Parallel Differential Evolution Meta-Heuristics and Modeling for Network Slicing in 5G Scenarios

Rayner Gomes
Information System Department
Federal University of Piauí (UFPI)
Picos, Brazil
email:rayner@ufpi.edu.br

Dario Vieira
Research and Innovation Department
Efrei Paris - France
Paris, France
email:dario.vieira@efrei.fr

Miguel Franklin de Castro
Computer Department
Federal University of Ceará (UFC)
Fortaleza, Brazil
email:miguel@ufc.br

Leonel Feitosa
Information System Department
Federal University of Piauí (UFPI)
Picos, Brazil
email:leonelfeitosa@ufpi.edu.br

Francisco Airton Silva
Applied Research to Distributed Systems Lab - PASID
Federal University of Piauí (UFPI)
Picos, Brazil
email:faps@ufpi.edu.br

***Abstract*—Network Slicing is the crucial component to face the Internet's ossification and support the heterogeneity of 5G scenarios. A slice is a virtual network mapped over a real network. Mathematically, the mapping is known as Virtual Network (VN) Embedding (VNE), which 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 $\mathcal{NP}$-hard problem, multiple efforts have been made to create VNE algorithms. Considering such efforts, this work presents: (i) a fitness function regarding multiobjective optimization to maximize the quantity of embedding; (ii) two new parallel Differential Evolution (DE) approaches to face the VNE problem; and (iii) a Stochastic Petri Net to model our approaches. We designed these two versions due to the lack of viable parallel solutions in the 5G scenario. We compared three approaches with two different infrastructure datasets and nine DE setups. The results demonstrate that the better parallel version reduced the runtime by 68.94% and 51.03% using datasets 1 and 2, respectively, and kept the same acceptance rate. The parallel performance decreases the runtime in certain conditions, and we explain in which scenarios the parallel approaches obtain advantages and disadvantages.**

***Index Terms*—network slicing; meta-heuristics; differential evolution; parallelization; 5G; VNE.**

I. Introduction

Intelligent connectivity for the smart world is presented in the fifth-generation mobile networks (5G) envision. A highly flexible 5G infrastructure is crucial to prevent the ossification problem in which the Internet suffers. 5G aims to support new markets such as device-to-device communication, machine-to-machine communication, and the Internet of Things (IoT). IoT is an excellent example of market diversity, such as smart homes, smart cities, smart grids, smart traffic, and other intelligent environments diffused in modern society.

Over a "softwarerized" network composed of Software-Defined Network (SDN), Network Function Virtualization (NFV), and Cloud paradigms, the 5G provider can create network slices, and each being is a portion of the network configured to meet all tenant requirements, overcoming the ossification problem [1]. The 5G provider seized by the Network Slice as Service (NSaaS) allows each tenant to require an end-to-end network, i.e., network slice. The NSaaS waits for some Virtual Network Requests (VNRs) that can arrive at any time. A VNR is a formal document described by network topology, and QoS demands. The NSaaS carries out the mapping as a primordial task to map virtual networks to physical resources, mathematically known as Virtual Network Embedding (VNE).

The VNE is a problem that has been receiving exhaustive attention from researchers. Its $\mathcal{NP}$-hardness can be proven by reducing it to *the multiway separator problem*, and [2] [3] have proved its complexity in different ways. VNE is solved by adopting the use of meta-heuristics. There are exact solutions in the literature; however, they are only used with small networks, which is inappropriate in 5G. As pointed out by [4], the heuristic solution aims to find a good solution with low execution time when facing realistic network scenarios. Another important point is that the meta-heuristic solution can improve the quality of the result by escaping from the local optimum.

The predominant evolutionary meta-heuristics approaches in Artificial Intelligence found in the literature to deal with VNE problems are Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Our previous work [5] presents a Differential Evolution (DE) approach to the VNE problem facing how to: (i) apply the DE heuristic to the VNE problem; (ii) map mathematical operations to node and link operations; and (iii) design a fitness function regarding different QoS parameters. Our last work focused on evaluation for better placements

 ISBN: 978-1-61208-965-2

to each requisition, and, for that, it proposed two measures named *node accuracy* and *accuracy gain*. However, this work seeks to enhance the runtime. Moreover, our mentioned work examined that the DE approach takes more time to be executed regarding the same number of repetitions and population size when compared to the GA. Hence, in this work, we extend our work [5] by presenting two parallel versions to reduce the execution time of the DE, and adopts two more complex network topologies. Furthermore, to the best of our knowledge, we are the first to present a Stochastic Petri Net (SPN) to model and evaluate the NSaaS system.

Our work [5] is the first one to apply DE in VNE problems. We demonstrated the capacity of its approach to select the best places in the network to a VNR. The 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 in a VNR. The measure function returns the better places before the execution and, for this reason, the network used in the simulations is small, with about 20 nodes. Four points distinguish this work from [5]: (i) it is not oriented to increase the accuracy, but the quantity of requisitions mapped; (ii) the evaluations use two large datasets, which is why we do not know a priory what part of the network is better for each requisition; (iii) it considers three DE versions, called Sequential, Vertical and Horizontal; and (iv) the main focus of this work is to reduce the runtime.

Finally, 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*. Sections VII presents the crucial artifacts for carrying out the assessments; Section VIII compares all the algorithms and analyzes the results; and, finally, Section IX concludes this work by showing some opportunities for future works.

II. Related Work

We are interested in the VNE-logic parallelization, where the meta-heuristic divides the solution into computation units to reduce the runtime. Fischer et al. [6] did a detailed survey in the network embedding area, and they proposed a taxonomy of embedding algorithms regarding three different dimensions: static and dynamic, centralized and distributed, and concise and redundant. Despite the scope of work, there is only one work dealing with parallelization, and that unique work considers the parallelization of mapping one node in two different substrates, and it is not about VNE-logic parallelization.

We have submitted a search on Scopus considering the title, abstract, and keywords with values "virtual network embedding and 5G" 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 parallelism.

One contribution of [7]–[9] is a solution to VNE based on a 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 [10] 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.

Regarding NSaaS' envision for the 5G system, the mapping processing must consider different QoS parameters. Our work is unique in that it considers the bandwidth, delay, and reliability at once. Moreover, unlike [7]–[9] we do not create a pool of paths. Instead, the path selections result from the nature of the DE approach. Thus, our solution is more scalable.

There are three approaches to dealing with VNE: exact, heuristic, and meta-heuristic. The exact solution is the optimal solution to the VNE problem, usually formulated by the optimization theory. The exact solution is fixated on the global optimum and can easily suffer from the problem of being stuck in a local optimum, which is far from the true optimum of the optimization theory. Finally, the heuristic solution aims at finding a suitable solution with low execution time under realistic network scenarios.

The meta-heuristic solution can improve the quality of results by escaping from the local optimum [4] [6]. Based on a review of 125 papers from [6], and our systematic review of 43 documents, the most evolutionary meta-heuristic used are GA, Ant-Colony, and PSO. In our previous work [5], we presented the adaptation of the DE to the VNE problem, and, in this work, we extended the number of parallel approaches. To the best of our knowledge, this work is the first to reveal two versions of parallel DE approaches to the VNE problem regarding 5G scenarios and presents an NSaaS model following the Stochastic Network Petri (SNP) modeling paradigm.

III. VNE Problem Description

A comprehensive modeling is presented in [5], 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 vertex 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. 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 δn_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 ∈ N to a physical node $v \in V$; (ii) 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. A slice behaves like a physical network for a slice user, 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 control of used resources. Moreover, VNE algorithms aim to find a feasible embedding respecting all QoS parameters demanded by a VNR. Duration d represents the time when a slice must exist, and this information is relevant to online operations, where each slice has a 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 this equation means we seek to maximize the quantity of mapping (σ) in the current cycle (c), thus, maximize the acceptance rate (AR). The constraint variables and problem formulation can be found in [5]; we have retained the same constraints.

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

IV. Parallel Differentiated Evolution to VNE Problems

As explained in Section II, there is a gap in the literature on parallel heuristics to tackle VNE problems. Hence, this work details two ways to parallelize the VNE-DE approach named the Vertical and Horizontal DE approaches. The main goal of implementing the parallel solutions is to try to reduce the execution time of the DE to perform the mapping. The nature of the heuristic/logic will determine how to make a parallelization. The main advantage of creating tasks/sub-processes to execute the mutation is that the parallel version can carry the mutation out simultaneously. The hypothesis is that the DE feature can allow creating a subprocess to reduce runtime.

This article evaluates three approaches called Sequential, Vertical, and Horizontal. The Sequential approach has the same logic as the VNE-DE approach presented in our previous work [5], and it is considered the benchmark algorithm. The difference between the last work and then is related to fitness functions.

The Vertical approach (see Algorithm 1) is a modification of our sequential algorithm presented in [5]. The modified algorithm has the outermost loop (line 3) and two internal loops (command *for*, line 4 e 8). The inner loops are the code blocks for the mutation (lines 4-6) and the selection phase (lines 8-10). In addition, DE has the particularity of performing each mutation separately. This feature favors the creation of parallel subprocesses. In the original code, the Sequential approach, these two blocks are sequential instructions. In the vertical parallelization approach, the main code is kept non-parallelize. However, it parallelized the inner codes in new tasks/sub-processes. The selection phase starts after the mutation of all individuals and it can be parallelized too.

Algorithm 1: Parallel Differentiated Evolution Algorithm

```
   Input : fitness, lb, ub, Np, T, F, Pc
1  P ← initPopulation(P);
2  Vi ← createDonor();
3  for t = 1 to T do
   |  /* Mutation block                                   */
4  |  for i = 1 to Np do
5  |  |  Pi ← process (Ui, i, runCrossover);
6  |  end
7  |  join (P);
   |  /* Selection block                                   */
8  |  for i = 1 to Np do
9  |  |  Pi ← process (Ui, Vi, i, lb, ub, bound, fitness,
   |  |    greedSelection);
10 |  end
11 |  join (P);
12 end
```

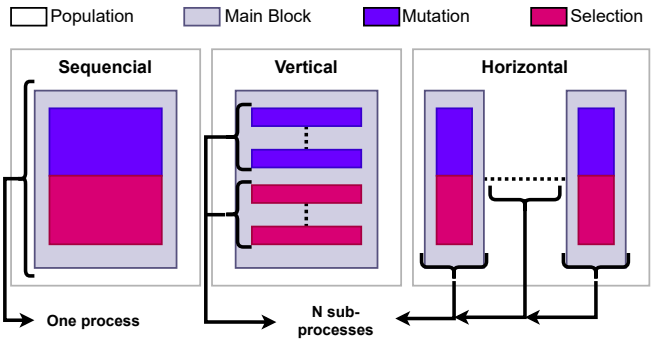

Figure 1: Difference in how the DE code blocks are processed.

The Horizontal approach divides the first population into groups of the same size. Then, it creates new processes to execute each group, in order that each group is tread with a Sequential Approach. The possible advantage of the Horizontal approach came from the aspect each subgroup can be performed in simultaneous CPUs. In the final process, the results from each sub-group are joined in one final population, shorter than the original, and this final population passes by the final sequential processing.

V. Fitness Function and its Behaviour

Our fitness function (2) is a special contribution. This equation 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 $+\infty$. Zero means that an individual does not meet all demands. The return value equals to 1 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.

Based only on the fitness, the DE can select more adapted individuals. The selection 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 $+\infty$, so 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. Each individual i is associated to a VNR.

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

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.

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

VI. NSaaS Modeling

This section presents the Stochastic Petri Net (SPN) model and the execution flows. Section VII presents two substrate datasets in which we submit the model. As described in Table II, each one has divergent properties.

The model is an essential artifact to compute the Mean Time to Absorption (MTTA) [11]. We used the Mercury Tool, version 5.0.1 [12] to design the Petri Net model, and through it, it was possible to carry out the data analysis. Mercury is a powerful tool and can be used to process availability, reliability, and other analysis.

TABLE I: Places and transitions denotations

Name	Denotation
P1	The repository of tokens, each token means a VNR.
Ti1	The immediate transition represents the entry of the request into the system. All transitions are immediate, and there is no delay associated with its execution.
T1	The preparation of the system to cope with the requisition.
T2	System obtaining infrastructure status.
Network	This place means the network substrate.
VNR	The repository of VNRs.
T3	Decoding the VNR.
T4	Creating the first population used in the VNE approaches.
Population	The place means the population element used in the approaches.
T5	Fitness calculation of each token.
P5[a,b,b,d]	There are specific places present in each approach. These places represent positions or steps used in each DE version (Sequential, Vertical e Horizontal).
T7a	They represent the flow of message processing after going through the mapping process. In this flow, the resources are allocated to the mapped VNR.
T7b	Similar to T7a, but in this flow, the resources are not allocated to the mapped VNR.
T8	Resources release to each VNR expired.
P13	Synchronization position, token means the system is ready to deal with another VNR.
T5a	Initialization of DE structures, trial, donor and target vectors.
T5b	Mutation of chromosomes.
T5c	Selection of individuals.
T6a	Checking if VNR can be meet.
T6b	Checking if VNR cannot be meet.
T5a1	Population division.
T5b[1-5]	Parallel mutation in each individual.
T5c[1-5]	Parallel selection in each individual.
T5d	Reordering individuals in the population when Vertical Embedding case.
T5d	Grouping sub-population in a new population when Horizontal Embedding case.
T5a2	Creation of population sub-groups.
T5a3	Creation of processes to deal with each sub-group of population.

The model has two main structures, named Base and Embedding. Base and Embedding compound the NSaaS system. The Base part, the blue rectangle in Figure 2, involves the NSaaS steps related to accepting the VNRs, the environment preparation to processing requests, and allocation and release of resources. The Embedding part, the red rectangle from the NSaaS rectangle, abstracts three VNE algorithms used in our tests. Thus, the Sequential, Vertical, and Horizontal Embedding are interchanged in the NSaaS. We have adopted this strategy to present the model to not redraw the whole Base for each Embedding due to space restrictions.

Table I describes the main places and transitions. We used the initial letters P to places, but all transitions start with the initial letter T.

 ISBN: 978-1-61208-965-2

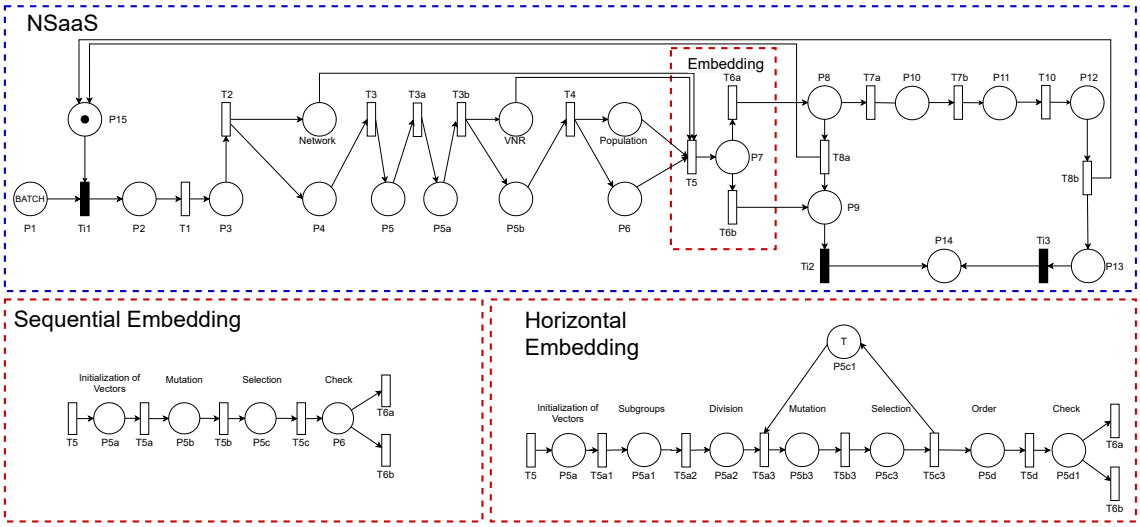

Figure 2: NSaaS (Base), Sequential, Vertical and Horizontal Embedding models.

VII. Artefacts and Scenarios for Evaluation

The datasets were created using the sequence of steps presented in [5]. 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. The existence of these datasets consists of an attempt to make the solution search harder. Table II 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 [5].

TABLE II: 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]		

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. One VNR is composed of (a) VNR identification, (b) virtual nodes demands (vnd), (c) links, (d) type of slice, (e) bandwidth demand, (f) delay demand, and (g) reliability demand. The structure of the VNR is presented in [5], and we kept the same in this work.

VIII. Results of executions

This section presents the actual results from the sequential, vertical, and horizontal DE implementations. It is important to note that the number of VNRs mapped must be the same before analyzing the approach runtimes. There is no advantage in reducing the runtime if the number of mapped VNRs decreases, i.e., if the derived approaches do not sustain the mapping performance. We do not expect any increment. What we desire is to reduce the runtime with these two new versions of DE.

Figure 3 contains four plots in two lines; the first and second one are the results using datasets 1 and 2, respectively. Each plot contains the minimal, first quartile, average, third quartile, and maximum quantity of VNRs mapped regarding nine tuples ([(5, 25), (50, 25), (100, 25), (5, 50), (50, 50), (100, 50), (5, 100), (50, 100), (100, 100)]). Considering Dataset 1, the Vertical and Horizontal obtained a slight advantage. Regarding Dataset 2, we can set a large area of intersection. The standard deviation (STD) of sequential, vertical, and horizontal are 3.8, 3.3, and 3.3, respectively. The approaches behave similarly while considering the number of mapped requisitions.

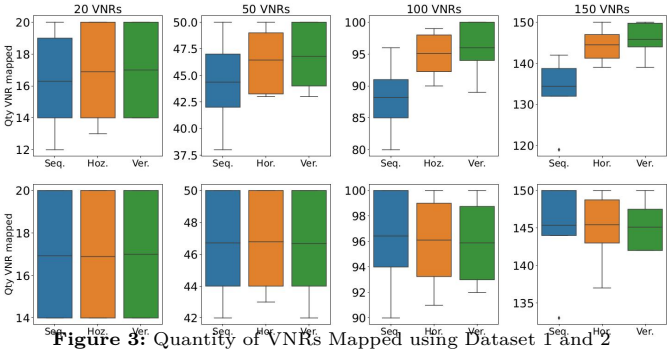

Figure 3: Quantity of VNRs Mapped using Dataset 1 and 2

The VNRs mapping results show one equilibrium in the accepted rate, and now our interest is to analyze the runtimes. Figure 4 presents the difference of runtime between the Sequential and Horizontal approaches. We excluded the Vertical approach because it reaches a high runtime. The bottleneck of the process creation in the Vertical approach causes these high values of the execution time. One way to overcome this bottleneck is that the Vertical approach should use Threads instead of Processes to multitask. However, the Python language does not im-

plement real parallelism with Threads. Both figures show the Horizontal approaches won in all sets of requisitions and both datasets. Regarding the *Setup* with repetition and population values equal to (5,50) and (50,50), the Horizontal obtained a runtime about 68.94% and 51.03% lower than Sequential.

Figure 4 presents how the parameter *Repetition* influences in the time. Each plot has four bars with Setup (repetition, population) equals to (5,50) and (50,50). In the tests, we keep the size of the population to 50. What we want to show is how the parameter repetition influences the runtime. Changing the repetition from 5 to 50, the Horizontal obtained an advantage of 68.94% to 51.03%. The advantage came from the system's ability to deal with up to the five processes simultaneously, and any value higher than five will trigger a wait-time among the processes. Therefore, the difference of 17.92% is due to concurrency. In an infrastructure with more available CPUs, e.g., in a Cloud, the Horizontal approach will behave better with more CPU allocations.

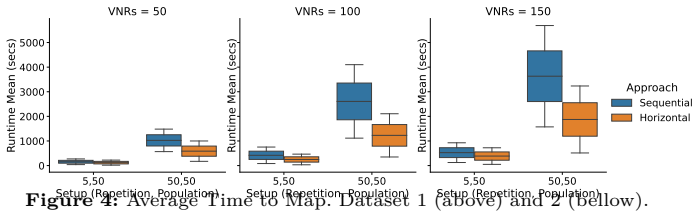

Figure 4: Average Time to Map. Dataset 1 (above) and 2 (bellow).

Figure 5 shows four evaluations between the model and the sequential and horizontal. Each evaluation considers four sets of requisitions (20, 50, 100, and 150).

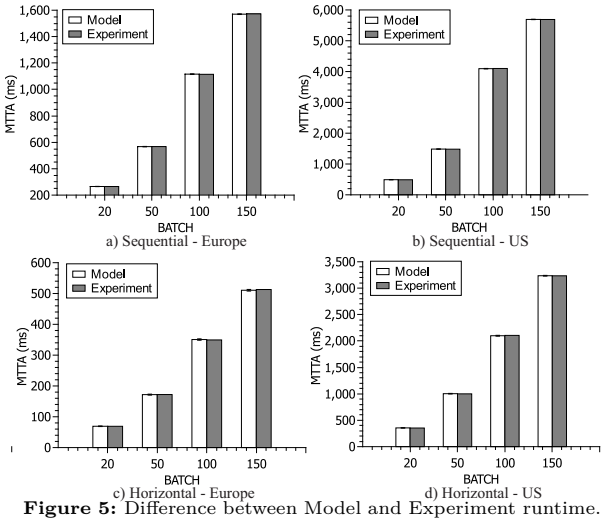

Figure 5: Difference between Model and Experiment runtime.

For the validation process, the *T Test* was performed for two samples where the MTTAs values found in the simulations and experiments were compared. The P-Values are greater than 0.05, implying that there is no evidence that both means are statistically different. Due to the limited space for writing in the article, the table with the respective values will not be displayed.

IX. Conclusion

In this work, we evaluated three versions of the Differential Evolution Meta-Heuristic (DE) applied in the VNE problem. The first version is presented in detail in our previous work [5]. The premise to adapt the DE in VNE is because of its efficiency and nature that favor its adaptation to parallel versions. We presented two ways to parallelize the DE approach. Also, we showed the two parallelization ways and their evaluations. Finally, we focused on reducing the runtime. We demonstrated that the two parallelized versions obtained, on average, the same quantity of VNRs, and it is a prerequisite; there is no reason for the provider to reduce the runtime if at least the efficiency was maintained. Due to our process capacity, we limited the quantity of sub-process to five, bottlenecking the Horizontal performance; however, as demonstrated in the results, it achieved the best results.

References

[1] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network Slicing for 5G: Challenges and Opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, 2017.

[2] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I. N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, and G. S. Paschos, "The Algorithmic Aspects of Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 112–119, 2017.

[3] H. Wu, F. Zhou, Y. Chen, and R. Zhang, "On Virtual Network Embedding: Paths and Cycles," *IEEE Transactions on Network and Service Management*, vol. 4537, no. c, pp. 1–14, 2020.

[4] H. Cao, L. Yang, Z. Liu, and M. Wu, "Exact solutions of VNE: A survey," *China Communications*, vol. 13, no. 6, pp. 48–62, 2016.

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

[6] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, 2013.

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

[8] K. Nguyen and C. Huang, "Distributed parallel genetic algorithm for online virtual network embedding," *International Journal of Communication Systems*, 12 2020.

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

[10] K. Salimifard and S. Bigharaz, "The multicommodity network flow problem: state of the art classification, applications, and solution methods," *Operational Research*, pp. 1–47, 2020.

[11] F. A. Silva, S. Kosta, M. Rodrigues, D. Oliveira, T. Maciel, A. Mei, and P. Maciel, "Mobile cloud performance evaluation using stochastic models," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1134–1147, 2017.

[12] A. Lobo, R. Matos, B. Silva, and P. Maciel, "Expolynomial modelling for supporting vanet infrastructure planning," in *2017 IEEE 22nd Pacific rim international symposium on dependable computing (PRDC)*. IEEE, 2017, pp. 86–91.

 ISBN: 978-1-61208-965-2