Vertical Parallelization of Differential Evolution Heuristic for Network Slicing in 5G Scenarios

Rayner Gomes[1], **Dario Vieira**[2], **Miguel Franklin de Castro**[3]

[1]Federal University of Piauí (UFPI)
Picos – PI – Brazil

[2]EFREI Paris (EFREI)
Paris – FR.

[3]Federal University of Ceará
Fortaleza, CE – Brazil

`rayner@ufpi.edu.br, dario.vieira@efrei.fr, miguel@ufc.br`

Abstract. *The 5G mobile network is based on a virtualized infrastructure and offers a virtual network (VN) creation service considering many new scenarios arising from the 5G vision. The diversity of scenarios and the instantiation of VN on-demand induce pressure on the virtual network embedding (VNE). VNE is the mapping of virtual nodes and links to real nodes and links obeying the QoS parameters present in the VN request and available resources. Since this is an optimization and NP-Hard problem, multiple efforts have been made to create VNE algorithms. Considering such efforts, this work presents: (i) a fitness function regarding multiobjective optimization; and (ii) a Parallel Differential Evolution (PDE) approach to face the VNE. We designed the PDE due to the lack of viable parallel solutions in the 5G scenario. We compared our approaches with different versions of Greedy, Stress, and Genetic Algorithms, totaling ten approaches. The results demonstrate that DE and its parallel version obtained a higher number of mapped requisitions. Also, the parallel performance decreases the execution time in certain conditions; in a favorable scenario, the parallel version obtains up 21.04% of runtime reduction.*

1. Introduction

Fifth-generation mobile network (5G) has the ambition to endure a wide range of services and applications. 5G paves the way to vertical markets such as smart-home, security and surveillance, transportation, industrial, retail, and healthcare. Each tenant will define a set of demands which will imply the type of services offered by the 5G provider. It is a fact that 5G will not design an entire architecture to comply with all sorts of vertical markets for different applications. Instead of this, 5G is designed to have a baseline architecture to offer high flexibility, built on software-defined network (SDN), network function virtualisation (NFV), and Cloud computing to allow a virtual network (VN) to each tenant as needed by the Network Slicing as a Service (NSaaS) [5GPPP 2019].

The 5G infrastructure has the NSaaS as a critical component, a mechanism adopted to address the diversity of requirements and specific services and applications. Network Slicing (NS) has been designed as a key enabler to allow 5G to handle Internet of Things (IoT) and other vertical markets. The NSaaS waits for some Virtual Network Requests (VNRs) that can arrive at any time. A VNR is a formal document described by a network topology and quality of services (QoS) demands. After the VNR is successfully mapped, its requirements are met, and all necessary resources are allocated to ensure its proper functioning. An accepted VNR becomes a virtual network that has a predefined duration. NSaaS is composed of a set of chained activities to yield a VN, e.g.: (i) allocating resources; (ii) installing new services; (iii) orchestrating network management components; (iv) signalling network controls; and others. Among these activities, a primordial one is the mapping of virtual networks to physical resources, mathematically known as Virtual Network Embedding (VNE).

The mapping of virtual networks to real ones can be done by starting with the mapping of the nodes then the links or vice versa. More elaborated algorithms map nodes and links at the same time. Mathematically, this process is called VNE, which is, essentially, an optimization problem. Considering only the node mapping, VNE can be associated with the classic travelling salesman problem [Wu et al. 2020]. Even when node mapping from virtual to physical is given, the problem of optimally allocating a set of virtual links to a single physical path is reduced to an unsplittable multicommodity flow problem, which is also *NP-hard* [Yu et al. 2008, Fischer et al. 2013, Vassilaras et al. 2017]. VNE is tackled with deterministic or heuristics ways. The goal is to map a set of a requisition taking into account some demands, such as: (i) a certain limit of time; (ii) maximum of embeddings; (iii) respecting all or part of QoS parameters. It is important to remember that NS is a service, and the slice clients (tenants) do not expect to wait a long time to receive its slice (virtual network), which is another requirement in 5G systems.

Although the researchers have addressed VNE over time, new trends arise from the various 5G scenarios, vertical market demands, and in-network applications innovations. Because of these trends' heterogeneity, academia and industry are embedded in proposing new VNE solutions coping with these novel aspects and constraints. The drive to increase the number of served requests and reduce the time of processing the mapping encourages continuous research in this area. While deterministic solutions tend to have a shorter runtime [Cao et al. 2016], on the other hand, heuristic-based solutions have a higher mapping success rate and better adaptation to more complex topologies and more stringent requirements [Fischer et al. 2013].

Regarding the aspects mentioned above, in this work, we extend our previous work [Gomes et al. 2021] in which it was the first to design and adapt the Differential Evolution (DE) to the VNE problem. We redesigned it to seek a way to reduce the runtime using a parallelization method, a method in which there is a lack in the literature regarding the evolutionary approaches. Also, we enhanced the scenario regarding a more complex infrastructure. Moreover, we magnified the embedding complexity considering node capacity, link bandwidth, and quality of services parameters such as delay and reliability. Further, the approach proposed in [Gomes et al. 2021] has a small number of requisitions and does not seek to increase the acceptance rate.

It is critical to perceive that reducing the time to perform the mapping and increasing the number of mapped requisitions represent an increase in profit for the provider. More elaborate algorithms allow for better resource selection to accommodate more virtual networks. Also, more involved algorithms are more processing consumers; one way to reduce the execution time considering the same heuristics is by code parallelization. Therefore, in this work, we propose a method to parallelize DE to minimize the runtime. The challenge is that the differential evolution algorithm is naturally a sequential logic.

Our previous work [Gomes et al. 2021] reveals that the DE takes longer to run for the same number of repetitions and population size than the Genetic Algorithm (GA). Hence, in this paper, we enhanced these points: **(i)** we seek to improve the number of mapped requisitions; thus, we conceived a new fitness function; **(ii)** we endeavor to reduce the runtime of DE by parallelization; **(iii)** we adopt two more complex network topologies in the tests; **(iv)** we amplify the number of approaches from one to ten, they derive from Greedy, Stress, GA, and DE approaches; **(v)** the parallelization is not a *silver-bullet*, so we present in what conditions the parallelization of DE has advantage and disadvantage; and **(vi)** we propose two ways of initializing the first populations.

This work is organized as follows: Section II compares this work to related works; the description of the mathematical problem is modeled in Section III; Section IV presents the possible ways to parallel DE. Section V covers competitor approaches; Section VI explains the structure of the *fitness function.* Section VII performs a comparison of all the algorithms and analyzes the results; and, finally, Section VIII concludes this work by showing some opportunities for future works.

2. Related Work

We are interested in a meta-heuristic in which divides the solution into computation units to reduce the runtime. Meta-heuristic adaptations to VNE problems that leverage the capacity of accelerating its computation by parallelisms in 5G NSaaS systems. [Fischer et al. 2013] made a comprehensive review surveying 125 works, and found only one work dealing with parallelization. However, that heuristic considers parallelization of the capacity of mapping one node in two different substrates, and it is not about VNE-logic parallelization. Therefore, we have submitted on Scopus a search taking into account the title, abstract, and keywords with values ”virtual network embedding” and ”5G” in the period ranging from 2018 to 2021. As a result, we received 43 papers related to VNE and 5G. Out of 43 articles, only 10 applied a meta-heuristic and only three used the parallelism.

One contribution of [Nguyen and Huang 2019, Nguyen et al. 2020] is a solution to VNE based on a Genetic Algorithm (GA) updated with some paralleled steps to reduce runtime. However, its non-parallelized step consists of building a collection of paths for each virtual link (path-pool) based on its source and destination pairs. As [Salimifard and Bigharaz 2020] demonstrated when mapping the VNE problem to the traveling sales, the path-pool formation is unfeasible in large networks since the possible combinations of paths are enormous in a vast infrastructure.

[Tasoulis et al. 2004] were the first authors to publish a version of the differential evolution parallel approach. Their solution takes a population and spawns in n sub-populations on a different processor. Each sub-population performs a complete DE step, and there is a mechanism to select the better individual for each group. Finally, if the stop criterion for the objective function is met, the controller process sends a termination signal to all the sub-populations. The literature is rich in other methods in fields distinct from VNE. Our approach is not the same as theirs. The difference is related to granularity, i.e., our solution does not create subgroups, so each individual is sent to a Thread or Sub-process, and the stop criterion is only based on the meta-parameter *repetition*. Another difference is related to mathematical operations, our DE version considers the node's geographic position; thus, calculations are performed based on the Cartesian plane.

To the best of our knowledge, there is no proposal for a parallel differential evolutionary approach applied to the VNE problem considering 5G scenarios. Despite numerous researches in the field of VNE, there is a lack of works dealing with parallel meta-heuristics. Another critical point concerns the limitations of the network. Our work is the only one that considers three network constraints in the optimization process: Bandwidth, Delay and Reliability. The Next Generation Mobile Network Alliance (NGMN), the Third Generation Project Partnership (3GPP), and the International Telecommunication Union (ITU) have proposed three types of slices: enhanced mobile broadband (eMBB), ultra-reliable and low latency communications (URLLC), and massive machine type communications (mMTC). Each slice type provides different network behavior based on the bandwidth, delay, and reliability values specified in each slice request [5G PPP Architecture Working Group 2016].

In our earlier work [Gomes et al. 2021] we set out some reasons for choosing the Genetic Algorithm (GA) as a good competitor: (i) it allows searching for the solution in a huge solution space; (ii) it does not restrict the search process to local domains; and (iii) it is flexible enough to be adapted to different scenarios. Moreover, the fitness function and chromosomal structure focus on adapting the heuristic GA to solve different problems. DE can derive all these advantages from GA due to their similarities, and this aspect led us to further adapt DE to VNE problems.

GA and DE have a similar evolution process. Due to its strengths mentioned above, GA is widely used for VNE [Han et al. 2018, Zhou et al. 2016], and thus we choose GA as one of the main competitors. Moreover, we can use the same fitness function, which allows a fair comparison between them during the process.

3. Modeling and Problem Description

A comprehensive modeling is presented in [Gomes et al. 2021], which is the same we followed in this work. A network infrastructure is represented by a non-directed graph denoted by $G = (V, E, \beta)$, in which G represents a physical network, V is a set of vertices in which $v_i \in$ V and represents a real device, an edge $e_i \in E$, where E is a set of edges and it represents a real link. Each vertex and edge is characterized by capacities $\beta \mathrm{v}_i^c \geq 0$ and $\beta \mathrm{e}_i^b \geq 0$. In the *online operation* of the network, $\beta \mathrm{v}_i^{res}$ and $\beta \mathrm{e}_i^{res}$ can play the role of residual capacity, which is the remaining resource of the vertices or edges after taking out the current utilization (Figure 1). There is only one edge between two vertices. The region/location is represented by $\beta \mathrm{v}_i^{re}$, which can define a geographic localization.

A virtual network is denoted by an undirected graph $H^i = (N, L, \delta)$ with virtual nodes $n_i \in N$, and links $l_i \in L$. Each access node has a total of $\delta \mathrm{n}_i^a$ capacities, and they can be decomposed to CPU, memory, and other device resources. Each virtual node n_i can be embedded in one physical node from a set of physical nodes V. A virtual node is associated with a single physical node, and a virtual link is associated with a single physical path, which is a set of physical links.

The embedding of H into G consists of a mapping as follows: (i) each virtual node n $\in$ N to a physical node $v \in V$; (ii) a virtual link (n_i, n_j) to a loop-free physical path, connecting physical nodes v_i and v_j to which the virtual nodes n_i and n_j have been mapped. For a slice user, a slice behaves like a physical network, and no difference should be noticed. The VNE is a process to associate each virtual node to a physical node and virtual links to physical links respecting that the sum of the demanded virtual resources is less than or equal to the available physical resources. The VNE must maintain the control of used resources. Moreover, VNE algorithms have the goal of finding a feasible embedding respecting all QoS parameters demanded by a VNR. Duration d represents the time in which a slice must exist, and this information is relevant to online operations, where each slice has lifetime.

Let σ be a mapping function, G a substrate and H a set of VNRs where H^i is a request from H. The main goal can be defined as (1), and that means we seek to maximize the quantity of mapping (σ) in the current cycle (c).

$$max \sum_{i}^{H} \sigma(H^i, G, c) \mid c \in [c_i, ..., c_j] \tag{1}$$

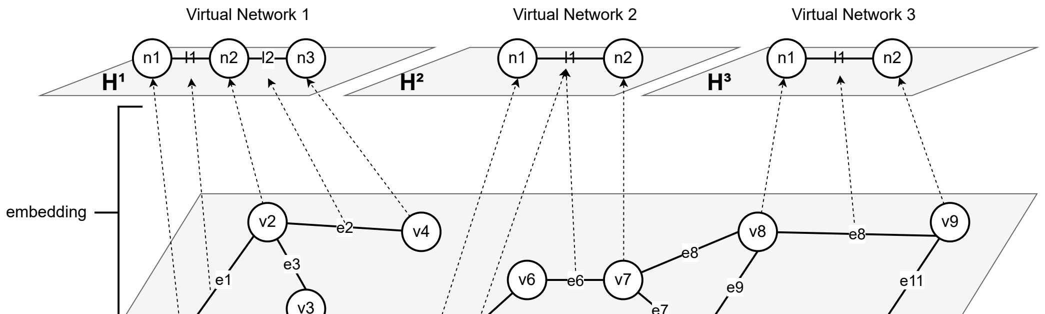
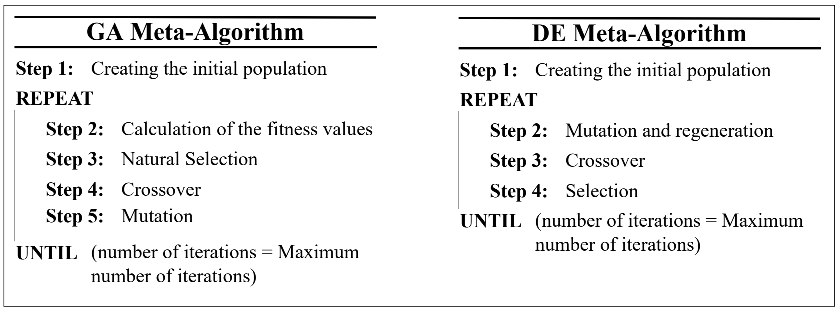


Figure 1. Embedding: virtual elements (nodes and links) associated with real ones. The mapping should respect the virtual network demands and infrastructure resources capacities.

4. Parallel Differentiated Evolution to VNE Problems

[Gomes et al. 2021] are the first one to apply DE in VNE problems. The authors demonstrated the capacity of its approach to select the best places in the network to a VNR. Their solution is based on a measurement to describe the accuracy of selection choices. The accuracy is calculated regarding the best places in the network, considering the bandwidth, delay, and reliability demands contained within a VNR. The measure function knows the better places before the execution and, for this reason, the network used in the simulations is small, with about 20 nodes.

Figure 2 summarizes the steps of GA and DE meta-algorithms. Price and Storn introduced DE in 1995. DE is a population-based optimization based on evolutionary algorithms in general. The crossover, mutation, and natural selection operators of GA are also included in DE. However, in DE the chromosomes are treated individually and a new individual is formed from three randomly selected chromosomes. These operations are performed using the mutation and crossover operators. The [Gomes et al. 2021] solution uses latitude and longitude characteristics during node operations. In this way, the geographic location is considered in addition and subtraction operations between two nodes.



```
GA Meta-Algorithm
Step 1:   Creating the initial population
REPEAT
    Step 2:   Calculation of the fitness values
    Step 3:   Natural Selection
    Step 4:   Crossover
    Step 5:   Mutation
UNTIL   (number of iterations = Maximum
        number of iterations)
```

```
DE Meta-Algorithm
Step 1:   Creating the initial population
REPEAT
    Step 2:   Mutation and regeneration
    Step 3:   Crossover
    Step 4:   Selection
UNTIL   (number of iterations = Maximum
        number of iterations)
```

Figure 2. GA and DE Meta-Algorithms

Four points distinguish this work from [Gomes et al. 2021]: (i) it is not oriented to increase the accuracy, but the quantify of requisitions mapped; (ii) the evaluations use two larger datasets, for which reason we do not know a priory what part of the network is better for each requisition; and (iii) it considers ten competitor approaches, while [Gomes et al. 2021] only two. As explained in Section 2, there is a gap in the literature on a parallel heuristic to tackle VNE problems. Hence, this work details two ways to parallelize the VNE-DE approach. The main goal of implementing the parallel solution is to try to reduce the runtime of the DE to perform the mapping.

Algorithm 1 is a modification of the VNE-DE in [Gomes et al. 2021]. The algorithm's internal functions are presented in [Gomes et al. 2021] and for the sake of space, we will not explain them in detail here. The algorithm has the outermost loop and two internal loops (command *for*). The internal loops are the code blocks for the mutation (lines 4-6) and the selection phase (lines 8-10). DE has the particularity that each individual mutation can be carried out separately. This feature favors the creation of parallel subprocesses. In the original code, these two blocks are sequential instructions. In the parallel version, the main code is kept non-parallelized, and the internal codes are parallelized in news tasks/sub-processes. The selection phase starts after the synchronization of ending of all threads/sub-processes created to carry out the mutation.

Algorithm 1: Parallel Differentiated Evolution Algorithm

```
    Input  : fitness, lb, ub, $N_p$, T, F, $P_c$
1   P ← initPopulation(P);
2   $V_i$ ← createDonor();
3   for $t = 1$ to T do
        /* Mutation block                                    */
4       for $i = 1$ to $N_p$ do
5           $P_i$ ← process ($U_i$, i, runCrossover);
6       end
7       join (P);
        /* Selection block                                   */
8       for $i = 1$ to $N_p$ do
9           $P_i$ ← process ($U_i$, $V_i$, i, lb, ub, bound, fitness,
             greedSelection);
10      end
11      join (P);
12  end
```

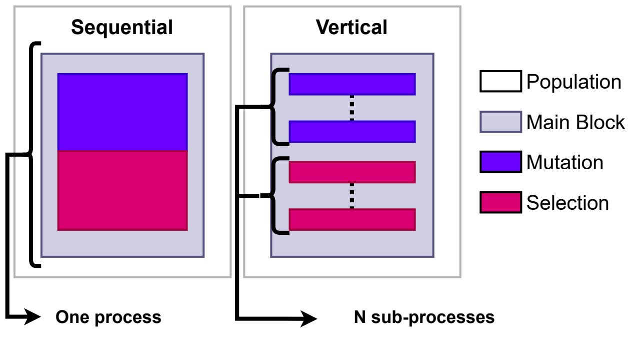


Figure 3. Parallelization of DE steps in subroutines.

The nature of heuristic/logic will determine which parallelization is possible. The DE is an evolutionary approach; therefore, each new generation is a consequence of the previous ones. In Algorithm 1, the first external loop is the code related to evolution and, because of this, it can not parallelize the external loop. The main advantage of creating tasks/sub-processes to execute the mutation is that the mutation process is a sequence of calculations and searches carried out simultaneously. It is essential to notice that DE has mutation before the crossover and selection differently of GA approach (Figure 3). The DE mutation is more CPU intensive than the GA approach. Therefore, DE mutation is the process that better leverages parallelism. Thus, the VNE-DE could have its final execution time reduced. One of the objectives of this work is to evaluate this hypothesis. Figure 3 illustrates the conceptual differences between DE Sequential and Vertical approaches as explained in this paragraph. Due to the particular nature of the DE mutation process, mutations of each individual in the population can be carried out in parallel without any competition. This work leverages this particular nature.

5. VNE Competitor Approaches

We compare our DE against GA, Stress, and Greedy virtual network embedding approaches (VNA). There are deterministic and heuristics approaches, each with two versions which differ in the first population formation. In the v1 version, the node selections are totally random and we named this process as *cold start*. In the v2 version, nodes are randomly selected from a pre-created group of nodes. This pre-created group contains a set of nodes from all links that contemplate the VNR demand, and we named this second process as *hot start*.

VNA Stress is presented in [Zhu and Ammar 2006], and is based on node and links stress. Stress is a measure that signals how used a resource is. The authors in [Zhu and Ammar 2006] have proved that it is impossible to obtain optimal nodes and links by mapping them at the same time. Thus, its solution seeks to allocate nodes and links equally, distributing the mapping along with the whole infrastructure. This distribution leverages the increase of VNR acceptance, and that behavior oriented us to select this approach as a strong competitor.

VNA-Greedy is a simple solution that groups real nodes based on their resources and the ones available on the adjacent links. After the grouping, it chooses the nodes that have more resources available to contemplate each requisition. Both VNA-Stress and VNA-Greedy are deterministic approaches. The variation of their results is a stochastic consequence of the random selection of nodes when they have the same available capacity.

Works [Qin et al. 2014, Han et al. 2018, Chen et al. 2019] cite the following GA strengths: (i) suitable for use when the solution space is vast; (ii) the search process is not restricted to local search spaces; (iii) it is flexible enough to be adapted to various scenarios. The evaluation function (*fitness*) and the chromosome structure concentrate the most sophisticated part of adapting the GA heuristic to solve a problem. Both the DE and the GA are evolutionary algorithms and share some concepts, the former also shares some advantages of the latter. Furthermore, the fitness evaluation is the same in GA and DE, but there are some differences: (i) the sequence of creating new individuals (offspring) and mutation is inverted; (ii) the mutation process is different; (iii) in GA, creation and mutation are totally random processes; and (iv) creation and mutation in DE are solution-oriented by the geographic space search.

An individual is a possible solution of mapping, and a chromosome represents it. All individuals are created when VNA-GA is performed. A chromosome is a set of genes, and each gene is a possible map between a virtual node and a real node. If the problem imposes any condition, it must be implicit into the chromosome's representation. The fitness function calculates the chromosome's evaluation, and the result is used to select the most adaptable individual during the offspring process.

The evaluations were carried out using the same set of requisitions and datasets. The requisition set is kept the same to avoid any injustice caused by any difference in resources demand. The repetition and population size parameters are crucial to the performance and efficiency of the evolutionary algorithm (EA). Thus, we chose the following values of the tuples (repetition, population) for the tests: (5,5), (50,5), (100,5), (5,50), (5,100), (10,10), (50,50) for all EA. The total of tests for 10 repetitions, 4 VNR datasets, 7 tuples, 2 network datasets, and 6 (GA, Stress, Greedy, DE, DE-Thread and DE-Processing) approaches, and two versions add up to 6,720 executions. We chose the set of tuples to increase the repetition and population gradually. Over the (100,100), there is no increment in the acceptance rate. The tuples are meta-parameters; consequently, they are not deterministic, particularly those of meta-heuristics.

For all evolutionary algorithms, the creation of the first population is the same. Thus, we guarantee the same initial characteristics for all. Moreover, the fitness function is applied to all heuristic algorithms. In this way, we were concerned with keeping a fair contest among them. The sequence is realized by an NSaaS provider, and it is kept for all VNE approaches. The inner functions embedded in our NSaaS simulator are: (i) recover the set of VNRs; (ii) increase use of resources after the mapping; (iii) release use of resources when a VNR expires; (iv) save performance information from the mapping execution.

6. Fitness Function and its Behaviour

Our fitness function (2) is a special contribution which considers all the QoS parameters (bandwidth, delay and reliability) required in VNRs. Fitness takes the individual's properties as its parameter. The return values are between 0 and $+\infty$, where *zero* means that an individual does not meet all demands. If the return value is 1, it means that an individual fully meets a VNR. Return values greater than 1 mean that an individual represents a larger set of resources than needed. The higher the value, the more unnecessary resources are selected. Based only on the fitness, the GA and DE can select more adapted individuals. The selection process does not exclude the individual whose score is less than 1. In the implementation, all individuals whose score is less than 1 are maintained in the population, and their score is updated with a value of $+\infty$. In this way, it is maintained in the population, and it obtains a lower chance to be selected in the offspring process.

The score is a value that expresses how efficient an individual is, and its equation is defined in 2. The $vMax(r)$ denotes the maximum capacity of a specific-resource in the whole infrastructure, R is a set of resources, such as $r \in R$. The $vGot(r)$ returns a value that denotes the maximum capacity of resource r that the mapping process found during its execution. The $vDesired(i, r)$ returns a demanded capacity of resource r which is required by individual i. The variable r can assume three specific-resources in this work: (i) bandwidth; (ii) (delay); and (iii) reliability. The score is a normalized measure, thus we can carry out different resources at once. Each QoS parameter present in the VNR has its own score. The individual's final score is the average of all scores plus the result of the score logarithm based on the number of hops. The symbol $\diamond$ is the addition operator when the variable (r) represents the bandwidth or reliability and it is the subtraction operator when the variable r represents the delay parameter. Lastly, the variable i denotes the VNR's identifier, which is used to obtain its demands.

$$score(i) = \frac{vMax(r) \diamond (vGot(i) - vDesired(i, r))}{vMax(r)} \tag{2}$$

Equation 3 denotes the final score of an individual. The function *c(r)* in (3) is the coefficient defined for each QoS parameter in a VNR, the variable P denotes a set of parameters from each type of slice. These values result from the provider and tenant negotiation. They are customized to meet the particularity of each tenant demand, e.g., massive machine type communication, ultra-reliable low latency, and enhanced mobile broadband. The coefficients range [0,1], and the greater the number, the higher the relevance. It is possible to create countless types of slices using these coefficients.

$$total = \frac{\sum_i^{|P|} c(P[i]) * score(i)}{|P|} + log(hops, score(i)) \quad (3)$$

Figure 4 illustrates the fitness function behavior based on equations 2 and 3. We have 20 hypothetical scenarios numerated from 1 to 20. The scenarios [11-20] represent situations where the solution searcher (i.e. the VNE approach) encountered resources with incrementally one unit values of bandwidth, delay and reliability greater than the ideal. The ideal is when the algorithm find exactly what the virtual network call for. On the other hand, the scenarios [0-9], represent situations where the solution searcher encountered resources with incrementally one unit values less than the ideal. That is, when the VNE approach find the resources which meet perfectly the virtual network demands its return is 1. All approaches in this work use the same fitness function, thus, we keep equality among the approaches.

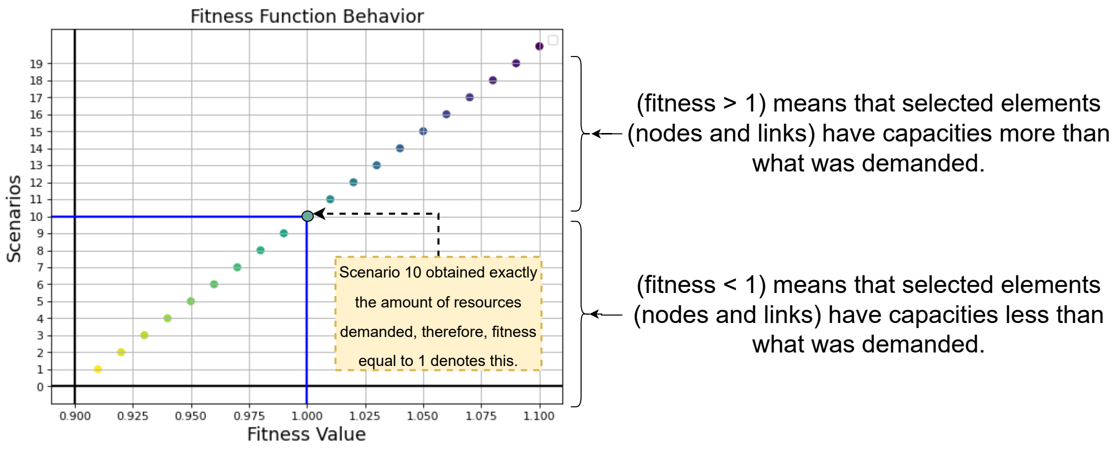


Figure 4. Fitness function behavior.

7. Artifacts, Scenarios and Evaluation

The sequence of tests was executed using two different datasets, which differ from each other in the variation of topology, bandwidth, delay and reliability capacity. There are four sets of VNRs and the number of requisitions is: (i) set 1 has 20; (ii) set 2 has 50; (iii) set 3 has 100; and (iv) set 4 has 150. Each set is kept the same for each different mapping algorithm. Table 1 presents the main properties of Datasets 1 and 2, which have a different number of nodes and links and the latter is more complex than the former. Our framework created these datasets with random values of bandwidth, delay, and reliability whose process is explained in [Gomes et al. 2021]. It is important to note that the values of nodes and links used in Dataset 2 exceed these parameters in all 78 works researched in [Fischer et al. 2013]. For example, about 67% of the works have a number of nodes between 10 and 100, and 51% of the datasets have a number of links between 0 and 400.

Table 1. Dataset properties used in the simulation.

Properties	Dataset 1	Dataset 2
Nodes	112	2,138
Links	125	2,395
Degree	12	50
Bandwidth	[103;1,640;9,881;2,807]	[100;1,215;9,988;2,338]
Delay	[1;121;294;103]	[1;113;300;98]
Reliability	[90;95.7;99;3]	[90;95;99;3]

Values = [minimal; average; maximum; standard deviation];

In the first scenario, all heuristic approaches using the *cold start* were able to map less than 3 VNRs. Thus, we excluded all results with the cold start in this section. In the figures, all heuristic approaches are the second version, which uses the *hot start.* Section 5 explains the cold and hot starts (bootstrap) concepts. Fig. 5 shows the minimal, first quartile, average, third quartile and maximum values of the executions of *Greedy*, *Stress*, DE_{v2t} and DE_{v2p}. Greedy was the only one whose acceptance rate was lower than in all VNR sets. DE_{v2t} and DE_{v2p} use thread and multiprocessing mechanisms, respectively. Using the small Dataset 1, the approaches DE_{v2}, DE_{v2t} e DE_{v2p} obtained similar results.

GA obtained better results in sets 20, 50, 100, and 150 with repetition and population equal (50,50), totaling 2,500 mutation instructions. DE received better results with the same set of VNRs with repetition and population equivalent (5,5), (10,10), (100,5), (100,5) to sets in sets 20, 50, 100, and 150, totaling in average with 281.25 instructions. The version DE using thread and processing reached 337.5 and 937.5 instructions to obtain a better acceptance rate. Regarding the runtime and the sets that received the better acceptance rate, the average runtime was 76.5, 1.1, 1.9, 26.4 minutes to GA, DE_v2, DE_{v2t}, and DE_{p2t}, respectively. Considering the small dataset, the original version of DE had the better runtime. Regarding the parallelization and the small size of Dataset 1, the DE version using Threads and Multiprocessing mechanisms did not reduce the runtime due to the computational effort to create and manage tasks and processes for a small dataset; and with a low value of repetition and population. The number of instructions denotes the efficiency of the algorithm, the smaller the more efficient.

Fig. 6 is related to the second scenario. In this scenarios, the quantity of nodes was increased from 112 to 2,138 nodes, turning the solution search harder. Fig. 6 shows the min, first quartile, average, third quartile and max values of executions of *Greedy*, *Stress*, DE_{v2} and GA_{v2}. The parallel versions were excluded in Fig. 6, due to their acceptance rate were similar to DE.

When the network infrastructure becomes more complex, the search for the solution becomes more challenging and the similarity of results achieved in the first scenario disappears in the second scenario. Fig. 6 shows the DE_{v2} obtained better results in average. Using the Dataset 2, the average of VNRs mapped for Greedy, Stress, DE_{v2} and GA_{v2} in the set with 20 VNRs are 6.0, 11.5, 17.0, and 15.3; in the set with 50 VNRs are 11.0, 38.5, 47.0, and 39.4; in the set with 100 VNRs

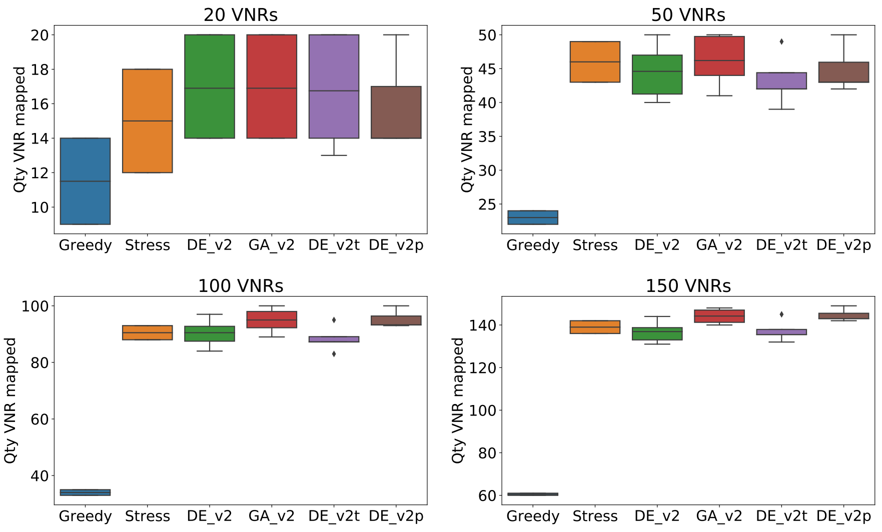


Figure 5. Distribution of VNRs Mapped using Dataset 1.

are 18.5, 78.5, 97.0 and 82.5; and in the set with 150 VNRs are 35.5, 118.5, 147.0, and 128.6. On average the DE approach obtained a acceptance rate better than Greedy, Stress and GA, with an improvement of 76.95%, 19.81%, and 13.70%, respectively. In the second scenario, the runtime of Stress, DE_{v2}, and GA_{v2} in the sets 20, 50, 100 and 150 were: [115, 16, 57]; [302, 13, 128]; [657, 7, 250]; [1123, 8, 387] minutes, respectively. The approach DE was 618.75%, 2,645.45%, 9,285.71% and 1,3937.50% better than Stress in sets 20, 50, 100 and 150, respectively. And DE was 256.25%, 1,063.64%, 3,471.43%, 4,737.50% better than GA in sets 20, 50, 100 and 150, respectively.

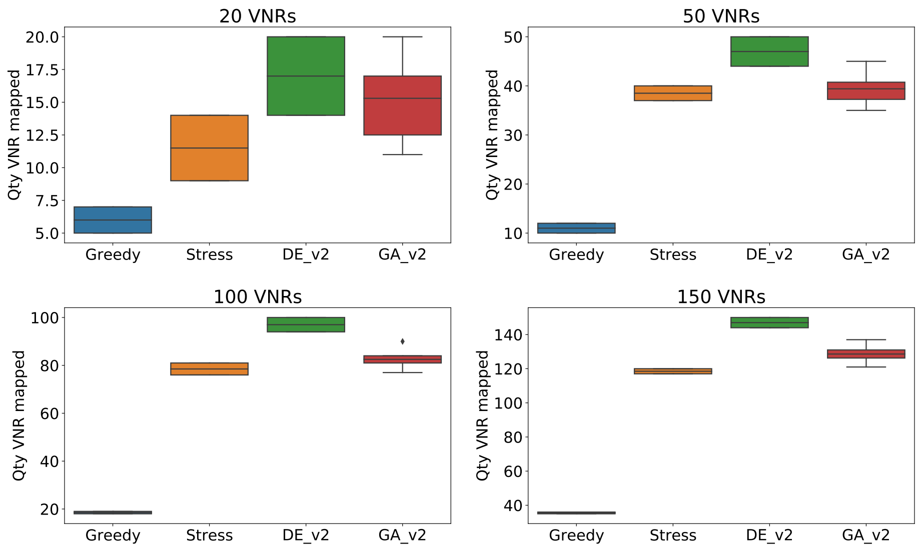


Figure 6. Distribution of VNRs Mapped using Dataset2.

Figure 7 reveals a particular behavior of parallel DE versions. There is a relation of performance with the repetition and population values (R, P). Ideally, parallelized versions have advantages with the increment of P, keeping the same number of sub-processes equals the number of CPUs. The no-parallel DE version has an advantage in performance when R and P are low. Regarding the set of 20 VNRs, Dataset 2, and the same population, the DE_{v2t} improved 9.52% from (5,5) to (50,5) and and 9.97% from (50,5) to (100,5). DE_{v2t} improved 21.04% from (50,5) to (100,5).

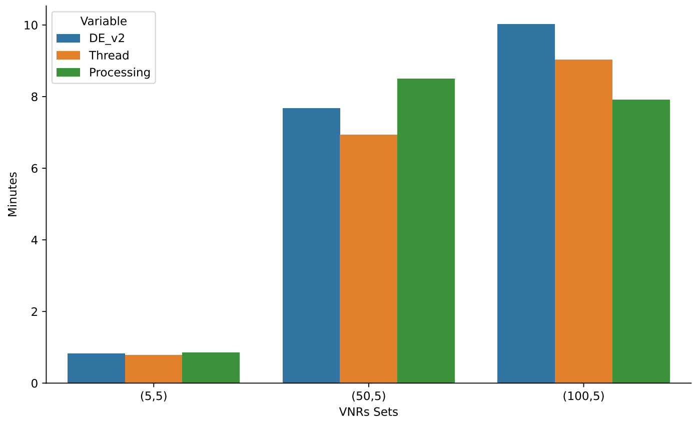


Figure 7. Elapsed time to map using the DE, Thread and Multiprocessing approaches using Dataset 2 and Set with 20 VNRs.

8. Final considerations and Future Work

We formulated a new fitness function to increase the number of requests taking into account several QoS parameters. Parallelism is only better when the number of subprocesses is equal to the number of CPU and when the value of the population increase. In tests, the unparalleled versions won only when the repetition and population have low values; as the population value increases, the parallel performance decreases the execution time (the smaller, the better). Also, we showed the conditions under which the parallelized versions obtain better results. A heuristic does not know the ideal repetition and population size in a given scenario; however, the provider can choose the best for these values based on a history of executions. Taking the results of mapping histories is attractive to be used in machine learning techniques, and we will consider this possibility for our future work. Finally, the GA approach only managed a competitive number of mappings with a high value of the population's size. Consequently, its execution time was longer than all the tested approaches. The results show that the randomness of the initial population is what more influences AG. At the same time, repetition favors the DE, which means that DE is more intelligent in searching for the ideal solution.

References

5G PPP Architecture Working Group (2016). View on 5G Architecture. *White paper*, (July).

5GPPP (2019). View on 5G Architecture. Technical Report June, 5GPPP.

Cao, H., Yang, L., Liu, Z., and Wu, M. (2016). Exact solutions of VNE: A survey. *China Communications*, 13(6):48–62.

Chen, L., Abdellatif, S., Simo Tegueu, A. F., and Gayraud, T. (2019). Embedding and re-embedding of virtual links in software-defined multi-radio multi-channel multi-hop wireless networks. *Computer Communications*, 145(July):161–175.

Fischer, A., Botero, J. F., Beck, M. T., De Meer, H., and Hesselbach, X. (2013). Virtual network embedding: A survey. *IEEE Communications Surveys and Tutorials*, 15(4).

Gomes, R., Vieira, D., and Franklin de Castro, M. (2021). Differential evolution for vne-5g scenarios. In *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–6.

Han, B., Lianghai, J., and Schotten, H. D. (2018). Slice as an Evolutionary Service: Genetic Optimization for Inter-Slice Resource Management in 5G Networks. *IEEE Access*, 6(c):33137–33147.

Nguyen, K., Lu, Q., and Huang, C. (2020). Efficient virtual network embedding with node ranking and intelligent link mapping. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–5. IEEE.

Nguyen, K. T. and Huang, C. (2019). An intelligent parallel algorithm for online virtual network embedding. In *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE.

Qin, Z., Denker, G., Giannelli, C., Bellavista, P., and Venkatasubramanian, N. (2014). A software defined networking architecture for the internet-of-things. *IEEE/IFIP NOMS 2014*.

Salimifard, K. and Bigharaz, S. (2020). The multicommodity network flow problem: state of the art classification, applications, and solution methods. *Operational Research*, pages 1–47.

Tasoulis, D., Pavlidis, N., Plagianakos, V., and Vrahatis, M. (2004). Parallel differential evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 2023–2029 Vol.2.

Vassilaras, S., Gkatzikis, L., Liakopoulos, N., Stiakogiannakis, I. N., Qi, M., Shi, L., Liu, L., Debbah, M., and Paschos, G. S. (2017). The Algorithmic Aspects of Network Slicing. *IEEE Communications Magazine*, 55(8):112–119.

Wu, H., Zhou, F., Chen, Y., and Zhang, R. (2020). On Virtual Network Embedding: Paths and Cycles. *IEEE Transactions on Network and Service Management*, 4537(c):1–14.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: Substrate support for path splitting and migration. *Computer Communication Review*, 38(2):19–29.

Zhou, Z., Chang, X., Yang, Y., and Li, L. (2016). Resource-Aware virtual network parallel embedding based on genetic algorithm. *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, 0:81–86.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. *Proceedings - IEEE INFOCOM*.