do**
    if $b_e < B_{req}$ ***or*** $w_i^e + c_i > q_i$ **then** **continue** ;
    $m \leftarrow Dest(e)$;
    if $m \in \mathcal{V}^* \cup \mathbb{B}$ **then** **continue** ;
    foreach *QoS measure* i **do**
      $u_i^{n,d} \leftarrow \max_{j\neq i} \tilde{c}_{j,i}^{n,d}$;
      if $\tilde{c}_{i,i}^{m,d} + w_i^{(n,m)} \leq u_i^{n,d}$ **then**
        $\mathcal{E}^* \leftarrow e$ and **break**;
      end
    end
  end
  if $\mathcal{E}^* = \emptyset$ **then**
    $\mathbb{B} \leftarrow \mathbb{B} \cup \{n\}$, $n \leftarrow$ last entry in $\mathcal{V}^*$;
    if $n \neq s$ **then continue**;
    else break;
  end
  if $\mathcal{E}^* \cap \tilde{\mathcal{E}} \neq \emptyset$ **then** Select e from $\mathcal{E}^* \cap \tilde{\mathcal{E}}$ randomly ;
  else Select e among entries in $\mathcal{E}^*$ randomly ;
  $n \leftarrow Dest(e)$, $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{w}^e$, and $\mathcal{V}^* \leftarrow \mathcal{V}^* \cup \{n\}$;
end
return $\mathcal{V}^*$;

Algorithm 2: Repairer

In line 11, the worst case complexity is $\mathcal{O}(\log|\mathcal{V}|)$ and it is $\mathcal{O}(M)$ for the process from line 12 to line 15. Thus, the overall worst case complexity of the main loop of repairing procedure (line 6 to line 20) is $\mathcal{O}(|\mathcal{V}|^3 (M + \log|\mathcal{V}|))$. The Dijkstra algorithm complexity (line 4) is $\mathcal{O}(M|\mathcal{E}|\log|\mathcal{V}|)$. Note that this loop runs once and can be reused for all individuals, while the main loop has to be executed for each individual.

C. Non-dominated crossover

A crossover is a process in which the bits of one parent are combined with those of another in order to create offspring that inherit characteristics of both parents. We propose a crossover scheme so as to generate offspring that are not dominated by their parents in this section.

Definition 1. A crossover point $n^{i,j}$ of a pair of paths (i, j) is a common vertex of both paths (excluding the source and the destination).

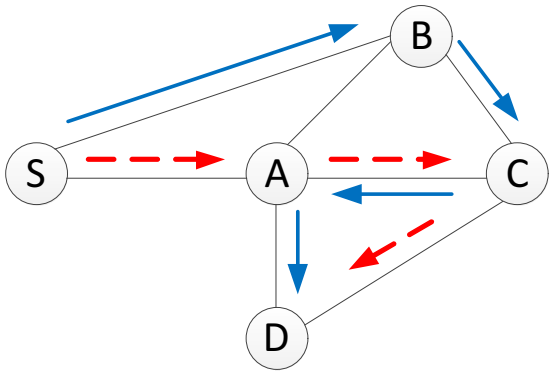


Fig. 3: Example of lemma 1

Definition 2. Sub-paths of path i created by crossover point m, denoted as $\mathcal{P}_{ia}(m)$ and $\mathcal{P}_{ib}(m)$, where $\mathcal{P}_{ia}(m)$ and $\mathcal{P}_{ib}(m)$ are the sub-paths from source s to m (inclusive) and from m (exclusive) to destination d, respectively.

Definition 3. A non-isolated crossover point of a pair of parents (i, j), $n^{i,j}$, is a crossover point of path $\mathcal{P}_i$ and $\mathcal{P}_j$ that $\mathcal{P}_{ia}(n^{i,j}) \cap \mathcal{P}_{jb}(n^{i,j}) \neq \emptyset$.

Definition 4. An isolated crossover point of a pair of parents (i, j), $m^{i,j}$, is a crossover point of path $\mathcal{P}_i$ and $\mathcal{P}_j$ that $\mathcal{P}_{ia}(m^{i,j}) \cap \mathcal{P}_{jb}(m^{i,j}) = \emptyset$

Note that a non-isolated (isolated) crossover point $n^{i,j}$ might be not a non-isolated (isolated) crossover point of (j, i) since it could happen that $\mathcal{P}_{ja}(n^{i,j}) \cap \mathcal{P}_{ib}(n^{i,j}) = \emptyset$ ($\mathcal{P}_{ja}(n^{i,j}) \cap \mathcal{P}_{ib}(n^{i,j}) \neq \emptyset$).

Lemma 1. Given an offspring A formed by a non-isolated crossover point $n^{i,j}$, there exists an offspring formed by an isolated crossover point that dominates A.

Proof. Since $n^{i,j}$ is a non-isolated crossover point, it means $\mathcal{P}_{ia}(n^{i,j}) \cap \mathcal{P}_{jb}(n^{i,j}) \neq \emptyset$. Let us denote $\mathcal{C} = \mathcal{P}_{ia}(n^{i,j}) \cap \mathcal{P}_{jb}(n^{i,j})$ as the common vertices of $\mathcal{P}_{ia}(n^{i,j})$ and $\mathcal{P}_{jb}(n^{i,j})$. Obviously, it exists an isolated crossover point $m^{i,j} \in \mathcal{C}$, and forms two new sub-paths $\mathcal{P}_{ia}(m^{i,j})$ and $\mathcal{P}_{jb}(m^{i,j})$. Due to $\mathcal{P}_{ia}(m^{i,j}) \cap \mathcal{P}_{jb}(m^{i,j}) = \emptyset$ and additive measures, $\mathbf{c}^{\mathcal{P}_{ia}(m^{i,j})} \leq \mathbf{c}^{\mathcal{P}_{ia}(n^{i,j})}$ and $\mathbf{c}^{\mathcal{P}_{jb}(m^{i,j})} \leq \mathbf{c}^{\mathcal{P}_{jb}(n^{i,j})}$. Consequently, the offspring formed by $(\mathcal{P}_{ia}(m^{i,j}), \mathcal{P}_{jb}(m^{i,j}))$ dominates the offspring $(\mathcal{P}_{ia}(n^{i,j}), \mathcal{P}_{jb}(n^{i,j}))$. □

Input: Linked list of vertices of parents $\mathcal{P}_1, \mathcal{P}_2$, QoS requests $\mathbf{q}$
Output: non-dominated offspring
Find common vertices $\rightarrow \mathcal{V}_c$;
List of swap nodes $\mathcal{S}_1, \mathcal{S}_2$;
if $\mathcal{V}_c = \emptyset$ **then** **return** $\mathcal{P}_1, \mathcal{P}_2$; ;
foreach $n \in \mathcal{V}_c$ **do**
  $\mathcal{P}_i^{'}(n) \leftarrow \emptyset, i = 1, 2$
  $\mathcal{P}_{ia}(n) \leftarrow \mathcal{P}_i.sublist\,(s, n)\,, i = 1, 2$;
  $\mathcal{P}_{ib}(n) \leftarrow \mathcal{P}_i.sublist\,(n.next(), d)\,, i = 1, 2$;
  if $\mathcal{P}_{1a}(n) \cap \mathcal{P}_{2b}(n) = \emptyset$ **then** $\mathcal{P}_1^{'}(n) = \mathcal{P}_{1a}(n).concatenate\,(\mathcal{P}_{2b}(n))$; ;
  if $\mathcal{P}_{2a}(n) \cap \mathcal{P}_{1b}(n) = \emptyset$ **then** $\mathcal{P}_2^{'}(n) = \mathcal{P}_{2a}(n).concatenate\,(\mathcal{P}_{1b}(n))$; ;
  if $\mathcal{P}_1^{'}(n) \neq \emptyset$ ***and*** *not dominated* ***and*** *satisfies* $\mathbf{q}$ **then** $\mathcal{S}_1 \leftarrow n$; ;
  if $\mathcal{P}_2^{'}(n) \neq \emptyset$ ***and*** *not dominated* ***and*** *satisfies* $\mathbf{q}$ **then** $\mathcal{S}_2 \leftarrow n$; ;
end
Select node n_1^* in $\mathcal{S}_1$ randomly $\rightarrow O1 = \mathcal{P}_1^{'}(n_1)$;
Select node n_2^* in $\mathcal{S}_2$ randomly $\rightarrow O2 = \mathcal{P}_2^{'}(n_1)$;
return O1 and O2;

Algorithm 3: Non-dominated Crossover

Fig. 3 describes an example of Lemma 1. The first path (red dashed line) comprises the following vertices S, A, C, D. The second path (blue solid line) traverses through S, B, C, A, D. Crossover points are A, C. Vertex C is non-isolated crossover point while A is isolated crossover point. The offspring formed by non-isolated crossover point (C) is S, A, C, A, D which is dominated by the offspring formed by isolated crossover point (A): S, A, D.

The proposed non-dominated crossover is presented in Alg. 3. First, the proposed crossover finds common nodes from parents (line 3). For each node in the set of common nodes, it checks if swapping can create the new offspring that are not dominated by their parents (line 7 to line 13). Then, it checks whether this node is an isolated crossover point (line 10 and line 11) because the non-isolated crossover points are dominated as shown in Lemma 1. The offspring are compared with their parents (line 12 and line 13). Isolated crossovers that form non-dominated offspring and satisfy QoS requirements are stored in $\mathcal{S}_1$ and $\mathcal{S}_2$ corresponding to the first and second offspring. Finally, the algorithm select arbitrary offspring among candidates (line 14 and line 15).

The computational complexity of finding common nodes in parents is $\mathcal{O}(|\mathcal{V}|\log|\mathcal{V}|)$. The worst case complexity of the process from line 10 to line 11 is $\mathcal{O}(|\mathcal{V}|\log|\mathcal{V}|)$. The comparison between offspring and parents has the complexity of $\mathcal{O}(M)$. Consequently, the loop from line 7 to line 13 has the complexity of $\mathcal{O}(|\mathcal{V}|\,(|\mathcal{V}|\log|\mathcal{V}|+M))$. The overall complexity of non-dominated crossover procedure is $\mathcal{O}(|\mathcal{V}|\,(|\mathcal{V}|\log|\mathcal{V}|+M))$.

VII. VNF-FGs ALLOCATION PROBLEM

In this section, we discuss the proposed algorithm to solve the VNF-FGs allocation problem, which is the main problem in this paper. Individuals consist of bits representing $\Phi_e^{e'}$ and $\Omega_n^{n'}$. With a given set of $\Omega_n^{n'}$, GAR presented in Section VI, will be called to identify a corresponding set of $\Phi_e^{e'}$.

A. Initialization

The initial population of VNF-FGs allocation problem is created by randomly assigning VNFs to hosts, which have adequate resources. It is to satisfy the constraints (2). First, the set of substrate nodes that can satisfy resource requests of VNF n', $\mathbb{S}_{n'}$, is identified. Each VNF is allocated at most to one substrate node (following the constraints (3)), while a substrate node can host multiple VNFs to reduce the end-to-end QoS. The host of n' is selected from $\mathbb{S}_{n'}$ to minimize the number of substrate hosts.

B. Non-dominated E2E Routing Algorithm

The initialization step provides a set of substrate nodes that are able to host VNFs. For each individual, we have to identify the paths connecting the hosts of VNFs. By executing GAR, we can identify the set of paths, called as sub-paths, connecting the substrate hosts of each virtual link. The algorithm 4 is to create a set of end-to-end paths for an VNF-FGs based on them.

Theorem 3. With additive metrics, a non-dominated path traversing through given nodes is the combination of non-dominated sub-paths.

Proof. Let us denote $\mathtt{h}_{n'}$ as the host of VNF n'. The set of non-dominated paths traversing through all $\mathtt{h}_{n'}, \forall n'$ is $\mathbb{P}^*$. For a virtual link $n'm'$, $\mathcal{P}^*_{\mathtt{h}_{n'}\to\mathtt{h}_{m'}}$ is a non-dominated path from $\mathtt{h}_{n'}$ to $\mathtt{h}_{m'}$. Let us assume there is a path $\mathcal{P}' \in \mathbb{P}^*$ comprising k dominated sub-paths and l non-dominated sub-paths. It exists a path $\mathcal{Q}^*_{\mathtt{h}_{n'}\to\mathtt{h}_{m'}}$ that dominates each sub-path $\mathcal{Q}'_{\mathtt{h}_{n'}\to\mathtt{h}_{m'}}$ in k dominated sub-paths. Let us denote $\mathbb{Q}^*$ as the set of paths that dominate k sub-paths of $\mathcal{P}'$. Consequently, $\mathcal{P}'$ is dominated by the path forming by l non-dominated sub-paths and k sub-paths $\mathbb{Q}^*$. □

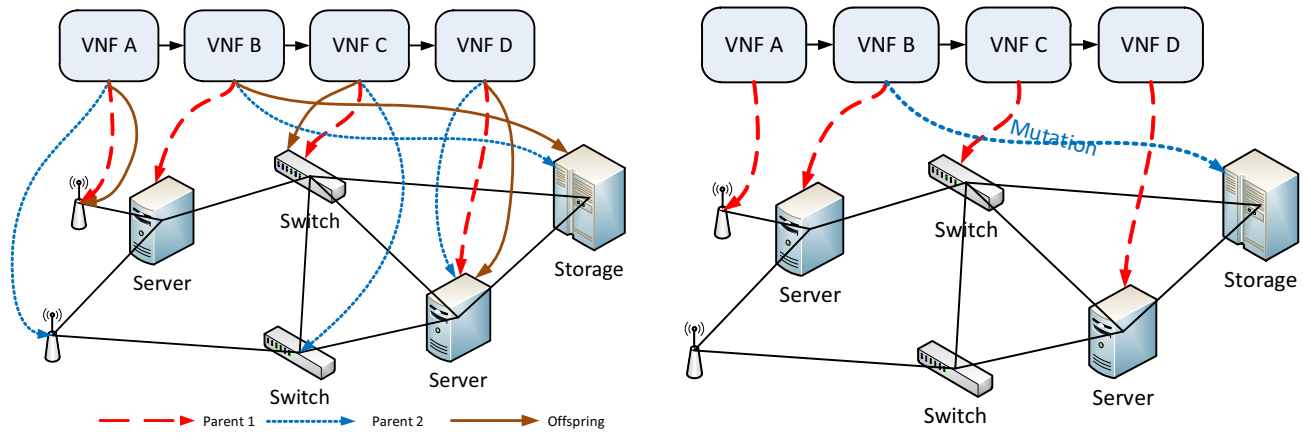
The input of the algorithm is the given values of $\Omega_n^{n'}$. Theorem 3 claims that an end-to-end non-dominated path traversing all given substrate nodes forming by the non-dominated sub-paths. Consequently, the algorithm starts by computing the set of non-dominated sub-paths of each virtual link. For each virtual link $n'm'$, the algorithm determines the source S and the destination D based on the given $\Omega_n^{n'}$ (line 6). Then, it search in the explored sub-paths database $\mathbb{Q}^{'}$. If the set of non-dominated paths from S to D exists, the algorithm will retrieve the set from the database (line 7 to 8) and remove paths violating QoS requirements (line 9). Otherwise, it calls the GAR to determine the set of non-dominated paths between S and D and add the set to $\mathbb{Q}^{'}$ (line 10 to line 12). Having the set of explored sub-paths helps to reduce the computation time by avoiding call GAR frequently. These new sub-paths will be combined with the set of current non-dominated paths, $\mathbb{P}^*$, which are the combination of sub-paths of previous virtual links (line 14 to 18). The new path formed by this combination is added into a set $\mathbb{C}$ if it satisfies bandwidth constraints (i.e. all links of the path have the greater bandwidth than the request). Following Theorem 3, the algorithm selects only non-dominated paths from $\mathbb{C}$ (line 20 to line 24) to update the set of predetermined non-dominated paths. The algorithm continues until all virtual links are processed. The output are the set of non-dominated paths that satisfy all QoS requests of virtual links as well as that of end-to-end.

C. VNF placement crossover and mutation

Due to the dependencies of bits in each individual, conventional crossover approaches, e.g. single-point crossover, are not adoptable. Consequently, a simple crossover that fits with the encoding is proposed. The crossover happens on a VNF n' on both parents at a given probability θ_c. When crossover operator is executed, the substrate node n_1 hosting n' in parent 1 is swapped with the substrate node n_2 hosting n' in parent 2.

Input: A graph of VNFs and their hosts $\left(\Phi_n^{n'}\right)$, QoS requests for each virtual link $\mathbf{q}_{n'm'}$ and end-to-end $\mathbf{q}_{e2e}$
Output: non-dominated path traversing through all hosts of VNFs
End-to-end non-dominated path $\mathbb{P}^* \leftarrow \emptyset$;
Explored sub-paths $\mathbb{Q}^{'} \leftarrow \emptyset$;
foreach *Virtual link* $n'm'$ **do**
  $S \leftarrow \mathrm{h}_{n'}$ and $D \leftarrow \mathrm{h}_{m'}$;
  if $\mathbb{Q}^{'}.\mathtt{find}(S, D) \neq \emptyset$ **then**
    $\mathbb{Q}^* \leftarrow \mathbb{Q}^{'}.\mathtt{find}(S, D)$;
    Remove paths which violate $\mathbf{q}_{n'm'}$ in $\mathbb{Q}^*$;
  end
  else
    $\mathbb{Q}^* \leftarrow \mathtt{GAR}(S, D, \mathbf{q}_{n'm'})$;
    $\mathbb{Q}^{'} \leftarrow \mathbb{Q}^{'} \cup \mathbb{Q}^*$;
  end
  Set of combinations $\mathbb{C} \leftarrow \emptyset$;
  foreach $\mathcal{P} \in \mathbb{P}^*$ **do**
    foreach $\mathcal{Q} \in \mathbb{Q}^*$ **do**
      $\mathcal{P}' \leftarrow \mathtt{concatenate}(\mathcal{P}, \mathcal{Q})$;
      $c_{\mathcal{P}'} \leftarrow$ Compute the cost of $\mathcal{P}'$;
      if $\mathcal{P}'$ *satisfies* $\mathbf{q}_{e2e}$ **then** $\mathbb{C} \leftarrow (\mathcal{P}', c_{\mathcal{P}'})$; ;
    end
  end
  $\mathbb{P}^* \leftarrow \emptyset$;
  foreach $\mathcal{P} \in \mathbb{C}$ **do**
    $n_{\mathcal{P}} = 0$;
    foreach $\mathcal{Q} \in \mathbb{C}$ **do**
      if $\mathcal{Q} \prec \mathcal{P}$ **then** $n_{\mathcal{P}} = n_{\mathcal{P}} + 1$; ;
    end
    if $n_{\mathcal{P}} = 0$ **then** $\mathbb{P}^* \leftarrow \mathcal{P}$; ;
  end
end

Algorithm 4: Non-dominated Routing Algorithm



(a) Crossover operator

(b) Mutation operator

Fig. 4: Examples of crossover and mutation operators

For each VNF-substrate node map $\Omega_n^{n'}$, a mutation can happen with a given probability θ_m. When a mutation operator occurs on a VNF-substrate node map, the substrate node will change to another substrate node in $\mathcal{S}_{n'}$ randomly. Fig. 4 shows an example of the crossover operator and the mutation operator. When crossover or mutation is applied, the non-dominated end-to-end routing algorithm should be executed to identify the new end-to-end path.

D. Genetic algorithm for VNF-FGs allocation (GAVA)

The proposed algorithm for VNF-FGs allocation problem is presented in Alg. 5. It requires information of VNF-FGs, thus it should be located in a service orchestrator (e.g. NFVO). The algorithm starts by initializing values of $\Omega_n^{n'}$ discussed in Section VII-A. The algorithm prefers assigning multiple VNFs at the same substrate node when it is possible to reduce the end-to-end QoS measure, although it may not optimize in terms of VNF deployment cost. Then, Alg. 4 is executed to determine a set of paths traversing through all given substrate hosts. For each generation, crossover operator and mutation operator (Section VII-C) will be applied to expand the searching space. Due to crossover and mutation, $\Phi_e^{e'}$ should be revised to be consistent with the new $\Omega_n^{n'}$. This could be done by running Alg. 4. The algorithm continues until the maximum number of generations is reached.

Input: The size of population N and the number of generations G, SFC
Output: Substrate hosts $\left(\Omega_n^{n'}\right)$ and links $\left(\Phi_e^{e'}\right)$ for SFC Initialize $n \leftarrow 0$, $g \leftarrow 0$;
Set of population $\mathbb{P} \rightarrow \emptyset$;
while $n < N$ **do**
  Initialize an individual p by assigning value for $\Omega_n^{n'}$ such that $\Omega_n^{n'}$ satisfies resource constraints and minimize number of substrate hosts;
  Run Algorithm 4 to identify a set of paths $\mathcal{P}$ traversing all host defined by $\Omega_n^{n'}$;
  Select a path from $\mathcal{P}$ randomly $\rightarrow \Phi_e^{e'}$;
  Increase n;
end
while $g < G$ **do**
  `Crossover` $\left(\Omega_n^{n'}\right) \rightarrow \Omega_n^{n'}$ `Mutation` $\left(\Omega_n^{n'}\right) \rightarrow \Omega_n^{n'}$ **if** *Mutation* $\neq$ *null* ***or*** *Crossover* $\neq$ *null* **then**
    foreach *individual* **do**
      Run Algorithm 4 to identify a set of paths $\mathcal{P}$ traversing all host defined by $\Omega_n^{n'}$;
      Select a path from $\mathcal{P}$ randomly $\rightarrow \Phi_e^{e'}$;
    end
  end
  Increase g;
end
return $\Omega_n^{n'}$ and $\Phi_e^{e'}$

Algorithm 5: Genetic algorithm for VNF-FG allocation (GAVA)

E. Upper-bound of Hypervolume

Due to difficulties in identifying Pareto solutions, this section will introduce the upper-bound of the Hypervolume. Let us denote $\mathtt{S}^*$, $\widehat{\mathtt{S}}$, and $\tilde{\mathtt{S}}^*$ as the sets of solutions of VNF-FGs allocation problem, routing problem, and substrate node selection problem respectively. An VNF-FGs allocation solution i, $S_i^* \in \mathtt{S}^*$ is formed by two sub-solutions $\widehat{S}_i^*$ and $\tilde{S}_i^*$.

Lemma 2. It does not exist an VNF-FGs allocation solution which has sub-solutions dominating all solutions of routing problem and substrate node selection problem

Proof. Let us assume that there is a sub-solution $\widehat{S}_0^*$ ($\tilde{S}_0^*$) which dominates all solutions in $\widehat{\mathtt{S}}$ $\left(\tilde{\mathtt{S}}\right)$. Since $\widehat{S}_0^*$ ($\tilde{S}_0^*$) is feasible, it should be an entry of $\widehat{\mathtt{S}}$ $\left(\tilde{\mathtt{S}}\right)$. Consequently, $\widehat{S}_0^*$ ($\tilde{S}_0^*$) cannot dominate all solutions in $\widehat{\mathtt{S}}$ $\left(\tilde{\mathtt{S}}\right)$. This contradicts the assumption. □

Let us denote $\mathtt{S}^c$ as the combination of solutions of routing problem and substrate node selection problem. The hypervolume of a solution $\mathtt{S}$ is denoted as $\mathtt{HV}(\mathtt{S})$.

Lemma 3. $\mathtt{HV}(\mathtt{S}^c) \geq \mathtt{HV}(\mathtt{S}^*)$

Proof. From Lemma 2, for each solution $S_i^* \in \mathtt{S}^*$, it exits a solution S_j^c such that S_j^c dominates S_i^*. Consequently, $\mathtt{HV}(\mathtt{S}^c) \geq \mathtt{HV}(\mathtt{S}^*)$ □

From Lemma 3, the upper-bound of the hypervolume is $\mathtt{HV}(\mathtt{S}^c)$.

VIII. Numerical results

This section evaluates the performance of GAR and GAVA. The probability of crossover operator (θ_c) and the probability of mutation operator (θ_m) are 0.9 and 0.1, respectively. The GA default algorithm is NSGA-II. Tab. II presents default parameters for simulations.

Parameters	Value
Population size	20
Number of generations	120
QoS metrics	Loss rate and Latency
Loss rate	$0.01\% - 0.1\%$ [29]
Number of runs per scenario	30
Confidence interval	95%
Simulation Environment	Intel Core i5-6300U, 8GB RAM
Genetic algorithm framework	jMetal [28]

TABLE II: Default simulation parameters

Name	# nodes	# directed links	Shortest path distance
Kdl (Kentucky Data Link)	754	1788	39
Colt (Colt telecom)	153	354	18
GtsCe (GTS Central Europe)	149	386	27
UsCarrier (US Carrier)	158	378	26
Deltacom (ITC Deltacom)	113	322	18

TABLE III: Topology parameters

Different scales of networks (hundreds of nodes) are considered. Network sizes and shortest distance between source and destination (number of hops) are shown in Table III. These topologies are provided by topology-zoo [27] with real latency.

We use a genetic algorithm framework provided by jMetal 5 [28]. We are aware that jMetal only parallelizes evaluation steps in GAs. However, the heaviest computing task in the proposed scheme is of the repairer and each individual could be repaired independently. Consequently, we modify the framework to enable parallel repairing process.

A. Routing problem

We evaluate the performance of GAR under different scales of networks and configurations in this section. For genetic algorithms, different combinations of proposed modules are considered: Non-dominated crossover and mutation (NDC-M), Single-point crossover and mutation (SP-M), Non-dominated crossover only (NDC), and Mutation only (M). In single-point crossover, parents are divided at an arbitrary point. Novel offspring are generated by concatenating the first part of one parent with the second part of the other. For NDC, it is not necessary to execute repairing process after crossover since the offspring is a feasible solution, therefore reducing computational complexity.

The conventional SAMCRA defines the length as the maximum of ratios of metrics and their upper-bounds. Its objective is to determine the shortest length path. Consequently, the solution of SAMCRA might not be a Pareto-optimal solution. As a result, we define the length as a vector of ratios instead of a linear or non-linear function to derive Pareto-optimal solutions. The modified SAMCRA with Look-ahead feature, denoted as SAMCRA-LA-MO, is used as reference because its output is the Pareto optimal solution.

First, we run simulations to confirm the advantages of the proposed parallel NSGA-II, denoted as p*NSGA-II, and the default parallel NSGA-II of jMetal, denoted as pNSGA-II. The network is Kdl. Number of generations and population are 60 and 10 respectively. The results are shown in Fig. 5. When only NDC is used, the difference in calculation time is not significant. It is because the repairing process is executed once when the initial population is generated. Meanwhile, there are remarkable differences in other configurations when the repairing process has to be executed in every generations. Generally, p*NSGA-II is able to reduce the runtime up to 30%.

Comparisons of four configurations in terms of normalized hypervolume and calculation time are shown in Fig. 6 and Fig. 7. Two QoS metrics are considered in these simulations. When the population and the number of generations is 10 and 60 respectively, there is a noticeable gaps between NDC and other configurations, especially in very complex networks, (e.g. Kdl)

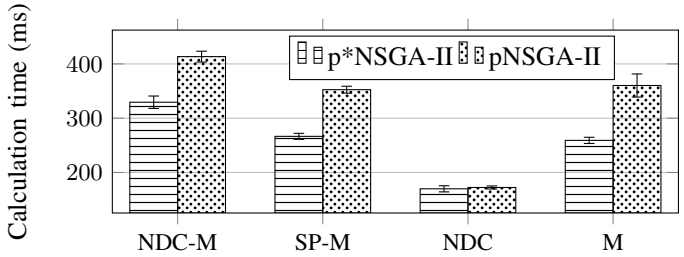
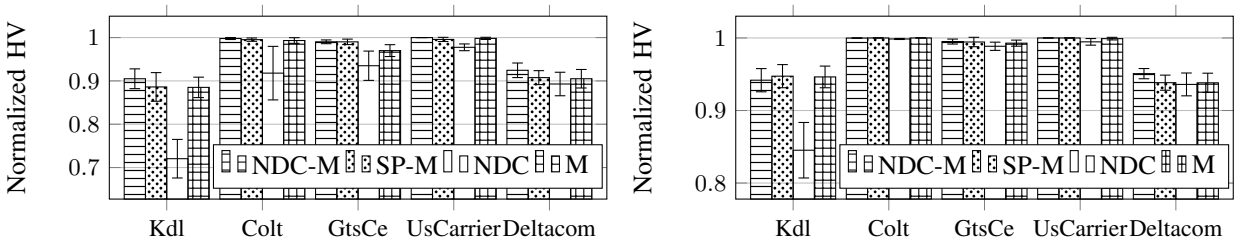


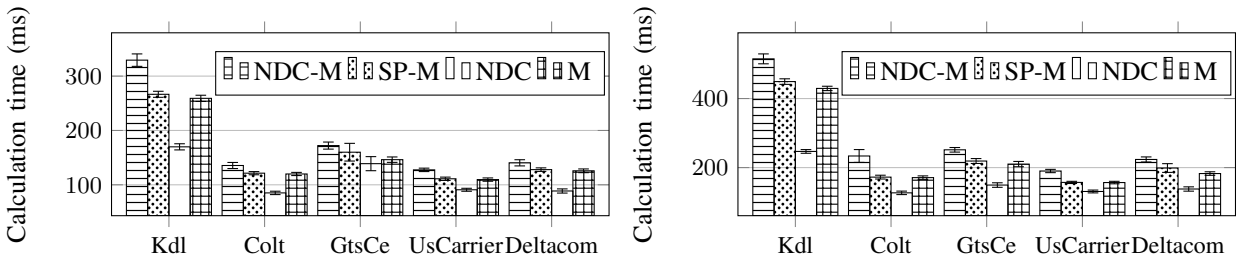
Fig. 5: Calculation time of p*NSGA-II and pNSGA-II



(a) Nomarlized Hypervolume with 10 individuals and 60 generations

(b) Normalized Hypervolume with 20 individuals and 120 generations

Fig. 6: Normalized Hypervolume with 2 QoS metrics



(a) Calculation Time with 10 individuals and 60 generations

(b) Calculation Time with 20 individuals and 120 generations

Fig. 7: Calculation time when there are 2 QoS metrics

since the missing of mutation and small population restricts the capability of finding a Pareto optimal solution of NDC. Due to simplicity of NDC, its calculation time is shortest. The calculation time gap is proportional to the complexity of networks because the high complex networks requires more calculation time for the repairing process. The combination NDC-M has better performance than NDC thanks to the mutation. The performance of NDC-M, nevertheless, is similar to SP-M and M while its runtime is noticeably longer. When the population and the number of generations are increased, the performances of all configurations in most cases (except Kdl) are similar to each other. The increase in population improves the chance to find out Pareto optimal solutions of NDC. The cost of obtaining better solution is the calculation time. By doubling both population and the number of generations, the calculation time increases by approximately 50%.

The latency and loss-rate of optimal solutions (Kdl-Pareto Front (Kdl-PF), Colt-Pareto Front (Colt-PF), GtsCe-Pareto Front (GtsCe-PF)) obtained by SCRM and GA-based solutions with NDC-M (Kdl-NDC-M, Colt-NDC-M, GtsCe-NDC-M) are shown in Fig. 8. The results are consistent with the normalized HV shown in Fig. 6. In Colt and GtsCe networks, the GA-based results are similar to optimal solutions. The performance of Kdl-NDC-M is near to the optimal solutions; however, it does not

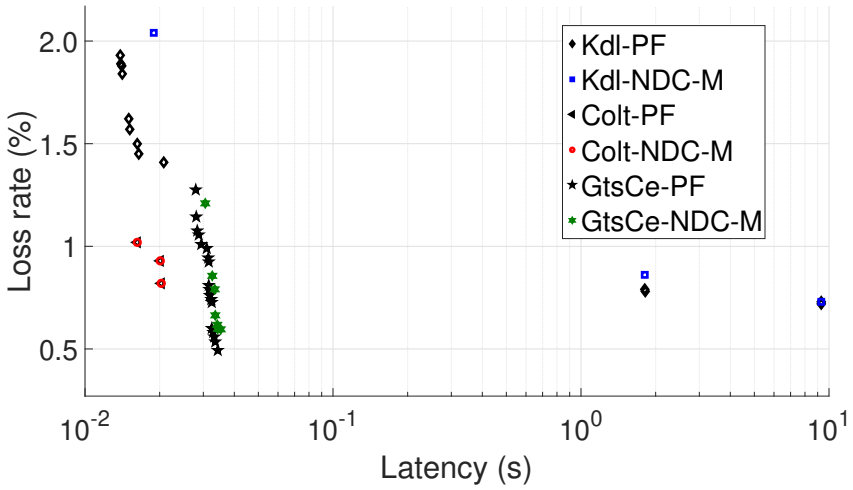
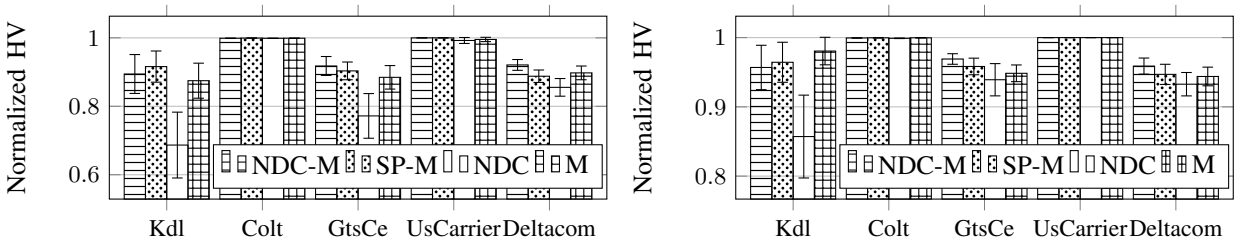


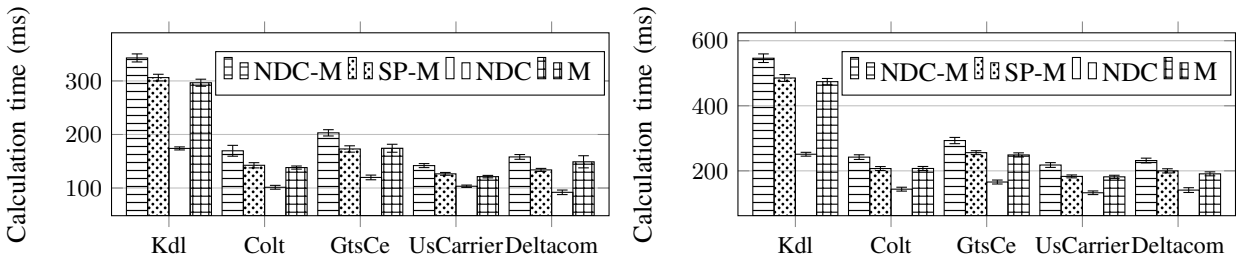
Fig. 8: Loss Rate and Latency performance



(a) Normalized Hypervolume with 10 individuals and 60 generations

(b) Normalized Hypervolume with 20 individuals and 120 generations

Fig. 9: Normalized Hypervolume, 3 QoS metrics



(a) Calculation Time with 10 individuals and 60 generations

(b) Calculation Time with 20 individuals and 120 generations

Fig. 10: Calculation time, 3 QoS metrics

cover all Pareto Front leading to the lower normalized HV in Fig. 6.

We run simulations when the number of QoS metrics is extended to 3 in order to verify the scalability of the proposed mechanism. The normalized hypervolume and calculation time are shown in Fig. 9 and Fig. 10. By adding QoS metrics, the number of non-dominated solutions could change (increase or decrease), which explains the reduction in the normalized HV of GtsCe and the increase in the normalized HV of Colt. When it comes to the calculation time, there is no significant changes when the number of QoS metrics changes.

Fig. 11 compares the proposed mechanism with the SAMCRA-LA-MO in terms of calculation time. The calculation time of SAMCRA-LA-MO is similar to SP-M when the sizes of networks is small or moderate. The calculation time of SAMCRA-LA-MO, nevertheless, is much greater than the proposed mechanism in a large-scale network scenario (e.g. Kdl). Consequently, the implementation of SAMCRA-LA-MO in large-scale networks is impractical. The proposed mechanism is able to offer similar normalized HV as shown in Fig. 6 within 1s.

Fig. 12 and Fig. 13 are the comparisons of the proposed mechanism with different GAs in Kdl scenario. Although NSGA-II has the worst performance in normalized HV, its calculation time is shortest thanks to its simplicity. SPEA2 has better normalized HV and lower calculation time than NSGA-III. Both SPEA2 and NSGA-III have the calculation time about 10 times that of NSGA-II.

B. VNF chain placement

We consider an VNF-FGs with 4 VNFs connecting serially. Each VNF requests four types of resources: computing, memory, storage, and radio. We assume that the first VNF is the radio head, so it requires radio resource while others do not. In addition,

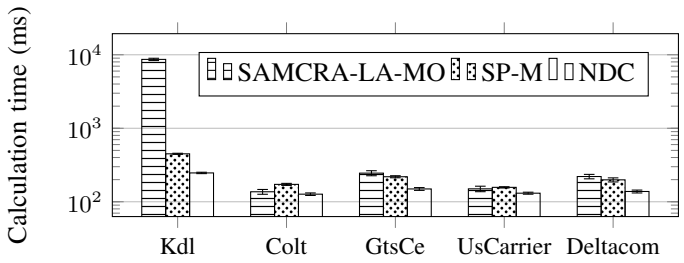


Fig. 11: Calculation Time of SAMCRA-LA-MO vs GAs (20 individuals and 120 generations), 2 QoS metrics

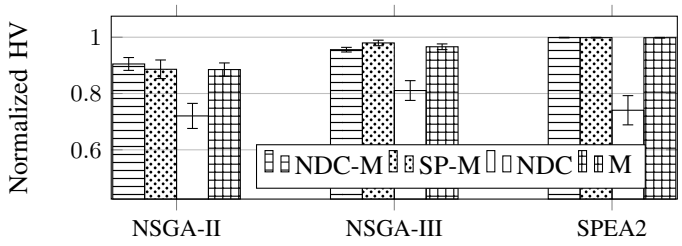


Fig. 12: Normalized HV (10 individuals and 60 generations), 2 QoS metrics of different GA algorithms

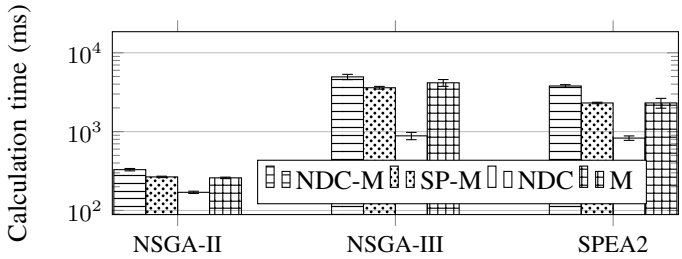


Fig. 13: Calculation time (10 individuals and 60 generations), 2 QoS metrics of different GA algorithms

the number of substrate node that can host the first VNF is limited due to the geographical location of users. For simplicity, we assume that the hosts of the first VNF and the last VNF are given.

For the VNF-FGs allocation problem, the default size of population (denoted as PS) is 20 and the number of generations (GS) is 120. Meanwhile, the size of population of routing problem (PR) is 10 and 20 with the corresponding numbers of generations (GR) 60 and 120. As discussed in the previous section, NDC-M will cost more in calculation time while offering similar performance in HV. Consequently, we exclude NDC-M in this section. The hypervolume results are normalized with the upper-bound presented in Section VII-E

As shown in previous section, the performance of NSGA-II is around 10% less than of NSGA-III and SPEA2. However, the very short calculation time of NSGA-II motivates us to focus on it, in this section. The performance of p*NSGA-II in the VNF-FGs placement problem is demonstrated in Fig. 14. In the previous section, it is shown that p*NSGA-II outperforms pNSGA-II in routing problem. The performance gap between them is even extended in VNF-FGs placement problem due to its high computational complexity. For instance, the calculation time of pNSGA-II in Kdl scenario is twice as long as of p*NSGA-II. In other scenarios, the calculation time values of pNSGA-II are 50% more than of p*NSGA-II.

As discussed in Section VII, GAR will be called to determine the set of non-dominated paths between two hosts of a virtual link. Consequently, it is necessary to examine impacts of routing problem to the VNF-FGs allocation problem. The normalized HV is computed as the ratio of the obtained HV and the upperbound HV introduced in Section VII-E. Fig. 15 presents the normalized HV under different types of GA routing and values of GR and PR. For normalized HV, there are significant differences between NDC and two other mechanisms when GR and PR are small. However, this gap is negligible when GR and PR increase to 120 and 20 respectively. The HVs of three scenarios: Colt, UsCarrier, and Deltacom are near to the upper-bound HV. However, the HVs of GtsCe and Kdl are far from the upper-bound HV. This is because the complexity of Kdl and GtsCe are higher than others (presented through the number of links and the shortest length of VNF-FGs). The calculation times of scenarios are presented in Fig. 16. Intuitively, the increase in the size of population and the number of generations prolong the calculation time. The calculation time is around 10s except Kdl scenario. Even in very high complex

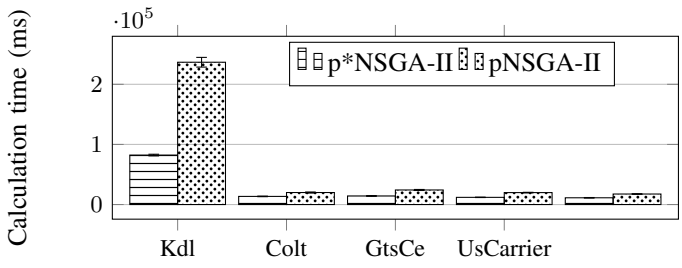
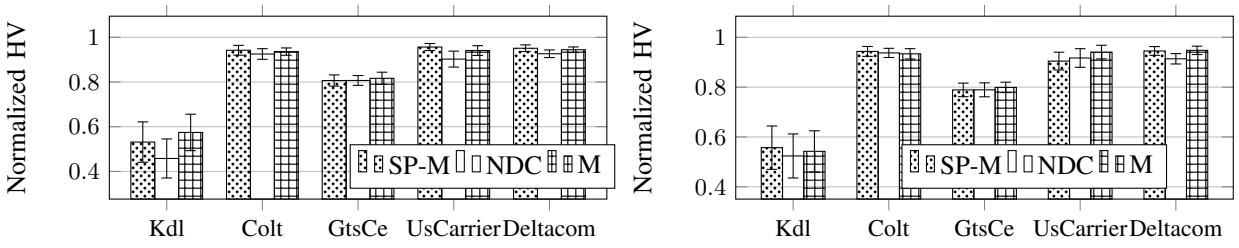


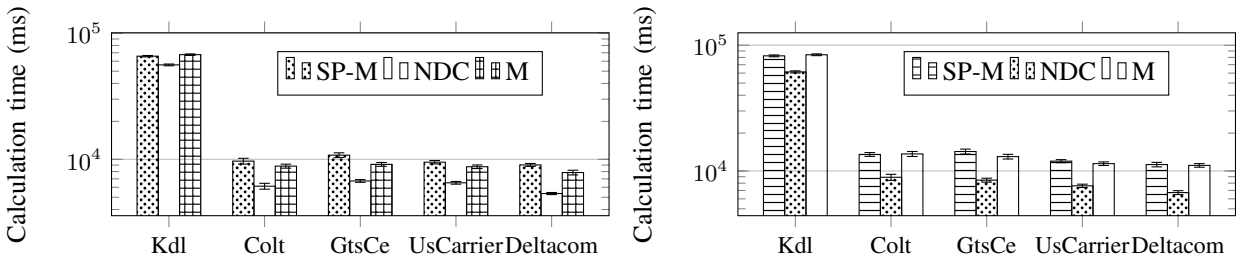
Fig. 14: Calculation time of p*NSGA-II and pNSGA-II



(a) Normalized Hypervolume of VNF-FGs allocation problem with PR=10 and GR=60

(b) Normalized Hypervolume of VNF-FGs allocation problem with PR=20 and GR=120

Fig. 15: Normalized Hypervolume of VNF-FGs allocation



(a) Calculation Time with 10 individuals and 60 generations

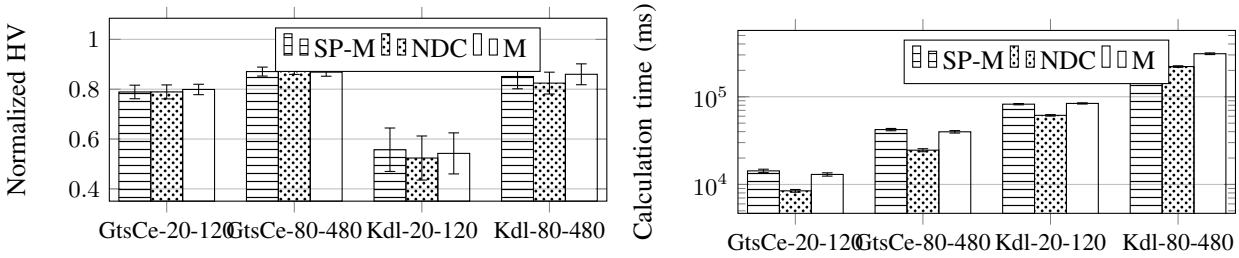
(b) Calculation Time with 20 individuals and 120 generations

Fig. 16: Calculation time

scenario (e.g. Kdl), the calculation time is still less than 100s in both configurations.

To deal with the highly complex topologies, one obvious approach is to increase the size of the population and the number of generations of the VNF-FGs problem, i.e. PS and GS. Fig. 17 shows the enhancements in HVs when $\texttt{PS} = 80$ and $\texttt{GS} = 480$ of Kdl and GtsCe scenarios. The HVs of both scenarios increase when PS and GS increase. While there is a slight increase in GtsCe scenario (around 10%), a remarkable increase is found in Kdl scenario. The calculation time in Kdl scenario with $\texttt{PS} = 80$ and $\texttt{GS} = 480$ is from 200s to 300s. Note that we can reduce it significantly by running on parallel computing supported platforms that are available in data center in practice. It is remarkable that the proposed approach outperforms existing works in terms of execution time. For instance, it takes 40 seconds to determine the placement of a 3-VNF chain with a small substrate topology, BT Europe (24 nodeS, 74 links), and 500 generations as shown in [30]. Meanwhile, our proposed approach can achieve a good NHV for a much larger substrate network, GtsCe (149 nodes, 386 links), in less than 20 seconds (with 120 generations). In [31], a 4 VNF chain placement on a small substrate network, Internet2 (12 nodes, 15 links), was studied. It takes around 2 seconds to achieve 90% of maximum hypervolume. Meanwhile, we can achieve a better performance $\texttt{NHV} = 90\%$ (the upperbound HV is even better than every obtainable maximum hypervolume) in a much larger network, Deltacom (113 nodes, 322 links) in 2 seconds. The loss rate, deployment cost, and latency of GAVA (SP-M) under different configurations of PS and GS in Kdl scenario is shown in Fig. 18. By increasing PS and GS, GAVA is able to enhance both quality and the number of solutions, thus NHV increases as shown in Fig. 17

Next, we conduct simulations under different lengths of VNF-FGs. It is to demonstrate how the performance of the proposed scheme changes when the complexity of VNF-FGs increases. Fig. 19 presents the performance of the proposed scheme in



(a) Normalized HV 80-480

(b) Calculation time 20-120 vs. 80-480

Fig. 17: Performance 20-120 vs. 80-480

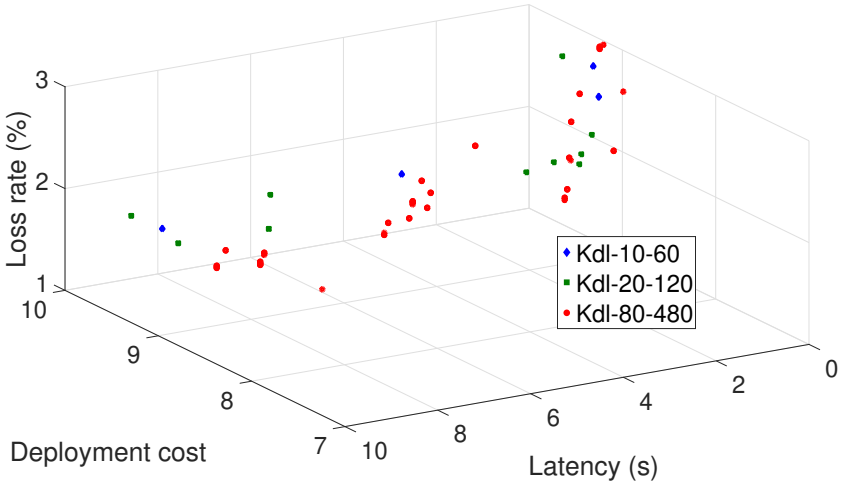
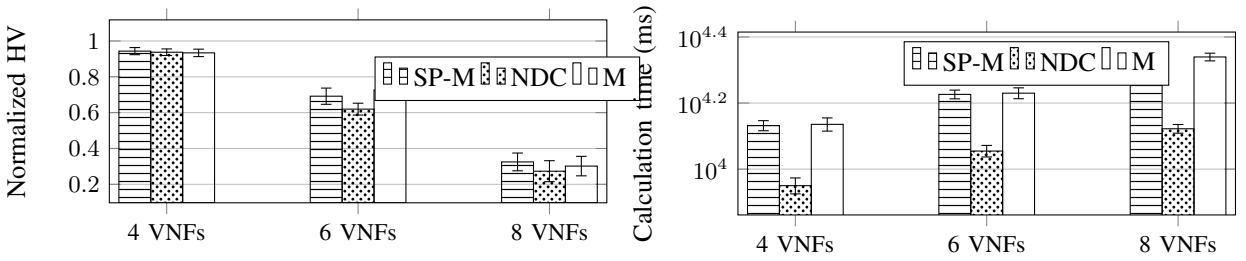


Fig. 18: Performance of GAVA in Kdl scenario



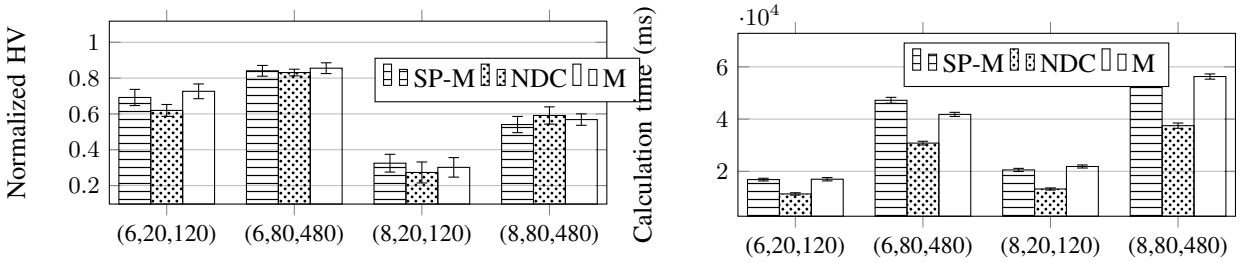
(a) Normalized HV Colt 20-120 under different numbers of VNFs

(b) Calculation time Colt 20-120 under different numbers of VNFs

Fig. 19: Performance Colt 20-120 under different numbers of VNFs

Colt scenario under different number of VNFs (4, 6, and 8). Regarding normalized HV, the increase in the number of VNFs leads to the degradation in normalized HV. Due to the constraint of location of VNFs, the difference between the solution and the optimal solution of routing problem ($\widehat{S}$) increases. In addition, the higher number of VNFs requires a greater number of population to explore more possibilities of placing VNFs. When it comes to the calculation time, Fig. 19b shows that the calculation time is linear to the size of VNFs. It confirms the scalability of the proposed scheme.

The low normalized HV of complicated VNF-FGs can be overcome by increasing the size of population and the number of generations. Fig. 20 demonstrates improvement in performance when we increase the number of generations and the size of population. In the figure, we use tuples (x, y, z) to denote the configurations of simulations, where x is the number of VNFs, y is PS, and z is GS. By increasing both PS and GS four times, we obtain a gain of about 20% in normalized HV. Obviously, the calculation time will also increase when we increase the number of generations and the population size as shown in Fig. 20b. However, the long calculation time can be shortened by deploying on many CPU hardware.



(a) Normalized HV Calculation time Colt 20-120 vs 80-480

(b) Calculation time Colt 20-120 vs 80-480

Fig. 20: Performance Colt 20-120 vs 80-480

IX. Conclusions

Network function virtualization plays a key role in next generation networks. In existing literature, the placement of VNFs has been studied intensively. However, it is lack of studies considering this problem with multiple objectives. Additionally, the extremely high computational complexity of this problem represents a real challenge in optimization of the operation of the networks. This paper proposed a framework of genetic algorithms in order to deal with the placement of VNF chains. The proposed framework is able to reduce the computation time whilst maintaining good solutions at the output. Although the optimization of multiple chain placement was not considered in this paper, it could be done by slightly extending the proposed framework. Additionally, it could require more computational resources due to its high complexity. These challenges will be considered in future.

References

[1] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992 – 1007, 2006, special Issue - Genetic Algorithms and Reliability.

[2] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[3] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106 – 127, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000291

[4] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213 – 220, 2016, iNOC 2015 – 7th International Network Optimization Conference.

[5] P. T. A. Quang, K. D. Singh, A. Bradai, and A. Benslimane, "QAAV: quality of service-aware adaptive allocation of virtual network functions in wireless network," in *IEEE ICC 2018*, Kansas City, USA, May 2018.

[6] P. T. A. Quang, A. Bradai, K. D. Singh, and R. Riggio, "A^2VF: adaptive allocation for virtual network functions in wireless access networks," in *IEEE WoWMoM 2018*, Chania, Crete, Greece, Jun. 2018.

[7] D. Li, J. Lan, and P. Wang, "Joint service function chain deploying and path selection for bandwidth saving and vnf reuse," *International Journal of Communication Systems*, vol. 31, no. 6, p. e3523, 2018, e3523 IJCS-17-0957.R1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3523

[8] C. Zhang, X. Wang, F. Li, M. Huang, and Q. He, "Network service chains deployment across multiple sdn domains," *International Journal of Communication Systems*, vol. 31, no. 18, p. e3826, 2018, e3826 dac.3826. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3826

[9] M.-T. Thai, Y.-D. Lin, and Y.-C. Lai, "Joint server and network optimization toward load-balanced service chaining," *International Journal of Communication Systems*, vol. 31, no. 10, p. e3556, 2018, e3556 dac.3556. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3556

[10] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, Sep. 2016.

[11] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for load balancing in NFV networks," in *Proc. of IEEE ICC*, May 2017, pp. 1–6.

[12] N. Srinivasan and K. Deb, "Multi-objective function optimisation using non-dominated sorting genetic algorithm," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[13] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr 2002.

[15] Z. He and G. G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 145–160, Feb 2016.

[16] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[17] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[18] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. of IEEE CEC*, vol. 1, May 2002, pp. 825–830.

[19] H. Seada and K. Deb, "U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization," *COIN report*, vol. 2014022, 2015.

[20] A. Ibrahim, S. Rahnamayan, M. V. Martin, and K. Deb, "EliteNSGA-III: An improved evolutionary many-objective optimization algorithm," in *Proc. of IEEE CEC*, July 2016, pp. 973–982.

[21] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," in *Proc. of IEEE CEC*, July 2016, pp. 3045–3052.

[22] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," European Telecommunication Standard Institute (ETSI), Group Specification (GS) NFV-MAN 001, 12 2014, version 1.1.1.

[23] P. V. Mieghem and F. A. Kuipers, "Concepts of exact qos routing algorithms," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 851–864, Oct 2004.

[24] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sep 1996.

[25] P. T. A. Quang, K. Piamrat, K. D. Singh, and C. Viho, "Video streaming over ad hoc networks: A qoe-based optimal routing solution," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1533–1546, Feb 2017.

[26] D. E. Goldberg, K. Deb, and D. Thierens, "Toward a better understanding of mixing in genetic algorithms," *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 10–16, 1993.

[27] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, October 2011.

[28] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jmetal multi-objective optimization framework," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1093–1100. [Online]. Available: http://doi.acm.org/10.1145/2739482.2768462

[29] E. Rodriguez-Colina, D. Gil-Leyva, J. L. Marzo, and V. M. RamosR., "A bit error rate analysis for tcp traffic over parallel free space photonics," *Telecommunication Systems*, vol. 56, no. 4, pp. 455–466, Aug 2014.

[30] S. Khebbache, M. Hadji, and D. Zeghlache, "A multi-objective non-dominated sorting genetic algorithm for vnf chains placement," in *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2018, pp. 1–4.

[31] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, "A multi-objective heuristic for the optimization of virtual network function chain placement," in *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1, Sept 2017, pp. 152–160.