Complete Edge Function Onloading for Effective Backend-driven Cyber Foraging

Flavio Esposito*, Andrej Cvetkovski†, Tooska Dargahi‡, and Jianli Pan§
*Saint Louis University, USA † Mother Teresa University, FYROM,
‡ CNIT - University of Rome Tor Vergata, Italy, §University of Missouri-St. Louis, USA.

***Abstract*—Edge computing, which is a fundamental component of emerging 5G architectures, involves onloading or offloading multiple virtual network functions from mobile devices to an edge network substrate. In this paper, we present a model for the complete edge function onloading problem, which consists of three main phases: (1) Cyber foraging, which involves discovery of resources monitoring the state of edge resources, (2) edge function mapping, which involves matching requests to available resources, and (3) allocation, which involves assigning resources to mappings. Using optimization theory, we show how these three phases are tightly connected, and how the wide spectrum of existing solutions that either solve a particular phase, or jointly solve two of the phases (along with their interactions), are incomplete and may lead to inefficiencies. Moreover, with extensive simulation experiments we demonstrate that joint optimization of all three phases enables the edge network to host a larger set of constrained edge function requests.**

I. Introduction

Edge computing is a paradigm in which much of the processing takes place in a process at the edge of the network, as opposed to in the core of the network as in the Cloud Computing paradigm. This (distributed) approach has been shown to improve user experience by reducing the perceived latency, and is growing in popularity because of the Internet of Things (IoT) and the vast amount of data that sensors generate. It is inefficient to transmit all the data a bundle of sensors creates to the cloud for processing and analysis; doing so requires a great deal of bandwidth and all the back-and-forth communication between the sensors and the cloud can negatively impact performance. In a challenged edge computing scenario, $e.g.$, in case of natural disaster, or for medical applications, multiple processes need to establish and maintain a set of virtual flows to guarantee a set of Service Level Objectives, $i.e.$, some acceptable levels of network performance, to accomplish a phase or, more generally, to provide a service. Several edge cloud infrastructures arose in the research community, see $e.g.$ [14], [20], as well as from industry initiatives, such as Microsoft's micro DCs [5] and ETSI Mobile Edge Computing [12]. Moreover, several 5G initiatives, $e.g.$, [1], [2] are basing their architecture on edge clouds to meet the EU 5G Vision requirements [3].

In contrast with the vast majority of these proposals, in this paper we argue that back-end driven onloading is a wiser alternative than client-driven offloading. Client-based offloading has many difficult challenges, mostly due to the diversity of devices: from the different operating systems, to the numerous apps running on them, to accurate code profiling, and gauging optimal offload conditions. Often a continuous monitoring of network conditions is required, and several recent offloading approaches ignore this crucial phase [5], [12]. Moreover, resource constrained of geographically distributed end user devices are often ignored in finding an optimal offloading strategy. The edge cloud instead is inherently designed to handle different types of devices, and practically has unlimited energy and computational resources. Furthermore, it has ability to accurately characterize access to users' location and time constraints [15]. One of the goals of edge computing is to release devices from the burden of (expensive) computations, so it is natural to think that such devices should not waste resources to decide where to offload. Most importantly, the edge cloud may benefit from an holistic view of where to run the edge functions, a view that has been surprisingly neglected in previous solutions. In this paper we focus on such view.

Our Contributions: In particular, in this paper we introduce, model and evaluate with a simulation campaign the impact of *the complete edge function onloading problem*, showing its significant efficiency gains. The problem, solved by processes in the edge cloud back-end, comprises three subproblems (or phases): cyber-forage, edge function mapping, and edge function binding or allocation. *Cyber-forage* is the process of discovering resources capable of hosting the phase or sequence of phases ($i.e.$ mechanisms), by monitoring the state of the substrate resources using sensors and other measurement processes. The monitored states include energy consumption, processor loads, memory usage, network statistics, etc.[1] *Edge Function Mapping* is the phase in charge of matching edge functions' requests with the available resources. Due to the possible combination of node and link constraints, this is the most complex step in the mapping problem. This problem is in fact NP-hard as it can be reduced from the multiway separator problem [7]. The constraints include intra-node ($e.g.$, desired edge location, processor speed, storage capacity, type of network connectivity), as well as inter-node constraints ($e.g.$, network topology). The last subproblem – *edge function binding* – involves assigning the resources that match the edge function queries to the appropriate virtual function, considering additional constraints, $e.g.$, location or infrastructure physical limits. This last phase also ensures that users will not exceed physical limits or their authorized resource usage. For

[1]The term cyber foraging is often used as synonym of offloading. We dissect this notion by clarifying its separation from the other two phases, and removing the assumption that such operation is managed by mobile devices.

978-1-5386-3839-2/17/$31.00 ©2017 IEEE

example, the system may decide not to allocate a service that has not yet been authorized, even though the virtual network could be physically mapped.

To our knowledge, this is the first study that captures the interactions among the onloading phases (see Figure 1) Previous onloading solutions, $e.g.$, [15], only formulate or partially address the problem, and always neglect at least one of the interdependencies. We argue that providers should not neglect such cooperations, as an efficient utilization of the edge network is one of the key factors not only for profit maximization, but also for avoidance of congestion on edge links and nodes, and therefore for minimizing virtualization artifacts experienced by edge computing applications.

Paper outline: The remainder of this paper is organized as follows: we discuss related solutions in Section II. We then introduce the three edge infrastructure services encompassing our problem in Section III, and we describe how our model unifies such phases holistically in Section IV. Then, we turn our attention to the development of efficient solution methods for our problem, exploiting the rich problem structure via the dual-decomposition in Section V. In Section VI we show with a simulation campaign how our approach improves the performance of network virtualization applications that require agility, and we conclude our work in Section VII.

II. Related Work

The problem of optimizing the transmission of a workload from a mobile device to the edge cloud is of central importance for an efficient coordination. We classify existing optimization solutions for edge offloading using five dimensions: i) optimize the offloading destination [8], [10], [19], ii) the load to balance [13], [24], iii) the user device mobility [22], [25], iv) the application partitioning [16], and v) the partition granularity [20]. Such classification covers the whole offloading process in a sequential order from mobile devices to the edge cloud. We only cite here representative solutions that help us define our contributions, but a more comprehensive reference list can be found in these surveys [4], [17], [23].

MAUI [10] and CloneCloud [8] discuss different optimization strategies for the offloading on a single-sever, while ThinkAir [19] solves the resource allocation problem by parallelizing the offloading on multiple virtual machines. We also consider an optimal offloading destination, but we consider also the cyber foraging and the final allocation phases, not merely the edge function mapping phase. Shifting the focus to problems that seek load balancing under limited resources and processing abilities on the edge cloud, we consider the work of Xia et al. [24], that proposed an online request admission algorithm to maximize the system throughput. In [13], instead, the authors proposed an algorithm for allocating VMs in a cloud which consists of geographically diversified small data centers close to users. The algorithm takes users' specified constraints into consideration including server geographic locations, with resource cost and communication latency constraints. Our model subsumes those approaches since it captures all these constraints, but does not ignore the interactions among the three subproblems. Moreover, using dual decomposition, as shown in Section V, our network utility maximization problem may also be easily distributed.

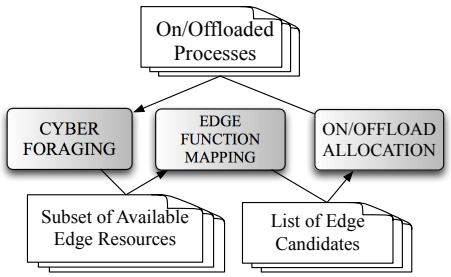


Fig. 1. Complete Edge Function Onloading: interactions and data exchanges.

The third modeling classification dimension that we consider is mobility. To improve the performance while satisfying the SLA of mobile applications, other authors [22], [25] proposed different mobile-friendly architectures; the two-tiered mobile cloud architecture in [25] for example, includes computation capabilities both at the edge and the central cloud. Both solutions aim at offloading location-aware phases of mobile applications to ensure, $e.g.$, fairness of energy consumption. Our model captures also their constraints, but defines a more flexible architecture in which each subproblem is solvable jointly in different tiers, or individually in a centralized or distributed fashion. Moreover, the existing approaches focus only on the location-aware mapping phase ignoring the cyber foraging (discovery) and the final binding phases.

III. Edge Function Onloading Subproblems

In this section we present a model for each of the three subproblems (phases) of the complete edge function onloading problem. To do so, we first define a network as follows:

Definition 1 (Network): A Network is given by an undirected graph $G = (N, L, C)$ where N is a set of nodes, L is a set of links, and each node or link $e \in N \cup L$ is associated with a set of constraints $C(e) = \{C_1(e), \ldots, C_m(e)\}$ where $m \geq 0$ is the number of constraints on element e. We denote a physical network as $G^P = (N^P, L^P, C^P)$, and an edge function to onload as a graph $G^V = (N^V, L^V, C^V)$.

Nodes in G represent standalone resources, whereas links represent relationships between resources. Examples of standalone resources (nodes in G) include processors and storage. Examples of relationships between resources include communication links, spatial or temporal relationships. Links may be directed or undirected. For instance, a virtual network function can be represented as a pipeline of processing units in which each pair of adjacent node is connected using a directed edge; a cluster of processing units would be instead represented using a connected subgraph, in which each pair of vertices are connected with an undirected edge. The labeled graph G_V models the set of edge cloud resources and underlying relationships that are necessary to support a specific user application or phase. Labels in G may decorate either nodes or edges. In a hosting graph, labels specify supply attributes such as unit capacities and unit prices of processing or communication links. In a requested graph of resources to

onload, labels specify demand attributes such as the minimum CPU utilization, storage or other hardware or software package needed by the standalone process to onload, or the minimum bandwidth tolerable by communicating processes. Notice that resources may have multidimensional capacities. In particular, let j denote an application requesting a phase to onload on the edge, $e.g.$, a computation. The request is defined as a triple $\{\gamma_j, \psi_j, C_j(e)\}$ composed of γ_j virtual nodes, ψ_j virtual links and a vector of constraints

$$C_j(e) =< C_j(e_1), \ldots, C_j(e_c) >$$

where e is a vector of $c = \gamma_j + \psi_j$ elements – nodes and links – of the edge function. γ_j and ψ_j represent the cardinality of N^V and L^V, respectively, while $C_j(e)$ is the vector of constraints summarized with the C^V notation.

A. Modeling Cyber Foraging of Onloader Candidates

To model the cyber foraging (discovery of resources able to execute tasks), we introduce two vectors of binary variables, n_i^P, $i = 1, \ldots, |N^P|$ and p_k, $k = 1, \ldots, |\mathcal{P}|$, where $\mathcal{P}$ denotes the set of all paths in G^P, that are equal to 1 if the i^{th} physical node and the k^{th} physical path, respectively, are available, and zero otherwise. An element is available if a cyber foraging operation is able to find it, given a discovery protocol policy, $e.g.$, find all (or the first k shortest) physical paths no later than a given deadline, or find as many available physical nodes as possible within a given number of hops. If the system does not return at least γ physical nodes and at least ψ available physical paths among all the possible $|N^P|$ nodes and $|\mathcal{P}|$ paths of the physical network G^P, then the user's request should be immediately discarded or stored in a queue for later allocation attempts. Among all possible resources, the system may choose to return a set that maximizes a given notion of utility. Those utilities may have the role of selecting the resources that are closer – with respect to some notion of distance – to the given set of constraints $C(e)$. We denote as $u_i \in \Re$ and $\omega_k \in \Re$ the utility of physical nodes and paths, respectively. Regardless of the type of technology (wireless or wired), the search algorithm (from flooding to unicast or anycast), the architecture of the cyber foraging system (unstructured, structured, hybrid, centralized or distributed), and the type of the resource to be discovered, we model the maximum number of nodes and paths that the protocol is able to discover with $\Gamma \in [0, |N^P|]$ and $\Psi \in [0, |\mathcal{P}|]$, respectively. Those variables represent the (limited) time interval after which the protocol has to return an answer to the system, or the restricted scope of the provider running this infrastructure service. Therefore, the cyber foraging phase can be modeled with the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & f(n_i^P, p_k) = \sum_{i \in N^P} u_i n_i^P + \sum_{k \in \mathcal{P}} \omega_k p_k \\ \text{subject to} \quad & \gamma \leq \sum_{i \in N^P} n_i^P \leq \Gamma \\ & \psi \leq \sum_{k \in \mathcal{P}} p_k \leq \Psi \\ & n_i^P, p_k \in \{0,1\} \ \ \forall i, \ \ \forall k \end{aligned} \tag{1}$$

After the cyber foraging phase is completed, the vectors of available physical resources (n^P, p) are passed to the virtual network mapper (bottom left block in Figure 1). Although the two discovery operations on nodes and paths are not independent, we assume that the path foraging, for example a k-shortest path algorithm [11], is run after the node discovery process is completed, so a path cannot be found if all the physical nodes comprising it have not been yet discovered. Moreover, we denote as (n_{ij}^P, p_{kj}) the resource status under a query from edge function j, where $j = 1, \ldots, |J|$, and J denotes the set of edge functions to be onloaded/offloaded. Note how problem (1) is an integer program. In general, the running time of those problems is in a worst-case exponential, and they can be solved with exact methods, $e.g.$, branch and bound or branch and cut (see, $e.g.$, [6]). However, since problem (1) is *separable*, its worst-case complexity is polynomial:

Proposition III.1. *The unconstrained cyber foraging problem (Problem 1) can be solved in polynomial time.*

Proof. First we note how problem (1) is separable, $i.e.$, if we define $f_1(n_i^P) = \sum_{i \in N^P} u_i n_i^P$ and $f_2(p_k) = \sum_{k \in \mathcal{P}} \omega_k p_k$, then the solution of problem (1) is $f^* = f_1^* + f_2^*$, where

$$f_1^* = \sup \left\{ f_1 | \gamma \leq \sum_{i \in N^P} n_i^P \leq \Gamma, \ \ n_i^P \in \{0,1\} \ \forall i \right\}$$

and

$$f_2^* = \sup \left\{ f_2 | \psi \leq \sum_{k \in \mathcal{P}} p_k \leq \Psi, \ \ p_k \in \{0,1\} \ \forall k \right\}.$$

Therefore, an algorithm ∇ to solve (1) runs the following steps: (i) run `quicksort` [21] independently on the vectors $\mathbf{u}$ and ω, (ii) set to one the first Γ n_i^P's and the first Ψ p_k's, (iii) compute f^*. Since the worst-case running time of `quicksort` is $O(n^2)$, step (i) has worst-case performance of $O(|N^P|^2 + |\mathcal{P}|^2)$, and so the running time of ∇ is polynomial. □

Note that even though the complexity is polynomial, it may still be impractical to run an $O(n^2)$ in the number of paths.

B. Modeling Edge Function Mapping

The edge function mapping phase takes as input all the available edge resources (subset of all the existing resources passed by the cyber foraging phase) $\mathcal{P}' \subseteq \mathcal{P}$ and $N^{P\prime} \subseteq N^P$, maps edge functions to edge nodes, links connecting edge nodes to physical paths, and returns a list of candidates – virtual nodes and virtual links – to the resource binder. We model the edge function mapping phase as follows:

Definition 2 (Edge Function Mapping): Given an edge function $G^V = (N^V, L^V, C^V)$ and a physical edge network $G^P = (N^P, L^P, C^P)$, a virtual network mapping is a mapping of G^V to a subset of G^P, such that each edge function process node is mapped onto exactly one edge cloud

node, and each virtual link is mapped onto at least one path p on the edge network. Formally, the mapping is a function

$$M : G^V \rightarrow (N^P, \mathcal{P}) \tag{2}$$

where M is called a *valid mapping* if all constraints of G^V are satisfied, and for each $l^v = (s^V, t^V) \in L^V$, $\exists$ a physical path $p : (s^P, t^P) \in \mathcal{P}$ with source s^P and destination t^P with $M(s^V) = s^P$ and $M(t^V) = t^P$ [2].

To model this phase for an onloading request j, $\forall j \in J$, we define two vectors of binary variables n_{ij}^V $\forall i \in N^{P\prime}$, and l_{kj} $\forall k \in \mathcal{P}'$. The variables $n_{ij}^V = 1$ if a virtual instance of node i is mappable – there exists a physical node that satisfies all constraints of user j request – and zero otherwise. While, $l_{kj} = 1$ if a virtual instance of the physical path k asked by edge function j is mappable – there exists at least a physical path p such that all the links in p satisfy the constraints of user j request – and zero otherwise. The edge function mapping can be modeled by the following network utility maximization problem:

$$\text{maximize} \quad g(n_{ij}^V, l_{kj}) = \sum_{j \in J} \Big(\sum_{i \in N^{P\prime}} \Theta_{ij} n_{ij}^V + \sum_{k \in \mathcal{P}'} \Phi_{kj} l_{kj} \Big) \tag{3}$$

subject to:

$$\sum_{i \in N^{P\prime}} n_{ij}^V = \gamma_j \;\; \forall j \in J \tag{4}$$

$$\sum_{k \in \mathcal{P}'} l_{kj} = \psi_j \;\; \forall j \in J \tag{5}$$

$$n_{ij}^V = n_{ij}^P \;\; \forall i \in N^{P\prime} \;\; \forall j \in J \tag{6}$$

$$l_{kj} \leq p_{kj} \;\; \forall k \in \mathcal{P}' \;\; \forall j \in J \tag{7}$$

$$n_{ij}^P, n_{ij}^V, p_{kj}, l_{kj} \in \{0, 1\} \;\; \forall i \;\; \forall j \;\; \forall k, \tag{8}$$

where Θ_{ij} is the system's gain when function j gets onloaded or offloaded to edge node i, and Φ_{kj} is the system's gain if the user j uses path k. The first two sets of constraints (4 and 5) enforce that all the virtual resources requested by each user are mapped. Constraint (6) ensures that the one-to-one mapping between virtual and physical nodes is not violated, and constraint (7) ensures that at least one hosting path is going to be assigned to each virtual link. This inequality would be strict if, for example, we force path splitting in the link assignment. Moreover, constraints (8) denote the non negativity domain of the the nodes and links variables $\mathbf{n^P}, \mathbf{n^V}, \mathbf{p}$ and $\mathbf{l}$.

C. Modeling Onload/Offload Allocation

As soon as the edge function mapping candidates have been identified, a *Set Packing Problem (SPP)* needs to be run, considering both edge function priorities and capacity constraints on the physical resources. When demand exceeds supply and not all the needs can be met, virtualization systems goals can no longer be related to maximizing utilization, but different policies to guide resource allocation decisions have to be designed. A natural policy is to seek efficiency, namely, to allocate resources to the set that bring to the system the highest utility. To such an extent, the research community has frequently proposed market-based mechanisms to allocate resources among competing interests while maximizing the overall user utility. A subclass of solutions dealing with this type of allocation is represented by auction-based systems.

Auction-Based Allocation. An auction is the process of buying and selling goods or services by offering them up for bid, taking bids, and then selling them to the highest bidder. The last phase of our problem builds on the *Combinatorial Auction Problem* (CAP). The CAP considers a set O of resources (or objects) and a collection of users J each asking for a subset of O. J is a subset of the *powerset* of O, $P(O)$ – the set of all the possible subsets of O, including the empty set and O itself. Each user has associated a non-negative utility w_j. The edge infrastructure provider that runs the binding phase seeks the largest utility collection of subsets, namely, it seeks to allocate the set of edge functions that maximize its aggregate utility. Let $y_j = 1$ if the j^{th} set in J with utility w_j is selected, and zero otherwise. Moreover, let $a_{ij} = 1$ if the j^{th} set in J contains the resource of type $i \in O$ and zero otherwise. If we also assume b_i copies of resource i, we have:

$$\begin{aligned} &\text{maximize} \quad \sum_{j \in J} w_j y_j \\ &\text{subject to} \quad \sum_{j \in J} a_{ij} y_j \leq b_i \;\; \forall i \in O \\ &\qquad\qquad\;\; y_j \in \{0, 1\} \;\; \forall j \in J \end{aligned} \tag{9}$$

which is a multi-resource SPP. As shown in [9], in fact, a CAP is NP-Hard and equivalent to a SPP. Intuitively, we need to check all the possible combinations of bids w_j to find out which is the set of bidders that maximizes the provider's profit.

Leveraging on Problem 9, we model the onload/offload allocation phase with the following mixed integer $|L^V|$-commodity flow problem:
Objective:

$$\text{maximize} \quad h(y_j) = \sum_{j \in J} w_j y_j \tag{10}$$

subject to:

$$\sum_{j \in J} n_{ij}^V \leq y_j C_i^n \;\; \forall i \in N^{P\prime} \tag{11}$$

$$\sum_{j \in J} \left(f_{i,w}^{kj} + f_{w,i}^{kj} \right) \leq y_j b(i, w) \; \forall i, w \in N^{P\prime}, \; i \neq w, \; \forall k \tag{12}$$

$$\sum_{w \in N^{P\prime}} f_{i,w}^k - \sum_{w \in N^{P\prime}} f_{w,i}^k = 0 \;\; \forall k \in \mathcal{P}', \;\; \forall i \in N^{P\prime} \setminus \{s^k, t^k\} \tag{13}$$

$$\sum_{w \in N^{P\prime}} f_{s^k,w}^k - \sum_{w \in N^{P\prime}} f_{w,s_k}^k = b(s^k, w) \;\; \forall k \in \mathcal{P}' \tag{14}$$

[2]Note how the mapping function M may accept any edge function graph, including a graph whose $|N^V| = 1$ and $|L^V| = 0$; assignments as $M(s^V) = s^P$ are well defined. Note also that even if a path may have intermediate nodes, we only need its starting and ending node to define the mapping.

$$\sum_{w \in N^{P\prime}} f^k_{t_k,w} - \sum_{w \in N^{P\prime}} f^k_{w,t_k} = -b(t_k, w) \;\; \forall k \in \mathcal{P}' \quad (15)$$

$$f^k_{i,w} \geq 0 \;\; \forall k, \forall \;\; i, w \in N^P \quad (16)$$

$$n^V_{ij}, y_j \in \{0, 1\} \;\; \forall i \in N^{P\prime} \;\; \forall j \in J \quad (17)$$

We model an edge function link to be a flow on a physical network. Therefore, we consider the allocator as a *flow allocator*, and we include the flow conservation conditions, that is, we enforce the net flow of all the physical nodes to be zero (constraint 13), except when a node is the source s^k (14) or the destination t^k (15) of the flow (or virtual link or physical path) k. Note how the flow constraints model the connection between a link and its end-nodes. Variable $b(e^{kj})$ represents the bandwidth necessary to allocate the virtual link e^k asked by user j. Moreover, constraints (16) denote the non-negativity domain of the flows $\mathbf{f}$, where C^n_i and C^l_k are the numbers of virtual nodes and links, respectively, that can be simultaneously hosted on the physical node i and physical path k, respectively, and y_j is a binary variable equal to 1 if user j has been allocated and zero otherwise. A weight w_j is assigned to each user j, and it depends on the allocation policy used. Note how, since the set packing problem is NP-Hard, problem (10) is also NP-Hard.

IV. Complete Edge Function Onloading

To clarify how the three phases of the complete edge function onloading problem interact, and how they may influence the efficiency of edge computing applications, we formulate a centralized optimization problem that treats the three phases as inseparable. We consider the objective function of the complete edge function onloading problem to be the sum of the objectives of its three standalone phases, under the union of their constraints, with the addition of a few coupling constraints:

$$\begin{array}{ll} \text{maximize} & \alpha \cdot f(n^P_{ij}, p_{kj}) + \beta \cdot g(n^V_{ij}, l_{kj}) + \delta \cdot h(y_j) \\ \text{subject to} & \text{constraints of problems (1), (3) and (10)} \\ & n^V_{ij} = n^P_{ij} \;\; \forall i \;\; \forall j \\ & l_{kj} \leq p_{kj} \;\; \forall k \;\; \forall j \\ & y_j \leq n^V_{ij} \;\; \forall i \in N^{P\prime} \;\; \forall j \in J \\ & y_j \leq l_{kj} \;\; \forall k \in \mathcal{P} \;\; \forall j \in J. \end{array} \quad (18)$$

The first set of constraints is identical to those described for problems (1), (3) and (10), while the last four constraints are coupling (or complicating) constraints, as they bind the three phases of the edge function onloading problem together. If those constraints were absent, the problem could be separated and each phase could be solved independently. Such constraints guarantee that an edge function is not mapped unless enough resources have been found by the cyber foraging phase, and that the mapped edge function components are all bindable. Finally, α, β and δ are weight factors whose values depend on the application. They model the general nature of the problem and they can be used to subsume different (existing or novel) edge function onloading or offloading strategies, depending on whether or not they capture each single phase. To our knowledge, in existing solutions, all the above constraints have never been simultaneously considered. From an optimization theory point of view, constraint addition or omissions in general may result in sub-optimal or infeasible solutions. For example, the cyber foraging constraints impact the other phases of the onloading process, since an edge resource not found certainly cannot be mapped or allocated. Moreover, we risk or running the edge function mapping phase on resources that can never be reserved because they would exceed the physical capacity constraints. Such inefficiencies are quantified, in our simulation study.

V. Decomposing the Complete Edge Function Onloading Problem

We now turn our attention to the development of efficient solution methods for Problem (18). Our approach is based on exploiting the problem structure via the dual-decomposition methods (see, $e.g.$, [6] Ch. 6). Due to the rich structure of this problem, there are many ways to formulate the dual problem, depending on which constraints are introduced with the Lagrange multipliers. To make the problem separable, we consider only dual problems created by introducing Lagrange multipliers for the coupling constraints. For example, if we relax constraints $y_j \leq n^V_{ij}$ and $y_j \leq l_{kj}$, then the problem becomes separable into two subproblems EFM and ALLOC. So we solve the primal problem:

$$\begin{array}{ll} \text{maximize} & \beta \cdot g(n^V_{ij}, l_{kj}) + \delta \cdot h(y_j) \\ \text{subject to} & y_j \leq n^V_{ij} \;\; \forall i \;\; \forall j \\ & y_j \leq l_{kj} \;\; \forall k \;\; \forall j \\ & \text{decoupled constraints in (18)} \end{array} \quad (19)$$

by computing the dual function $V(\lambda, \mu)$, $i.e.$, the objective function of the dual problem:

$$\begin{array}{rl} V = \sup_{n^V_{ij}, l_{kj}, y_j} & L(n^V_{ij}, l_{kj}, y_j, \lambda_{ij}, \mu_{kj}) \\ \text{subject to} & \lambda_{ij} \geq 0 \;\; \forall i \;\; \forall j \\ & \mu_{kj} \geq 0 \;\; \forall j \;\; \forall k \\ & \text{decoupled constraints in (18)} \end{array} \quad (20)$$

where the partial Lagrangian function is given by:

$$L = \beta g(n^V_{ij}, l_{kj}) + \delta h(y_j) \quad (21)$$

$$- \sum_{i \in N^{P\prime}} \sum_{j \in J} \lambda_{ij}(y_j - n^V_{ij}) - \sum_{k \in \mathcal{P}'} \sum_{j \in J} \mu_{kj}(y_j - l_{kj}) = \quad (22)$$

$$\sum_{j \in J} \sum_{i \in N^{P\prime}} \beta \Theta_{ij} n^V_{ij} + \sum_{j \in J} \sum_{k \in \mathcal{P}'} \beta \Phi_{kj} l_{kj} + \quad (23)$$

$$\sum_{i \in N^{P\prime}} \sum_{j \in J} \lambda_{ij} n^V_{ij} + \sum_{k \in \mathcal{P}'} \sum_{j \in J} \mu_{kj} l_{kj} + \quad (24)$$

$$\sum_{j \in J} w_j y_j - \sum_{i \in N^{P\prime}} \sum_{j \in J} \lambda_{ij} y_j - \sum_{k \in \mathcal{P}'} \sum_{j \in J} \mu_{kj} y_j = \quad (25)$$

$$\left[\sum_{j\in J}\sum_{i\in N^{P\prime}}(\beta\Theta_{ij}+\lambda_{ij})n_{ij}^{V}+\sum_{j\in J}\sum_{k\in\mathcal{P}'}(\beta\Phi_{kj}+\mu_{kj})l_{kj}\right]+ \quad (26)$$

$$\left[\sum_{j\in J}(w_j-\sum_{i\in N^{P\prime}}\lambda_{ij}-\sum_{k\in\mathcal{P}'}\mu_{kj})y_j\right] \quad (27)$$

Since the dual function is always convex [6], this is a convex optimization problem. Note that the dual function can be evaluated separately in the mapping and the allocation variables:

$$V(\lambda_{ij},\mu_{kj}) = V_{EFM}(\lambda_{ij},\mu_{kj}) + V_{ALLOC}(\lambda_{ij},\mu_{kj}) \quad (28)$$

where:

$$\begin{aligned} V_{EFM} = \sup_{\mathbf{l},\mathbf{n^V}\geq 0} \quad & \text{Equation (26)} \\ \text{subject to} \quad & \lambda_{ij}\geq 0 \ \ \forall i \ \ \forall j \\ & \mu_{kj}\geq 0 \ \ \forall j \ \ \forall k \\ & \text{EFM decoupled constraints in (18)} \end{aligned} \quad (29)$$

and

$$\begin{aligned} V_{ALLOC} = \sup_{\mathbf{y}\geq 0} \quad & \sum_{j\in J}(w_j-\sum_{i\in N^{P\prime}}\lambda_{ij}-\sum_{k\in\mathcal{P}'}\mu_{kj})y_j \\ \text{subject to} \quad & \lambda_{ij}\geq 0 \ \ \forall i \ \ \forall j \\ & \mu_{kj}\geq 0 \ \ \forall j \ \ \forall k \\ & \text{ALLOC decoupled constraints in (18)} \end{aligned} \quad (30)$$

As we are solving the dual problem instead of the original primal, we know from duality theory that this problem will only give us appropriate results unless strong duality holds. This is the case because the original relaxed problem is convex and there always exist a strictly feasible solutions ($i.e.$, the Slater's condition is verified [6]). The Slater's condition in our case means that there exists a feasible solution such that $y_j < n_{i,j}^V$ and $y_j < l_{k,j}$ that is, it is possible, for example, to have no users ($\mathbf{y}=\mathbf{0}$) allocated even if some edge function node could be mapped. The dual decomposition results in each user solving, for the given Lagrangian multipliers,

$$y_j^*(\lambda,\mu) = \arg\max_{y_j\geq 0}[(w_j-\sum_{i\in N^{P\prime}}\lambda_{ij}-\sum_{k\in\mathcal{P}'}\mu_{kj})y_j] \quad (31)$$

which is unique, due to the strict concavity of w_j.

VI. Performance Evaluation

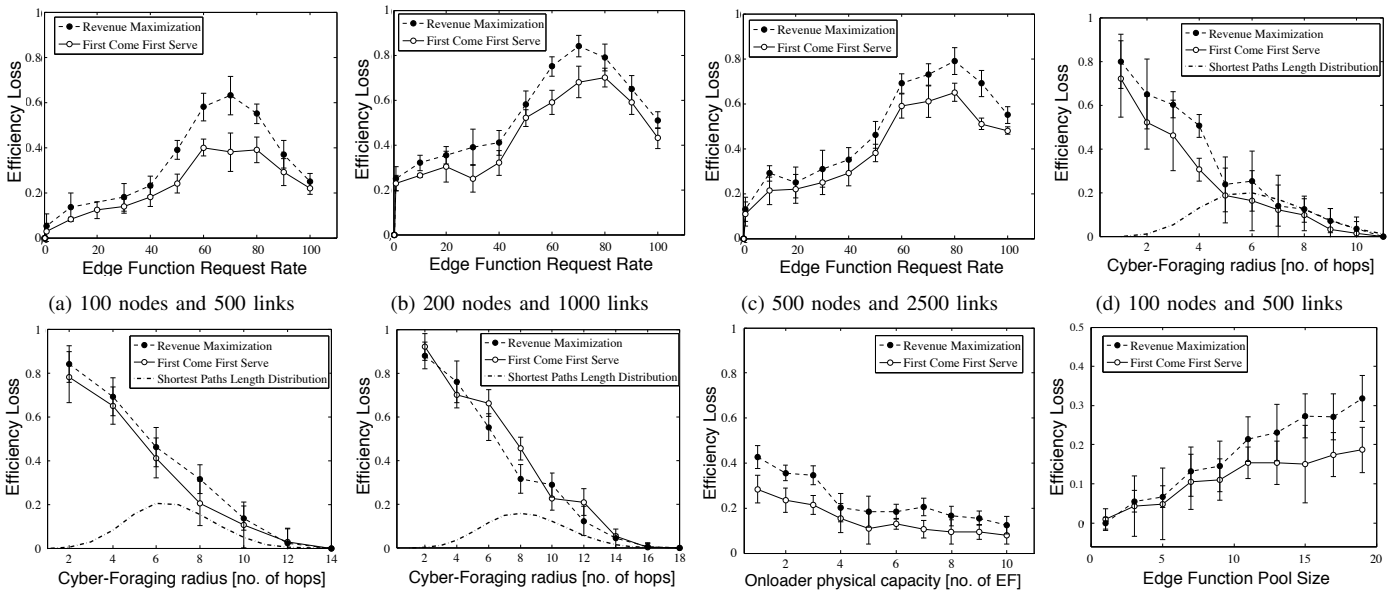
Evaluation Environment. To validate our model, we implemented an event driven simulator assessing the impact of the cooperation gain among the edge function onloading phases. Our simulator captures the three phases as shown in Figure 1. To generate our hosting physical network, we use the BRITE topology generator [18] following a flat Waxman topology generation model. BRITE has been commonly used in research that requires practical network topology generation. Unless otherwise specified, the substrate edge network has 100 nodes and 500 links, a scale that corresponds to a medium-sized Internet Service Provider (ISP). Each end-user process to be onloaded has a CPU requirement, chosen uniformly at random between 1 and 100 units. The CPU resources at the edge nodes follow a uniform distribution between 50 to 200 units, and the link bandwidths follow a uniform distribution from 1 to 10 units.

Evaluation Metrics. The common goal of the proposed solution is the maximization of a notion of onloader utility, which can be, in its simplest case, the number of concurrent allocated edge functions. The aim of our evaluation is to show how each singleton phase of the complete edge function onloading problem alone is insufficient to guarantee an efficient cyber foraging, since each phase influences the decision space of the others. We test our idea implementing two allocation policies: *First Come First Serve (FCFS)* and *Revenue Maximization*. In FCFS, each request is onloaded without any notion or vision of future requests. We picked this algorithm as a baseline being the simplest packing heuristic. In the Revenue Maximization heuristic instead, the edge function whose utility is maximized is picked first, given a pool of simultaneously considered functions to onload. After the node mapping phase, we try to map each virtual link to the first or second shortest path that satisfies the capacity constraints, or else we reject the requested edge function. Results obtained increasing k in a k-shortest path virtual link allocation are similar, and we omit them for lack of space. We now summarize our evaluation results with a few take-home messages.

Evaluation Results. *(1) Simultaneously optimizing the three phases of the complete edge function onloading problem increases global resource utilization.* In Figure 2(a-c) we show the impact of the edge function request rate over the efficiency loss when the cyber foraging phase is ignored. The x-axis shows the edge function request rate. The y-axis shows the fraction of allocated edge function (or efficiency loss), when the feedback-loop depicted in Figure 1 is open, that is, when the virtual edge function mapping and binding (allocation) algorithm is unaware of all the available resources $\{n_{ij}^P, p_{kj}\}$ provided by the cyber foraging service. We then compare the two edge function mapping policies (revenue maximization and First Come First Serve) on two different sets of nodes and edges; first, assuming full knowledge of the physical edge resources, and hence using the sets $(N^P,\mathcal{P})$ – the set of all the possible physical nodes and physical paths in the network – and then, using (N',P'), a subset of discovered physical nodes and paths. A path p is discovered if each node in $p\in N^{P\prime}$. A physical node $n_i^P\in N^{P\prime}$ if it is $\bar{h}$ hops away from a random node of the physical topology, when a discovery-protocol event is triggered. The value of $\bar{h}$ will depend on the application. We fixed $\bar{h}=\bar{p}+1$ where by $\bar{p}$ we denote the average length of a shortest path. With these settings, given our physical network topologies, about 70% of the paths are available after a foraging operation. The efficiency loss is then computed as the ratio between the number of *Mappable Edge Function* (MEF) computed on the set tuple $(N^{P\prime},\mathcal{P}')$, $MEF_{(N^P,\mathcal{P})}$ minus $MEF_{(N^{P\prime},\mathcal{P}')}$, normalized by $MEF_{(N^P,\mathcal{P})}$, that is:

$$\text{Efficiency Loss} := \frac{MEF_{(N^P,\mathcal{P})}-MEF_{(N^{P\prime},\mathcal{P}')}}{MEF_{(N^P,\mathcal{P})}}. \quad (32)$$

We can see how, initially, as the onloading request rate



(a) 100 nodes and 500 links (b) 200 nodes and 1000 links (c) 500 nodes and 2500 links (d) 100 nodes and 500 links

(e) 200 nodes and 1000 links (f) 500 nodes and 2500 links (g) 500 nodes and 2500 links (h) 500 nodes and 2500 links

Fig. 2. (a-c) Impact of the edge function offload request rate on the efficiency loss for different algorithms, and increasing number of physical network size. (d-f) Impact of the discovery radius on efficiency loss for different virtual network mapping algorithms and increasing number of topology size. (g) Impact of the physical node capacity on the Edge Function Mapping – Allocation cooperation. (h) Impact of the pool size on the efficiency loss.

increases, the impact that the cyber foraging phase has on the mapping-allocation algorithm increases too. This translates to greater resource occupancy, and hence a larger global utilization (Figure 2d-f). It is interesting to note however, that, *(2) the efficiency loss exhibits a maximum for high request rates*. This is because, when an edge function phase is completed, the cyber foraging service is not immediately notified of this resource availability.

(3) *Efficient path management solutions are more important than efficient node mapping:* Note how, for smaller physical graphs, the number of available resources decreases and so does the efficiency loss. That seems obvious as fewer resources are available. However, it is interesting to note, for medium and larger physical networks, the efficiency loss does not significantly change (compare Figures (2b) and (2c)). To explain this behavior, we had to observe the number of available distinct physical paths for each physical network size, whose length distribution is shown in Figure 2d-f (dashed lines); given our simulation settings, there are not enough paths in the largest (BRITE generated) physical network to allocate more edge onload requests, even if more edge physical nodes are available. This result suggests that, when devising heuristics or approximation algorithms for the NP-Hard edge function onloading problem, we should keep in mind that the physical paths are the real scarce resource in the mapping process.

Resource Discovery Interaction. The three Figures 2(a-c), highlight the benefit of a cooperation between the edge function mapping and the cyber foraging phases on different physical substrates, with equivalent physical edges over physical nodes ratio: in 2(a) we used 100 nodes and 500 links, in 2(b) 200 nodes and 1000 links, while in 2(c) 500 nodes and 2500 links. Note from the drawn shortest path length distribution that the average length increases from around 6 in 2(d) to around 8 in 2(f). On the x-axis we have the radius –number of hops that the resource discovery has reached from a fixed random physical node. For the network that produced Figure 2(d), $e.g.$, a cyber foraging radius of 11 hops is enough to cover the whole physical network. Intuitively, such restricted knowledge forces some of the variables n_{ij}^P and p_{ij} to zero, reducing the probability that a user j can be mapped or even allocated. On the y-axis we show again the efficiency loss as defined previously. The first take-home message is that the same efficiency improvements apply similarly to both small and large scale physical topologies. This is again due to the fact that the link mapping strategy strongly depends on the number of available physical paths in $\mathcal{P}'$, whose size is similar although the size of the topologies changes. Note how the efficiency loss deriving from a (lack of) cooperation with the cyber foraging phase increases when the information available would be otherwise limited, and more interestingly, when only few updates on the availability of the resources are permitted.

Edge Function Mapping - Allocation Interaction. We now show how, with full awareness of the underlying edge network topology, the cooperation between virtual network mapping and allocation could lead to efficiency improvement as well. Referring to Figure 1, in the previous simulation scenario we have allowed full exchange of the set of candidates, $i.e.$, variables $\{n_{ij}^V, l_{kj}\}$, and we have analyzed the performance restricting the knowledge of the available resources, $i.e.$, variables $\{n_{ij}^P, p_{kj}\}$. This time instead, we show what happens when, to the contrary, the set of candidates is unknown due to lack of cooperation between the edge function mapping and

the binding phase, but the algorithm has full knowledge of the physical topology. We do not study the case where the knowledge of both physical and virtual resources are partial (since the results are not surprising), but we focus on analyzing the effect of each individual cooperation.

In Figure 2g, we show the impact of the physical node capacity on the efficiency loss, this time computed as follows:

$$\text{Efficiency Loss} := \frac{MEF_{(N^{P\prime},\mathcal{P}')} - OEF_{(N^{P\prime},\mathcal{P}')}}{OEF_{(N^{P\prime},\mathcal{P}')}} \quad (33)$$

where, $MEF_{(N^{P\prime},\mathcal{P}')}$ is defined as in Equation (32), $OEF_{(N^{P\prime},\mathcal{P}')}$ is the set of onloaded edge function when the set of available nodes is $N^{P\prime}$, and the set of available paths $\mathcal{P}'$, and, in this case study, $(N^P, \mathcal{P}) \equiv (N^{P\prime}, \mathcal{P}')$. [3]

The edge function mapping phase, in this scenario, only checks whether or not there exists at least a node whose (total and not residual) CPU capacity is larger than it was requested for each edge function process – third constraint of Problem (3) – and whether or not there exists a physical path whose (total and not residual) bandwidth capacity on each link is greater than or equal to that requested for each virtual link – fourth constraint of Problem (3). When both constraints are satisfied, the edge function is mappable. On the x-axis we have the maximum node capacity: $x = 10$ means that the edge node capacity is assigned uniformly at random between 1 and 10 while the capacity of a single link is set to 5, that is, each physical link can host at most 5 virtual edge function links. Note that the range of efficiency loss goes from 12% to a peak of 43% when the Revenue Maximization policy is used.

In Figure 2h we show the impact of the pool size – the number of simultaneous edge function requests collected before running the last two edge onloading phases – on the efficiency loss. Note how, as the allocation candidates increase, the allocation policy used changes the efficiency performance. This observation suggests an important principle in designing centralized or distributed solutions for the complete edge function onloading problem, that is, *without cooperation among the edge onloading phases, complex mapping algorithms do not help increase the global system utilization*.

VII. Conclusions

In this paper, we presented a model for the complete edge function onloading problem, a series of interconnected infrastructure management services consisting of three main phases: cyber forage, edge function mapping, and binding or allocation. We modeled via optimization each of the three phases, and we showed that such phases are tightly connected, that they should be run by the back-end infrastructure, and that the solutions that either solve a particular phase, or jointly solve multiple phases along with the interactions between them are incomplete and may lead to inefficiencies. We carried out a simulation study using our own event driven simulator to demonstrate that the proposed cooperation enables a substrate (mobile) edge cloud provider to host a larger set of constrained virtual network requests, and that the design of protocols that consider interaction among the phases may be as crucial as the adoption of a more efficient offloading or onloading algorithm.

[3]Note how, in the previous simulation scenario, we always have $MEF_{(N',\mathcal{P}')} \equiv OEF_{(N^{P\prime},\mathcal{P}')}$.

Acknowledgment

This work has been partially supported by the National Science Foundation award CNS-1647084, and EU H2020 Superfluidity Project (grant agreement No.671566).

References

[1] EU H2020 SESAME Project, http://www.sesame-h2020-5g-ppp.eu/.
[2] EU H2020 SUPERFLUIDITY Project, http://superfluidity.eu/.
[3] 5G Vision–The 5G Infrastructure Public Private Partnership: the next generation of communication networks and services. Technical report, 5G-PPP, 2015.
[4] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *2Proc. of the ISCO'16*, pages 1–8, Jan 2016.
[5] V. Bahl. Micro data centers for mobile edge computing (keynote talk) https://goo.gl/fkwpzm, 2015.
[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. http://www.stanford.edu/people/boyd/cvxbook.html, 2004.
[7] B. Chun and A. Vahdat. Workload and failure characterization on a large-scale federated testbed. Technical report, IRB-TR-03-040, Intel Research Berkeley, 2003.
[8] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. CloneCloud: Elastic Execution Between Mobile Device and Cloud. In *Proc. of the EuroSys '11*, pages 301–314, 2011.
[9] S. de Vries and R. V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, (3):284–309, 2003.
[10] E. Cuervo et al. MAUI: Making Smartphones Last Longer with Code Offload. In *Proc. of the MobiSys '10*, pages 49–62, 2010.
[11] D. Eppstein. Finding the k shortest paths. *IEEE Symposium on Foundations of Computer Science*, 1994.
[12] ETSI. Mobile Edge Computing https://goo.gl/zgnft4, 2017.
[13] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee. Online allocation of virtual machines in a distributed cloud. *IEEE/ACM Transactions on Networking*, PP(99):1–12, 2016.
[14] Ketan Bhardwaj et al. AppFlux: Taming App Delivery via Streaming. In *Proc. of the Usenix TRIOS*, 2015.
[15] Ketan Bhardwaj et al. Fast, scalable and secure onloading of edge functions using airbox. In *IEEE/ACM Symp. on Edge Computing*, 2016.
[16] Li, Zhiyuan et al. Computation offloading to save energy on handheld devices: A partition scheme. In *Proc. of the 2001 Inter. Conf. on Compilers, Arch., and Synthesis for Embedded Systems*, CASES, 2001.
[17] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, PP(99):1–1, 2017.
[18] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a users perspective. Technical report, BUCS-TR-2001-003, CS Department, Boston University, 2001.
[19] S. Kosta et al. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proc. of the INFOCOM'12*, pages 945–953, March 2012.
[20] M. Satyanarayanan, P. Bahl, R. Cceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.
[21] R. Sedgewick. Implementing quicksort programs. *Commun. ACM*, 21:847–857, October 1978.
[22] T. Taleb and A. Ksentini. An analytical model for follow me cloud. In *Proc. of the GLOBECOM'13*, pages 1291–1296, Dec 2013.
[23] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, PP(99):1–1, 2017.
[24] Q. Xia, W. Liang, and W. Xu. Throughput maximization for online request admissions in mobile cloudlets. In *Proc. of the LCN'13*, pages 589–596, Oct 2013.
[25] Q. Xia, W. Liang, Z. Xu, and B. Zhou. Online algorithms for location-aware task offloading in two-tiered mobile cloud environments. In *Proc. of the UCC'14*. IEEE/ACM, 2014.