Multi-Objective Placement of Virtual Network Function Chains in 5G

Osama Arouk
INRIA, Université Côte d'Azur
2004 Route des Lucioles,
06902 Sophia Antipolis Cedex, France
Email: osama.arouk@inria.fr

Navid Nikaein
Communications Systems Department
EURECOM, Biot, France
Email: navid.nikaein@eurecom.fr

Thierry Turletti
INRIA, Université Côte d'Azur
2004 Route des Lucioles,
06902 Sophia Antipolis Cedex, France
Email: thierry.turletti@inria.fr

***Abstract*—In this paper we propose a novel algorithm, namely Multi-Objective Placement (MOP), for the efficient placement of Virtualized Network Function (VNF) chains in future 5G systems. Real datasets are used to evaluate the performance of MOP in terms of acceptance ratio and embedding time when placing the time critical radio access network (RAN) functions as a chain. In addition, we rely on a realistic infrastructure topology to assess the performance of MOP with two main objectives: maximizing the number of base stations that could be embedded in the Cloud and load balancing. The results reveal that the acceptance ratio of embedding RAN functions is only 5% less than the one obtained with the optimal solution for the majority of considered scenarios, with a speedup factor of up to 2000 times.**

***Index Terms*—5G; C-RAN; NFV; function split; function placement**

I. Introduction

The evolution of cellular mobiles networks toward 5G is driven by many issues, such as the explosive demand of traffic and the arrival of a myriad of Internet of Things (IoT) devices (e.g., 50 billion or even more), in addition to other emerging technologies. An approach to support such scenarios in 5G networks is by using dense, cooperative, and heterogeneous wireless networks [1]. Unfortunately, this approach is costly in terms of CAPEX and OPEX, especially when deploying small cells [2]. In order to support the increasing demand of traffic with a minimal cost, a Centralized (or Cloud) Radio Access Network (C-RAN) architecture can be adopted [3]. In this architecture, a subset of network functions, after being virtualized, is moved from the cell sites to the cloud, named as Base Band Unit (BBU), and the remaining functions are placed at the cell site close to the radio frontend and antennas, named as Remote Radio Unit (RRU). This is called functional split and provides the required flexibility to enable various use cases considered in 5G. In the recent 3GPP standardization (Rel. 13 and 14), three elements have been defined, RRU, Distribution Unit (DU) which hosts time-critical L1/L2 functions and aggregates a subset of RRUs (equivalent to BBU), and centralize unit (CU) for the remaining functions. RRU and DU can be defined as a logical unit including a subset of base station functions located at the cell edge, while the other functions are located at the CU in the Cloud.

Despite its advantages, functional split comes with many challenges, such as very high capacity requirement of the fronthaul (FH) (i.e., transport network between RRU and BBU). Since PHY and higher layers processing is moved to the Cloud, strict latency requirements have to be met, especially for certain functional split such as LTE (inverse) Fast Fourier Transform (IFFT/FFT) and operations like LTE Hybrid Automatic Repeat Request (HARQ) [3]. Relaxing fronthaul requirements can be done in two ways: fronthaul compression and functional split [4]. Although the fronthaul compression is able to reduce the capacity requirements [5], [6], it may increase the complexity, especially at the RRU side. Moreover, this solution only solves the problem of capacity requirement and to some extend the latency, while jitter issues are still remaining. The other approach consists in shifting some of the baseband processing functions from the BBU to the cell site using functional split between BBU and RRU. Not only the functional split relaxes the fronthaul requirements, but it also adds flexibility in the deployment of various use-cases.

Placement of a network function chain in the Cloud is another challenging issue facing the adoption of C-RAN type architecture. While in the literature many works address the problem of elastic network function placement [7], [8], there exists few work considering the placement of time critical network functions and the associated service chain. The authors in [9] formulate an Integer Linear Programming (ILP) model to maximize the number of cloudified BBUs with unlimited resources. Another approach proposed for the problem of BBU placement is the one presented in [10], where a graph-based model is introduced to decide whether the BBU functions need to be placed in the cloud or in the cell site. However, in the previous works resources are considered to be unlimited and the placement of network function inside the cloud is not addressed.

In this paper, we propose the Multiple Objective Placement (MOP) algorithm for the efficient placement of virtualized network function chains in future 5G systems. The main idea of MOP is to cluster the nodes into different distinct regions (or classes) according to the latency requirements of the VNFs. A VNF is then embedded on a node chosen from its related region, without violating the requirement of the VNF. The advantage of MOP is that it can be adapted to many objectives by simply changing the conditions on which nodes are selected to embed VNFs. However, maximizing

the number of cloudified BBUs and load balancing are the considered objectives in the current paper. Our contributions are as follows:

- we present the placement model for network service chain and the problem formulation for Integer Linear Programing (ILP) (Sections II and III, respectively),
- we propose our model MOP and show its relevance and applicability in the context of 5G networks (Sections IV),
- using extensive simulations on realistic topology, we show that significant performance improvements could be achieved compared to current approaches (Sections V).

II. RELATED WORK

Once shifted to the Cloud, the VNFs must be efficiently placed to optimize the resource utilization. Note that when embedding a Service Function Chain (SFC), the latency requirement of the whole chain has to be met, in addition to the latency requirement of each VNF composing it. Moreover, there may be some interactions between chains to accomplish certain tasks, e.g., when using Coordinated Multipoint (CoMP), which imposes strict latency on certain functions.

In the literature, many works tackle the problem of BBU placement. The authors in [10] propose a graph-based model and formulate an optimization problem to minimize the computational cost of the fronthaul. They use a Genetic Algorithm (GA) to find a sub-optimal solution because a general analytical solution for such a problem is difficult to obtain. As minimizing the fronthaul cost and the computation cost follow contradictory objectives, they analyze a trade-off between the centralization and decentralization by tuning a trade-off coefficient. Although the authors take into consideration that the latency for the chain as a whole is constrained, they do not account for the latency constraint of the functions in the chain, which may be strict too. Moreover, they suppose that the Cloud has ideal characteristics, with neither processing delay nor computational cost (i.e., the cost of embedding a BBU function) in the Cloud, which is not realistic. The authors of [9] tackle the problem of BBU hoteling, i.e. finding the best location for the BBU to be placed in the Central Office (CO). Their model aims at minimizing both the number of the deployed fibers and the number of locations where the BBUs have to be placed, i.e. the degree of consolidation. However, they do not consider the limited amount of resources available in the CO. Furthermore, they tackle the problem of BBU placement as a whole, without considering the constraints on each BBU function. Moreover, placing a whole BBU in a single location does not allow to use in an optimal way the available resources, especially when the latter are heterogeneous. Therefore, in the following, a novel algorithm for the functional placement is introduced.

III. PLACEMENT MODEL AND PROBLEM FORMULATION

In this section, we list the VNF requirements considered in the placement algorithm, and then we present an Integer Linear Programming (ILP) formulation of the VNF placement problem.

A given VNF can be characterized by the following parameters, considering the chain Γ_i with a length h_i (see Table II):

1) Input rate of the VNF $\lambda_{\Gamma_i}(f)$: $\sigma^r_{\Gamma_i}(f-1, f)$. All the VNFs placed at the cell site are assigned to $\lambda_{\Gamma_i}(0)$, where this VNF does not have input $\sigma^r_{\Gamma_i}(-1, 0) = 0$, and does not need any computation. Note that we tackle the problem of VNF placement in the Cloud. Therefore, all the VNFs placed at the cell site are not taken into consideration.
2) Output rate of the VNF, which is the input rate of the following VNF in the chain: $\sigma^r_{\Gamma_i}(f, f+1)$. Alternatively, the last VNF in the chain is considered without output.
3) CPU requirements: $\lambda^c_{\Gamma_i}(f)$.
4) Memory requirements: $\lambda^m_{\Gamma_i}(f)$.
5) Latency constraint for each function: $\lambda^\tau_{\Gamma_i}(f)$.

TABLE I: Substrate Network Parameters

Notation	Definition
$\Phi = \{\varphi(p); p = 1{:}N\}$	Substrate network graph. N is Nb. nodes
$\varphi(p) \in \Phi$	Physical node p
$L = \{l(\varphi(p), \varphi(q)) \varphi(p), \varphi(q) \in \Phi\}$	Substrate links
$l(\varphi(p), \varphi(q)) = l(p, q)$	Link between two nodes $\varphi(p), \varphi(q) \in \Phi$
$\varphi^c(p)$	Total available CPU
$\varphi^m(p)$	Total available memory
$\varphi^r(p, q)$	Available rate of the link between the nodes p and q
$\kappa_c(p)$	Cost of a CPU unit
$\kappa_m(p)$	Cost of a memory unit
$\kappa_r(p)$	Cost of a link rate unit for the link $l = l(p, q)$
t_l	Link's delay
$t_{\lambda_{\Gamma_i(f)} \to \varphi(p)}$	Processing time of the function $\lambda_{\Gamma_i(f)}$ if it is mapped on the node $\varphi(p)$
d_p	Minimal (propagation) delay among the different links between the cell site and the node p
$\omega^c(p)$	CPU resources that are already used on the node p
$\omega^m(p)$	Memory resources that are already used on the node p
$\omega^{r_{in}}(q_{f-1}, p)$	Rate, which is already used, of the link between node q_{f-1} on which the function $(f-1)$ is already mapped and p on which the function f is expected to be mapped.
$\omega^{r_{in}}(p, *)$	Rate, which is already used, of the links connected to the node p

TABLE II: SFC parameters

Notation	Definition
$\Gamma = \{\Gamma_i; i = 1{:}M\}$	Set of service function chains to be embedded. M is Nb. chains
$\Gamma_i = \{\lambda_{\Gamma_i}(f); f = 1{:}h_i\}$	Service function chain with a length h_i
$\lambda_{\Gamma_i}(f)$	f^{th} function of the chain Γ_i
$\Sigma = \{\Sigma_i; i = 1{:}M\}$	Set of virtual links for all the chains
$\Sigma_i = \{\sigma_{\Gamma_i}(f-1, f); f = 1{:}h_i\}$	Set of virtual links for the chain Γ_i
$\lambda^c_{\Gamma_i}(f)$	CPU requirements of the f^{th} function from the chain Γ_i
$\lambda^m_{\Gamma_i}(f)$	Memory requirements of the f^{th} function from the chain Γ_i
$\sigma^r_{\Gamma_i}(f-1, f)$	Link rate requirements of the link between the functions $(f-1)$ and (f) from the chain Γ_i
$\lambda^\tau_{\Gamma_i}(f)$	Latency requirements of the function f belonging to the chain Γ_i
$\Delta(f \to p)$	Mapping the function f on the physical node p

The SFC embedding problem can be formulated as an ILP with the objective to reduce the cost of VNF chain embedding on a substrate network. This can be expressed as follows:

$$\min \sum_{n=1}^{N} \sum_{j=1}^{M} \sum_{i=1}^{h_j} \left\{ \kappa_c(n) \lambda^c_{\Gamma_j}(i) + \kappa_m(n) \lambda^m_{\Gamma_j}(i) \right\} \Delta(i \to n) + \sum_{l \in L} \sum_{j=1}^{M} \sum_{i=1}^{h_j} \left\{ \kappa_r(l) \sigma^r_{\Gamma_j}(i-1, i) \right\} \Delta(\sigma^r_{\Gamma_j}(i-1, i) \to l) \quad (1)$$

where $\Delta(x \to y)$ is the unit function and is expressed as follows:

$$\Delta(x \to y) = \begin{cases} 1 \text{ if the function } x \text{ is mapped on node } y \\ 0 \text{ otherwise} \end{cases}$$

In addition to the resource utilization cost (i.e., the amount of resources that would be allocated for a VNF or a chain) [11], the cost of CPU/memory/link unit, i.e., $\kappa_c(p)/\kappa_c(p)/\kappa_c(p)$, by itself may vary from a location to another one [10]. For example, the cost of resource unit could vary according to the virtualization environment, cell site rent or the electricity consumption of the site.

During the mapping phase and in order to make sure that the requirements of the VNFs mapped on a node would not exceed the available resources at the considered node, the following conditions must be applied when solving Equation 1:

$$\sum_{j=1}^{M}\sum_{i=1}^{h_j}\left\{\kappa_c(p)\lambda^c_{\Gamma_j}(i)\right\}\Delta(i\rightarrow p)\ \leq\ \varphi^c(p)\ \forall\varphi(p)\in\Phi$$

$$\sum_{j=1}^{M}\sum_{i=1}^{h_j}\left\{\kappa_m(p)\lambda^m_{\Gamma_j}(i)\right\}\Delta(i\rightarrow p)\ \leq\ \varphi^m(p)\ \forall\varphi(p)\in\Phi$$

$$\sum_{j=1}^{M}\sum_{i=1}^{h_j}\left\{\kappa_r(l)\lambda^r_{\Gamma_j}(i-1,i)\right\}\Delta(i-1;i\rightarrow l)\ \leq\ \varphi^r(l)\ \forall l\in L$$

where these equations represent the conditions on CPU, memory, and links, respectively. Moreover, the latency requirements of the VNFs must be satisfied too:

$$\sum_{g=1}^{f}(\sum_{n=1}^{N}t_{\lambda_{\Gamma_i}(g)\rightarrow\varphi(p)}\Delta(g\rightarrow n)+\sum_{l\in L}t_l\Delta(\sigma^r_{\Gamma_j}(g-1,g)\rightarrow l))$$
$$;\ 1\leq j\leq M,\ \ 1\leq g\leq h_j \qquad (2)$$

As solving an ILP problem is generally intractable [12], in the following we introduce the MOP algorithm for efficient embedding. Notice that, for the sake of simplicity, the cost of functions embedding and of the corresponding links are considered to be the same in the whole considered topology. Thus, the values of κ_* are set to one.

IV. Multi-Objective Placement (MOP)

This section describes the MOP algorithm that aims to find the best candidate node for embedding each VNF of a chain. This algorithm consists of three steps:

1) for every VNF having distinct latency constraint in the chain, determine the Eligible Regions (ER) corresponding to the set of nodes that satisfy the latency requirements of the VNF;
2) determine the Candidates Group (CG), which is composed of the nodes from the ER that satisfy all the other VNF requirements, e.g. CPU/memory and input/output rate requirements;
3) select the best node among CG according to the considered objective.

In the first step, each node is mapped to a certain ER based on the latency that it can support, where the number of ERs is equal to the number of VNFs with distinct latencies, see Fig. 1. If two VNFs have the same latency, then there will be only one ER. The reason behind the classification of nodes based on VNF latency requirements is to avoid a situation where all

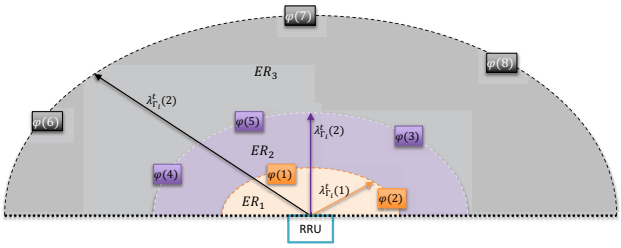


Fig. 1: Eligible Regions: nodes distribution based on the latency requirements of the VNFs $\lambda^\tau_{\Gamma_i}(f)$

the chains are embedded to the edge of the network, i.e. as close as possible to the cell site, when the network is lightly loaded. Without caution, it may not be possible to embed new chains when there is a surge in the traffic without violating the latency constrains of the chains.

Algorithm 1 Select the best candidate node among the available ones to embed function $\lambda_{\Gamma_j}(f)$ of chain $\Gamma_i(f)$

Determine_ER() :
for n = 1:N **do** ▷ physical nodes
  for f = 1:h_j **do** ▷ functions
    $T = d_n + \sum_{g=1}^{f} t_{\lambda_{\Gamma_i}(g)\rightarrow\varphi(n)}$
    if f == 1 **then**
      if $T \leq \lambda^\tau_{\Gamma_i(f)}$ **then**
        add (n, ER_f), and **Break**
      end if
    else
      if $\lambda^\tau_{\Gamma_i(f-1)} < T \leq \lambda^\tau_{\Gamma_i(f)}$ **then**
        add (n, ER_f), and **Break**
      end if
    end if
  end for
end for

Determine_CG_node() :
for f = 1:h_j **do** ▷ functions
  $CG = \phi$; $r = i + 1$
  while $(CG = \phi)\,\&\,(r > 0)$ **do**
    r = r - 1
    for $j = 1 : SizeOf(ER_r)$ **do**
      If $ER_r(j)$ satisfies remaining requirements of f
      add $(ER_r(j), CG)$
    end for
  end while
  if $SizeOf(CG) = 0$ **then**
    skip(f) ▷ f can not be embedded
  else if $SizeOf(CG) = 1$ **then**
    embed (f, CG(1))
  else
    if $OBJ = MAX.\ EMBED\ CHAINS$ **then**
      Apply AHP
    else
      Apply MSE
    end if
  end if
end for

After determining the eligible regions, the candidates group is created. In this step, and for each VNF, the search task to find possible candidates starts from its eligible region (i.e., the ER corresponding to the VNF latency requirement). If the search task could not find any candidate in its eligible region, it then goes to the inferior ER, i.e. the ER corresponding to

the VNF just before in the chain. The third step consists in choosing the best node among the candidate ones based on the considered objective. As shown later, the decision is taken to achieve a specific objective like load balancing or maximizing the number of BBUs that could be supported by the Cloud.

Fig. 1 shows the distribution of nodes when they are organized in ERs according to the latency requirements of the PHY-layer VNFs. In this example, it is assumed that there are 8 nodes and 3 BBU VNFs. This figure is obtained using the first part of Algorithm 1, i.e. *determine_ER()*.

In the second part of the algorithm, i.e. *determine_CG_node()*, the objective of lines 1-10 is to select, from the candidates group, the nodes that can support: i) the required CPU/memory resources of the VNF, ii) the VNF input and output rates requirements.

After going through this procedure, if there is more than one candidate (lines 11-18 of the second part of the algorithm), the mechanism used to choose the node to embed the VNF depends on the target objective. In this paper, two objectives are considered: maximizing the number of BBUs to embed in the Cloud, and load balancing. In the first case, we use the Analytic Hierarchy Process (AHP) [13] to choose the best candidate node, considering the three following criteria: CPU, input rate and output rate requirements of the function. It is worth noting that AHP is a well-known tool for decision making, especially in presence of multiple conflicting criteria. In the second case, i.e. load balancing objective, we use the Mean Square Error (MSE) in the following way. Let us first define the percentage of resources already used at node p: $\alpha_p^c = \omega^c(p)/\varphi^c(p)$. Then, let us assume that D candidate nodes are found to place the VNF. When placing the VNF on node d (from the set of D candidate nodes), the percentage of CPU resources used for this node becomes:

$$\alpha_p^{c'} = \frac{\lambda_{\Gamma_i}^c(f) + \omega^c(p)}{\varphi^c(p)}$$

To perform load balancing among nodes, we compute the MSE as follows:

$$MSE_d = \frac{1}{D}\sum_{v=1}^{D}\left(\alpha_d^{c'} - \alpha_v^c\right)^2$$

where MSE_d is the MSE when placing the VNF on node d. The final decision consists in placing the VNF on node (x) for which the MSE_x is the minimum among all the computed MSE values. From the above discussion, it is clear that the only change in the algorithm pertains to the lines 16-17, while the rest of the algorithm is unchanged.

Regarding the complexity, it is easy to find that it can be expressed by the following equation:

$$\mathcal{O}\left(N(h_{Tx} + h_{Rx})\right) + \mathcal{O}\left(M(h_{Tx} + h_{Rx})(L + 2N)\right)$$

where h_{Tx}/h_{Rx} are the length of Tx/Rx. From this equation, it is clear that the complexity of MOP is linear with respect to the number of BBUs to be embedded. Moreover, the complexity of MOP increases linearly with the number N of nodes in the topology. Notice that the effect of N is more visible than that of M, since it affects the complexity of both creating the eligible regions (the first part of Equation IV) and the placement (the second part of Equation IV). Given the linearity in its complexity, MoP is able to efficiently place network service chain in heterogeneous infrastructure with divers constraints in the context of disaggregated 5G networks.

V. Performance Evaluation

In the first part of this section, we provide a general description of the topology and we present the different simulation scenarios along with the performance metrics used to evaluate the efficiency of MOP. Finally, we discuss the results obtained for the two target objectives: maximizing the number of cloudified BBUs and load balancing. Without loss of generality, we focus on PHY-layer VNFs placement as they have more stringent requirements.

Fig. 2 represents a realistic infrastructure topology of that is used in the simulation, which consists of three main parts: the distributed cell sites, the local Cloud which is the one closer to the cell site represented by the nodes [N1, N6], and the farther macro Cloud represented by the nodes [N7, N14]. We assume the presence of low-latency fronthaul links like optical fibers [14]. The parameters used in the simulation are summarized in Table III [15], [16]. Regarding the PHY-layer VNFs, the three most greedy resource PHY-layer VNFs for Tx/Rx are considered: IFFT/FFT, Mod/Demod, and Encod/Decod, see the analysis done in [2], [15]. Furthermore, we consider the following VNF requirements: CPU and input/output rates. Concerning the network configuration, we consider 20 MHz bandwidth with peak traffic rate. Note that the requirements of the three aforementioned functions are gathered from OpenAirInterface [17].

Furthermore, we consider two cases to assess the performance of MOP: 1) the Homogeneous case where the capabilities of nodes in every level of the tree topology are the same and 2) the non-Homogeneous case where the nodes in the same level of the tree topology have different capabilities, as shown in Table III. Note that the values of mesh levels 1, 2, and 3 for the mesh index are binary values set to 1 when there is a mesh in that level and zero otherwise.

A. Performance Metrics

Results obtained with ILP are used to benchmark the MOP algorithm, since it gives the optimal solution. Due to lack of space, only two parameters are considered for the comparison: acceptance ratio and embedding time. The acceptance ratio is computed based on the upper bound of the number of BBUs (each one has two chains: Tx and Rx) that could be supported with the set of available resources. Assuming that enough capacity is remaining on the links, the upper bound is calculated as follows:

$$N_{BBUs}^c = \frac{\sum_{n=1}^{N}\varphi^c(n)}{\sum_{i_1=1}^{h_{ul_{k-1}}}\lambda_{\Gamma_{k-1}}^c(i_1) + \sum_{i_2=1}^{h_{dl_k}}\lambda_{\Gamma_k}^c(i_2)} \quad (3)$$

where $h_{ul_{k-1}} = h_{k-1}$ and $h_{dl_k} = h_k$ stand for the length of the uplink and downlink chains, respectively. The embedding

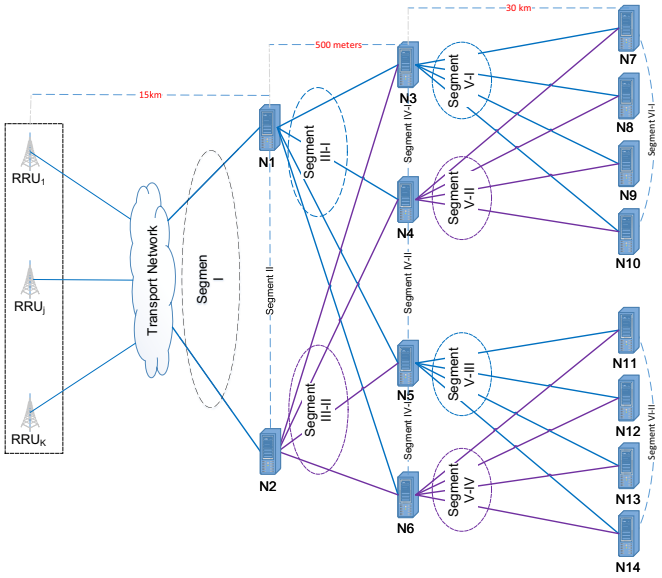


Fig. 2: Network infrastructure topology

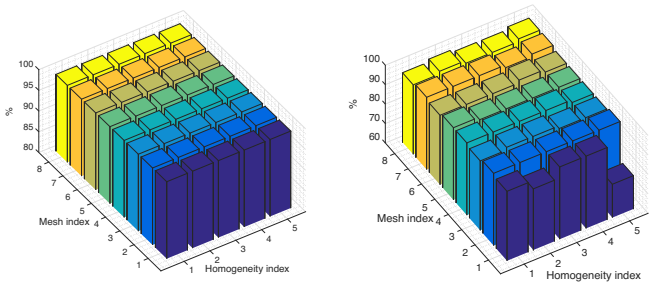
time corresponds to the amount of time required to solve the ILP (with CPLEX) and the time required to run MOP (with MATLAB).

B. Performance Evaluation

1) Maximize the number of embedded BBUs: Based on the parameters considered in Table III, the upper bound of the number of BBUs that can be cloudified is 70. Fig. 3a and 3b illustrate the acceptance ratio for ILP and MOP, respectively. As expected, ILP gives the best solution whatever the considered case, with an acceptance ratio equals to 98.6%, see Fig. 3a. As the considered number of BBUs to be embedded is upper bounded, it is not always possible to embed all the chains corresponding to these BBUs.

Regarding MOP, it generally obtains a high acceptance ratio, where the difference, compared to ILP, is less than 5 % in the majority of the considered scenarios. However, this difference increases for the tree topology (i.e., when there is no mesh at any level), especially for the last homogeneous index. This phenomenon could be explained by the fact that the resources available in the topology are distributed in a way that it is not always possible to use these nodes because this would violate the latency requirements of some of the VNFs. Note that this problem could be solved by relying on the multi-knapsack problem.

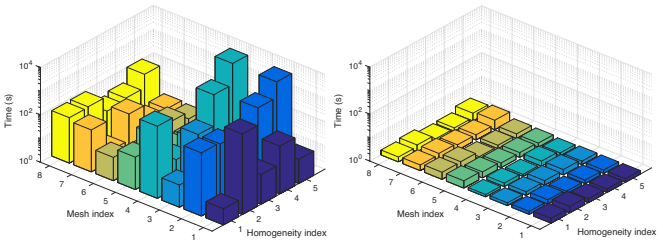
Even though its acceptance ratio is slightly lower than that for ILP, MOP outperforms ILP regarding the necessary time to embed BBUs. The difference observed in embedding time is illustrated in Fig. 4a and 4b for ILP and MOP, respectively. We can note that the average time required to find an efficient placement for one BBU (i.e., the chain of VNFs composing a BBU) is less than 5 seconds, Fig. 4b. However, this time could reach more than 2 hours for ILP (Fig. 4a), which is not acceptable in practice, since the network should respond quickly to network changes (e.g., service chain definition,



(a) ILP (b) MOP

Fig. 3: Acceptance Ratio

traffic load variability). Such a short embedding time makes MOP a relevant candidate for placement algorithm within the network service orchestration logic in future disaggregated 5G networks.



(a) ILP (b) MOP

Fig. 4: Embedding time

2) Load Balancing: Fig. 5 illustrates the average resource utilization for every node in the considered topology. From these figures, it can be seen that the proposed algorithm for load balancing generally achieves a good balance. However, few fluctuations can be observed for certain nodes (e.g., the nodes 8 and 12) for the two considered homogeneous indexes, respectively, when embedding 20 BBUs. Such fluctuations could be explained when placing functions with very heterogeneous requirements. Note that the fluctuations are lower when placing 30 BBUs. This could be explained by the fact that the number of VNFs whose requirements are much higher than the others increases. In this case, load balancing performs better since the VNFs are distributed over more nodes. Another important issue that could affect load balancing is related to latency constraints. Sometimes, a VNF should be placed on a particular node to achieve better load balancing. However, such a placement is not always possible due to latency issues, i.e. the latency requirement of this VNF will be violated if it is placed on the considered node. An important observation from these figures is that non homogeneous topologies (e.g., for Homogeneous index 5) achieve better load balancing compared to homogeneous topologies (i.e., for Homogeneous index 1). Moreover, it is found that the majority of IFFT/FFT VNFs are located at the local cloud closer to RRU. However, this result is not surprising as latency and rate constraints of these VNFs are directly related to the fronthaul constraints.

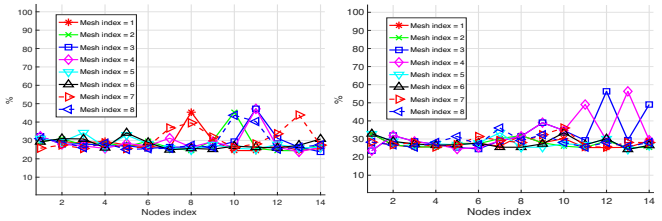
TABLE III: Configuration parameters related to topology

Nodes	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
CPU: @core	8	8	4	4	4	4	2	2	2	2	2	2	2	2
Homogeneity index														
Index 1	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
Index 2	$N_1+N_2/2$	$N_2/2$	$N_3/2$	$N_4/2$	$N_5+N_3/2$	$N_6+N_4/2$	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
Index 3	N_1	N_2	$N_3+N_5/2$	$N_4+N_6/2$	$N_5/2$	$N_6/2$	$N_7/2$	$N_8/2$	$N_9/2$	$N_{10}/2$	$N_{11}+N_7/2$	$N_{12}+N_8/2$	$N_{13}+N_9/2$	$N_{14}+N_{10}/2$
Index 4	$N_1+N_2/2$	$N_2/2$	N_3	N_4	N_5	N_6	$N_7/2$	$N_8/2$	$N_9/2$	$N_{10}/2$	$N_{11}+N_7/2$	$N_{12}+N_8/2$	$N_{13}+N_9/2$	$N_{14}+N_{10}/2$
Index 5	$N_1+N_2/2$	$N_2/2$	$N_3/2$	$N_4/2$	$N_5+N_3/2$	$N_6+N_4/2$	$N_7+N_{11}/2$	$N_8+N_{12}/2$	$N_9+N_{13}/2$	$N_{10}+N_{14}/2$	$N_{11}/2$	$N_{12}/2$	$N_{13}/2$	$N_{14}/2$

Links parameters: Capacity: Gbps Length: meter		Level 0 -> Level 1	Mesh level 1	Level 1 -> Level 2	Mesh level 2		Level 2 -> Level 3		Mesh Level 3	
	Segment	I	II	III-I, III-II	IV-I	IV-II	V-I, V-II	V-III, V-IV	VI-I	VI-II
	Capacity	80	20	20	20	20	10	10	20	20
	Length	15000	100	500	25	100	30000	30000	25	25

Mesh Index								
index	1	2	3	4	5	6	7	8
Mesh level 1	0	1	0	1	0	1	0	1
Mesh level 2	0	0	1	1	0	0	1	1
Mesh level 3	0	0	0	0	1	1	1	1

Concerning the other VNFs like Encod/Decod, they are located in both local and macro Clouds. The reason is that, for the considered parameters, the available resources in the macro Cloud are not sufficient to support all of them.



(a) Nb. BBUs = 20 (b) Nb. BBUs = 20

(c) Nb. BBUs = 30 (d) Nb. BBUs = 30

Fig. 5: Resource utilization for every node; Homogeneous index = 1 and 5 for 5a, 5c, and 5b, 5d, respectively

VI. Conclusion

In this paper we propose a low-complexity multi-objective placement algorithm to efficiently embed BBU chains of VNFs in the context of three tier cloud RAN architecture. Two objectives are considered: maximizing the number of BBUs that could be embedded in the Cloud and distributing the load evenly across the set of computing nodes. Simulation results, carried out on a realistic infrastructure topology with real datasets of RAN functions requirements, show that MOP can achieve a high percentage of acceptance ratio (less than 5% for non-homogeneous topologies compared to the optimal solution), with less than 5 seconds to embed an entire BBU chain. Moreover, the proposed algorithm offers load balancing in most of scenarios. As future work, we plan to consider the placement of the whole set of RAN functions with dynamic splits as well as the cost minimization when embedding the RAN chains.

Acknowledgments

Research and development leading to these results have received funding from the French Government (National Research Agency, ANR) through the Investments for the Future Program reference ANR-11-LABX-0031-01 and the European Framework Program under H2020 grant agreement No. 762058 for the 5GPicture project.

References

[1] S. Zhou *et al.*, "CHORUS: a framework for scalable collaboration in heterogeneous networks with cognitive synergy," *IEEE Wireless Communications*, vol. 20, no. 4, pp. 133–139, August 2013.

[2] N. Nikaein, "Processing radio access network functions in the Cloud: Critical issues and modeling," in *MCS 2015, 6th International Workshop on Mobile Cloud Computing and Services*, 2015.

[3] C. Mobile, "C-RAN: the road towards green RAN," *White Paper, ver*, vol. 2, 2011.

[4] N. Alliance, "Suggestions on Potential Solutions to C-RAN," 2013.

[5] B. Guo *et al.*, "LTE/LTE-A signal compression on the CPRI interface," *Bell Labs Technical Journal*, vol. 18, no. 2, pp. 117–133, Sept 2013.

[6] C.-Y. Chang *et al.*, "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN," in *ICC 2017, IEEE International Conference on Communication, May 21-25, 2017, Paris, France*, 2017.

[7] Y. Xie *et al.*, "Service Function Chaining Resource Allocation: A Survey," *CoRR*, vol. abs/1608.00095, 2016. [Online]. Available: http://arxiv.org/abs/1608.00095

[8] D. Bhamare *et al.*, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138 – 155, 2016.

[9] F. Musumeci *et al.*, "Optimal BBU Placement for 5G C-RAN Deployment Over WDM Aggregation Networks," *Journal of Lightwave Technology*, vol. 34, no. 8, pp. 1963–1970, April 2016.

[10] J. Liu *et al.*, "Graph-based framework for flexible baseband function splitting and placement in C-RAN," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 1958–1963.

[11] X. Cheng *et al.*, "Virtual Network Embedding Through Topology-aware Node Ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[12] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[13] T. L. Saaty, *What is the Analytic Hierarchy Process?* Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 109–121.

[14] J. Bartelt *et al.*, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Communications*.

[15] "Small cell virtualization: Functional splits and use cases," Small Cell Forum, Tech. Rep., January 2016.

[16] "Technical Specification Group Radio Access Network; Radio Frequency (RF) system scenarios," 3GPP, Tech. Rep., January 2016.

[17] "OpenAirInterface: 5G software alliance for democratising wireless innovation," http://www.openairinterface.org/.