Optimal VNF Chains Management for Proactive Caching

Gao Zheng, Anthony Tsiopoulos, Vasilis Friderikos
Centre for Telecommunications Research, King's College London, London WC2B 4BG, England
E-mail: {gao.zheng, anthony.tsiopoulos, vasilis.friderikos}@kcl.ac.uk

***Abstract*—Notwithstanding the significant attention that Network Function Virtualization (NFV) architectures received over the last few years little attention has been placed in cases where proactive caching is considered within a service chain. Caching algorithms have been developed independently from virtual network function (VNF) chaining schemes and as we explain in detail in this paper such operation might lead to sub-optimal overall network and service performance. Since caching of popular content is envisioned to be one of the key adopted technologies in emerging 5G networks to increase network efficiency and overall end user perceived quality of service we explicitly consider the interplay and subsequent optimization of caching based VNF service chains. To this end, a mathematical programming framework is proposed tailored to VNF caching chains and in addition detail a scale-free heuristic algorithm to provide competitive solutions for large network instances since the problem itself can be seen as a variant of the classical *NP-hard* Uncapacitated Facility Location (UFL) problem. A wide set of numerical investigations are presented for characterizing the attainable system performance of the proposed schemes.**

***Index Terms*—Network Function Virtualization, 5G networks, proactive caching, integer linear programming.**

I. Introduction

It is well accepted that current mobile network architectures suffer from insufficient scalability and flexibility to speedily accommodate new services and ability to embrace vertical industries [1]. To address these challenges, applying software defined networking (SDN) [2] principles in emerging architectures towards 5G networks is gaining significant momentum recently [3]. This goes hand-in-hand with the currently heavily studied network function virtualization (NFV) [4] architectures, that together with SDN, can be considered as the two enablers towards flexible 5G networks, where full virtualization and efficient network slicing according to the needs of different tenants can be implemented. Such an SDN/NFV-enabled network is in essence able to decouple network functions (NFs) from the underlying physical devices, thereby, NFs can be virtualized, creating the so-called virtual network functions (VNFs). The benefit stems from the fact that VNFs can be flexibly controlled/assigned/moved within the network using Virtual Machines or (docker) containers. In NFV framework, an end-to-end network service (e.g., rich voice/data) is described by an VNF forwarding graph, where a number of VNFs (possibly distributed in various physical nodes in the network) need to be visited in certain predefined order [5]. To be more precise, the sequenced VNFs of a service request form a service chaining as the service flow passes through an ingress or egress point in a virtual network device. An illustrative example of such service chain is shown in figure 1, where caching is considered as one of the VNFs[1] that constitute the overall service chain; these VNFs might be located in different nodes in the network. Our aim is to consider caching and the other possible VNFs that might be required for the service in an integrated manner in order to increase network efficiency.

Undoubtedly, among different VNFs, it is expected that caching would emerge as one of the potential key network elements to be supported in emerging and future wireless/mobile networks. Viral and popular video streams dominate aggregate mobile Internet traffic[2] and it is an application well suited to various different caching strategies. In that respect, caching of popular content deserves paying a special attention in terms of VNF hosting location and chaining. This is because in the most general case, a cached content must be visited before other VNFs can be applied and this service flow might originate from different possible network locations depending on the caching strategy. In other words, this type of service does not need to reach a gateway node or a specific cloud but can originate at any node that host the required cached content (which, most probably, be topologically closer to the end user). Therefore, the location of caches in a VNF service chain, greatly affects the overall VNF chain orchestration as well as the aggregate traffic dynamic in the network, since links of higher aggregation (deeper in the network) can reduce their utilization levels. However, efficient caching in mobile networks can be deemed as a highly challenging task since the optimality of the cache locations are dependent on the movement/mobility patterns of the end users. Notably, to significantly reduce access delays to highly popular content caching content close to the end user without considering the effect of mobility might lead to a degradation of the performance. In this case, caching popular content closer to the end user might inevitably require more frequently changes of the cache location to keep providing optimal performance. As a result, the proactive caching location and the associated VNF chaining need to be jointly considered to avoid sub-optimal cases, especially under network congestion episodes where performance can be significantly penalized. To summarize,

[1] The terms VNF and NF are used interchangeably in the rest of the paper, except where differentiation is required.

[2] Mobile video traffic accounts for 60 percent of total mobile data traffic according to the CISCO Global Mobile Data Traffic Forecast Update that has been released in February 2017.

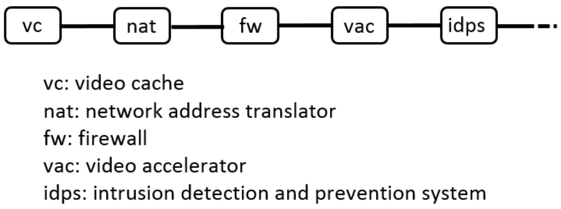

vc
nat
fw
vac
idps
vc: video cache
nat: network address translator
fw: firewall
vac: video accelerator
idps: intrusion detection and prevention system

Fig. 1: An example of caching in conjunction with other virtual network functions.

the focus and main motivation of the paper is on enhancing proactive caching policies by taking into account the whole VNF chain.

II. Motivation and Illustrative Examples

As already eluded, in this paper, we propose a Proactive caching-chaining (PCC) scheme to enhance the mobility support of SDN-enabled/NFV service chaining in mobile networks. To motivate the research we detail hereafter illustrative examples of the possible cross issues between caching and VNF chaining with the aim to shed further light on some of the key challenges. To start with, Figure 2 shows the case of a service with two VNFs where the first one is caching and the other one is assumed to be a video acceleration network function. As can be seen from the figure, Case I entails a sub-optimal allocation when mobility of the end user is not taken into account. However, Case II shows a more suitable VNF location where after the mobility event the cache and chain location is topologically closer to the end user; note that in Case II the VNFs are located 3 hops away from the end user after the mobility event whereas in Case I, which is a mobility oblivious allocation, the VNFs are located 4 hops away. Figure 3 shows the case where VNF chaining and pro-active caching take place independently. The figure shows potential pro-active caching locations but not in all of those pre-selected locations from the caching algorithm it is possible to host the other NFs due to numerous reasons such as for example reservation policies, placement based on affinity and/or anti-affinity rules and overall resource usage of the virtual machines [6]; for example only in one of those locations the two VNF can be co-located (node b). Furthermore, as shown in figure 4 the optimal location of caching and the other VNF in the service chain might be different; in the figure shown the optimal location of caching is in node (b) whereas the VNF for video acceleration is located at node (d) (assuming each node can only host one NF). It is therefore important to consider the issue of caching and service chaining in a holistic integrated manner and subsequently to optimize the location and chaining of the different VNFs in order to increase overall network performance.

Based on the above discussion, the proposed scheme performs proactive caching and VNF chaining so that overall network performance is optimized whilst end user receive their service requests seamlessly. Notably, we take VNF chaining allocation and proactive caching as a joint problem and formulate it as an Integer Linear Programming (ILP) problem with the objective to minimize the combined cost of VNF placement, chaining and routing. We also investigate the performance obtained of a proposed scale-free heuristic algorithm since the problem resembles the NP-hard uncapacitated facility location problem.

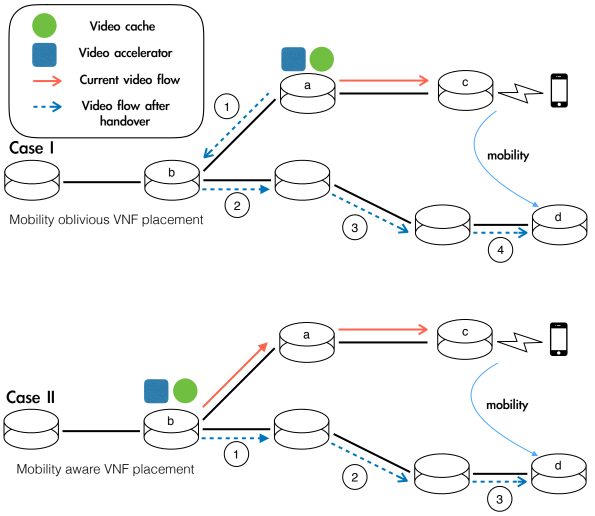


Fig. 2: Effect of mobility on the joint caching VNF chaining problem.

In summary, we hereafter make the following key contributions, We firstly, propose a novel VNF chaining placement scheme, namely, proactive caching-chaining (PCC) that improves the mobility support for the up coming SDN-enabled/NFV network framework. Secondly, we model and formulate the VNF chaining problem for proactive caching to obtain optimal routing and placement cost and based on that we devise a scalable heuristic approach and evaluate the performance of the system. Finally, we meticulously examine the performance of the proposed schemes under various network conditions. This paper is extended from our previous work [7], in which two scalable heuristics are proposed for seeking the suitable VNF chains for proactive caching. We extend the study by finding the theoretical optimal solution to calculate the difference of cost performance between the proposed heuristics and the theoretical optimal value. Moreover, the analysis on a comprehensive set of practical network scenarios explains to what extent the proposed heuristics can find approximative solutions effectively.

III. Previous Closely Related Research Work

The overall logical/functional architecture of the VNF Management and Orchestration (MANO) framework has been mainly an industry-lead initiative and has been defined within ETSI [4]. An example of a VNF orchestrator, which is called *Stratos* is presented in [8] and is built on top of a *Floodlight*[3] controller. The work in [9] can be considered as another effort to provide orchestration between virtualized NFs especially with emphasis on issues such as Virtual Machine (VM) migration and split/merging of service flows. An overview of

[3] `www.projectfloodlight.org`

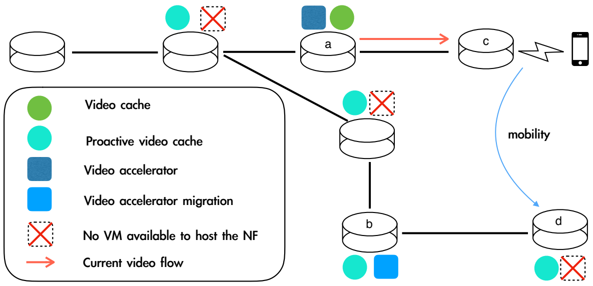


Fig. 3: Limited availability of resources (in terms of Virtual Machines for example) in the candidate pro-active caching locations to host the required VNFs for the service.

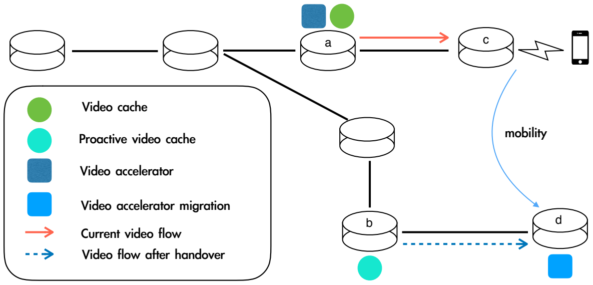


Fig. 4: An optimal VNF chain NF are located in different nodes in the network.

the challenges emerging in virtual network function scheduling is presented in [10]; in this paper the authors explain the application of SDN and NFV technologies with emphasis on backbone networks.

In terms of caching there has been recently a significant amount of work. A caching scheme suitable for mobile networks that takes into account user mobility has been proposed in [11] where the idea is to predict the mobility pattern of users and opportunistically cache content along the predicted path of users. A scheme that pro-actively cache content using transportation and focusing on video content has been presented in [12]. The idea is to utilize the almost deterministic mobility of users in transportation systems such as trains to proactively cache popular content that the users might request upon their arrival. The ideas on proactive caching in this paper resemble more closely the work in [13], [14] which propose a set of mobility-aware caching schemes.

On the other hand, VNF placement problem has recently received wide attention from both industry and academia. In work [15], the authors address VNF placement for unordered service chains in the cloud with the objective to satisfy as many tenants' requests as possible. In [16], the optimization problem VNP-OP minimizes the cost (which can be further broken down into deploy cost, energy cost and cost of forwarding traffic) by carefully placing VNFs as well as their forwarding traffic through best available paths. The work focuses on providing VNF location optimization for fixed networks thus, mobility of end-users are not considered.

However, none of previous research works make caching decisions on a view of the whole VNF service chain, especially for mobile networks. To the best of our knowledge this is the first work to consider in an explicit and integrated manner proactive caching as part of a VNF chain. In most practical cases, this simple cache moving could lead to inefficient routing of a mobile user to receive a service. Fig 1 gives an example of the inefficient routing problem where firewall as a NF must also be visited and only cache is moved [4]. It is apparent that, in order to improve the mobility support of SDN-enabled networking, other NFs on a same VNF service chain must also be moved, with the decision of caching. A close related work can be found in [17] which aims to assign VNFs into given SDN-enabled networks. However, it does not take routing and location of VNFs into consideration.

IV. Network Modeling and Proactive Chaining with Caching

An arbitrary mobile network is modeled as an undirected graph $G = (\mathbf{N}, \mathbf{E})$, where $\mathbf{N}$ denotes the set of nodes and $\mathbf{E}$ denotes the set of links in the network. By $\mathbf{F}$, we denote the set of NFs and f_i represents the specific NF$_i$. Each f_i, if activated, consumes/requires some physical resources (i.e., CPU cycles, DRAM memory). We uniformly describe these resource requirements as a single column matrix u_i, meanwhile, the amount of available resources of node k, which is able to host VNFs, is denoted using the single column matrix U_k.

The term "chain" in the so-called service chaining represents the different middleboxes that the service should traverse, with a specific order, across the network using software provisioning. This is the case under the proposed NFV architecture, where new services and/or network slices can be instantiated as software-only, running on commodity hardware on top of virtual machines or containers. To provide a service request $r \in \mathbf{R}$ (with $\mathbf{R}$ we denote the set of requests[5]) for a mobile user and/or tenant, a network function forwarding graph (VNF-FG) [18] needs to access a set of corresponding NFs that are visited in a pre-defined order (which the VNF orchestrator should preserve). In this paper, we consider the form of service request r as the set $r = \{f_1, f_2, \cdots, f_i\}$ where the function appearance sequence expresses the visiting order of the different network functions while the function index expresses a specific network function. The proposed optimization scheme provides a batch processing based service that the requests are handled in batches, such that the number of requests processed in each batch is $|\mathbf{R}|$. For modeling simplification reasons, the corresponding relationship of a NF and its order in a request can be represented by a binary matrix V_{ril} as follows,

$$V_{ril} = \begin{cases} 1 & \text{if the } l^{th} \text{ NF of request } r \text{ is NF}_i. \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Hereafter, we consider the scenario where a mobile user and/or tenant connected to node o and requesting $\mathbf{R}$ services.

[4] NF movement in this paper refers to any approach that occurs the change of the function's location. (e.g., proactive caching)

[5] As part of the standard [19], VNF chain deployment in this paper is on a per request basis. Nevertheless, coarser grain can be applied without change to our model by referring the request as a set of traffic. This is based on the granularity of VNF deployment and usage.

As presented in Figure 2, caching as a NF, is the head of a service request chain and it is denoted as f_0. We define a candidate node set $\mathbf{K} \subseteq \mathbf{N}$ that consist of the potential candidate nodes of hosting NFs. By $\mathbf{D}$, we define a set of potential destinations that mobile users might move due to their inherent mobility. Using historical data available to mobile network providers it is feasible to estimate such probabilities of end users moving from their current location to an adjacent candidate destination node d. We denote this probability of changing their serving access router with ρ_d. As eluded, we assume that ρ_d is predefined by using available historical data from operators so this assumption can be deemed as realistic due to vast available data which can provide accurate characterization of user mobility patterns. With known candidate cache locations, which can be done using for example a proactive caching technique such as PCWR [13]), PCC aims to proactively place network functions $f_i \in \mathbf{F}$ into the set of nodes $\mathbf{K}$. To be more precise, we define by $\mathbf{S_r}$ to be the set of initiating nodes (i.e., proactive caching locations) of a service chain r, with $\mathbf{H}$ denoting the set of $\mathbf{S_r}$. Given $\mathbf{H}$ and $\mathbf{D}$, the proposed scheme returns the optimal proactive allocation of the NFs that minimizes the joint cost of routing, location and chaining. To sum up, the key notations we used in this paper are listed in I

TABLE I: Notations

Notation	Meaning
$\mathbf{R}$	Set of requests arrive in a batch
$\mathbf{K}$	Set of NFV enabled candidate hosts
$\mathbf{F}$	Set of Network Functions
$\mathbf{S_r}$	Set of caching points for request r
$\mathbf{D}$	Set of potential destinations
C_i^k	The cost for placing VNF i at node k
ρ_d	The probability of a mobile user moving to destination d
P_{km}	The routing cost of the path from node k to m
V_{ril}	Indicator of VNF i if it is the l^{th} function of request r
u_i	Physical resource requirement of VNF i
U_k	Physical resource capacity of node k
λ_r	Flow rate requirement of request r
Λ_{km}	Link capacity of the path from node k to m
x_{ri}^k	Decision variable indicates whether VNF i is placed at k for request r
y_{ri}^{ksd}	Decision variable indicates whether VNF i of request r with caching point s and destination d is visited from k
z_{rij}^{kmsd}	Auxiliary variable defined as $z_{rij}^{kmsd} = y_{ri}^{ksd} y_{rj}^{msd}$

A. Proactive chaining-caching problem

Based on the previously described network settings we define the following binary decision variables,

$$x_{ri}^k = \begin{cases} 1 & \text{if NF}_i \text{ is placed at } k \text{ for request } r. \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$$y_{ri}^{ksd} = \begin{cases} 1 & \text{if NF}_i \text{ of request } r \text{ with head } s \text{ and} \\ & \text{destination } d \text{ is visited from } k. \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The optimal VNF location and chaining for the proactive caching problem is defined as the following non-linear integer optimization problem,

$$\begin{aligned} \min_{x_{ri}^k, y_{ri}^{ksd}} & \sum_{r\in\mathbf{R}} \sum_{k\in\mathbf{K}} \sum_{i\in\mathbf{F}} C_i^k x_{ri}^k + \sum_{r\in\mathbf{R}} \sum_{s\in\mathbf{S_r}} \sum_{d\in\mathbf{D}} \sum_{k\in\mathbf{K}} \sum_{i\in\mathbf{F}} \rho_d P_{sk} V_{ri1} y_{ri}^{ksd} \\ & + \sum_{r\in\mathbf{R}} \sum_{s\in\mathbf{S_r}} \sum_{d\in\mathbf{D}} \sum_{k,m\in\mathbf{K}} \sum_{i,j\in\mathbf{F}} \sum_{l=1}^{L-1} \rho_d P_{km} V_{ril} y_{ri}^{ksd} V_{rj(l+1)} y_{rj}^{msd} \\ & + \sum_{r\in\mathbf{R}} \sum_{s\in\mathbf{S_r}} \sum_{d\in\mathbf{D}} \sum_{k\in\mathbf{K}} \sum_{i\in\mathbf{F}} \rho_d P_{kd} V_{riL} y_{ri}^{ksd} \end{aligned} \tag{4}$$

$$\text{S.t.} \sum_{r\in\mathbf{R}} \sum_{i\in\mathbf{F}} u_i x_{ri}^k \le U_k, \forall k \in \mathbf{K} \tag{4a}$$

$$\sum_{r\in\mathbf{R}} \sum_{d\in\mathbf{D}} \sum_{i\in\mathbf{F}} \lambda_r V_{ri1} y_{ri}^{ksd} \le \Lambda_{sk}, \forall r \in \mathbf{R}, k \in \mathbf{K}, s \in \mathbf{S_r} \tag{4b}$$

$$\sum_{r\in\mathbf{R}} \sum_{s\in\mathbf{S_r}} \sum_{d\in\mathbf{D}} \sum_{i,j\in\mathbf{F}} \sum_{l=1}^{L-1} \lambda_r V_{ril} V_{rj(l+1)} y_{ri}^{ksd} y_{rj}^{msd} \le \Lambda_{km}, \quad \forall k, m \in \mathbf{K} \tag{4c}$$

$$\sum_{r\in\mathbf{R}} \sum_{s\in\mathbf{S_r}} \sum_{i\in\mathbf{F}} \lambda_r V_{riL} y_{ri}^{ksd} \le \Lambda_{kd}, \forall k \in \mathbf{K}, d \in \mathbf{D} \tag{4d}$$

$$\sum_{k\in\mathbf{K}} \sum_{i\in\mathbf{F}} V_{ril} y_{ri}^{ksd} \ge 1, \ \forall r \in \mathbf{R}, s \in \mathbf{S_r}, d \in \mathbf{D}, \quad l = 1, \ldots L \tag{4e}$$

$$y_{ri}^{ksd} - x_{ri}^k \le 0, \forall r \in \mathbf{R}, i \in \mathbf{F}, k \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{4f}$$

$$x_{ri}^k \in \{0, 1\}, \quad \forall i \in \mathbf{F}, k \in \mathbf{K} \tag{4g}$$

$$y_{ri}^{ksd} \in \{0, 1\}, \quad \forall r \in \mathbf{R}, i \in \mathbf{F}, k \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{4h}$$

where C_i^k is the cost of placing NF$_i$ at k. While P_{sk}, P_{km} and P_{kd} are the shortest path routing costs between the candidate nodes. Accordingly, Λ_{sk}, Λ_{km} and Λ_{kd} are the remaining link capacities of a path (i.e., the bottleneck link capacity). Further, notice that Λ_{km} can be seen as flow rate tolerant of a node when $k = m$. λ_r denotes the flow rate requirement of request r. Constraint (4a) is the VNF processing capacity constraint which takes into account the CPU cycles associated with the Virtual Machine(VM) allocated to a VNF and the memory capacity for a specific VNF. (4b)-(4d) are the QoS constraints related to the service chain such that the requests can be properly assigned based on the flow rate requirements and the link capacity. (4e) enforces that each NF in a requested chain must be visited at least once. (4f) is a binding constraint that insures the availability of a NF at a node is valid only when the NF is hosted at the node.

The first term of the objective function is the placement cost of hosting VNFs at a node. The rest of the terms in the objective function reflect the accumulative routing cost of each hop on the VNF-FG of a requested chain. To linearize the optimization problem, we replace the product of binary decision variables $y_{ri}^{ksd} y_{rj}^{msd}$ with an auxiliary variable z_{rij}^{kmsd}, which is defined as follows,

$$z_{rij}^{kmsd} = \begin{cases} 1 & \text{if request } r \text{ with head } s \text{ and destination } d \\ & \text{visits NF}_i \text{ at node } k \text{ and NF}_j \text{ at node } m. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Hereafter, the optimization problem is converted to the integer linear programming problem shown as follows,

$$\begin{aligned} \min_{x_{ri}^k, y_{ri}^{ksd}} & \sum_{r\in\mathbf{R}}\sum_{k\in\mathbf{K}}\sum_{i\in\mathbf{F}} C_i^k x_{ri}^k + \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S_r}}\sum_{d\in\mathbf{D}}\sum_{k\in\mathbf{K}}\sum_{i\in\mathbf{F}} \rho_d P_{sk} V_{ri1} y_{ri}^{ksd} \\ & + \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S_r}}\sum_{d\in\mathbf{D}}\sum_{k,m\in\mathbf{K}}\sum_{i,j\in\mathbf{F}}\sum_{l=1}^{L-1} \rho_d P_{km} V_{ril} V_{rj(l+1)} z_{rij}^{kmsd} \\ & + \sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S_r}}\sum_{d\in\mathbf{D}}\sum_{k\in\mathbf{K}}\sum_{i\in\mathbf{F}} \rho_d P_{kd} V_{riL} y_{ri}^{ksd} \end{aligned} \tag{6}$$

$$\text{S.t.} \sum_{r\in\mathbf{R}}\sum_{i\in\mathbf{F}} u_i x_{ri}^k \le U_k, \forall k \in \mathbf{K} \tag{6a}$$

$$\sum_{r\in\mathbf{R}}\sum_{d\in\mathbf{D}}\sum_{i\in\mathbf{F}} \lambda_r V_{ri1} y_{ri}^{ksd} \le \Lambda_{sk}, \forall r\in\mathbf{R}, k\in\mathbf{K}, s\in\mathbf{S_r} \tag{6b}$$

$$\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S_r}}\sum_{d\in\mathbf{D}}\sum_{i,j\in\mathbf{F}}\sum_{l=1}^{L-1} \lambda_r V_{ril} V_{rj(l+1)} y_{ri}^{ksd} y_{rj}^{msd} \le \Lambda_{km}, \quad \forall k, m \in \mathbf{K} \tag{6c}$$

$$\sum_{r\in\mathbf{R}}\sum_{s\in\mathbf{S_r}}\sum_{i\in\mathbf{F}} \lambda_r V_{riL} y_{ri}^{ksd} \le \Lambda_{kd}, \forall k \in \mathbf{K}, d \in \mathbf{D} \tag{6d}$$

$$\sum_{k\in\mathbf{K}}\sum_{i\in\mathbf{F}} V_{ril} y_{ri}^{ksd} \ge 1, \ \forall r \in \mathbf{R}, s \in \mathbf{S_r}, d \in \mathbf{D}, \quad l = 1, \dots L \tag{6e}$$

$$y_{ri}^{ksd} - x_{ri}^k \le 0, \forall r \in \mathbf{R}, i \in \mathbf{F}, k \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6f}$$

$$z_{rij}^{kmsd} \le y_{ri}^{ksd}, \forall r \in \mathbf{R}, i \in \mathbf{F}, k \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6g}$$

$$z_{rij}^{kmsd} \le y_{rj}^{msd}, \forall r \in \mathbf{R}, j \in \mathbf{F}, m \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6h}$$

$$z_{rij}^{kmsd} \ge y_{ri}^{ksd} + y_{rj}^{msd} - 1, \forall r \in \mathbf{R}, i, j \in \mathbf{F}, k, m \in \mathbf{K}, \quad s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6i}$$

$$x_{ri}^k \in \{0, 1\}, \quad \forall i \in \mathbf{F}, k \in \mathbf{K} \tag{6j}$$

$$y_{ri}^{ksd} \in \{0, 1\}, \quad \forall r \in \mathbf{R}, i \in \mathbf{F}, k \in \mathbf{K}, s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6k}$$

$$z_{rij}^{kmsd} \in \{0, 1\}, \quad \forall r \in \mathbf{R}, i, j \in \mathbf{F}, k, m \in \mathbf{K}, \quad s \in \mathbf{S_r}, d \in \mathbf{D} \tag{6l}$$

where (6f)-(6i) are binding constraints that insure z_{rij}^{kmsd} taking the same value as product $y_{ri}^{ksd} y_{rj}^{msd}$.

V. Scale Free Heuristic Approaches

The PCC problem falls within the family of $\mathcal{NP}$-hard problems (the detailed proof can be seen in the Appendix). As a result, heuristics becomes the only viable option of finding competitive feasible solutions for real time operation. Therefore, we present three heuristic approaches for finding caching points and VNFs allocations that ponder features including user mobility, network capacity utility and request importance. Nevertheless, the presented schemes differ in the performance of obtaining lower routing cost and the ability of providing smaller pro-active service missing rate.

First, we propose Probability-prior proactive caching-chaining (PPCC) which aims at minimizing the overall network traffic cost with the awareness of end user mobility. Based on user moving trend, PPCC places VNF chains between a pro-active caching point and the potential user moving destination that with highest ρ_d.

The second algorithm, hereafter called Shortest Path Based Allocation (SPBA) also allocates caching as well as VNFs along the shortest path from pro-active caching points to serving access routers but without taking user mobility into consideration. The SPBA is presented as a mobility-unaware baseline where allocation decision is made by assuming the current accessing node is the permanent destination.

Finally, in the last algorithm, henceforth called All in Gateway (AGW), hosts all content caches and VNFs at the network gateway. With a straightforward decision, AGW shows a lower bound on the network traffic cost performance, in which no optimization techniques are applied.

A. PPCC algorithm

In order to handle end user mobility feature, we propose PPCC approach in which caching and VNF chains allocations are decided based on user moving trend. In particular, the main philosophy of the proposed PPCC heuristic is to create a set of candidate pro-active caching points for each possible visited access router and then weighted by the probability of visiting each access router and explore node combinations for creating the service chains. This approach is highly efficient when the user movement is predicable. i.e., fixed route public transportations.

1) For any request r, select the target node $d \in \mathbf{D}$ by highest ρ_d and find the closest starting node $s \in \mathbf{S_r}$ by minimum shortest path routing cost P_{sd};
2) On the shortest path from the selected s and d, find all candidate nodes by $\mathbf{K}$;
3) Choose the closest k from the selected s on the path to host the NF$_i$ with the lowest visiting order sequence in request r if there are enough resources (including both the VNF processing capacity and the link capacity) to support the function, otherwise, host the sub-lowest function, until running out of resources;
4) Repeat step 2 and 3 until all NFs of request r are hosted.

In step 3, depending on different consideration of the features of network capacity utility and request importance, PPCC also provides two sub modes, namely, PPCC node first mode and PPCC function first mode, which are denoted as PPCC-k and PPCC-f respectively. In PPCC-f, step 1,2 and 4 are identical to PPCC-k and are omitted:

3) Choose the NF$_i$ with the lowest visiting order sequence in request r to host in the closest k from the selected s on the path if there are enough resources to support the function, otherwise, host in the sub-closest node, until the last node;

PPCC is detailed in the pseudocode Algorithm I below. In short, the main principle of PPCC-k is making use of the network capacity by risking VNF visiting order. While PPCC-f ensures the serving quality for a set of requests by over using some network resources.

B. SPBA algorithm

Similar to PPCC, SPBA moves towards the same objective of minimizing the network traffic routing cost. The distinction between it and PPCC is that SPBA calculates the allocation solution without the requirement of knowing the end user mobility trend. Accordingly, SPBA creates a set of candidate pro-active caching points for the original access router and then explores node combinations for creating the service chains.

1) For any request r, select node o and find the closest starting node $s \in \mathbf{S_r}$ by minimum shortest path routing cost P_{so};
2) On the shortest path from the selected s and o, find all candidate nodes by $\mathbf{K}$;
3) Choose the closest k from the selected s on the path to host the NF$_i$ with the lowest visiting order sequence in request r if there are enough resources to support the function (including both the VNF processing capacity and the link capacity), otherwise, host the sub-lowest function, until running out of resources;
4) Repeat step 2 and 3 until all NFs of request r are hosted.

Just as PPCC, SPBA also supports the two sub modes. SPBA is detailed in the pseudocode Algorithm II below.

C. AGW management scheme

With no optimization techniques involved, AGW provides a quick and simple solution, that for each request $r \in \mathbf{R}$, places all the requesting VNFs and content caches at the network gateway. As a result, all benefits of using pro-active caching techniques are lost. Moreover, with a mass of VNF entities running on the network gateway, the network resources of the gateway becomes the bottleneck.

VI. Calculation complexity analysis

In this section, we present the computation complexity analysis for the three heuristics. The analysis provides the incurred computational burden for each proactive caching and chaining enabled node and captures both the node choosing phase and the chaining allocation phase. The most important parameter of each approach is the total computing duration time for generating a chaining allocation, which is the sum of the calculation time of the two phases.

As mentioned in V-A, PPCC selects a target node in $\mathbf{D}$ and a caching point in $\mathbf{S_r}$. We define such phase as the node choosing phase for the three algorithms. For PPCC, a maximum value finding problem and a minimum value finding problem are involved in the node choosing phase. Therefore, finding the appropriate target node and caching node, PPCC needs an upper bound of $|\mathbf{D}|-1$ and $|\mathbf{S_r}|-1$ comparisons for each request $r \in \mathbf{R}$, that is, examining each element of the set in turn and keep track of the largest/smallest element seen so far. As a result, the computation complexity of the node choosing phase of PPCC is $O(|\mathbf{D}|) + O(|\mathbf{S_r}|)$.

Algorithm 1: PPCC

```
Input  : $\mathbf{G}$; $\mathbf{D}$; $\mathbf{R}$; $\mathbf{K}$; $\mathbf{F}$; $\mathbf{H}$; attaching node $o$;
Output: VNF allocation: $x^k_{ri}$; PPCC cost: PPCC;
PPCC← 0;
for $k \in \mathbf{K}$ do
    Remaining utility of node $k$: $RU_k \leftarrow U_k$
end
Initialize all path bottlenecks: $R\Lambda \leftarrow \Lambda$;
for $i \in \mathbf{D}$ do
    if $\rho_i == max(\rho_i)$ then
        Destination node: $d \leftarrow i$;
    end
end
for $r \in \mathbf{R}$ do
    Starting node:$s \leftarrow$ find closest node $s$ to $d$ in $S_r$ with
      minimum $P_{sd}$;
    candidate node priority list: $CPL \leftarrow \emptyset$;
    $CPL \leftarrow$ sort $k \in \{ \{n|n$ is on the shortest path from
      s to d$\} \cap \mathbf{K} \}$ by the distance between $k$ and $s$
      from low to high;
    VNF priority list: $FPL \leftarrow \emptyset$;
    $FPL \leftarrow$ sort $f_i$ by its visiting sequence $l$ of $r$;
    former VNF location: $m \leftarrow s$ ;
    :// PPCC-k
    for $k \in CPL$ do
        for $f_i \in FPL$ do
            if $u_i \leq RU_kn$ & $\lambda_r \leq R\Lambda_{km}$ then
                host $f_i$ at $k$: $x^k_{ri} \leftarrow 1$ ;
                $RU_k \leftarrow RU_k - u_i$;
                PPCC ← PPCC+$C^k_i$;
            end
        end
    end
    :// PPCC-f
    for $f_i \in FPL$ do
        for $k \in CPL$ do
            if $u_i \leq RU_k$ & $\lambda_r \leq R\Lambda_{km}$ then
                host $f_i$ at $k$: $x^k_{ri} \leftarrow 1$ ;
                $RU_k \leftarrow RU_k - u_i$;
                PPCC ← PPCC+$C^k_i$;
            end
        end
    end
    update $R\Lambda$;
end
for $d \in \mathbf{D}$ do
    for $r \in \mathbf{R}$ do
        Length of the chain requested by $r$: $L_r \leftarrow$ the
          number of requested VNFs by $r$;
        VNF chaining list: $I \leftarrow$ sort $f_i$ by its visiting
          sequence $l$ of $r$;
        Chaining Routing cost between $I(j)$ and $I(j+1)$:
          $CR_{j,j+1} \leftarrow$ cumulative $P_{km}$ where $k$ hosts $I(j)$
          and $m$ host $I(j+1)$ which is given by $x^k_{ri}$;
        Chaining Routing Cost: $CRC \leftarrow 0$;
        $CRC \leftarrow$ find the cost of shortest chaining
          routing path from $I(0)$ to $d$;
        PPCC← PPCC+$\rho_d CRC$;
    end
end
```

Algorithm 2: SPBA

```
Input  : $\mathbf{G}$; $\mathbf{D}$; $\mathbf{R}$; $\mathbf{K}$; $\mathbf{F}$; $\mathbf{H}$; attaching node $o$;
Output: VNF allocation: $x^k_{ri}$; SPBA cost: SPBA;
SPBA← $0$;
for $k \in \mathbf{K}$ do
    Remaining utility of node $k$: $RU_k \leftarrow U_k$;
end
Initialize all path bottlenecks: $R\Lambda \leftarrow \Lambda$;
for $r \in \mathbf{R}$ do
    Starting node:$s \leftarrow$ find closest node $s$ to $o$ in $S_r$ with
      minimum $P_{so}$;
    candidate node priority list: $CPL \leftarrow \emptyset$;
    $CPL \leftarrow$ sort $k \in \big\{ \{n|n$ is on the shortest path from
      s to o$\} \cap \mathbf{K} \big\}$ by the distance between $k$ and $s$
      from low to high;
    VNF priority list: $FPL \leftarrow \emptyset$;
    $FPL \leftarrow$ sort $f_i$ by its visiting sequence $l$ of $r$;
    former VNF location: $m \leftarrow s$ ;
    :// SPBA-k
    for $k \in CPL$ do
        for $f_i \in FPL$ do
            if $u_i \leq RU_k$ & $\lambda_r \leq R\Lambda_{km}$ then
                host $f_i$ at $k$: $x^k_{ri} \leftarrow 1$ ;
                $RU_k \leftarrow RU_k - u_i$;
                SPBA ← SPBA+$C^k_i$;
            end
        end
    end
    :// SPBA-f
    for $f_i \in FPL$ do
        for $k \in CPL$ do
            if $u_i \leq RU_k$ & $\lambda_r \leq R\Lambda_{km}$ then
                host $f_i$ at $k$: $x^k_{ri} \leftarrow 1$ ;
                $RU_k \leftarrow RU_k - u_i$;
                SPBA ← SPBA+$C^k_i$;
            end
        end
    end
    update $R\Lambda$;
end
for $d \in \mathbf{D}$ do
    for $r \in \mathbf{R}$ do
        Length of the chain requested by $r$: $L_r \leftarrow$ the
          number of requested VNFs by $r$;
        VNF chaining list: $I \leftarrow$ sort $f_i$ by its visiting
          sequence $l$ of $r$;
        Chaining Routing cost between $I(j)$ and $I(j+1)$:
          $CR_{j,j+1} \leftarrow$ cumulative $P_{km}$ where $k$ hosts $I(j)$
          and $m$ host $I(j+1)$ which is given by $x^k_{ri}$;
        Chaining Routing Cost: $CRC \leftarrow 0$;
        $CRC \leftarrow$ find the cost of shortest chaining
          routing path from $I(0)$ to $d$;
        SPBA← SPBA+$\rho_d CRC$;
    end
end
```

After selecting an appropriate target node and caching point, PPCC starts seeking for a set of candidate nodes $k \in \mathbf{K}$ along the shortest path from the selected caching point to the selected target to allocate the requesting VNFs. We define such phase as the chaining allocation phase. In order to minimize the probability of having packets being switched back and forth along the path, PPCC greedily gives the highest priority to the candidate node that has the closest distance from the caching point to hold VNFs. Hence, a sorted list of the candidate VNF holding nodes is required for the PPCC caching allocation phase. Notice that, the calculation complexity of solving a sorting list problem is verified based on different solutions. Through out the paper, we use quick sorting for the sorting list problems, which give us an average of $O(\kappa \log \kappa)$ calculation complexity and $O(\kappa^2)$ complexity in the worst case, where κ denotes the number of candidate nodes k along the shortest path between the caching point and the target point. On the other hand, the priority of the requesting VNF to hold is scaled down by its visiting order in the corresponding request. This is a result of trying to preserve the sequence of a chain such that path segment overlap is kept at a minimum. As a result, the computation complexity for sorting VNF to hold is $O(\phi \log \phi)$ in average and $O(\phi^2)$ for the worse case, where ϕ denotes the number of requested VNF f of a request r. Notice that, the node first mode and the function first mode are interchanging the processes of looping over the two sorted lists. Hence, the computation complexity of the two sub modes are identical.

Unlike PPCC, SPBA does not choose any target node in $\mathbf{D}$ in its node choosing phase, instead, it finds the closest caching points $s \in \mathbf{S_r}$ to node o for each request $r \in \mathbf{R}$. In this case, $|\mathbf{S_r}| - 1$ comparisons are needed for finding the closest caching point for each request r, which leads to a calculation complexity of $O(|\mathbf{S_r}|)$. In the chaining allocation phase, SPBA is identical with PPCC and therefore, the calculation complexity of the chaining allocation phase of SPBA is $O(\kappa \log \kappa) + O(\phi \log \phi)$ in average and $O(\kappa^2) + O(\phi^2)$ for worse case as well.

AGW has constant calculation complexity for both the node choosing phase and the chaining allocation phase as it holds every VNF in the gateway. In summary the calculation complexity of each heuristic approaches is presented in Table II.

VII. Optimal Solution Scalability Analysis

As discussed earlier, the proactive chaining-caching problem resembles the *NP-hard* UFL problem. In this section, we focus on the scalability analysis for the optimal solution against the increasing problem size. Precisely, both the calculation scalability and the storage space scalability are in the scope of our discussion.

Here we denote a proactive chaining-caching and routing problem by $\pi(\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F})$, where the size of this problem is determined by the network size $\mathbf{K}$, the request size $\mathbf{R}$, the caching nodes of a request $\mathbf{S_r}$, the number of access nodes in $\mathbf{D}$, the length L of a chain (the number of VNFs in a chain) and the VNF size $\mathbf{F}$. For legibility, hereafter π and $\pi(\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F})$ are used interchangeably unless when differentiation is required.

TABLE II: Calculation Complexity

Heuristic	Node Choosing Phase	Chaining Allocation Phase (Avg.)	Chaining Allocation Phase (worst case)
PPCC-k	$O(\lvert\mathbf{D}\rvert) + O(\lvert\mathbf{S_r}\rvert)$	$O(\kappa \log \kappa) + O(\phi \log \phi)$	$O(\kappa^2) + O(\phi^2)$
PPCC-f	$O(\lvert\mathbf{D}\rvert) + O(\lvert\mathbf{S_r}\rvert)$	$O(\kappa \log \kappa) + O(\phi \log \phi)$	$O(\kappa^2) + O(\phi^2)$
SPBA-k	$O(\lvert\mathbf{S_r}\rvert)$	$O(\kappa \log \kappa) + O(\phi \log \phi)$	$O(\kappa^2) + O(\phi^2)$
SPBA-f	$O(\lvert\mathbf{S_r}\rvert)$	$O(\kappa \log \kappa) + O(\phi \log \phi)$	$O(\kappa^2) + O(\phi^2)$
AGW	$O(1)$	$O(1)$	$O(1)$

A. Calculation Scalability

Given problem π, its ILP problem can be presented in the standard matrix form. By Π we denote the constraint coefficients matrix of π. While $Row(\Pi)$ and $Col(\Pi)$ are the functions that return the number of rows and columns in Π respectively. Let $\Omega(\pi)$ be the function returns the number of computations for finding the optimal solution of π. In this case, we measure the computational growth of the proactive chaining-caching problem by the scalability factor α_π, which is defined as follows,

$$\alpha_\pi = \frac{\Omega(\pi)}{\Omega(\pi_o)} \tag{7}$$

where $\Omega(\pi_o)$ is the calculations of the original problem and it is given by:

$$\Omega(\pi_o) = \Omega(\pi(1,1,1,1,1,1)) \tag{8}$$

A solution of problem π is the set of the binary variables i.e., x, y and z in formula (6). Since verifying an optimal solution requires two processes, the feasibility verification for each solution and the minimum value searching among the solutions. Therefore, for any problem π, we derive that $\Omega(\pi)$ holds:

$$\Omega(\pi) = \Omega_{fv}(\pi) * \Omega_{fm}(\pi) \tag{9}$$

where $\Omega_{fv}(\pi)$ returns the number of calculations for verifying the feasibility for each solution of π and $\Omega_{fm}(\pi)$ returns the number of calculations for finding the minimum cost solution for π.

For each solution of π, the feasibility verification needs to exhaustively traverse each constraint in the worst case. Consequently, $Row(\Pi)$ calculations are involved and hence we have

$$\Omega_{fv}(\pi) = Row(\Pi) \tag{10}$$

From formula (6), we know

$$\begin{aligned} Row(\Pi) = n|\mathbf{K}| + |\mathbf{RS_rK}| + |\mathbf{DK}| + |\mathbf{RS_rD}L| \\ +|\mathbf{K}^2| + |\mathbf{RS_rDKF}| + 3|\mathbf{RS_rDK}^2\mathbf{F}^2| \end{aligned} \tag{11}$$

where n is the size of u_i or U_k, i.e., the number of dimensions of the resources to hold a VNF.

On the other hand, finding the objective value of each solution involves one calculation. While searching for the optimal solution given the amount of the solutions requires a number of calculations that equals to the amount of the solutions in the worst case. Since for any problem π, the variables are binary, we derive the number of candidate solutions of a problem π is $2^{Col(\Pi)}$ and hence

$$\Omega_{fm}(\pi) = 2^{Col(\Pi)} \tag{12}$$

From formula (6), we know

$$Col(\Pi) = |\mathbf{RKF}| + |\mathbf{RS_rDK}| + |\mathbf{RS_rDK}^2\mathbf{F}^2| \tag{13}$$

Using Eq. (8) to (13), we obtain:

$$\Omega(\pi_o) = (n+8) * 2^3 \tag{14}$$

By replacing Eq. (10), (12) and (14) into (7), we derive the calculation scalability factor of proactive chaining-caching:

$$\alpha_\pi = \frac{Row(\Pi) * 2^{Col(\Pi)}}{(n+8) * 2^3} \tag{15}$$

In summary the calculation scalability of the proactive chaining-caching problem is presented in Table III.

TABLE III: Calculation Scalability

$\pi(\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F})$	$\Omega_{fv}(\pi)$	$\Omega_{fm}(\pi)$	α_π
(1,1,1,1,1,1)	$n+8$	2^3	1
(1,1,1,1,1,2)	$n+18$	2^8	$\frac{(n+15)*2^8}{(n+8)*2^3}$
(1,1,1,1,1,3)	$n+34$	2^{15}	$\frac{(n+31)*2^{15}}{(n+8)*2^3}$
……	…	…	…
$\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F}$	$Row(\Pi)$	$2^{Col(\Pi)}$	$\frac{Row(\Pi)*2^{Col(\Pi)}}{(n+8)*2^3}$

B. Memory Space Scalability

In order to solve the ILP, a computer needs to assign a certain amount of memory space to hold the coefficients matrices for π. To present a upper bound of the memory size requirement of π, we focus on the case of using full matrices. Without loss of generality, we use the constraint coefficients matrix memory space scalability to represent the memory space scalability of π, as it is the dominating storage space consumer.

Based on the above settings, we define $\Theta(\pi)$ the function returns the storage size requirement of the constraint coefficients matrix of a problem π. Similar to the calculation scalability, we measure the storage size requirement growth of π by the scalability factor β_π, which is defined as follows,

$$\beta_\pi = \frac{\Theta(\pi)}{\Theta(\pi_o)} \tag{16}$$

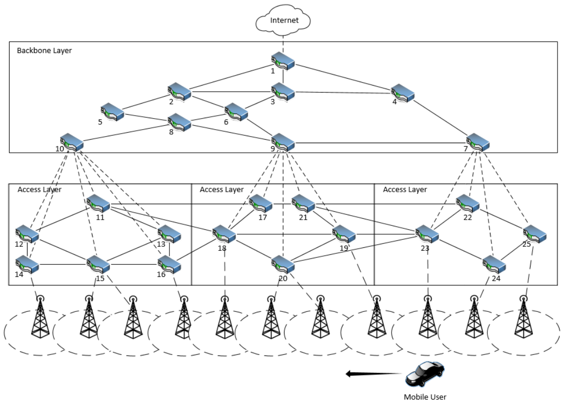


Fig. 5: A City Network like Topology Example

Using full matrix, we have

$$\Theta(\pi) = Row(\Pi) * Col(\Pi) \tag{17}$$

By replacing Eq. (11), (13) and (17) into (16), we derive the memory space scalability factor of proactive chaining-caching:

$$\beta_\pi = \frac{Row(\Pi) * Col(\Pi)}{(n+8)*3} \tag{18}$$

In summary the calculation scalability of the proactive chaining-caching problem is presented in Table IV

TABLE IV: Memory Space Scalability

$\pi(\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F})$	$\Theta(\pi)$	β_π
(1,1,1,1,1,1)	$(n+8)*3$	1
(1,1,1,1,1,2)	$(n+18)*8$	$\frac{(n+18)*8}{(n+8)*3}$
(1,1,1,1,1,3)	$(n+34)*15$	$\frac{(n+34)*15}{(n+8)*3}$
$\cdots\cdots$	$\cdots$	$\cdots$
$\mathbf{K}, \mathbf{R}, \mathbf{S_r}, \mathbf{D}, L, \mathbf{F}$	$Row(\Pi) * Col(\Pi)$	$\frac{Row(\Pi)*Col(\Pi)}{(n+8)*3}$

VIII. Numerical Investigations

The numerical investigation part consists of two sets of simulated network settings. The first part is devoted to examine the performance under different network topology types, in which we apply both optimal and heuristics on a set of random generated networks including a hybrid network, a star network and a ring network. In the second part, to further understand the performance under more general and practical network scenarios, we then carry out our simulations on a range of random network graphs that are generated with the well-used technique outlined in [20]; the key idea is that an edge between two nodes are generated based on a probability that is related to the distance between the two nodes.

Since the proposed scheme is highly effective in edge networks, we estimate the simulated network size by referencing the advanced 5G Ultra-Dense CellUlar network architecture with the single gateway model [21]. The applied random networks contain up to 400 small cells [21]. In typical settings, the number of deployed small cell per marco cell is 8 [22], however it can reach 20 if dense small cell deployment scenarios are considered [23] [24]. Together with the 1 : 1 router to marco cell base station ratio [25], we set the number of the candidate VNF hosting nodes $|\mathbf{K}|$ to be between 20 to 50. Moreover, the number of starting points and destination points are set from 1 to 5.

As to the number of requests in the system, we assume that a total of 200 requests per second [26] are generated and we take 0.25 to 1 seconds for each batch which converts to a total number of 50 to 200 requests per batch. We also assume that the flow rate requirement of a request varies from 64Kbps [27] (e.g. audio traffic) to 10Mbps [28] (e.g. 4K video streams). In terms of mobility, the moving probabilities to candidate destination nodes are randomly generated between 0 and 1 (in a non independent manner however, since all should add up to one). In order to flesh out in a clear manner the effect of routing we set the VNF placement cost to zero. As eluded in previous sections the proposed framework is generic one to encapsulate various different shortest path definitions; such as for example delay, congestion level at the node and/or energy consumption. However, and without loss of generality, we choose to use in the numerical investigations section typical metrics used in Open Shortest Path First (OSPF) or Enhanced Interior Gateway Routing Protocol (EIGRP) protocols. To maintain the link diversity, we normalize the routing metric in the range from 1 to 100. In terms of physical resources of candidate VNF hosting node, we assume that each candidate node has 8 to 16 GByte memory capacity and 32 virtual CPU cores (e.g. a CPU with 4 cores 8 threads). While each VNF consumes memory in a range from 10 to 50 MBytes and uses 0.125 to 0.25 cores (i.e., each virtual CPU supports 4 to 8 VMs). As for the link capacity, we assume each link has a capacity of 2Gbps.

To measure the end-user experience, we define the blocking chance as the probability of a request failing to build a complete proactive chain. We further assume the blocked requests are routed to the Internet via the gateway to simulate the impact of the broken chains that caused by mobility. In this case, the routing cost of a blocked request is consist of the destination-gateway shortest path cost and a penalty cost in a range from 500 to 700, which imitates a general Internet routing cost. All results presented hereafter are obtained by averaging 100 Monte Carlo simulations. Also we calculate the optimal allocations by utilizing MATLAB and its build-in Mixed-Integer Linear Programming solver. Further details and to sum up, the parameters that have been used in the investigations are presented in Table V.

A. *Performance evaluation for practical networks*

In this first part of the numerical investigations, we apply the Proactive chaining for caching technique on an urban network like topology as shown in figure 5. Without loss of generality, we assume that a mobile user moves across a number of Base Stations (BS)/Access Routers (AR) within an urban environment following a pseudo-predictable path (i.e., using navigation information from GPS). With proactive caching techniques, the pre-caching points are set to be at nodes 7,9

TABLE V: Simulation Parameters

Parameter	value
Number of candidate hosting nodes($\lvert\mathbf{K}\rvert$)	20-50
Degree per node	2-5
Moving probability (ρ_d)	0-1
Number of starting points per request ($\lvert\mathbf{S_r}\rvert$)	1-5
Number of destination points per user ($\lvert\mathbf{D}\rvert$)	1-5
Number of requests per batch ($\lvert\mathbf{R}\rvert$)	50-200
VNF number ($\lvert\mathbf{F}\rvert$)	10
Maximum number of VNF in a chain (L)	3-5
Routing cost per link (P_{km})	1-100
Routing penalty per blocking request	500-700
Cost per node to host VNF (C_i^k)	0
Memory capacity per candidate node	8-16 GByte
Number of virtual CPU cores per candidate node	32
Memory requirement per VNF	10-50 MByte
CPU core requirement per VNF	0.125-0.25
Flow rate requirement per chain (λ_r)	0.064 - 10 Mbps
Capacity per link (Λ_{km})	2 Gbps

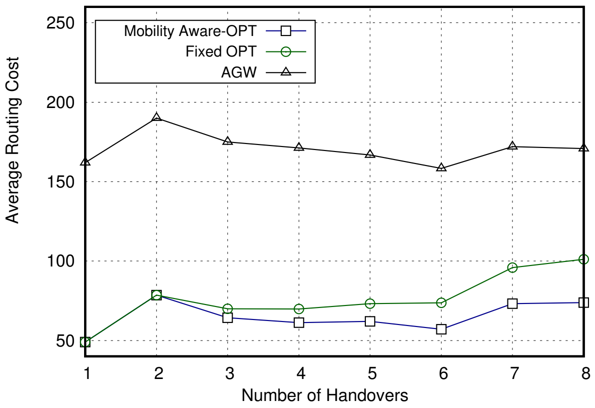


Fig. 6: Average gains of using proactive chaining for caching

and 10 respectively. In that respect, before the mobile user leaving for the next the BS, the proactive chaining for caching technique pre-allocate the request service chains along the nodes between the proactive caching point and the destination node. Figure 6 depicts the average gains when using proactive chaining for caching in that specific use case. Note that that the average routing cost when using proactive chaining for caching is significantly lower than in the case where we have a static VNF caching chain (i.e., mobility agnostic scheme). Also, as expected, when the mobile user move further away from its original connected BS the gains of the proposed optimal mobility-aware allocation scheme are increased compared to the baseline scheme.

To further study the performance of the proposed optimal and heuristic schemes on practical network topologies, we apply the proposal schemes in three different topologies; a random generated tree-like topology consisting of 25 nodes, a star network consisting of 26 nodes and a ring network with 13 nodes. Since the original optimization problem is $NP-hard$, we scale down the request size to 20 per batch in order to obtain the optimum. The network capacity is set to be approximately supporting 10 to 12 requests simultaneously. The rest of the network settings remain identical to the case as described in the previous experiments and summarized in Table IV. The aim is to investigate the impact of the topology type on the performance gains related to routing and blocking as shown in Table VI. The table depicts routing cost and blocking probability in three groups for low (R=5), mid (R=12) and high (R=20) number of requests. From the hybrid network we observe the optimal solution can achieve more than 75% gains against the sub-optimal heuristic for the low number of requests scenario whereas the gain is increased further in the other two scenarios. Unlike hybrid networks, in the star topology we obtain in average a 60% gain with the optimal allocation. This reduced gap between the optimal and heuristic solutions in star topology can be explained due to the more limited number of cache points for each access router. This trend can be found by comparing the costs listed in the Star and Hybrid fields of Table VI. In the star network, edge nodes have less degrees leading to that a limited set of links are concentrated in cache-destination paths.

The heuristic routing performance is further approximating the optimal one in the ring topologies scenarios. Precisely, the difference between the optimal solution and PPCC-k is less than 20%. It is also noteworthy that, due to each node has only 2 degrees in a ring network, the optimal allocation is highly identical to the PPCC solution and hence the routing deviations shown in ring scenarios are much smaller than that in other scenarios. Therefore, the result indicates that, in a small network with its node degrees are low, the PPCC scheme can very well balance the routing performance and the computational complexity.

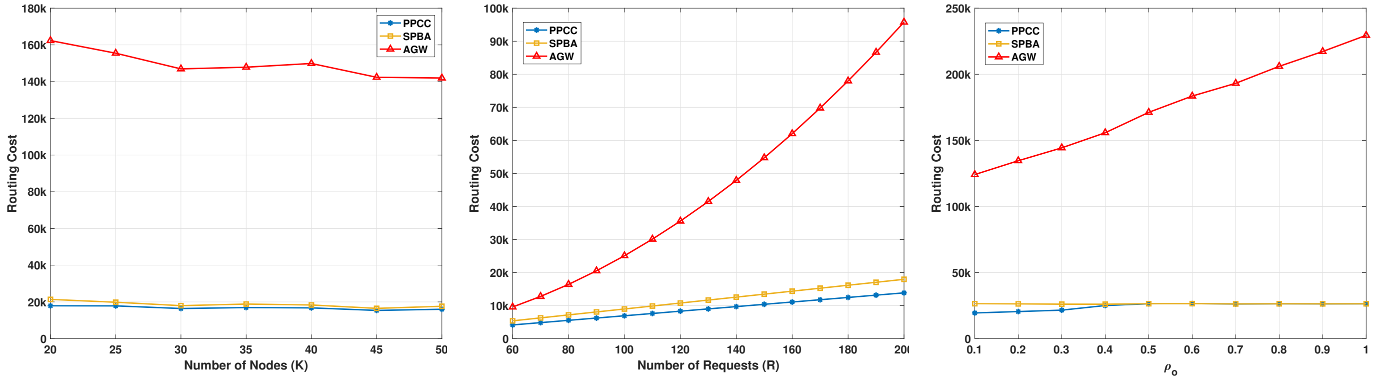
B. *Performance evaluation for scenarios of random generated topologies*

To better understand the practical implement of the proposed proactive chaining for caching technique, we test our proposed heuristics on a wider set of random generated network graphs. Figure 7-8 depicts the routing performance comparisons for different scenarios of random generated topologies. We divide the simulation into two cases. In the first case, which is called uncapacitated, every candidate node has a relatively high capacity e.g. memory requirement of each VNF is low, such that none of the requesting VNFs will be blocked in the sense that there are no available resources in the hosted nodes. While in the other case, which we call it as capacitated, we assume a congestion episode scenario where nodes do not have sufficient resource to support all the requesting VNFs simultaneously.

In figure 7, we depict the routing performance for the random generated networks in the uncapacitated case. For the simulations on the topologies that have sufficient capacity, the two sub modes of PPCC return identical solutions and SPBA holds this feature as well. As such we do not distinguish the two sub modes in the plot. Figure 7 (a) shows the performance of the proposed scheme compared to the previous mentioned baseline techniques for different number of nodes in the network. As can be seen from the figure, a performance gain of around 10% being mobility-aware (i.e., PPCC vs SPBA) can be achieved which remains robust against different network

TABLE VI: Routing and Blocking Performance of different schemes

		R	OPT	PPCC-k	PPCC-f	SPBA-k	SPBA-f	CAGW
Hybrid	Routing cost (k)	5	0.18	0.93	0.85	1.23	1.1	1.5
		12	0.41	1.9	2.1	3.16	4.46	12.61
		20	0.69	4.37	7.79	6.82	10.06	65.10
	Blocking Probability	5	0	0	0	0	0	0
		12	0	0	0.06	0	0.06	0.29
		20	0	0.16	0.44	0.16	0.44	0.58
Star	Routing cost (k)	5	0.19	1.80	1.54	2.90	2.71	4.60
		12	0.45	3.53	3.60	6.44	6.58	28.95
		20	0.75	4.62	10.52	10.17	13.55	108.70
	Blocking Probability	5	0	0	0	0	0	0
		12	0	0	0.02	0	0.02	0.30
		20	0	0	0.39	0	0.37	0.58
Ring	Routing cost (k)	5	0.18	0.35	0.26	2.06	1.58	4.15
		12	0.43	1.54	3.74	4.25	3.95	26.35
		20	0.72	5.59	10.68	7.97	10.92	101.29
	Blocking Probability	5	0	0	0	0	0	0
		12	0	0	0.06	0	0.06	0.29
		20	0	0.15	0.43	0.15	0.43	0.58

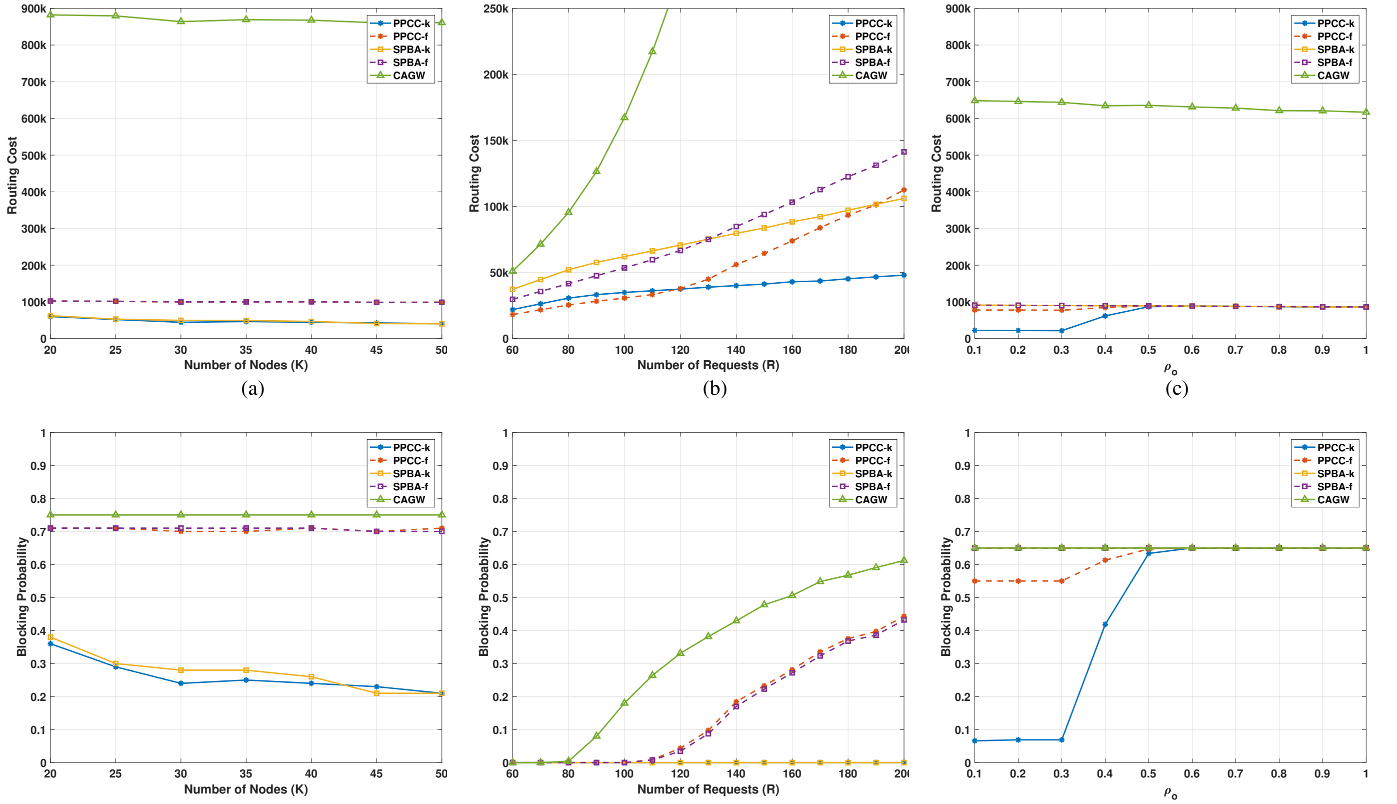


(a) (b) (c)

Fig. 7: Random generated networks with sufficient capacity: (a) Performance of the proposed scheme with different number of nodes in the network. (R=200); (b) Performance of the proposed scheme with increased number of service requests in the network. (K=20); (c) Performance of the proposed scheme for different values of the parameter ρ_0. (K=20, R=200)

sizes. This is expected by the fact that as the network resources are preserved for the most probable routing path, lower routing cost is obtained. A similar observation can be made from figure 7 (b), which shows the performance for different number of requests. With increased number of requests, i.e., more constrained allocations, the performance gains increase from 22% to 23%. It is noteworthy that, the performance metric shows a linear growth trend due to that each request are technically identical i.e., the number of VNFs in the chain is similar for each request. As a result, a higher gain is expected in the case of dense mobile networks where the number of arriving requests in a certain amount of time is large. Finally, in figure 7 (c) we show the performance of the proposed scheme for different mobility use cases. The figure shows the performance gains as a factor of the parameter ρ_o. This parameter is defined as follows $\rho_o = 1 - \sum_{d \in \mathbf{D}} \rho_d$, which means that as ρ_o reaches close to 1 there is no mobility of the end-user, i.e., there is no change on the serving access router. As expected, there are no gains when there is no mobility, but as the mobility increase the gains reach more than 26%. The result suggests that, the proposed scheme is ideal for public transport mobile scenarios where mobile users have almost predictable mobility paths. Last but not least, the two proactive schemes achieve overall exceptional routing performance in compared with AGW.

In figure 8, we examine the impact of network capacity. In the capacitated case, we also assign the capacity size that approximately supports 75% requests. Due to the resource limitation, only a selective set of requests are assisted with the service of proactive unbroken chains. In this setting, we depict the topology size impact on the routing performance of the proposed heuristics under capacitated scenarios in figure8 (a). The curves in the plot indicate that the Capacitated All from Gateway (CAGW) scheme has the highest cost and it is robust against the network topology. Since putting every VNF in the gateway node consumes a large amount of network resources, CAGW receives also the highest blocking chance which cause huge routing penalties. In comparison, the node



(a) (b) (c)

(d) (e) (f)

Fig. 8: Random generated networks with insufficient capacity: (a) Routing performance of the proposed scheme with different number of nodes in the network. (R=200); (b) Routing performance of the proposed scheme with increased number of service requests in the network. (K=20); (c) Routing performance of the proposed scheme for different values of the parameter ρ_0. (K=20, R=20);(d) Blocking performance of the proposed scheme with different number of nodes in the network. (R=200); (e) Blocking performance of the proposed scheme with increased number of service requests in the network. (K=20); (f) Blocking performance of the proposed scheme for different values of the parameter ρ_0. (K=20, R=200)

first mode (the k mode) of both PPCC and SPBA scheme can achieve in average 50% routing gains against the function first mode (the f mode). This comes from the fact that the k mode consumes less capacity compare to the f mode, which can be justified from the blocking curves shown in figure8 (d). It is worth noting that, with limited resource to place the requested VNFs, the blocking probability and routing cost decrease with the increasing topology size. This explains that in capacitated case, the routing performance of PPCC and SPBA is topology sensitive. As a result, if the path from the caching point to the most potential destination has higher blocking probability due to insufficient capacity, PPCC might receive higher routing cost against SPBA.

In figure 8 (b) and (e), we describe the impact of the request size on the routing and blocking performance in capacitated case. We notice that the routing cost exhibits relatively linear behavior when no blocking involved however larger growing slopes show when blocking comes. As can be seen in figure 8 (e), the blocking probability of each scheme goes higher with the increase of the request size. As expected, the blocking probability of CAGW grows much faster against that of other schemes and the k mode performs better than the f mode in terms of blocking. It makes sense because the f mode excessively place the VNFs among the cache-destination path for ensuring the routing performance of a selective set of requests. This trade off can be clearly observed in figure 8 (b), as the f mode holds more redundancy VNFs, it could potentially generate shorter service chains against the k mode, therefore, the f mode of both PPCC and SPBA start at a lower routing cost and end up at a higher position against the k mode. Moreover, it is noteworthy that, the routing cost curves of the two sub modes of the same scheme cross at R=120 when the f mode has blocking.

In figure 8 (c) and (f), we investigate the impact of mobility on the routing and blocking performance in the capacitated scenario. It can be observed from figure 8 (c), the routing cost merges when ρ_0 goes high. However, with respect to mobility, PPCC could reach over 75% gains against SPBA. In comparison with the gain under the uncapacitated scenario, the enlarged gain mainly attributes to the blocking margin.

In figure 9 we show the execution efficiency of heuristics PPCC, SPBA and AGW. The computation complexities of the two sub modes are identical, hence we do not distinguish them in this part. Evidently, the calculation time of the three

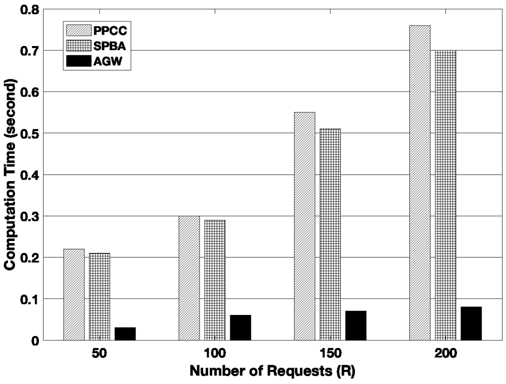


Fig. 9: Average execution efficiency of different schemes against Number of Requests

schemes increases with larger request size. Although AGW is much faster in execution, it sacrifices both routing and blocking performance. The PPCC is in general 5.6% slower than SPBA in VNF allocation however, it achieves significant gains in terms of routing performance. The PPCC and SPBA schemes are slower than AGW, nevertheless their execution time is smaller than the preset batch length. In the simulation, we use a PC with 8GB memory and 2.9GHz Intel Core i5 processor, thus, the execution time should be much lower if the proposed schemes are applied on industrial level machines.

IX. Conclusions

In this paper, the rational of VNF location and chaining for proactive caching has been presented together with some key observations on this problem and the general principle of optimizing cache specific VNF service chains. Based on those observations an optimization framework using integer linear mathematical programming has been detailed that integrates VNF chaining and proactive caching. In addition, since the problem resembles the UFL problem, which is *NP-hard*, some scale-free heuristic algorithms have been presented that can be applied in large network instances amenable for real time implementations. The calculation complexity is analyzed as well as the calculation and memory space scalability of the formulated optimization problem.

Finally, the attainable performance of the proposed proactive caching service chains schemes was investigated. Our numerical results provide evidence that mobility-aware approaches receive significant performance benefits especially during network congestion episodes and high mobility network scenarios. In particular, the proposed PPCC-f mode provides the best performance in terms of reducing routing cost but requires sufficient resources across the VNF service chain path. While another option that balances routing and resource consumption is offered by PPCC-k mode which also achieves a competitive performance. In more detail, our mobility-aware heuristics is in general 60% larger than the theoretical optimal performance however in specific networks i.e., ring topology networks, this approximation gap can be significantly decreased to under 20%.

Acknowledgment

Partially funded by the EC H2020-ICT-2014-2 project 5G NORMA (`www.5gnorma.5g-ppp.eu`)

Appendix

A. The PCC problem is NP-hard

Lemma 1. *Defining the following Linear programming formulations of the uncapacitated facility location (UFL) problem:*

$$\min_{\chi_a,\upsilon_{ab}} \sum_{a\in\mathbf{A}} \Gamma_a\chi_a + \sum_{a\in\mathbf{A}}\sum_{b\in\mathbf{B}} \Upsilon_{ab}\upsilon_{ab} \tag{19}$$

$$S.t. \quad \sum_{a\in\mathbf{A}} \upsilon_{ab} \geq 1, \forall b\in\mathbf{B} \tag{19a}$$

$$\upsilon_{ab} - \chi_a \leq 0, \forall a\in\mathbf{A}, b\in\mathbf{B} \tag{19b}$$

$$\chi_a \in \{0,1\}, \quad \forall a\in\mathbf{A} \tag{19c}$$

$$\upsilon_{ab} \in \{0,1\}, \quad \forall a\in\mathbf{A}, b\in\mathbf{B} \tag{19d}$$

Here, $\mathbf{A}$ *and* $\mathbf{B}$ *are the set of facilities and the set of customers. Accordingly,* Γ_a *and* Υ_{ab} *are the cost for opening facility* a *and the cost for customer* b *to access facility* a*. The two decision variables:* χ *and* υ*.* $\chi_a = 1$ *denotes facility* a *is opened,* 0 *otherwise;* $\upsilon_{ab} = 1$ *denotes customer* b *accesses the item from facility* a*,* 0 *otherwise. Given such a UFL it is* NP-hard *to find its optimum.*

Proof. The Set Covering Problem (SCP) which is one of Karp's 21 NP-complete problems shown to be *NP-complete* [29] can be reduced to the UFL, i.e., SCP <= UFL. The detailed proof is provided in Chapter 3 of [30]. □

Theorem 1. *Given an instance of the PCC problem shown as (4). It is* NP-hard *to find an optimal VNF allocation such that the objective function is minimized.*

Proof. We proof the theorem by showing the UFL is reducible to our PCC. Given an instance of the UFL, we can construct an instance of the PCC such that an optimal solution of the PCC gives an optimal solution to the UFL. First we construct a UFL instance as in (19) and a PCC instance as in (4). For each input variable χ_a of UFL, we construct the input variable x^k_{ri} of PCC by setting r and i as 1 such that x^k_{ri} is basically x_k. Similarly, we construct y^{ksd}_{ri} by forcing r,i,s to be 1 for each υ_{ab} such that y^{ksd}_{ri} can be expressed as y_{kd}. Accordingly, we make $|\mathbf{R}|$, $|\mathbf{I}|$, $|\mathbf{S_r}|$ as 1 and $|\mathbf{K}| = |\mathbf{A}|$, $|\mathbf{D}| = |\mathbf{B}|$. The size of the corresponding cost parameters of the PCC problem keeps in-line with the size of the variables (e.g. $C^k_i = C^k_1 = C^k$ to just mention a few). For each k and d, we let $C^k = \Gamma_a$, $P_{kd} = \Upsilon_{ab}$ and $P_{sk} = P_{km} = 0$. Furthermore, for each d and l, let $\rho_d = 1$ and $V_{ril} = V_l = 1$. Besides, for each k, i and r, we set u_i, U_k, λ_r to be 0 respectively. At last, we make $\Lambda_{sk} = \Lambda_{km} = \Lambda_{kd} = 0$, for all $s\in\mathbf{S_r}$, $k,m\in\mathbf{K}$ and $d\in\mathbf{D}$.

By such conversion, we translate the UFL as a special case of PCC. Thus, if an algorithm can find the optimal solution for our PCC then it can be used to calculate the optimum of the UFL as well. Since all the above conversions can be done in polynomial time, we can conclude that UFL is reducible to PCC. Combining with **Lemma** 1, we can conclude that $SCP <= UFL <= PCC$ and it proofs the theorem. □

References

[1] R. Mijumbi et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, Vol. 18, No. 1, 2016.

[2] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, 2012.

[3] X. Jin, L. Erran Li, L. Vanbever and J. Rexford, "SoftCell: Scalable and flexible cellular core network architecture", *Proc. 9th Int. Conf. Emerging Netw. Exp. Technol.*, pp. 163-174, 2013.

[4] Network Function Virtualisation: ETSI introductory white paper, http://portal.etsi.org/NFV/NFV_White_Paper.pdf, October 2012.

[5] "ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV); Architectural framework," ETSI Ind. Spec. Group (ISG) Netw. Functions Virtualisation (NFV), Sophia-Antipolis Cedex, France, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[6] S. Lee, S. Pack, M-K. Shin, E. Paik, R. Browne, "Resource Management in Service Chaining", Internet Research Task Force (IRTF), Internet-Draft, July 2015.

[7] G. Zheng, A. Tsiopoulos, V. Friderikos, "Dynamic Placement of VNF Chains for Proactive Caching in Mobile Edge Networks" *arXiv preprint arXive*: 1807.10736, 2018.

[8] A. Gember, A. Krishnamurthy, S. St. John, R. Grandl, XY.Gao, A. Anand, T.Benson , V. Sekar, A.Akella, "Stratos: A Network-Aware Orchestration Layer for Virtual Middleboxes in Clouds", Technical Report, 2013.

[9] S. Rajagopalan, D. Williams, H. Jamjoom, A. Warfield, "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes", in *ACM USENIX Symposium on Networked Systems Design and Implementation*, 2013.

[10] F. Riera, et al., "On the complex scheduling formulation of virtual network functions over optical networks", in *16th International Conference on Transparent Optical Networks (ICTON)*, 2014.

[11] T.Han and N.Ansari, "Opportunistic content pushing via WiFi hotspots", in *Proc. 3rd IEEE IC-NIDC*, Sep. 2012, pp. 680-684.

[12] K. Kanai et al., "Proactive Content Caching for Mobile Video Utilizing Transportation Systems and Evaluation Through Field Experiments", in IEEE Journal on Selected Areas in Communications, vol. 34, no. 8, pp. 2102-2114, Aug. 2016.

[13] G. Zheng, V. Friderikos, "Optimal proactive caching management in mobile networks" in *Proc. IEEE ICC*, 2016.

[14] V. A. Siris, X. Vasilakos, and G. C. Polyzos, "Efficient proactive caching for supporting seamless mobility," *arXiv preprint arXiv*:1404.4754, 2014.

[15] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghay, D. Li, G. Wilfong, Y. R. Yang, and C. Guo, "Pace: policy-aware application cloud embedding" in *Proc. IEEE INFOCOM*, 2013.

[16] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, O. C. M. B. Duarte, "Orchestrating virtualized network functions", *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725-739, Dec. 2016.

[17] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. CloudNet*, Oct. 2014, pp. 7-13.

[18] "ETSI GS NFV 001 V1.1.1: Network Functions Virtualisation (NFV); Use Cases," ETSI Ind. Spec. Group (ISG) Netw. Functions Virtualisation (NFV), Sophia-Antipolis Cedex, France, Oct. 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf

[19] Framework for SDN: Scope and Requirements. *Technical Recommendation*. Version 1. 0. ONF, 2015.

[20] B. M. Waxman, "Routing of multipoint connections" *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[21] X. Ge, S. Tu, G. Mao, C. Wang and T. Han, "5G Ultra-Dense Cellular Networks", *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72-79, 2016.

[22] M. Wang, H. Gao, T. Lv, "Energy-Efficient User Association and Power Control in the Heterogeneous Network," *IEEE Access*, Vol. 5, pp. 5059-5068, 2017.

[23] 3GPP standardization, "Scenarios and requirements for small cell enhancements for E-UTRA and E-UTRAN," TR 36.932 v12.1.0, Mar. 2013, http://www.3gpp.org/.

[24] T. Yamamoto and S. Konishi, "Impact of Small Cell Deployments on Mobility Performance in LTE-Advanced Systems," Proc. *IEEE 24th Int'l. Symp. Personal, Indoor and Mobile Radio Commun.*, Sept. 2013, pp. 189-93.

[25] J. Li, J. Chen and K. Xiao, *Comprehensive Carrier Network Planning and Design Handbook*, 1st ed. Bei Jing: The People's Posts and Telecommunication Press, 2015.

[26] V.Sourlas,L.Gkatzikis,P.Flegkas,andL.Tassiulas,"Distributedcache management in information-centric networks", *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, 2013.

[27] Grzech , A. , P. Swiatek , and P. Rygielski, "Dynamic Resources Allocation for Delivery of Personalized Services" In *Software Services for e-World* , edited by W. Cellary and E. Estevez , 17-28. Berlin : Springer, 2010.

[28] G. Zheng, C. Chen, V. Friderikos, M. Dohler, "High Mobility Multi Modal E-Health Services" in *Proc. IEEE ICC*, 2018.

[29] Richard M. Karp (1972). "Reducibility Among Combinatorial Problems" In *Complexity of Computer Computations*. New York: Plenum. pp. 85-103.

[30] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*. John Wiley & Sons, New York, 1990.

Gao Zheng has received MSc degree in Telecommunications and Internet Technology from King's Collage London (2014) where he is currently working toward the Ph.D degree with Centre for Telecommunications Research. His research interests includes Future Internet Technologies, Network Function Virtualization, Information Centric Networks and Mobile Edge Computing. The emphasis is on the network optimization for routing, caching, resource allocation in virtualized wireless networks with application to practical implementations.

Anthony Tsiopoulos has received an MSc from the University of Sussex in Scientific Computation and Mathematics (2014) and an MSc in Computing and Security from King's College London (KCL)(2015) where he then continued his research in Cloud Technology, Telecommunications, Virtualization and Software Implementation in the KCL 5G Telecommunications Laboratory. His research interests include Security and Privacy, Distributed Networks and Software Implementation of Network Programming and Virtualization.

Vasilis Friderikos published 200 research papers in flagship IEEE, Elsevier, Springer journals, international conferences, book chapters and patents. He has been program co-chair of IEEE ICT'16 and co-chair at the IEEE WCNC 2010 conference (acting technical program committee member for IEEE Globecom, IEEE ICC and several other flagship international conferences). He has also been organizing committee member of the Green Wireless Communications and Networks Workshop (GreeNet) during VTC Spring 2011. He has been teaching advanced mobility management protocols for the Future Internet at the Institut Supérieur de l'Electronique et du Numérique (ISEN) in France during autumn 2010. Received two times best paper awards in IEEE ICC 2010 and WWRF conferences respectively. He has been visiting researcher at WinLab in Rutgers University (USA) and recipient of the British Telecom Fellowship Award in 2005. Vasilis is a member of IEEE, member of IET and member of the INFORMS section on Telecommunications. His research interests lie broadly within the closely overlapped areas of wireless networking, mobile computing, and architectural aspects of the Future Internet. The emphasis is on the design and analysis of algorithms for scheduling, routing, admission control, load and power management in virtualized wireless networks with application to both centralized and distributed implementations.