OKpi: All-KPI Network Slicing Through Efficient Resource Allocation

OKpi: All-KPI Network Slicing Through Efficient Resource Allocation

J. Martín Peréz†, F. Malandrino‡, C. F. Chiasserini∗, C. J. Bernardos†
† Universidad Carlos III de Madrid, Spain – ‡ IEIIT-CNR, Torino, Italy – ∗ Politecnico di Torino, Torino, Italy

***Abstract*—Networks can now process data as well as transporting it; it follows that they can support multiple services, each requiring different key performance indicators (KPIs). Because of the former, it is critical to efficiently allocate network and computing resources to provide the required services, and, because of the latter, such decisions must jointly consider all KPIs targeted by a service. Accounting for newly introduced KPIs (e.g., availability and reliability) requires tailored models and solution strategies, and has been conspicuously neglected by existing works, which are instead built around traditional metrics like throughput and latency. We fill this gap by presenting a novel methodology and resource allocation scheme, named OKpi, which enables high-quality selection of radio points of access as well as VNF (Virtual Network Function) placement and data routing, with polynomial computational complexity. OKpi accounts for *all* relevant KPIs required by each service, and for any available resource from the fog to the cloud. We prove several important properties of OKpi and evaluate its performance in two real-world scenarios, finding it to closely match the optimum.**

I. Introduction

Network Function Virtualization (NFV) enables mobile networks to expand their capabilities beyond data transport and to support software-based applications on demand. Under such a paradigm, third party industries ("verticals") specify their services through a graph of virtual network functions (VNFs), then it is the mobile network's task to run such services. This requires *selecting* suitable radio points of access (PoAs) as well as *placing* and *connecting* the VNFs across computing and network resources[1], in order to deliver the vertical services with the required quality of service. Importantly, different resources can be employed to achieve this goal, ranging from those in the cloud to the ones at the edge of the network infrastructure (i.e., through multi-access edge computing (MEC)) or in the fog (i.e., in devices such as smartphones, vehicles, robots). Note that *which* and *how many* resources are used is a critical issue, as their associated cost, performance, and availability vary significantly. This is confirmed by [1]–[4], highlighting how cost is a grave concern for both the mobile operators owning the cellular infrastructure and the verticals paying for the services mobile systems should support.

In spite of the fact that the VNF placement problem has been already addressed in the literature (see Sec. II for a more in-depth discussion), virtually all existing approaches only focus on throughput and latency as performance metrics, ignoring what is one of the most disruptive innovations of 5G. Indeed, new generation networks have been conceived with the goal of serving multiple use cases (summarized in the ubiquitous ITU "pyramid" [5]) whose requirements are both diverse and heterogeneous. *Diverse* reflects the fact that, for example, the latency requirements of different use cases can vary by several orders of magnitude. *Heterogeneous* refers to the important fact that 5G introduces several new performance metrics (or KPIs, Key Performance Indicators), including service availability (in both space and time) and service reliability, as exemplified in Fig. 1. These new KPIs are all but ignored by existing VNF placement algorithms, which may thus be unable to honor the verticals' requirements. We underline that, as discussed in the next section, accounting for all relevant KPIs requires introducing a new problem formulation and a new solution, which cannot be a mere extension of previous work.

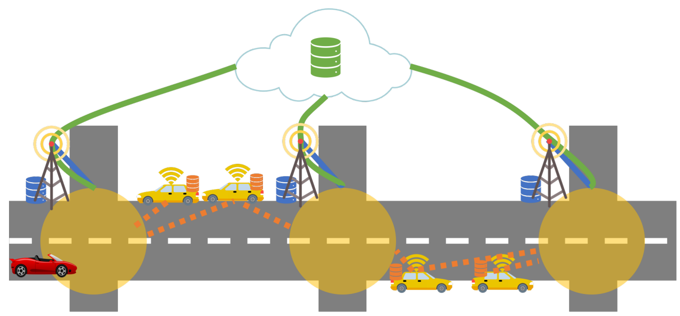

Fig. 1. As per geographical availability requirements, a safety service (e.g., warning vehicles on a collision course [6]) must be provided at the highlighted intersections with high reliability. This can be obtained by deploying: (a) four service instances at parked cars (fog resources), at lower cost but also lower reliability, hence, needing redundancy to meet reliability constraints (orange option); (b) three instances at the points of access (PoA), e.g., eNBs, covering the intersections (MEC resources, blue option); (c) deploying only one instance in the cloud, but with larger delay (green option).

Furthermore, existing studies have limited or no support for several specific aspects of slicing-based networks, including (i) the opportunity to reuse existing VNF instances ([3] reuses VNFs within a single data center, while [7] only focuses on cost), (ii) the possibility of combining cloud-based and MEC-based services (with the exception of [8], which however only deals with caching), and (iii) the need to make decisions on how to place *and* connect VNFs, thus jointly addressing VNF placement and data routing ([9]–[12] do so, but without considering PoAs or VNF re-usage, and under some limiting assumptions, e.g., on the number of VNF instances).

[1]Although memory and storage resources have been omitted for brevity, our framework can handle them as well.

Our contribution and methodology. We fill this gap by introducing OKpi, an efficient framework able to create high-quality, end-to-end network slices. Specifically, OKpi advances the state-of-the-art in the following main ways:

(i) it effectively tackles the 5G-defined network slicing KPIs;
(ii) it leverages fog, MEC, and cloud resources, allowing VNFs to be placed at any layer of the network topology;
(iii) it accounts for the fact that already-deployed VNF instances can be reused for newly-requested services[2];
(iv) in such a general setting, it makes *joint* decisions on PoA selection, VNF placement, and traffic routing, which minimize the cost of the resources, thus addressing both mobile operators and verticals' concerns;
(v) it exhibits a low, namely, polynomial, complexity.

We remark that, not only the problem we pose is novel, but also our methodology to solve it blends together graph theory and optimization, in a unique fashion. In particular, we describe possible decisions through a graph that reflects their effect on KPIs. Such a graph is then translated into a multi-dimensional expanded graph, which allows us to efficiently find feasible decisions leveraging simple shortest-path algorithms. The expanded graph can be built with different levels of detail and size, which results into a tuneable tradeoff between computational complexity and optimality.

In the remainder of the paper, we first review related works in Sec. II, highlighting which KPIs they consider and the novelty of our study. After that, we introduce the system model in Sec. III, and the problem formulation in Sec. IV. The OKpi solution and algorithm are described in Sec. V; then Sec. VI proves several relevant properties of OKpi and discusses its computational complexity. Sec. VII compares OKpi against the optimum in a small-scale, yet practically relevant robotics scenario, and shows its performance in a real-world, large-scale automotive scenario. Finally, Sec. VIII concludes the paper and highlights current research directions.

II. Related work

One of the pioneering works on VNF placement is [13], which casts placement as a generalized assignment problem (GAP) and proposes a near-optimal solution based on bi-criteria approximation. Very recent works [10], [12], [14] focus on the mutual influence of VNF placement and traffic routing. Others tackle the VNF placement problem through graph theory [9], [15] and set-covering [4], obtaining very good competitive ratios (constant in specific cases for [4]).

In the context of MEC, some works tackle tasks different from sheer data processing; as an example, [11], [16] aim at jointly optimizing computation and caching offloading between cloud-based and MEC-based infrastructure. Others focus on additional decisions that can be made in slicing scenarios, e.g., priority assignment in [3]. A body of works considers incremental deployment, i.e., service requests arriving at different times: in this case, it is possible to share existing VNF instances [3], [7], [17], augment routing paths instead of computing them from scratch [7], [17], and minimize the difference between current and future network configuration [17]. Among the few works tackling non-functional requirements, [2] performs resilient VNF placement, to achieve robustness to equipment failures. More recently, [18] considered the problem of jointly placing the VNFs and the data they need.

VNF placement, along with the closely-related problem of VNF chaining, has been studied in the software-defined networking and cloud-computing contexts as well. For instance, [19] focuses on updating the placement in order to react to traffic changes, and [20] deals with the parallelization opportunities offered by VNF graphs. Other works [10], [21] focus on the choice between MEC- and cloud-based computation resources, while [8] studies which cache storage (i.e., MEC- or cloud-based) to access, balancing miss probability and cost.

Several works aim at simplifying the problem of VNF placement by characterizing and/or predicting the traffic demand. In particular, [22] exploits the spatial and temporal variability of traffic demand to serve it with as little resources as possible; as for demand prediction, popular approaches include reinforcement learning [23]. In a similar spirit, [24] estimates the resources needed by an incoming service request before deciding whether or not it shall be accepted.

Finally, a body of work addresses the slicing of the radio access network; in particular, [25] proposes solutions that let different virtual operators use the radio resources without interfering, while [26] develops a stochastic model to investigate the throughput and delay of a slice as functions of the cell parameters. Although such specific aspects are out of the scope of our work, we do tackle the problem of selecting radio technologies and points of access that honor the required KPI targets and minimize the cost.

Novelty. Remarkably, *none* of the existing works accounts for fundamental KPIs in network slicing such as availability or reliability: due to their pioneering nature, such works focus on traditional performance metrics, namely, service latency and/or network throughput. Although some approaches could be extended to account for additional KPIs, such extensions would not be trivial and would, in general, jeopardize their complexity and/or competitive ratio properties. OKpi, on the contrary, is designed from the start to support multiple, heterogeneous KPIs in an effective and efficient manner, beside accounting for all types of resources and their location, from the fog to the cloud. Our methodology combining graph theory and optimization is also unique, and provides an effective way to tradeoff optimality and complexity.

III. System model

Our model concisely describes the two main components of mobile, slicing-based networks: the services they support (Sec. III-A), and the computing and network resources they include (Sec. III-B). Each of them is modeled through a graph – the service graph and the physical graph, respectively. We then describe how such graphs can be combined in Sec. III-C.

[2]This is possible if the services share a common subset of VNFs and no service isolation constraints exist.

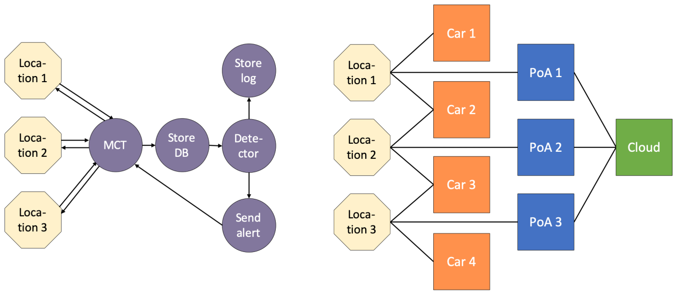

Fig. 2. Service (left) and physical (right) graphs corresponding to the example in Fig. 1. Messages periodically sent by vehicles are collected through the Mobile Communication Transport (MCT), e.g., virtual eNB and vEPC, then stored in a database and used for detecting vehicles on a collision course. The latter are warned by sending them an alert. In the service graph, vertices are endpoints (yellow) and VNFs (purple), edges are directed and correspond to flows l. In the physical graph, vertices are endpoints in $\mathcal{E}$ or nodes in $\mathcal{C}$, and edges correspond to links in $\mathcal{L}$; colors correspond to those in Fig. 1 and refer to the different resource locations: fog (orange), MEC (blue), cloud (green).

A. *Services*

A vertical service $s \in \mathcal{S}$ is described through a *service graph* where vertices are VNFs, $v \in \mathcal{V}$, and edges specify in which order the VNFs should process the related data traffic (i.e., how data shall be routed from a VNF instance running on a network node to the next). Note that VNFs can also represent database-related functionalities [18], requiring storage resources: like other VNFs, they must be placed on a node and consume resources therein.

A service s is associated with one or more *KPIs*, namely,

- the required bandwidth, or expected traffic load l to be transferred and handled by the VNFs composing the service;
- the maximum allowed delay $D(s)$;
- the minimum level of reliability $H(s)$;
- the required geographical availability at a subset of locations, $A(s) \subseteq \mathcal{A}$, where $\mathcal{A} = \{\alpha\}$ represents the set of all possible locations in the considered region. As an example, $A(s)$ can represent the urban intersections where an automotive vertical wants to provide a safety service (Fig. 1), or the areas where robots should move within a warehouse. We refer to the combination of a service and a location as an *endpoint* $e = (\alpha, s) \in \mathcal{E} \subseteq \mathcal{A} \times \mathcal{S}$;
- the lifetime (or temporal availability) $\varphi(e) \subseteq \mathcal{T}$, corresponding to a subset of all time steps $\mathcal{T}$ during which the service must be available at endpoint e.

As foreseen by standards [27], services may be associated with one or more of these requirements, i.e., not all KPIs have to be specified for all services. Also, without loss of generality, we consider that the traffic associated with a service is generated at endpoint e and has to be processed by the VNFs in the service graph; in Fig. 2(left), this would correspond to uplink data transfers. Note however that our model is general and can also capture downlink as well as bidirectional traffic patterns.

The quantity of traffic originated at endpoint $e \in \mathcal{E}$, that has been processed last at VNF v_1, and will be next processed at VNF v_2 is denoted with $l(e, v_1, v_2)$ (with $l(e, v, v)$ being the traffic that will be processed for the first time at v). After a traffic flow is processed at a VNF, the outgoing traffic can increase, decrease, or be split among several other VNFs, according to the service graph. Parameters $\chi(v_1, v_2, v_3)$ express the fraction of the traffic that was last processed (or originated) at $v_1 \in \mathcal{V} \cup \mathcal{E}$, that is currently processed at v_2, and that will next be processed at v_3. For instance, if v_2 is a deep packet inspector, $\chi(v_1, v_2, v_3) = 1$; but if v_2 is a firewall, then $\chi(v_1, v_2, v_3) \leq 1$.

The need for such χ-parameters is due to the fact that, as discussed in the previous example, there is no flow conservation on the service graph. Instead, the following *generalized flow conservation* law holds:

$$l(e, v_2, v_3) = \sum_{v_1 \colon v_1 \neq v_2} l(e, v_1, v_2)\chi(v_1, v_2, v_3) + l(e, v_2, v_2)\chi(e, v_2, v_3), \quad \forall v_2, v_3 \in \mathcal{V} : v_2 \neq v_3 \quad (1)$$

The intuitive meaning of (1) is that traffic traveling from VNF v_2 to VNF v_3 must either come from another VNF v_1 and then it is transformed in v_2 according to the χ-coefficients (first term of the second member), or it has just originated at e and is processed for the first time at v_2 (second term).

B. *Radio coverage and Fog/MEC/cloud resources*

Network nodes, with switching or computing capabilities, are denoted by $c \in \mathcal{C}$, while endpoints, which are origins or destinations of service traffic, are denoted by $e \in \mathcal{E}$. Nodes may be equipped with different resources, e.g., CPU or memory; the set of resources is identified by $\mathcal{K} = \{\kappa\}$. The quantity of resource type κ available at node c is specified through parameters $k(\kappa, c)$, hence, $k(\kappa, c) = 0\, \forall \kappa$ for pure network equipment like traditional, non-software, switches. Also, binary parameters $R_i(c)$ express whether node c is equipped with radio interface $i \in \mathcal{I}$ or not. A radio interface available at node c determines which locations, hence endpoints, node c covers – an important feature of fog and MEC nodes.

Radio coverage, fog, MEC, and cloud resources can then be represented through a *physical graph* whose vertices are the network nodes and the endpoints, and the edges $(i, j) \in \mathcal{L} \subseteq (\mathcal{C} \cup \mathcal{E})^2$ represent the physical links connecting them, as per the network topology and the coverage provided by the radio interfaces. Each edge (i, j) is associated with delay $D_{i,j}$ and traffic capacity $C_{i,j}$. Any node c and link (i, j) are associated with *reliability* values $\eta(c, t)$ and $\eta(i, j, t)$, respectively, which express the probability that a specific node or link works as intended at time $t \in \mathcal{T}$. The fact that reliability values are time dependent models real-world aspects like the fleeting quality of communication links involving fog nodes, e.g., robots or cars, as in Fig. 1.

One of the main decisions to make through our model is where to process and route the service traffic. To this end, we introduce variables $\tau_{i,j}(e, v_1, v_2)$ representing the flows over the physical graph, or, more specifically, the traffic originated at $e \in \mathcal{E}$, traversing $(i, j) \in \mathcal{L}$, last processed at v_1, and to be next processed at v_2. Such traffic can be either processed at j, or just transiting through j; these two options are described by the two real variables $p_{i,j}(e,v_1,v_2)$ and $t_{i,j}(e,v_1,v_2)$ and by imposing: $\tau_{i,c}(e,v_1,v_2) = p_{i,c}(e,v_1,v_2) + t_{i,c}(e,v_1,v_2)$. Furthermore, the traffic going out of c must be equal to the sum of that transiting through c and that just processed at c:

$$\sum_{(c,h)\in\mathcal{L}} \tau_{c,h}(e,v_2,v_3) = \sum_{(i,c)\in\mathcal{L}} \Big[t_{i,c}(e,v_2,v_3) + p_{i,c}(e,v_2,v_2)\cdot \chi(e,v_2,v_3) + \sum_{v_1\in\mathcal{V}} p_{i,c}(e,v_1,v_2)\chi(v_1,v_2,v_3)\Big]. \quad (2)$$

Finally, each physical link (i,j) cannot carry more traffic than its capacity, i.e., $\sum_e \sum_{v_1,v_2} \tau_{i,j}(e,v_1,v_2) \leq C_{i,j}$.

C. Deploying VNFs and assigning resources

A node can process traffic of a VNF if it hosts an instance of that VNF; this is modeled through binary variables $\rho(v,c) \in \{0,1\}$, expressing whether VNF v is deployed at node c. Variables $a_c(e,v,\kappa)$, instead, express the quantity of resources of type κ that are assigned to the instance of VNF v deployed at node c and used for traffic generated at endpoint e. Such quantities cannot exceed the node capabilities, i.e., for any c and κ, $\sum_{e\in\mathcal{E}}\sum_{v\in\mathcal{V}} a_c(e,v,\kappa) \leq k(\kappa,c)$.

Importantly, for any $\kappa \in \mathcal{K}$, the quantity of traffic processed by v at node c cannot exceed the ratio between the quantity $a_c(e,v,\kappa)$ of resource type κ assigned to the VNF, and the quantity $r_\kappa(v)$ of resource type k needed by VNF v to process one unit of traffic:

$$\sum_{(i,c)\in\mathcal{L}}\sum_{e\in\mathcal{E}}\sum_{v_1\in\mathcal{V}} p_{i,c}(e,v_1,v_2) \leq \frac{a_c(e,v,\kappa)}{r_\kappa(v_2)} \quad \forall\kappa\in\mathcal{K}. \quad (3)$$

Also, node c's resources can be assigned to a VNF v only if the latter is deployed therein: $a_c(e,v,\kappa) \leq \rho(v,c)k(\kappa,c)$, for any c, κ, and v. These conditions imply that no traffic is processed at a node where no instance of a VNF is deployed.

Last, we ensure that VNFs are placed only at nodes where all the needed radio interface(s) are available, e.g., an MCT may work only at nodes equipped with specific radio interfaces. Thus, for any node c, interface i, and VNF v, we have: $\rho(v,c)r_i(v) \leq R_i(c)$, where $r_i(v) \in \{0,1\}$ are parameters specifying whether interface i is needed by VNF v, and $R_i(c)$ specifies whether such an interface is available at c.

Matching service and physical flows. Since our system model includes two graphs, we must ensure that service flows l and physical flows τ match. To this end, we impose that the flow incoming the first VNF of a service graph corresponds to one or more traffic flows on the physical graph:

$$l(e,v,v) = \sum_{(e,c)\in\mathcal{L}} \tau_{e,c}(e,v,v), \forall e\in\mathcal{E}, v\in\mathcal{V}. \quad (4)$$

Once (4) is met, then (1) and (2) ensure that the traffic on subsequent links is processed as specified by the χ-parameters.

IV. Problem Formulation

In this section, we formalize the problem of creating end-to-end network slices that meet all the required KPI targets (Sec. IV-A) while minimizing the total cost (Sec. IV-B). The problem complexity is then discussed in Sec. IV-C.

A. Meeting service KPIs

To handle more easily the service KPIs, we define a *string*, $w \in Wc$, over the physical graph as a sequence of physical links traversed by a flow, with the first item of the string being an endpoint. Similarly to [28], the possible strings can be pre-computed and stored for later usage.

Since a service flow can be split across different strings, we define $f(e,v_1,v_2,w)$ as the fraction of service flow $l(e,v_1,v_2)$ traversing string w. Clearly, such fractions must sum to 1. We also introduce string-wise equivalents to $\tau_{i,j}(e,v_1,v_2)$, $t_{i,j}(e,v_1,v_2)$, and $p_{i,j}(e,v_1,v_2)$. Specifically, $\tau_{i,j}(e,v_1,v_2,w)$ represents the traffic of service flow $l(e,v_1,v_2)$ traversing link (i,j) on its journey through string $w\in\mathcal{W}$, and then impose $\tau_{i,j}(e,v_1,v_2) = \sum_{w\in\mathcal{W}} \tau_{i,j}(e,v_1,v_2,w)$. Similar definitions and conditions hold for $t_{i,j}(e,v_1,v_2,w)$ and $p_{i,j}(e,v_1,v_2,w)$.

Furthermore, the fraction of service flow over a certain string w must match the physical traffic on the corresponding links, i.e., for all endpoints, VNFs v_1 and v_2, links, and strings, we have: $f(e,v_1,v_2,w)l(e,v_1,v_2) = \tau_{i,j}(e,v_1,v_2)\mathbb{1}_{w(i,j)}$, where $\mathbb{1}_{w(i,j)}$ denotes that link $(i,j)\in\mathcal{L}$ belongs to w.

1) Service latency: it has two components: network delay due to traffic traversing links and switches, and processing times at the nodes hosting VNF instances. Given endpoint e, the average network delay can be computed as the weighted sum of the delays associated with the individual strings taken by the traffic originated at e:

$$d_{\text{net}}(e) = \sum_{w\in\mathcal{W}}\sum_{v_1,v_2\in\mathcal{V}} f(e,v_1,v_2,w) \sum_{(i,j)\in w} D_{i,j}. \quad (5)$$

As for the processing time, let $\hat{f}_c(e,v_1,v_2)$ be the fraction of the service traffic flow $l(e,v_1,v_2)$ processed at the instance of VNF v_2 located at node c. Then the quantity of traffic $\lambda_c(e,v_2)$ originating at e and processed at the instance of v_2 in c is:

$$\lambda_c(e,v_2) = \sum_{v_1\in\mathcal{V}} \hat{f}_c(e,v_1,v_2) l(e,v_1,v_2).$$

Note that such traffic may come from different physical links.

Next, we model VNF instances as M/M/1-PS queues (see, e.g., [13]); the choice of the processor sharing (PS) policy closely emulates the behavior of a multi-threaded application running on a virtual machine. Hence, the total processing time at the instance of v_2 deployed at node c is: $1/(a_c(e,v_2,\text{cpu}) - r_{\text{cpu}}(v_2)\lambda_c(e,v_2))$. Summing over all flows, the total processing delay incurred by traffic originating at e can be written as:

$$d_{\text{proc}}(e) = \sum_{v_1,v_2\in\mathcal{V},c\in\mathcal{C}} \hat{f}_c(e,v_1,v_2) \frac{1}{a_c(e,v_2,\text{cpu}) - r_{\text{cpu}}(v_2)\lambda_c(e,v_2)}. \quad (6)$$

Combining the above equation with (5), and recalling that $D(s)$ is the maximum target delay for service s, the service latency constraint for its endpoints can be stated as:

$$d_{\text{net}}(e) + d_{\text{proc}}(e) \leq D(s), \quad \forall e\in\mathcal{E}.$$

Finally, note that the relationship between assigned CPU and processing time in the expression of $d_{\text{proc}}(e)$ also means that the CPU has a different role from the other types of resources. Indeed, for resources other than CPU, we can assign to each VNF instance exactly the amount needed to honor (3), as a greater amount would yield no benefit. With CPU, instead, there is an additional degree of freedom we can play with: assigning more CPU results in shorter processing times, but higher costs.

2) Service geographical availability: by service availability requirements, all locations in $A(s) \subseteq \mathcal{A}$ must be covered by service s. In other words, for all endpoints $e = (\alpha, s)\colon \alpha \in A(s)$, there must be a link (e, c) on the physical graph to a node c that is equipped with a radio interface covering α and that runs (or it is connected to another node running) the first VNF of the service graph.

3) Service reliability and temporal availability: the reliability of a string can be computed as the product between the reliability values of all links and nodes belonging to it. We can therefore ensure that the reliability $H(s)$ required for service s is honored, by considering a weighted sum of the per-string reliability values. In symbols, $\forall e \in \mathcal{E}, t \in \varphi(e)$,

$$\prod_{v_1,v_2\in\mathcal{V}} \sum_{w\in\mathcal{W}} f(e,v_1,v_2,w) \prod_{(i,j)\in w} \eta(j,t)\eta(i,j,t) \geq H(s)\,.$$

Note that imposing the above constraint for every time instant during the service lifetime also ensures that the target temporal availability is met.

B. Objective

As mentioned in Sec. I, cost is one of the main concerns related to service virtualization and network slicing. Such cost mainly comes from using network and computation resources. To model this issue, we define:

- a fixed cost $c_c(v)$, due to the creation at node c of a VNF instance v; this cost is null if an existing VNF instance can be reused;
- a cost $c_c(\kappa)$, incurred when using a unit resource κ at node c;
- a cost $c_{i,j}$, incurred when one unit traffic traverses link (i, j).

Then, upon receiving a request to deploy a service instance s, we formulate the following cost-minimization problem:

$$\min \sum_c \sum_v \left[c_c(v) + \sum_e \sum_\kappa c_c(\kappa) a_c(e, v, \kappa) \right] + \sum_{(i,j)} \sum_e \sum_{v_1,v_2} c_{i,j} \tau_{i,j}(e, v_1, v_2)$$

subject to the constraints reported in Secs. III-A–IV-A.

We recall that the endpoints e to consider depend on the service and on its geographic availability requirements, while the VNFs are those specified by the service graph. Furthermore, a solution to the above problem will always opt for reusing an existing instance of a VNF, whenever possible, as this would nullify the instantiation cost $c_c(v)$.

C. Nature and complexity of the problem

The problem of jointly making VNF placement and data routing decisions is notoriously hard; indeed, simpler versions thereof (considering one KPI only) have been proven to be NP-hard via reductions from the generalized assignment [13], [14] and set covering [4] problems. Thus, directly solving the above problem is impractical for all but very small instances.

We also observe that our problem can be seen as a more complex version of a multi-constrained path (MCP) problem, where the cost (hence, the weight of the edges in the MCP graph) *changes* at every hop. Although known solutions to the MCP problem, e.g., [29], are not applicable, such a similarity motivates us to propose an effective and efficient heuristic, called OKpi, for which we prove that:

- it provides high-quality VNF placement and data routing decisions, whose feasibility is guaranteed;
- such decisions are made in polynomial time;
- under mild homogeneity assumptions, decisions are optimal;
- in the general case, they can be arbitrarily close to the optimum.

V. The OKpi solution

Our solution includes three main steps. First, by leveraging the physical graph, we create a *decision graph* $\widetilde{G} = (\widetilde{N}, \widetilde{E})$, summarizing the service deployment decisions that can be made and their effect on the KPIs (Sec. V-A). Then we translate this graph into an *expanded graph*, and use the latter, along with the service graph in Sec. III-A, to identify a set of feasible decisions as well as to select, among them, the lowest-cost one (Secs. V-B and V-C). For presentation clarity, we present OKpi in the case where the service graph is a chain starting from an endpoint e and including N VNFs $v_1 \dots v_N$, and only one instance of each VNF can be placed. As discussed in Sec. V-D, both limitations can be dropped: OKpi works with arbitrary service graphs requiring any number of instances of each VNF.

A. The decision graph

Given the physical graph modeling the service endpoints and the fog, MEC, and cloud resources, we build the decision graph $\widetilde{G}$ with the aim to represent the possible service deployment decisions and their effects on the service KPIs.

As a preliminary step, we consider the *computation-capable* nodes in the physical graph (hence, a subset of $\mathcal{C}$), and for each of them we create $(|\mathcal{V}| - 1)$ replicas. Consistently, we create auxiliary edges (i) connecting each node c and its replicas in a chain fashion, and assign them zero delay, infinite capacity, and reliability 1, and (ii) connecting any replica of c with any computing node d, for which a link $(c, d) \in \mathcal{L}$ exists. Crucially, introducing node replicas enables us to account for the possibility to deploy multiple VNFs at the same node without introducing self-loops in the decision graph. Indeed, as it will be more clear later, given that a VNF is placed in c, each replica thereof represents the possibility to deploy the next VNF again in c.

Let then $\widetilde{G} = (\widetilde{N}, \widetilde{E})$ be the decision graph where:

- $\widetilde{N}$ includes the endpoints in $\mathcal{E}$, and the computation-capable nodes in the physical graph as well as their replicas;
- $\widetilde{E}$ is the set of (i) the aforementioned auxiliary links, and (ii) the virtual links (i.e., single physical links or sequences thereof) connecting the vertices in $\widetilde{N}$.

Every edge $(\tilde{n}_1, \tilde{n}_2)$ in $\widetilde{E}$ representing a virtual link has the following properties:

- its capacity $\widetilde{C}_{\tilde{n}_1,\tilde{n}_2}$ is set to the minimum of the individual capacities of the physical links composing the virtual link;
- its delay $\widetilde{D}_{\tilde{n}_1,\tilde{n}_2}$ is set to the sum of the individual delays of the physical links composing the virtual link;
- its reliability $\widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}$ is set to the product of the reliability values of physical links and nodes (both computation and pure-routing capable) included in the virtual link.

We stress that, when some services are already active in the network, we build the decision graph considering the *residual* capabilities of physical links and nodes, i.e., those not assigned to already-running services. Similarly, in case of virtual links sharing the same physical links, their capacity is updated as traffic is allocated to the physical links.

B. The expanded graph: finding decisions honoring availability and additive KPIs

Given the decision graph $\widetilde{G}$, our first purpose is to identify a set of *feasible* service deployment decisions that are consistent with the target KPIs. To this end, as a preliminary step, we ensure to meet the geographical and temporal availability requirements by pruning from $\widetilde{G}$ the vertices and edges that do not satisfy such constraints.

Then, for the additive KPIs, we proceed as follows. To any edge $(\tilde{n}_1, \tilde{n}_2)$ in the decision graph, we assign a *multi-dimensional weight* $\tilde{w}(\tilde{n}_1, \tilde{n}_2)$, defined as:

$$\tilde{w}(\tilde{n}_1, \tilde{n}_2) = \left(\frac{\widetilde{D}_{\tilde{n}_1,\tilde{n}_2}}{D(s)}, \frac{\log \widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}}{\log H(s)} \right). \quad (7)$$

The intuition behind (7) is that the weight of edge $(\tilde{n}_1, \tilde{n}_2)$ corresponds to the fraction of the target delay and reliability that will be "consumed" by taking that edge, i.e., by deploying a VNF at $\tilde{n}_1$ and the subsequent one at $\tilde{n}_2$. We stress that using logarithms in the second term of the weight allows us to translate a multiplicative performance index (namely, reliability) into an additive one[3].

Next, we take an approach inspired by [29] and build a multi-dimensional, *expanded graph*. Specifically, given a positive integer value of resolution γ:

1) for each vertex $\tilde{n}$ in the decision graph, create $(\gamma+1)^2$ vertices[4] $\tilde{n}^{0,0}, \tilde{n}^{0,1} \ldots, \tilde{n}^{0,\gamma} \ldots, \tilde{n}^{\gamma,\gamma}$;
2) for every edge $(\tilde{n}_1, \tilde{n}_2) \in \widetilde{E}$ with capacity $\widetilde{C}_{\tilde{n}_1,\tilde{n}_2}$ greater or equal to the amount of traffic to process, create directed edges from each vertex $\tilde{n}_1^{i,j}$ to vertex $\tilde{n}_2^{i+\lceil \gamma w(\tilde{n}_1,\tilde{n}_2)[0] \rceil, j+\lceil \gamma w(\tilde{n}_1,\tilde{n}_2)[1] \rceil}$ (if such a vertex exists), where the two superscripts refer to delay and reliability, respectively.

[3] It is easy to see that $\widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2} \widetilde{\eta}_{\tilde{n}_2,\tilde{n}_3} \geq H(s)$ translates into $\frac{\log \widetilde{\eta}_{\tilde{n}_1,\tilde{n}_2}}{\log H(s)} + \frac{\log \widetilde{\eta}_{\tilde{n}_2,\tilde{n}_3}}{\log H(s)} \leq 1$.

[4] The exponent 2 corresponds to the number of additive KPIs.

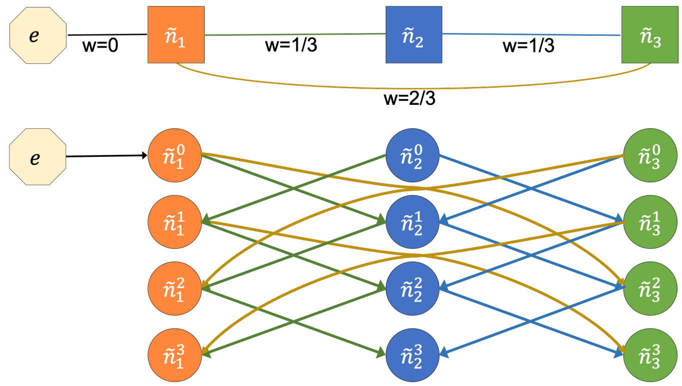

Fig. 3. Decision graph (top) and expanded graph (bottom) when only delay is considered as a KPI and $\gamma = 3$. In the decision graph, edges $(\tilde{n}_1, \tilde{n}_2)$ and $(\tilde{n}_2, \tilde{n}_3)$ have delay of 1 ms, while $(\tilde{n}_1, \tilde{n}_3)$ has delay of 2 ms (vertices representing replica nodes are omitted for simplicity); the target delay is 3 ms.

We stress that the expanded graph has *no weights* on its edges: the delay and reliability information that is expressed by weights in the decision graph is now represented by the topology of the expanded graph. A one-dimensional (i.e., one-KPI) example of decision graph and corresponding expanded graph is depicted in Fig. 3.

Finally, we identify a set of possible service deployments, i.e., VNF-to-compute node assignments and the corresponding data routing. We do so by looking for the shortest paths[5] in the expanded graph that (a) begin at endpoints, and (b) contain as many edges as there are VNFs to place. We underline that the latter is trivially required by the need to deploy all VNFs on the service graph, and by the fact that placing more VNFs at a physical node is allowed thanks to the replica nodes and auxiliary edges, as discussed in Sec. III-B.

In the following, we make several fundamental remarks on the expanded graph and on the paths, hence, the deployment decisions they correspond to. Given a KPI, we define as *depth* of a vertex in the expanded graph the quantity in the corresponding superscript; note that, by construction (see point 1 above), the maximum value of depth is γ. Also, let the *steepness* of an edge be the difference in depth between its target and source vertices. Considering the one-KPI example in Fig. 3(bottom), vertex $\tilde{n}_1^0$ has depth 0, vertex $\tilde{n}_3^2$ has depth 2, and the edge between the two has steepness $2 - 0 = 2$, i.e., equal to $\lceil \gamma w(\tilde{n}_1, \tilde{n}_3) \rceil$.

By construction, for a given KPI, the ratio between the steepness of an edge and γ is greater or equal to the weight component on the corresponding edge of the decision graph, which in turn is the fraction of the KPI target values consumed by making that decision (see (7)). As an example, considering

[5] In our implementation, we use the Bellman-Ford algorithm.

edge $(\tilde{n}_1^0, \tilde{n}_3^2)$ in Fig. 3(bottom), we have:

$$\frac{\text{steepness}}{\gamma} = \frac{2}{3} \geq w(\tilde{n}_1^0, \tilde{n}_3^2) = \frac{D_{\tilde{n}_1,\tilde{n}_3}}{D(s)} = \frac{2}{3}. \qquad (8)$$

The observations above allow us to state a very relevant property of the decisions corresponding to the paths on the expanded graph.

Lemma 1. *The decisions corresponding to any path on the expanded graph honor all additive KPIs.*

Proof: By definition, the depth of a vertex corresponds to the total steepness of the path required to reach it from endpoint e. Given that the maximum depth in the expanded graph is γ, there is *no path* with total steepness[6] greater than γ. Thanks to the relation between weight and KPI targets (exemplified in (8)), this implies that, given a path on the expanded graph, the sum of the weights of the corresponding edges in the decision graph cannot exceed 1, i.e., the corresponding decisions honor additive KPIs (including, thanks to the logarithmic weights, reliability). ■

Importantly, the smaller the resolution γ, the fewer the possible values of depth and steepness in the expanded graph, the fewer the levels of consumption of the KPI target values we are able to distinguish, which corresponds to introducing an error, akin to quantization. Indeed, $\gamma+1$ can be seen as the number of quantization levels[7] we admit: in the extreme case of $\gamma = 1$, all edges would have a steepness of 1, which also corresponds to exhausting the whole KPI target in one hop. Such a quantization error may lead to discarding some feasible solutions, and thus, in the most general case, may jeopardize the optimality of OKpi. However, two important facts stand out: *(i)* even enumerating all feasible paths in the decision graph is NP-hard, as proven in [29], hence, quantization is necessary; *(ii)* by increasing γ, OKpi can get *arbitrarily close to the optimum* (at the price of higher complexity).

Finally, we remark that all paths on the expanded graph honor additive KPIs constraints, *with the possible exception of delay*. Indeed, unlike other KPIs, whether or not the delay target is violated depends not only on the network latency, hence, the VNF placement, but also on the processing time, i.e., the quantity $a_c(e, v, \text{cpu})$ of CPU assigned to each VNF, which in turn impacts the deployment cost. We can account for this important aspect thanks to the M/M/1-PS model used for (6). In particular, below we show how to determine, given a possible deployment, whether there is a CPU assignment consistent with the target delay, and the cost thereof.

C. Minimizing the cost

We now need (i) for every path found in Sec. V-B, to identify the minimum-cost CPU assignment, i.e., the optimal values of the $a_c(e, v, \text{cpu})$ variables – if such an assignment exists –, and (ii) to determine the path that minimizes the overall cost.

[6]The steepness of a path should be not confused with the length of a path.

[7]Using logarithms for reliability values, which are all typically very close to 1, is akin to performing adaptive quantization.

To this end, for each path (hence, for a fixed e and for VNFs $v_1 \dots v_N$ to be deployed at computing nodes $\tilde{n}_1 \dots \tilde{n}_N$, respectively), we solve the following problem:

$$\min \sum_{\tilde{n},v} a_{\tilde{n}}(e, v, \text{cpu}) c_{\tilde{n}}(\text{cpu}) \qquad \text{s.t.} \qquad (9)$$

$$\sum_{\tilde{n}_1,\tilde{n}_2} \left(D_{\tilde{n}_1,\tilde{n}_2} + \frac{1}{a_{\tilde{n}_2}(e, v_2, \text{cpu}) - r_{\text{cpu}}(v_2)\lambda_{\tilde{n}_2}(e, v_2)} \right) \leq D(s).$$

If the problem above is infeasible for a given path, then that path (and the corresponding decisions) is incompatible with the delay target KPI and must be discarded.

Once the problem in (9) is solved for all paths identified in the expanded graph, we compute the total cost associated with each path (including all components defined in Sec. IV-B) and select and enact the lowest-cost deployment, thus fulfilling OKpi's purpose. Importantly, the problem is *convex*, hence, it can be efficiently solved in polynomial time [30]. The proof simply follows from observing that the objective in (9) is linear and the second derivatives of the delay constraint are positive in the decision variables, hence, the constraint itself is convex.

D. General scenarios

We now show how OKpi tackles arbitrary scenarios.

1) Arbitrary service graphs: If the service graph is more complex than a chain, as in Fig. 2(left), we can proceed by (i) decomposing the graph into a set of chains (e.g., one in uplink, from the MCT to the DB, and one in downlink, from the detector back to the MCT). OKpi is then applied subsequently to each chain, and the deployment decisions are cascaded. The case where multiple endpoints have to be covered, as in Fig. 2(left), is handled in the same way.

2) Multiple VNF instances: If the problem described in Sec. V-C is infeasible for *all* possible paths found in Sec. V-B, a possible reason could be the need to split the processing burden across multiple instances of the same VNF. This case is handled by first identifying the bottleneck VNF, i.e., taking the longest to process the service traffic, and then increasing by one the number of instances of that VNF in the service graph. OKpi is then re-run on the modified service graph.

VI. OKpi analysis

In this section, we prove several properties about OKpi; we start with the most essential aspect related to its effectiveness, i.e., its ability to meet all service KPIs.

Property 1. *OKpi's decisions honor all KPI targets.*

Proof: By Lemma 1, all decisions honor the additive KPIs. Concerning delay, it is guaranteed that such a KPI target is met, thanks to the delay constraint imposed while performing the CPU assignment. As noted in Sec. V-C, decisions resulting in an infeasible problem are discarded, hence, the selected decision honors the delay target. Finally, the availability constraints are satisfied through the initial selection of the vertices of the decision graph (see Sec. V-B). ■

Next, we address the computational complexity of OKpi.

Fig. 4. Service graph specifying the robot-based, smart factory application.

Property 2. *The worst-case computational complexity of OKpi is polynomial.*

Proof: To prove the property, we show that each of the steps described in Sec. V has a polynomial runtime. Specifically, (i) creating the expanded graph (Sec. V-A) requires creating at most $\gamma^2(|\mathcal{V}||\mathcal{C}|+|\mathcal{E}|)$ nodes and at most $\gamma^2|\mathcal{V}||\mathcal{L}|$ edges, where $|\mathcal{V}|$ is the number of VNFs specifying the service and, given the service, is a constant. (ii) Finding the possible decisions (Sec. V-B) implies computing the shortest paths between any endpoint (i.e., vertex meeting the availability constraints) and any other node in the expanded graph, which, in the worst case, has complexity [31] $o(n^{2.3})$ with n being the number of nodes in the expanded graph. (iii) Computing the optimal CPU assignments (Sec. V-C) has cubic complexity [30] in the problem size; indeed, convex problems are routinely solved in embedded computing scenarios. ■

Finally, under reasonable assumptions about the homogeneity of the physical graph, we can prove that OKpi can actually return the optimal solution.

Property 3. *If all links and nodes have the same capabilities and cost, then the output of OKpi is optimal.*

Proof: There is only one point in the procedure we described where, in general scenarios, we may overlook the optimal solution. As remarked in Sec. V-B, finite γ values may cause a quantization-like error: in general scenarios, only for $\gamma \to \infty$, we could consider all possible ways to move from one node of the decision graph to another. In the special case of homogeneous links and nodes, however, no such different possibilities exist, hence, taking a finite value of γ is enough to consider all possible choices the system offers and to make an optimal decision. Note that restricting our attention to shortest paths on the expanded graph does not harm optimality, as adding hops implies consuming a higher (or equal at best) fraction of KPI targets and cannot decrease the cost. ■

VII. Numerical results

Here, we first focus on a small-scale scenario and a robot-based smart factory application (Sec. VII-A), and compare the performance of OKpi against the optimum obtained via brute force. Then we move to a large-scale scenario and real-world automotive application (Sec. VII-B), and we characterize how the quantity of traffic to serve and the maximum delay impact the decisions made by OKpi.

A. Small-scale scenario: comparison against the optimum

We consider the robot-based, smart factory application [32], whose service graph is depicted in Fig. 4. A room (hence, an endpoint) contains three robots, with different levels of reliability: $\eta(\text{robo1}) = 0.999999$, $\eta(\text{robo2}) = 0.99999$, and $\eta(\text{robo3}) = 0.9999$. Two of these three robots must be used to perform an operation, hence, run the robo-master and robo-slave VNFs. The communication between the two selected robots can take place through three types of PoAs, with different levels of reliability (micro-cell: 0.999999, pico-cell: 0.99999, femto-cell: 0.9994), and costs as reported in [33]. The offered traffic is 1 Mb/s per robot, as specified in [32].

Fig. 5 depicts the results when OKpi's resolution is set to $\gamma = 10$. A first aspect we are interested in is the relationship between the target KPIs and cost: as we can see from Fig. 5(left) and Fig. 5(center), a longer allowable delay results in a lower cost; conversely, a higher traffic load or a higher target reliability both result in higher costs. Intuitively, this is due to the fact that cheaper resources (e.g., robot 3) tend to have lower reliability and/or capacity, hence, it is impossible to use them when the KPI targets become very strict.

Interestingly, in both Fig. 5(left) and Fig. 5(center), OKpi matches the optimum in *all* cases. Indeed, as discussed in Sec. V-B, OKpi always matches the optimum if the resolution γ is high enough; in the small-scale scenario we consider for Fig. 5, $\gamma = 10$ is sufficient to this end.

Fig. 5(right) shows the effect of setting a lower resolution, namely, $\gamma = 3$. As we can see by comparing the left and center bars, a lower value of γ results in suboptimal, higher-cost decisions. Specifically, the difference is due to the fact that, when $\gamma = 3$, a higher-cost PoA is selected, namely, the pico-cell *in lieu* of the femto-cell. This happens because, for $\gamma = 3$, the edges corresponding to the femto-cell in the expanded graph have steepness $\left\lceil \gamma \frac{\log 0.9994}{\log 0.999} \right\rceil = 2$. Considering that (i) all other edges have steepness 1 and (ii) OKpi seeks for paths composed of three edges (same as the number of VNFs to place) with a total steepness not exceeding $\gamma = 3$, the edges corresponding to the femto-cell will never be selected, hence, the corresponding decision is never considered. In summary, as discussed in the previous sections, using a too-low γ made us overlook a feasible – and, in this case, optimal – solution.

B. Large-scale scenario: impact of traffic and delay

Our large-scale scenario depicts a urban environment where the vehicle collision avoidance system depicted in Fig. 1 [6] has to be provided. Based on a real-world road topology (see Fig. 7), a total of 9 intersections (hence, endpoints) are covered by a combination of PoAs, namely, macro-, micro- and pico-cells, whose coverage is shown in Fig. 7.

Different PoAs have different reliability, latency, and cost values, as reported in Tab. I. The front- and back-haul network topology is based on [34] and ITU standard [35], and includes MEC, aggregation and core nodes with features as summarized in Tab. I [36]. The service graph to deploy is represented in Fig. 2(left), the total service traffic is 1.5 Mb/s, and γ is set to 40 (higher values do not improve the performance).

Fig. 6(left) shows that, as one might expect, a shorter target delay results in higher costs. It is also interesting to observe the behavior of the intermediate curve, corresponding to $H(s) = 0.9999$: when the target delay is very short, its associated cost is almost the same as for $H(s) = 0.999999$

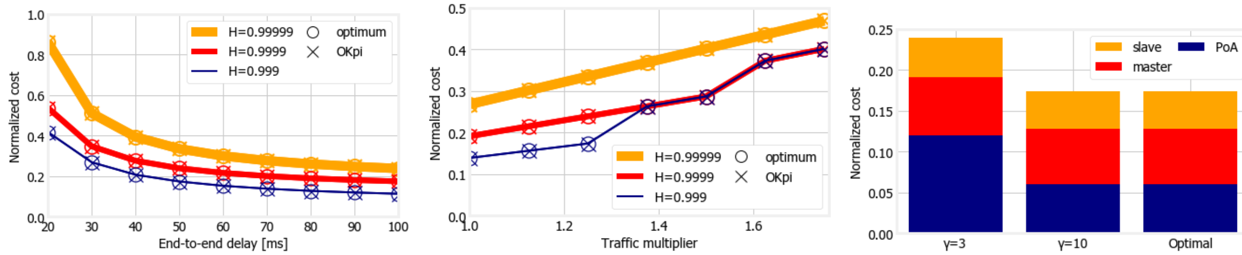

Fig. 5. Small-scale scenario: cost as a function of the maximum delay (left) and of the traffic load (center), for different values of target reliability; cost breakdown (right) when the target reliability is 0.999, the maximum delay is 50 ms, the traffic multiplier is 1, and γ varies.

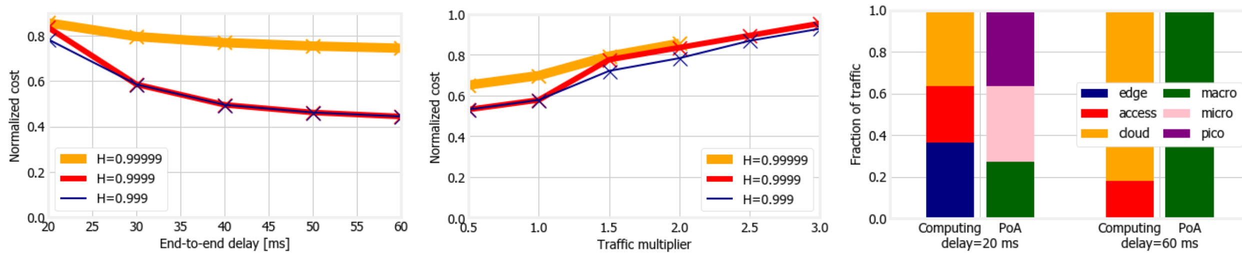

Fig. 6. Large-scale scenario: cost as a function of the maximum delay (left) and of the traffic load (center), for different values of target reliability; fraction of traffic (right) traversing different PoAs and computing nodes when the target reliability is 0.999, the traffic multiplier is 1, and the target delay varies.

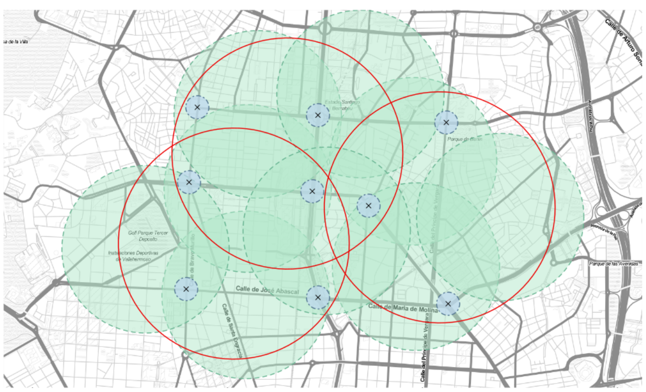

Fig. 7. Road topology used in the large-scale scenario. The nine crossings correspond to endpoints; red, green, and blue circles represent the coverage areas of macro-, micro- and pico-cells, respectively.

TABLE I

LARGE-SCALE SCENARIO: POINTS OF ACCESS AND COMPUTING NODES

Item	Reliability	Latency	Cost
Points of access			
macro-cell	0.99999999	6 ms	1.02 USD/Gbit
micro-cell	0.9999999	3 ms	2.31 USD/Gbit
pico-cell	0.999999	2 ms	3.80 USD/Gbit
Computing nodes			
cloud ring (Azure DataBox)	0.99999999	8 ms	2.23 USD/Gbit
aggregation ring (PowerEdge)	0.9999999	3 ms	5.23 USD/Gbit
MEC ring (small data center)	0.999999	1 ms	10.47 USD/Gbit

case; as the target delay increases, its cost drops to the same level as the $H(s) = 0.999$ case. This bespeaks the complexity of the decisions OKpi has to make, and their sometimes counter-intuitive effects.

In Fig. 6(center), the traffic load is multiplied by a factor ranging between 0.5 and 3. We can again observe that to a higher traffic corresponds a higher cost, even though the growth is less than linear, owing to the fixed costs described in Sec. IV-B. Also notice how the yellow curve in Fig. 6(center), corresponding to the highest reliability level, stops at a multiplier of 2: for higher traffic demands, the network capacity is insufficient to provide the service with the required reliability.

Fig. 6(right) shows which PoAs and computing nodes are selected for the minimum and maximum target delay values. Interestingly, in the presence of tight delay constraints, different PoAs and resources are all used (left bars). On the contrary, for the largest target delay, the cheapest options – cloud and macro-cells – are preferred.

VIII. CONCLUSION AND FUTURE WORK

We identified in the support for a limited set of KPIs one of the main shortcomings of present-day approaches to network slicing. To address this issue, we proposed OKpi, an efficient and effective solution strategy able to jointly make PoA selection, VNF placement, and data routing decisions, while natively accounting for all KPIs and for all resources, from the fog to the cloud. Importantly, OKpi draws on a novel methodology that blends together graph theory and optimization in a unique manner, and exhibits several desirable properties. Among these, we showed that OKpi has polynomial computational complexity and its performance can get arbitrarily close to the optimum. Our performance evaluation, carried out using two real-world scenarios, confirms that OKpi closely matches the optimum and consistently provides very good performance.

Future work will focus on enhancing the expanded graph presented in Sec. V-B by allowing edge steepness to take arbitrary real values, and on characterizing the competitive ratio of OKpi as a function of the parameter γ.

References

[1] ETSI. (2018) MEC in 5G networks. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf.

[2] P. Zhao and G. Dán, "A Benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds," *IEEE Transactions on Network and Service Management*, 2018.

[3] F. Malandrino and C.-F. Chiasserini, "Getting the Most Out of Your VNFs: Flexible Assignment of Service Priorities in 5G," in *IEEE WoWMoM*, 2019.

[4] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM*, 2017.

[5] "IMT Vision: Framework and overall objectives of the future development of IMT for 2020 and beyond," *ITU Recommendation M.2083-0*, 2015.

[6] Q.-H. Nguyen, M. Morold, K. David, and F. Dressler, "Adaptive Safety Context Information for Vulnerable Road Users with MEC Support," in *IEEE/IFIP WONS*, 2019.

[7] T. Lukovszki, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *Elsevier Journal of Parallel and Distributed Computing*, 2018.

[8] I. Cohen, G. Einziger, R. Friedman, and G. Scalosub, "Access Strategies for Network Caching," in *IEEE INFOCOM*, 2019.

[9] S. Dräxler, H. Karl, and Z. Á. Mann, "Jasper: Joint optimization of scaling, placement, and routing of virtual network services," *IEEE Transactions on Network and Service Management*, 2018.

[10] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks," in *IEEE INFOCOM*, 2019.

[11] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM*, 2018.

[12] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF Placement and Resource Allocation for the Support of Vertical Services in 5G Networks," *IEEE/ACM Transactions on Networking*, 2019.

[13] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *IEEE INFOCOM*, 2015.

[14] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *IEEE INFOCOM*, 2017.

[15] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic aware placement of interdependent NFV middleboxes," in *IEEE INFOCOM*, 2017.

[16] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, "Edge-CoCaCo: Toward joint optimization of computation, caching, and communication on edge cloud," *IEEE Wireless Communications*, 2018.

[17] T. Lukovszki, M. Rost, and S. Schmid, "It's a match!: Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Computer Communication Review*, 2016.

[18] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation, caching and forwarding in data-centric computing networks," in *ACM MobiHoc*, 2019.

[19] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, "NFVnice: Dynamic backpressure and scheduling for NFV service chains," in *ACM SIGCOMM*, 2017.

[20] C. Sun, J. Bi, Z. Zheng, H. Yu, and H. Hu, "NFP: Enabling network function parallelism in NFV," in *ACM SIGCOMM*, 2017.

[21] Y. Guo, A. L. Stolyar, and A. Walid, "Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud," *IEEE Transactions on Cloud Computing*, 2015.

[22] M. Bouet and V. Conan, "Mobile edge computing resources optimization: a geo-clustering approach," *IEEE Transactions on Network and Service Management*, 2018.

[23] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-TORCH: An automated NFV orchestration and monitoring solution," *IEEE Transactions on Network and Service Management*, 2018.

[24] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A Utility-Driven Multi-Queue Admission Control Solution for Network Slicing," in *IEEE INFOCOM*, 2019.

[25] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *IEEE INFOCOM*, 2019.

[26] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Slicing cell resources: The case of HTC and MTC coexistence," in *IEEE INFOCOM*, 2019.

[27] 5G PPP. Key Performance Indicators. https://5g-ppp.eu/kpis/.

[28] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," *ACM SIGCOMM computer communication review*.

[29] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," *IEEE/ACM Transactions on Networking*, 2007.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[31] R. Seidel, "On the all-pairs-shortest-path problem in unweighted undirected graphs," *Journal of computer and system sciences*, 1995.

[32] "5G-CORAL initial system design, use cases, and requirements," *5G-CORAL deliverable 1.1*, 2018, http://5g-coral.eu/wp-content/uploads/2018/04/D1.1_final7760.pdf.

[33] V. Nikolikja and T. Janevski, "A cost modeling of high-capacity LTE-Advanced and IEEE 802.11ac based heterogeneous networks, deployed in the 700 MHz, 2.6 GHz and 5 GHz bands," *Procedia Computer Science*, 2014.

[34] J. Martín-Pérez, L. Cominardi, C. J. Bernardos, A. De la Oliva, and A. Azcorra, "Modeling mobile edge computing deployments for low latency multimedia services," *IEEE Transactions on Broadcasting*, 2019.

[35] "Consideration on 5G transport network reference architecture and bandwidth requirements," *ITU Contribution 0462*, 2018.

[36] "Final system design and techno-economic analysis," *5G-TRANSFORMER deliverable D1.4*, 2019.