# Tenant-Oriented Resource Optimization for Cloud Network Slicing with Performance Guarantees

Lucian Beraldo
*Department of Computer Science*
*Federal University of São Carlos (UFSCar)*
São Carlos, SP, Brazil
lucian@estudante.ufscar.br

Angelos Pentelas
*Department of Applied Informatics*
*University of Macedonia*
Thessaloniki, Greece
apentelas@uom.edu.gr

Fábio Luciano Verdi
*Department of Computer Science*
*Federal University of São Carlos (UFSCar)*
Sorocaba, SP, Brazil
verdi@ufscar.br

Panagiotis Papadimitriou
*Department of Applied Informatics*
*University of Macedonia*
Thessaloniki, Greece
papadimitriou@uom.edu.gr

Cesar A. C. Marcondes
*Computer Science Division*
*Aeronautics Institute of Technology (ITA)*
São José dos Campos, SP, Brazil
cmarcondes@ita.br

***Abstract*—Cloud Network Slicing (CNS), emerging alongside the 5G mobile network, comprises a paradigm shift in the way networks are provisioned, managed, and operated. Fundamentally, CNS fosters the deployment of a multitude of modern applications, *e.g.*, virtual and augmented reality, 4K video streaming, and autonomous vehicles, which require ultra-low latency, high bandwidth consumption, or both. Slicing promotes the realization of such services through the allocation of computing and network resource bundles, which, as CNS mandates, are isolated from the rest of the network. Typically, such resources are arranged into wide geographical areas (*e.g.*, into multiple countries or even continents), which implies that it is possible to allocate from multiple infrastructure providers. This exacerbates the already challenging problem of maximizing resource allocation efficiency, a feature commonly addressed by CNS architectures.**

**In this respect, we study the optimized embedding of slices across multiple domains. Therefore, we account for slices as a collection of computing and network parts. Given specific resource requirements from slice tenants and potentially multiple offers per slice part, we model the problem as a Mixed Integer Linear Program (MILP). We further design two heuristic algorithms in order to mitigate the complex intricacies that would be perceptible in large problem instances. Our evaluation results, based on a simulation environment aligned with the NECOS slicing architecture, indicate that the MILP approach yields better efficiency compared to both heuristics, with respect to client expenditure with a fair amount of performance parameters in an adequate execution time. Our main contribution lies in the optimization methods based on the split and combine approach, integrated into the NECOS CNS architecture.**

***Index Terms*—Cloud computing, network softwarization, slicing, linear programming, optimization**

## I. INTRODUCTION

The slice abstraction represents a fundamental paradigm shift in the relationship between clients and service providers. Bandwidth-demanding applications, such as 4K and Virtual Reality (VR) video streaming are now becoming widespread, stressing the need for better-prepared infrastructures [1], [2]. With the increasing popularity of devices capable of reproducing such content, *e.g.* smart TVs, smartphones, tablets, VR headsets, in a Video on Demand (VoD) fashion, such demand is even higher. To guarantee the Quality of Experience (QoE) of those services, computer networks' availability, reliability, and stability are crucial points. Additionally, such characteristics, coupled with the ultra-low latency requirement, are vital to actualize the mass adoption of connected vehicles, *e.g.*, cars and delivery drones.

Cloud Network Slicing (CNS) is a concept emerging with the fifth-generation mobile network (5G), which proposes to isolate network traffic, infrastructure resources (*i.e.*, both network and computing) and replicate content across multiple domains and federations. This concept mainly relies on the efficient choice of resources across several domains, federations, and providers, sometimes posing as hard to manage. A key-enabler of effective network slicing consists in the efficient selection of resources across several domains, federations, and providers, which adds further complexity to the already challenging problem of resource allocation in virtualized infrastructures. The appropriate choice of resources can effectively reduce the load from bustling servers to conserve computing power and guarantee response times close to 1 ms. This could also lead to efficient power consumption and enable the choice of energy-efficient computing equipment for tenants, moving towards sustainability [3] in green computing [4].

In this work, we present a split and combine optimization model, which aims at near-optimal slice performance. In particular, our model accounts for network and cloud *Resource Options* from multiple providers, which renders it aligned to a real-life scenario. This augments the exploration of solutions that result in low cost, while adhering to performance and capacity requirements, imposed by the slice constituents, *i.e.*, nodes and edges. In order to fulfill this objective, we leverage on the 5G slicing architecture from the NECOS project, which offers a resource pool marketplace [5]. From an algorithmic

This work was partly supported by the H2020 4th EU-BR Collaborative Call, under the grant agreement no. 777067 (NECOS - Novel Enablers for Cloud Slicing), funded by the European Commission and the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through RNP and CTIC and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

perspective, we tackle this problem by designing a Mixed-Integer Linear Program (MILP) and two heuristics. The former encompasses the necessary resource constraints, as well as an objective function that aims at minimizing the cost of slice deployment. The two heuristics are mainly developed to cope with the problem's complexity, which will be perceptible in large problem instances, *e.g.,* many slice parts and many *Resource Options*.

Our results indicate the MILP as the best choice for selecting *Resource Options* with the lowest monetary cost, while providing sufficient infrastructure resources for the tenant. Nevertheless, these efficiency gains come at the expense of higher solver runtime and CPU consumption, compared to both heuristics. Yet, we demonstrate that the load caused by our algorithm on the workstation used to run the proposed scenarios was manageable. The experiments show promising results on optimal resource allocation with a minimum financial burden to the tenant. Finally, this work's critical contribution is the new MILP method for choosing providers' *Resource Options* in the CNS context. In particular, the main contribution of our work stands on the optimization methods based on the split and combine approach, introduced within the novel NECOS' CNS architecture.

In the rest of the paper, we delve into the details of the CNS architecture upon which our work is based, in Section II. Subsequently, we elaborate on related work in Section III. In Section IV, we present our mathematical model and our optimization methods, as well as our constraint reasoning. Then, in Section V, we present some realistic CNS scenarios and perform simulations using the developed models. Finally, we discuss future work plans and conclusions in Section VI.

## II. 5G CLOUD SLICING ARCHITECTURE

NECOS proposes novel techniques and proofs of concept to make CNS tangible. While NECOS introduces a holistic approach to address many aspects concerning CNS, our work assumes a simplified version of it. To delve into NECOS details, we refer to the project deliverables[1]. Hence, in this simplified version, the main actors are a slice tenant and multiple infrastructure providers. The tenant, who can be an individual or an organization, requests a slice instantiation. The slice represents the resource requirements across the network, which could be potentially scattered within a large geographical area and could be owned by different providers. Consequently, the tenant may subdivide the request into what we refer to as *Slice Parts*, *i.e.* the constituent components of a CNS. How the tenant should decompose its request into multiple *Slice Parts* is not considered in this paper. However, a work addressing this problem is [6].

The submitted tenant's request is organized as a YAML file. It is converted into a *Partially Defined Template (PDT)* (see Listings 1 and 3), which is forwarded to the *Slice Broker*. The *Slice Broker* is responsible for choosing *Slice Agents*, the provider's side elements of the NECOS architecture, who will compete with offers for the *Slice Part* request by replying with *Resource Options* comprising pricing and resource details. The individual *Slice Agents* (one per participating infrastructure provider) search within their respective infrastructures, in order to discover resources and push them to fulfill the tenant's request. In this sense, there are no resource requests made to the *Slice Agents*, so they advertise newly available resources to the *Slice Broker* according to the provider scheduling. Such offerings are conveyed to the *Slice Broker* in the form of *Resource Options*. Providers send their *Resource Options* to the *Slice Broker* using another YAML file called *Partially Defined Template with Resource Alternatives* (or SRA), depicted in Listing 2 and Listing 4. The *Slice Broker* joins all SRAs collected from the various providers and forwards them in a single YAML file to the *Slice Builder*. This paper's methods and discussions revolve around this final step. The *Slice Builder* uses the PDT and the SRA YAML files to choose the best *Slice Part* for the tenant, and thus optimization is required.

```
dc-slice-part:
  name: dc-slice1
  slice-constraints:
    geographic: brazil.sp.saocarlos
  resource-priority:
    memory-mb: 0.2
    storage-mb: 0.2
    cpu-number: 0.2
    power-kwh-day: 0.4
  cost:
    dc-model:
      model: \
COST_PER_PHYSICAL_MACHINE_PER_DAY
      value_euros: {<=: 10}
  slice-timeframe:
    service-start-time: {100918:10pm}
    service-stop-time: {101018:10pm}
  vdus:
   - dc-vdu:
     id: load_balancer
     epa_attributes:
       host_epa:
         cpu-model: PREFER_COREi9
         cpu-arch: PREFER_X86_64
         cpu-vendor: PREFER_INTEL
```

Listing 1: *DC Slice Part* tenant's request YAML.

```
dc-slice-part:
  name: dc-slice1
  slice-constraints:
    geographic: brazil.sp.saocarlos
  cost:
    dc-model:
      model: \
COST_PER_PHYSICAL_MACHINE_PER_DAY
      value_euros: 9
  slice-timeframe:
    service-start-time: {100918:10pm}
    service-stop-time: {101018:10pm}
  dc-slice-controller:
    dc-slice-provider: IBM
  vdus:
   - dc-vdu:
     id: load_balancer
     epa_attributes:
       host_epa:
         cpu-model: COREi9
         cpu-arch: X86_64
         cpu-vendor: INTEL
         storage-mb: 6000
         memory-mb: 8092
         power-kw-day: 30
```

Listing 2: *DC Slice Part* offer YAML.

Source: Example Adapted from [7]

```
net-slice-part:
  name: net-slice1
  slice-constraints:
    geographic: brazil.sp.saocarlos
  cost:
    dc-model:
      model: \
COST_PER_PHYSICAL_MACHINE_PER_DAY
      value_euros: {<=: 45}
  slice-timeframe:
    service-start-time: {100918:10pm}
    service-stop-time: {101018:10pm}
  links:
   - dc-part1:
     name: dc-slice1
   - dc-part2:
     name: dc-slice2
  constraints:
    bandwidth: 10 #Minimum
```

Listing 3: *Net Slice Part* tenant's request YAML.

```
net-slice-part:
  name: net-slice1
  slice-constraints:
    geographic: brazil.sp.saocarlos
  ...
      model: \
COST_PER_PHYSICAL_MACHINE_PER_DAY
      value_euros: 45
  ...
  wan-slice-provider: telefonica
  links:
   - dc-part1:
     name: dc-slice1
     dc-slice-provider: IBM
   - dc-part2:
     name: dc-slice2
     dc-slice-provider: Amazon
  constraints:
    bandwidth: 10
```

Listing 4: *Net Slice Part* offer YAML.

We should note that the allocation of the tenant's request is an immensely compounded task. A CNS with too many *Slice Parts* could lose practicality since there could be no physical parts that fulfill the performance requirements. Another architectural concern is the need for providers to conceal resource information details from their competitors [8]. To this end,

[1]http://www.h2020-necos.eu/

the providers only disclose information that is relevant to the tenant's request.

## III. RELATED WORK

Before we delve into our optimization model, to assess an overall state-of-the-art in optimization methods for selecting cloud infrastructure resources, we used a systematic review in the literature. This intended to search and organize the related work. Next we present the most relevant papers used for our work.

NESTOR [6] comprises, among others, a MILP approach to choose between VNF resources. It relies on a network service composition layer between providers and tenants to guarantee the privacy and fair competition. The results show that the method utilizes a ranking method for VNFs, and maps them in different datacenters controlled by different providers. This leads to a much larger search space for the computation of embeddings, as opposed to this work where resource offerings are queried and bound to each particular request. Another approach, focusing on intra-datacenter allocation efficiency, is [9]. *Pentelas et. al* also use the concept of a network service to organize the VNFs. However, their algorithms strive to maximize the profit for infrastructure providers by maximizing their respective resource utilization. Thus they do not account for any aspects pertaining to service tenants (besides satisfying resource requirements).

According to the systematic review of the literature addressed, there is no previous work focusing on CNS resource allocation. Although there are some focusing on Network Function Virtualization (NFV) and datacenter virtualization, they do not have an explicit way of ordering the offers by their performance capacity and power consumption, hence our main contribution. In this paper, we advance the state-of-the-art by coupling a tenant's performance requirements with the resource costs disclosed by various infrastructure providers. Our approach consists of subdividing the slice specification into pieces (which we term as *split* step), finding the optimal set of resources for these pieces jointly, and combining them (which we term as *combine* step).

## IV. MILP FORMULATION

In this section, we discuss our mathematical model by presenting the MILP objective function and restrictions used in the process of choosing *Slice Parts* for composing the CNS. For the formulation, we assume the existence of a physical topology YAML file (SRA) with the *Slice Parts* offered by the providers and an abstract topology YAML file (PDT) with the *Slice Parts* specified by the tenant through NECOS Marketplace in a single request.

With the request of this abstract topology (PDT) in mind, the infrastructure providers supply *Resource Options* through their *Slice Agents*. These *Resource Options* generate a physical topology (SRA). The physical *DC Slice Parts*, *i.e.* the ones offered by the providers, are contained in the set $D$ and the physical *Net Slice Parts* are contained in the set $E$. The physical *DC Slice Parts* are associated with the following set $R = [v, s, t, e]$, containing the different possibilities of infrastructure resources for the *DC Slice Part*, *i.e.* the amount of RAM in megabytes (MB) ($v$), the amount of storage in MB ($s$), the number of CPU cores, ($t$) and the power consumption in kilowatt-hour (kWh) per day ($e$).

In order to concatenate the previous set of abstract and physical *Net Slice Parts* with its correspondent symmetric set, we introduce the definition in Equation 1. As such, we provide a way for the optimization method to consider all of the possibilities given by the tenant in the abstract topology (PDT) and the providers in the physical topology (SRA). Equation 2 represents an array with the amount of resources $m$ of type $k$ from every infrastructure provider offer $i$ for the abstract *DC Slice Part* $u$.

$$\begin{aligned} E_V = E_V^{\frown} E_V^{-1}, E_V^{-1} = [(v,u) : (u,v) \in E_V] \\ E = E^{\frown} E^{-1}, E^{-1} = [(j,i) : (i,j) \in E] \end{aligned} \tag{1}$$

$$m_{ik}^u = [m_{1k}^u, m_{2k}^u, ..., m_{nk}^u], \forall u \in D_V, k \in R \tag{2}$$

Every resource amount $m$ of resource type $k$ within the physical *DC Slice Part* $i$ of the abstract *DC Slice Part* $u$ must be as high as the requirement specified by the tenant $t$ for the same resource $k$ in the abstract *DC Slice Part* $u$. This information is contained in the YAML file generated by the tenants request (PDT). Its mathematical representation is depicted in Equation 3.

$$m_{ik}^u >= m_{uk}^t, \forall i \in D, u \in D_V, k \in R \tag{3}$$

Equation 4 denotes the normalization for the amount of resource of type $k$ offered from each provider for the *DC Slice Part* $i$. The information could be interpreted as the offer from the infrastructure provider for the resource $k$ in the physical *DC Slice Part* $i$. The value of this normalization ranges between 0 and 1 for every resource type. This normalization serves the purpose of maintaining the proportionality between the values of the different resources. It avoids giving more priority to higher scale parameters, *e.g.* the amount of the RAM could vary between 1 gigabyte (GB) and 1000 GB, being orders of magnitude higher than CPU cores, usually ranging between 1 and 10 cores, in average.

$$m_{ik}^{'u} = \frac{m_{ik}^u - min(m_k^u)}{max(m_k^u) - min(m_k^u)}, \forall i \in D, k \in R, u \in D_V \tag{4}$$

The risk factor for the *DC Slice Part* in Equation 5 represents how suitable the *DC Slice Part i* is, offered by the provider to the tenant, with respect to all *Resource Options* from the other providers, *i.e.* the closer the risk factor gets to 1 the greater is the risk for the tenant; whenever the risk factor gets closer to 0, the lower the risk for the tenant, since there will be sufficient resources. This level of risk occurs because the physical *DC Slice Part* will have allocated the required amount of resources for its application, which could be problematic if the amount of the resources is inaccurately dimensioned. The $d$s are the priorities for each performance

indicator, *i.e.*, to express if the indicator has high or low priority in the selection compared to the others.

$$r_i^u = \sum_{k \in \{v,s,t\}} (d_k(1 - m_{ik}^{'u})) + d_e m_{ie}^{'u}, \forall i \in D, k \in R, u \in D_V \tag{5}$$

It is worth mentioning that power consumption could be considered a piece of sensitive information. Some infrastructure providers would prefer not to disclose it. In those cases, this parameter should be removed from the risk factor in Equation 5. Equation 6 represents that the summation of all priority parameters is equal to 1, *i.e.*, if the priority of one parameter is increased, the priority of the other parameters will decrease.

$$\sum_{k \in R} d_k = 1 \tag{6}$$

Equation 7 represents that the final cost $c_i^u$ for the *DC Slice Part* $i$ from the provider is dependent on the price $c$ offered by the provider to the tenant and the risk factor $r$. The price $p_i^u$ must be at maximum the one specified by the tenant's requirement in the PDT, as in Equation 8.

$$c_i^u = p_i^u(1 + r_i^u), \forall i \in D, u \in D_V \tag{7}$$

$$p_i^u <= p_u^t, \forall i \in D, u \in D_V \tag{8}$$

As depicted in Equation 9, the final cost $c_i^u$ for the physical *DC Slice Part* $i$ and the final cost $c_{ij}^{uv}$ for the physical *Net Slice Part* are contained in the set of real and positive numbers. We assume that the price of $p_i^u$ is informed by the infrastructure provider in the SRA file. Hence, the way it is calculated inside the provider is not considered in this work.

$$(c_i^u, c_{ij}^{'uv}) \in \mathbb{R}_+^* \tag{9}$$

In Equation 5, it is also calculated the final cost $c_{ij}^{uv}$, using the bandwidth of the physical *Net Slice Part* with a risk factor in Equation 10. This represents a link connecting the physical *DC Slice Parts* $i$ to $j$, which are the SRAs for the abstract *DC Slice Parts* $u$ connected to $v$. With this combination, the provider could have the chance to offer interconnected *DC Slice Parts* of its infrastructure, obviating the need for the traffic to flow using a third party link. In this case, the link will have a low price because the traffic will be confined within the provider's domain. Together with the price provided by the infrastructure provider, in Equation 10, it is used the bandwidth amount of $b_{ij}^{uv}$ to define the risk factor and to differentiate links that support more network flows.

$$c_{ij}^{'uv} = p_{ij}^{uv}(1 + \frac{1}{b_{ij}^{uv}}), \forall (i,j) \in E, (u,v) \in E_V \tag{10}$$

If there is more than one offer for the same *Net Slice Part*, the minimum value of the final cost $c$ will be chosen. This approach focuses on reducing the objective function's complexity and fostering choosing better *Network Slice Parts* for the tenant. Similarly to the *DC Slice Part*, the financial price $c$ of the *Net Slice Part* must be at maximum the one that the tenant desires to spend, as in Equation 12. The bandwidth in the SRAs must be at minimum the one specified in the PDT as in Equation 13, to have enough performance for the tenant's service to work correctly.

$$c_{ij}^{uv} = min[c_{1ij}^{'uv}, c_{2ij}^{'uv}, ..., c_{nij}^{'uv}], \forall (i,j) \in E, (u,v) \in E_V \tag{11}$$

$$c_{ij}^{uv} <= p_{uv}^t, \forall (i,j) \in E, (u,v) \in E_V \tag{12}$$

$$b_{ij}^{uv} >= b_{uv}^t, \forall (i,j) \in E, (u,v) \in E_V \tag{13}$$

Finally, the MILP's minimization objective function in Equation 14 could be divided in two parts: the left part of the sum referring to the *DC Slice Parts* and the right part to the *Net Slice Parts*. In the left part (DC) we multiply the selection variable $x$ by the final cost parameter $c_i^u$. In the right part (Net) we have the $y$ selection variable multiplying the final cost parameter $c_{ij}^{uv}$ of the *Net Slice Part*, which connects the pair of physical *DC Slice Parts* $(i,j)$. The pair $(i,j)$ represents the provider's *Resource Options* for the abstract *DC Slice Parts* $(u,v)$. We assume that the providers will submit *Resource Options* to interconnect the physical *DC Slice Parts* $(i,j)$, if and only if they are able to interconnect them, using the provider's own infrastructure or a third party one. The objective function has the normalization and other constraints presented in Equation 15, 16, 17, 18 and 19.

Minimize:

$$\sum_{u \in D_V} \sum_{i \in D} x_i^u c_i^u + \sum_{(u,v) \in E_V} \sum_{(i,j) \in E} y_{ij}^{uv} c_{ij}^{uv} \tag{14}$$

Subject to:

$$x_i^u + x_j^v \leq 1 + y_{ij}^{uv} \tag{15a}$$

$$y_{ij}^{uv} \leq x_i^u \tag{15b}$$

$$y_{ij}^{uv} \leq x_j^v, \forall (i,j) \in E, (u,v) \in E_V \tag{15c}$$

$$\sum_{i \in D}^{n} x_i^u = 1, \forall u \in D_V \tag{16}$$

$$x_i^u \in \{0,1\}, \forall i \in D, \forall u \in D_V \tag{17}$$

$$\sum_{(i,j) \in D}^{n} y_{ij}^{uv} = 1, \forall (u,v) \in D_V \tag{18}$$

$$y_{ij}^{uv} \in \{0,1\}, \forall (i,j) \in D, \forall (u,v) \in D_V \tag{19}$$

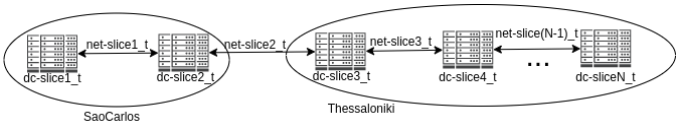
The binding between the *Net Slice Part* $y_{ij}^{uv}$ and the *DC Slice Parts* $x_i^u$ and $x_j^v$ is represented in Constraint 15, *i.e.* if the *Net Slice Part* is selected, both the above mentioned *DC Slice Parts* must be also selected. Only one *Resource Option* for the abstract *DC Slice Part* $u$ must be selected and this is offered by the provider $i$. This is specified by Constraint 16 and Constraint 17.

Constraint 17 also represents that the selection variable $x$ will have the value 1 or 0, *i.e.* the selection or not of a physical *DC Slice Part*, respectively. Each provider has only one *Resource Option* (physical topology SRA) per abstract *DC Slice Part*. In a similar way, Constraints 18 and 19 represent that only one *Net Slice Part* in the physical topology (SRA) will be selected to connect two *DC Slice Parts* $i$ and $j$. This formulation was implemented in Python programming

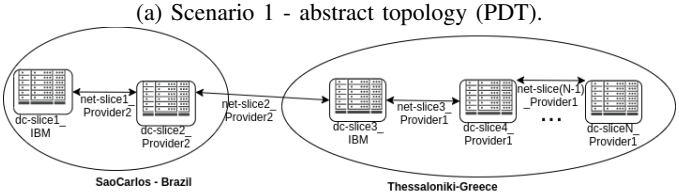
language with the *IBM ILOG CPLEX Optimization Studio v12.8 - Student*[2] for the MILP approach. Because the IBM CPLEX needs a software license, the execution environment could not be publicly disclosed, but the implementation code in Python was made public [10] and the results are shown in the next section.

## V. RESULTS

To evaluate our proposed method using the MILP formulation presented in the previous Section IV, we use four different scenarios. Those scenarios intend to compare the proposed MILP with two other heuristic methods developed: **Heuristic 1** selects the physical *DC Slice Parts* (SRAs) based on their final cost, *i.e.* the risk factor together with the financial price (Equation 7); the method **Heuristic 2** uses only the financial price in Euros for the selection.




(a) Scenario 1 - abstract topology (PDT).




(b) Scenario 1 - Physical topology.

Fig. 1: Scenario 1 - Multiple *DC Slice Parts* and one offer from infrastructure providers.

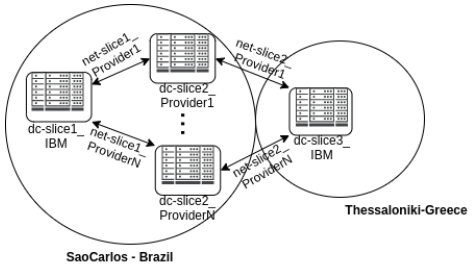



Fig. 2: Scenario 2 - Three *DC Slice Parts* and multiple *Resource Options* from providers in one node.

To guarantee that all the physical *DC Slice Parts* participating in the competition are interconnected, *i.e.* there's no loose end, we use the *DC Slice Parts* contained in the set of physical *Net Slice Parts*. All the resulting data from the tests presented here and the analysis of the resources' footprint of the computer used to run the experiments are publicly available [10]. The Scenario 1 depicted in Fig. 1 is composed of a varying number of *DC Slice Parts* and only one offer for each part, *i.e.* a big slice with no competition between the infrastructure providers. We use Scenario 1 as a baseline for the YAML file parsing' impact and information gathering between the three methods. Fig. 1a is the tenant's request and Fig. 1b is the infrastructure providers' *Resource Options*, where there are 3, 10, 50, 100, 150 and 200 *DC Slice Parts* in both topologies.

On the other hand, Scenario 2 shown in Fig. 2 is composed by a fixed number of three different *DC Slice Parts*, both in the abstract and physical topology, and a varying number of *Resource Options* for the second *DC Slice Part*, *i.e.* lots of providers competing for the same part. This scenario intends to compare how each method chooses the providers' *Resource Options* and the optimization impact in the workstation's resources by varying the number of providers' *Resource Options* in the physical topology between 3, 10, 50, 100, 150, and 200 *DC Slice Parts*.

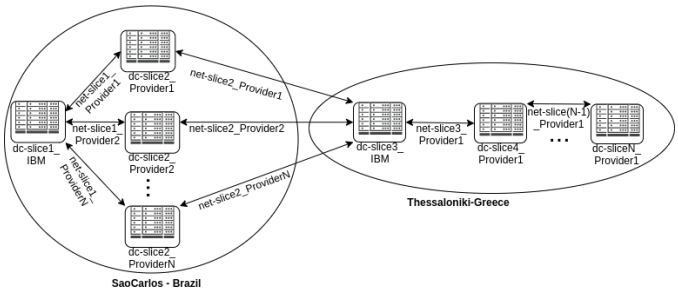



Fig. 3: Scenario 3 - Multiple *DC Slice Parts* and three *Resource Options* from providers in one node.

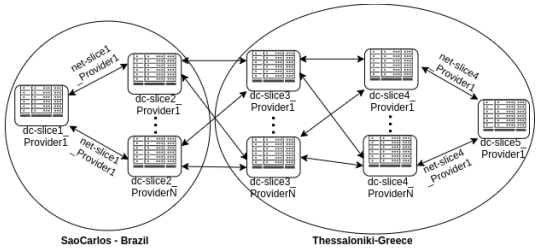



Fig. 4: Scenario 4 - Three *DC Slice Parts* and multiple *Resource Options* from providers in one node.

To evaluate the impact of a more realistic approach, Scenario 3 increases the number of *DC Slice Parts* in the whole slice abstract and physical topologies, and the number of providers *Resource Options* for only the second *DC Slice Part*. Therefore, it allows us to observe the effects of both Scenario 1 and Scenario 2 together. We used the abstract topology (PDT) from Fig. 1a as the tenant's request.

Finally, Scenario 4 from Fig. 4 tries to illustrate aspects of a real situation, where the providers have *Resource Options* for different abstract *DC Slice Parts*, *i.e. Resource Options* for more than one abstract *DC Slice Parts*. This scenario has a fixed number of only five abstract *DC Slice Parts* in the tenant's request. Its focus is on evaluating an exponential number of possibilities between *DC and Net Slice Part Resource Options*. Due to the need for generating a YAML file with all the *DC and Net Slice Parts Resource Options* as input for the implementation, to keep it simple, we define the number of *Resource Options* between 10 and 25. Those *Resource Options* are randomly distributed between the *DC Slice Parts*, and then the *Net Slice Parts* are generated to

[2]https://www.ibm.com/analytics/cplex-optimizer.

interconnect all the topology. In the next section, we present the experiments' results using the four previously defined scenarios and discussions about each one of the experiments.

### A. Experiment 1 - The influence of risk factor in the chosen Slice Parts

Experiment 1 presented in Fig. 5 was designed to understand how the risk factor (Equation 5), based on the number of resources (CPU, RAM, storage, power, and price) influences the three optimization methods compared (MILP, Heuristic 1 and Heuristic 2). Those resources are contained in the physical *DC Slice Part* offered by the infrastructure providers. To check this behavior, we implemented Scenario 3 with a fixed number of both physical and abstract *DC Slice Parts*. Therefore, the whole slice size does not grow, and the number of providers' *Resource Options* is always the same.

This scenario was executed multiple times (100) to avoid bias, each batch with a different set of parameters, saving the number of resources of the selected *DC Slice Part* in each execution. This bias avoidance allows us to check if some of the resources or the financial price influence the selection. To randomly generate the set of parameters for the *Slice Parts*, the Linux pseudo-random number generator[3] was used as input, changing the values within the range specified in each of the result graphs. The set of parameters are generated and specified in the YAML file from the provider's *Resource Options* in each execution. It is essential to mention that the *Slice Parts* financial prices offered by the providers are not related to the number of resources it has. *I.e.* a potent *Slice Part* does not mean an expensive *Slice Part*. This mechanism stimulates competition between the providers, such as a low price on some select dates.

The results presented in the candlestick graph in Fig. 5d shows a slight tendency to lower values for the power consumption in MILP optimization method as expected, *i.e.* the third quartile of MILP (69) is lower than the Heuristic 1 (73) and Heuristic 2 (75). The result can be explained because this parameter's priority is greater (0.4) than the other performance parameters (0.2). *I.e.*, the tenant prefers to give more priority to lower power consumption parts. In contrast, in Fig. 5a, Fig. 5b and Fig. 5c, the minimum, maximum and the quartiles are almost the same, which represents a random distribution for the choosing of those parameters. The explanation is due to the same priority specified by the tenant in this request. Finally, Fig. 5e elucidates the choosing of low prices in all the optimization methods, avoiding the *Resource Options* with more than 10 Euros of price, with a very similar minimum, maximum, and quartiles. The result confers the financial price is the main parameter for the *DC Slice Parts* competition between the providers, *i.e.* the price influences more than the other parameters in the selection, even with the weight of the performance parameters being the same. It must be pointed out that with a different situation like Scenario 4, the risk factor of the *Net Slice Part*, which interconnects the *DC Slice Parts* might have a higher impact on the competition.

[3]https://bit.ly/3lcwtS4

### B. Experiment 2 - Performance of the MILP method

Intending to test how efficient our MILP optimization method is compared to Heuristic 1 and Heuristic 2, we implemented Scenario 4 presented in Fig. 4 with a fixed size of 5 *DC Slice Parts* and 10 *Resource Options* from providers spread through different nodes. It was executed 100 times for each method, each time with a different price in Euros for each *DC Slice Part* and *Net Slice Part*. These executions represente 100 different tenant's requests.

In Fig. 6, we present two candlestick graphs with the distribution for the summation of the final costs and the price, respectively, of all the *Slice Parts* chosen by each method. Fig. 6a shows the summation of the final costs of all chosen *Slice Parts* by each method. It is worth pointing out that the developed MILP optimization has a better performance than the others, *i.e.* in most cases the tenant will pay a lower price and have more resources available for his slice. In the same way, Fig. 6b shows that the tenant will pay fewer Euros in most cases using our MILP approach.

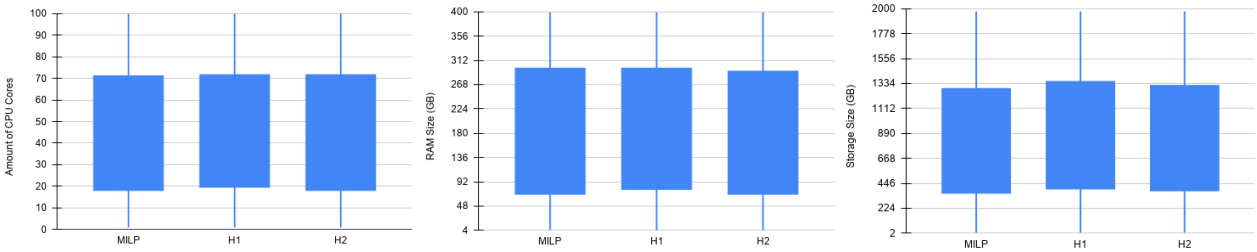
## VI. CONCLUSIONS

This paper focuses mainly on fulfilling the gap in the field of study of new methods for optimization in the context of Cloud Network Slices. We created the model based on a 5G network slicing proposal from the NECOS project. To investigate the model's performance and to address the complexity of this optimization problem, we developed our MILP as well as Heuristic 1 and Heuristic 2. Our experiments showed that the MILP method has a better performance in choosing *Resource Options* with the lowest financial price and good infrastructure resources. However, it generates more load in the workstation compared to the heuristic methods.

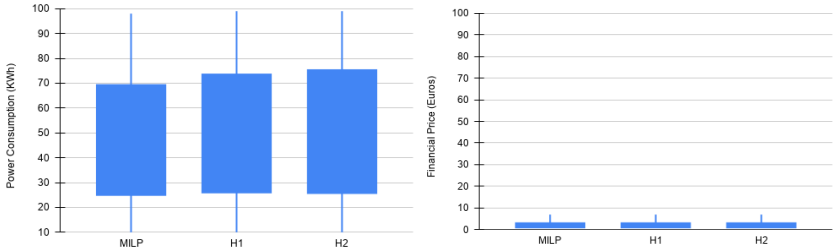
In our preliminary performance workload experiments [10] using a slightly different restriction in Equation 15, we executed the code in a bare-metal commodity workstation. The time spent in the optimization method is very small compared to the whole program. Such a stage can still be improved with better programming procedures and multi-threading techniques, which were not used. In comparison, the Python MILP optimization function has the improvements limited by the CPLEX solver.

In short, the number of resources consumed during the running of the experiments was not high for the presented scenarios due to the low amount of combinations generated in these scenarios. Despite this, we foresee a much higher resource consumption with more *Resource Options* in Scenario 4 and a new scenario with more *DC Slice Parts* in a big slice (*e.g.* more than 10 *DC Slice Parts* and 10 *Resource Options*). With our experiments, it is possible to observe that increasing the *Slice Parts* and the *Resource Options* also increases the resources consumed by our algorithm.

At this time, we did not consider the geographic location of *Slice Parts* to enable short response times. It is also planned to add more *Resource Options* into Scenario 4, which is expected to grow exponentially. With this addition, we plan to identify the behavior of the MILP and heuristics implementations.

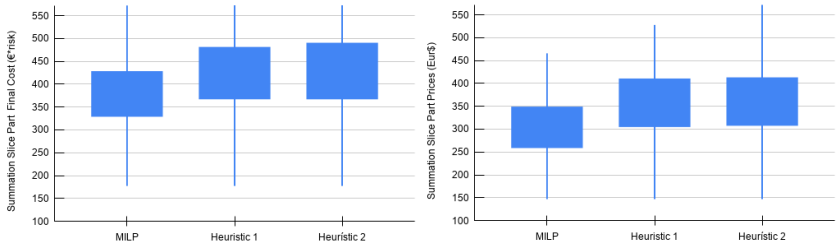



(a) Preference: number of CPU cores. (b) Preference: amount of RAM. (c) Preference: storage size.




(d) Preference: power consumption. (e) Preference: financial price.

Fig. 5: Distribution for the amount of resources chosen in Scenario 3.




(a) Summation of final costs (Euro*risk) (b) Summation of prices (Euro)

Fig. 6: Total price and cost comparison between optimization methods in Scenario 4.

Another aspect is to propose Scenario 5, where both the providers' *Resource Options* and the topology's size increase in all the *Slice Parts*. Moreover, it would be interesting to compare the MILP approach presented here with other state-of-the-art, like NESTOR [11].

It is also important to point out our work's limitations, considering that it is still necessary to scale the number of combinations to stress out the methods and check their behavior. Also, the power consumption parameter could not be feasible when infrastructure providers refuse to inform it. So, it is necessary to investigate further the possibility of removing it from the formulation. Although we still have some limitations in our approaches, we understand our main objectives were fulfilled.

## References


[1] Y. Hu, S. Xie, Y. Xu, and J. Sun, "Dynamic vr live streaming over mmt," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2017, pp. 1–4.

[2] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward qoe-assured 4k video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct 2017.

[3] N. Bocken, S. Short, P. Rana, and S. Evans, "A literature and practice review to develop sustainable business model archetypes," *Journal of Cleaner Production*, vol. 65, pp. 42 – 56, 2014.

[4] E. Hachicha, K. Yongsiriwit, and W. Gaaloul, "Energy efficient configurable resource allocation in cloud-based business processes (short paper)," in *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*. Cham: Springer International Publishing, 2016.

[5] P. D. Maciel, F. L. Verdi, P. Valsamas, I. Sakellariou, L. Mamatas, S. Petridou, P. Papadimitriou, D. Moura, A. I. Swapna, B. Pinheiro, and S. Clayman, "A marketplace-based approach to cloud network slice composition across multiple domains," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, June 2019, pp. 480–488.

[6] D. Dietrich, A. Abujoda, A. Rizk, and P. Papadimitriou, "Multi-provider service chain embedding with nestor," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 91–105, March 2017.

[7] P. Valsamas, P. Papadimitriou, I. Sakellariou, S. Petridou, L. Mamatas, S. Clayman, F. Tusa, and A. Galis, "Multi-pop network slice deployment: A feasibility study," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, Nov 2019, pp. 1–6.

[8] A. Abujoda and P. Papadimitriou, "Midas: Middlebox discovery and selection for on-path flow processing," in *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2015, pp. 1–8.

[9] A. Pentelas, G. Papathanail, I. Fotoglou, and P. Papadimitriou, "Network service embedding with multiple resource dimensions," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, April 2020, pp. 1–9.

[10] L. Beraldo, "Tenant-Oriented Resource Optimization for Cloud Network Slicing with Performance Guarantees (experiments)," Mar. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4582772

[11] D. Dietrich, A. Abujoda, and P. Papadimitriou, "Network service embedding across multiple providers with nestor," in *2015 IFIP Networking Conference (IFIP Networking)*, May 2015, pp. 1–9.