Computer Communications 157 (2020) 1–19

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Latency and energy-aware provisioning of network slices in cloud networks

Piotr Borylo [a,*], Massimo Tornatore [b], Piotr Jaglarz [a], Nashid Shahriar [c], Piotr Chołda [a], Raouf Boutaba [c]

[a] AGH University of Science and Technology, Department of Telecommunications, Poland
[b] Politecnico di Milano, Italy
[c] David R. Cheriton School of Computer Science, University of Waterloo, Canada

ARTICLE INFO

Keywords:
5G network slicing
Distributed computing
Multi-objective optimization
VNF placement

ABSTRACT

Modern network services are constantly increasing their requirements in terms of bandwidth, latency and cost efficiency. To satisfy these requirements, the concept of *network slicing* has been introduced in the context of next-generation 5G networks. However, to successfully provision resources to slices, a complex optimization problem must be addressed to allocate resources over a cloud network, i.e., a distributed computing infrastructure interconnected through high-capacity network links.

In this study, we propose two new latency and energy-aware optimization models for provisioning 5G slices in cloud networks comprising both distributed computing and network resources. The proposed approaches differ from other existing solutions since we conduct our studies with respect to the end-to-end latency. Relevant models of latency and energy consumption are proposed based on a comprehensive review of the state-of-the-art. To effectively solve those optimization problems, a configurable heuristic is also proposed and investigated over different network topologies. Performance of the proposed heuristic is compared against near-optimal solutions. Moreover, we assess the importance of matching between resource provisioning algorithms and architectural assumptions related to 5G network slices and a proper problem modeling.

1. Introduction

Global mobile data traffic is expected to increase seven-fold between 2017 and 2022, reaching 77.5 exabytes per month by 2022 [1]. The increasing number of end users is not the only reason for this phenomenon, although end users are also increasing their bandwidth demands to get access to several new bandwidth-hungry cloud applications. However, increasing traffic volume is not the only challenge for future cloud networks. Other challenges arise from the heterogeneous and strict requirements imposed by new network services. For example, in 5G networks, considered in this work as a preeminent form of cloud networks, high bandwidth, low latency and energy-efficiency services are critical. To meet these requirements, *network slicing* has been proposed and defined as generating a logical network to provide specific network capabilities and network characteristics in 5G architectures [2]. This definition can be further generalized to cloud networks, where a 'slice' is defined as a set of virtual resources (either network or computing) provisioned for the purpose of a particular request [3]. Slicing is not achievable in traditional architectures characterized by limited control capabilities, since it calls for a more sophisticated control plane and distributed computing resources.

Software defined networking (SDN), together with cloud orchestration, ensures programmable and automated control with a global view over the infrastructure composed of both network and computing resources. Furthermore, *network function virtualization* (NFV) enables scalable deployment of network services [3] through interconnected *virtual network functions* (VNFs) that form *virtual network function chains* (VNFCs) to satisfy service requirements. VNFs can be located not only in large centralized *data centers* (DCs), but also in smaller facilities distributed over edge locations, including base stations or micro data centers in operators' central offices. Such a distributed computing infrastructure is denoted as *edge* or *fog computing*, depending on context. All of those technologies and concepts enable slicing in cloud networks.

The main contributions of this paper can be summarized as follows.

(a) We formulate two multi-objective optimization problems for provisioning 5G slices in a cloud network. The related optimization goals reflect requirements faced by 5G network slices which must ensure sufficiently-low latency while minimizing resource utilization and energy consumption. Per-service provisioning is assumed as a slice reserved for each consecutive request. The first problem assumes only a single computing location, whereas the second one considers a more distributed scenario with several cloud and edge computing facilities.

[*] Corresponding author.
E-mail address: borylo@agh.edu.pl (P. Borylo).

https://doi.org/10.1016/j.comcom.2020.03.050
Received 13 October 2019; Received in revised form 16 March 2020; Accepted 31 March 2020
Available online 4 April 2020

(b) Due to the combinatorial characteristics of the two models and the resulting computational complexity, we also propose a time-efficient heuristic to obtain satisfactory sub-optimal results. Note that our heuristic shall also account for network limitations in terms of capacity. The proposed algorithm is based on the shortest path algorithm performed on an auxiliary graph with numerous novel aspects introduced. These are additional stages of the heuristic and sub-algorithms being parts of those particular stages.

(c) Energy and latency models are provided for both computing infrastructure and network resources. Latency and energy consumption are studied under assumption that end-to-end (E2E) latency originates simultaneously from both **network** (transmission) and **computing** domains.

(d) Numerous experiments are conducted to provide a comprehensive assessment. First of all, different configurations of the proposed heuristic are investigated in different network topologies to verify the ability of adjustment to the latency requirements of 5G network slices. Second, by comparing the two optimization models, we assess the importance of matching between resource provisioning algorithms and architectural assumptions related to 5G network slices. Finally, we compare the performance of the proposed heuristic against near-optimal solutions. For all of the experiments, realistic assumptions and configuration values are ensured based on our comprehensive review of available literature.

The paper is organized as follows. In Section 2, we present the related work and current state-of-the-art. In Section 3, the two optimization models are introduced. In Section 4, a heuristic is proposed to effectively obtain a sub-optimal solution for realistic problem instances. In Section 5, we describe the research environment, methodology, and discuss the assumptions and values of computation parameters. Section 5.2 presents the results along with their comprehensive analysis. Finally, Section 6 concludes the paper.

2. Related work

One advantage of distributed computing is the ability to limit latency during service provisioning by directing computing tasks and VNFs to edge facilities located closely to end users, instead of locating them in distant DCs. Such an approach is referred in the literature as *computation offloading* [4]. However, it comes at the cost of increased energy consumption and required investments in an edge infrastructure [5]. That is why joint optimization model of communication and computation resources assuming limited energy and sensitive latency was proposed in [4] while some online heuristics are presented in [5]. Those works show how to balance the trade-off between computation in more efficient, yet farther, DCs and closely located edge facilities. Similarly, energy vs. latency trade-off has been considered in multilayer data processing stacks comprising end devices, edge/fog layers and cloud data centers [6], all of those important aspects are also addressed in our work.

In our work, we also consider a hybrid infrastructure composed of cloud DCs and distributed edge resources interconnected through the SDN network. Our research assumes that E2E latency originates from both domains: network (transmission) and computing. Our approach is an important improvement in comparison to the current state-of-the-art studies, especially when considered under assumptions related to 5G infrastructure [3]. For example, in a very recent work [7], E2E latency comprising both infrastructure domains was pointed to be a crucial and under-investigated research area. The authors proposed E2E analytical queue-based latency model applying packet level considerations and validating the model by simulations. In [3], a framework was proposed to place VNFs in an 5G network infrastructure composed of both edge and core cloud servers, while respecting requirements of 5G slices. The authors also provide convincing arguments that no previous work was adequate for the VNF placement problem for 5G network slices. However, their aim is to minimize throughput degradation caused by VNF consolidation without considering energy and latency aspects. Contrary to the presented theoretical solutions, experimental aspects are addressed in [8]. The paper presents details of implementation experiences when deploying streaming and augmented reality services over 5G network slices utilizing extended open-source software solutions.

SDN with all of its capabilities (e.g., a central controller and programmability and its flexibility in treating various flows in a differentiated way) creates a great opportunity to introduce energy- and latency-awareness into the optimization process [9]. Reductions in terms of energy consumption may significantly help cut down operational costs and limit a negative impact of IT infrastructures on the natural environment by decreasing carbon dioxide emission [10]. Similarly, improvements in terms of latency are especially crucial for modern services offered in 5G networks utilizing the NFV approach [7].

Furthermore, SDN controllers can be easily integrated with cloud and edge orchestrators in order to provision a complete E2E service. This aspect of different orchestration capabilities is also tackled in our work by considering two different optimization models. It is extremely important that our proposed modeling is adequate to the ecosystem and fundamental assumptions. For example, deploying computing jobs in various locations should be enabled by the optimization model. However, such an approach to distributed computing requires integration between controllers handling different facilities and a WAN SDN controller.

Other optimization models have already been proposed to solve problems related to the one considered in this work. For example, in [11], the authors formulated a problem which aims to deploy VNFs in nodes along the path that minimizes latency comprising network and processing factors. An auxiliary graph is defined to ensure a proper order of functions in the chain and the so-called Resource Constrained Shortest Path problem is formulated. However, energy aspects are not considered. The latency aspect is differently addressed in [12], where, cost is minimized while latency is considered as a constraint. The cost comprises physical servers, links, and licensing for virtual servers. Additionally, service differentiation is ensured and failures are considered. However, the problem is solved in two stages and the some important aspects are moved to the post-processing phase.

Energy consumption is an important issue in the context of distributed computing and it has been studied both with and without taking into account latency constraints. The later approach has been taken by authors in [13], where the VM placement problem is converted to a routing problem and computing resources are transferred to the network domain. Only intra-DC networks are considered in this work. On the other hand, latency is considered as a constraint in [14], yet delays originate solely from packet transfers in DC-internal networks. In [14], energy consumption in the combined network and computing infrastructure is optimized when co-located virtual machines communicate with each other. An additional contribution of this work is related to the fact that validation is conducted utilizing real-life traffic traces (to/from Wikipedia, Yahoo!, and IBM services). The approach proposed in [14] is based on correlations between traffic generated by different VMs, while the intermediate aim is to consolidate the load and VM deployments. A joint VNF placement and routing problem which also minimizes energy under latency constraints is proposed in [15]. Energy consumption and latency come from both network and computing domains. Moreover, distinctive features regarding the utilization of network topology properties, as betweenness centrality and presence of blocking islands, are taken into account. Despite numerous similarities, the crucial difference between the model proposed in [15] and in our work lies in the fact that we consider latency as a component of the optimization goal instead of considering latency as a constraint.

Other works focus only on latency, while neglecting energy aspects. Interesting models of network functions, NFC requests, and corresponding heuristics are proposed in [16]. The optimization aim comprises latency originating in the network and service provisioning domains. Latency is also the main indicator in [17] which is one of the first that considers the VNF placement problems in multi-layer infrastructures. Latency originates from traffic grooming performed by the network nodes in the upper layer of the model. The study is conducted in a metro-like topology which is most suitable for 5G slices deployment. Some other works, even through they consider neither latency nor energy issues, are mentioned in the following due to their contribution in the optimization field strongly connected to our work. One recent example is [18], where computing and network resources are reserved for incoming demands to deploy VNFCs. The problem is constructed based on the expanded network and is solved with the use of four different heuristics. Ref. [19] is a second example where overall CPU and bandwidth utilization is the optimization factor. An on-demand VNFC provisioning problem for a 5G environment is modeled using linear programming and solved by novel adaptive deep Q-learning based approach being significantly advantageous over reference solutions. On the other hand, Elastic Optical Network slicing problem for 5G networks under strict latency requirements has been studied in [20].

Finally, some studies focus solely on task scheduling in distributed systems, without considering network aspects. Despite this important difference from our assumptions those works remain important as such a task scheduling is a sub-problem of issues addressed in our work. In [21], the authors aim at minimizing energy consumption under deadline constraints. The main contribution are eight heuristics to solve the problem. Those proposals were thoroughly assessed solely in the computing domain. The network domain is simplified to the virtual connection between the cloud and fog infrastructures in [22]. The aim is to reduce energy consumption and latency in the fog computing environment. The authors proposed two algorithms: (1) machine learning-based algorithm that minimize latency in physical layer analyzing user behavior; and (2) intelligent task offloading between fog nodes with a limited battery and processing power. Another approach was proposed in [23], where the authors focus on the costs related to sharing of computing resources between different VNFs: the upscaling cost and the context switching cost. The latter is especially valuable when modeling impact of computing resource utilization on the latency. The aim of the proposed optimization model is to minimize a number of active computing nodes under latency constraints obligatory to satisfy the demands. A very recent study [24] formulates the NP-hard optimization problem modeling deployment of VNFCs with the aim to reduce the overall cost comprising energy and distance (latency) aspects. Two solving methods were considered. One algorithm is based on the Markov Approximation (MA) and is further improved by the novel approach combining MA with the matching approach which solves the problem iteratively.

To sum up, to the best of our knowledge, it is the first work that jointly optimize energy and latency when solving multi-objective optimization problem for provisioning of 5G slices in a cloud network. The novelty is further extended as energy and latency models are provided as well as a time-efficient heuristic is proposed.

3. Problem formulation

In this section, we formulate two novel optimization models. Both models use node-link notation, but they formally differ as one of the models (the less restrictive one) ignores two additional constraints enforced in the more restrictive model. Therefore, to avoid repetitions, we present both models using a unified notation, and report only additional comments regarding the constraints responsible for the difference. As explained in Section 1, the rationale is to investigate scenarios in which resource provisioning methods are not directly adjusted to the 5G architecture and concept of network slicing. The more restrictive model enforces that all computing jobs must be performed in a single computing node, and, as such, is better suited for traditional applications as grid computing. The second model removes that limitation, hence it lends itself to more modern applications as those envisioned in customized 5G network slices. The proposed models significantly extend the model introduced in our previous work [25].

The optimization models capture both topology planning and flow routing problems. Moreover, computing resources are assumed to be interconnected through a programmable SDN network creating a cloud network. The demand modeling is adequately extended to properly model also processing in the computing domain and, as a result, a set of virtual resources in each domain is provisioned to satisfy each request. This slice definition is a natural extension of the slice definition in a 5G architecture [2] when applied in cloud networks. Furthermore, latency issues are modeled, assuming that delay is introduced by both computing and networking elements in the following way: (a) with a linealized delay function based on queuing traffic in link interfaces; (b) with a latency that inversely proportional to the assigned resources in computing elements. Finally, energy consumption of the cloud network is modeled by a non-proportional function of network link load and computing resource consumption. To model both latency and energy, we introduce binary variables into the multi-objective optimization problem, that make the optimization task hard, or even impossible, to solve on large networks. As a consequence, some heuristics are also proposed to effectively find a sub-optimal solutions.

Note that the following optimization models solve a flow assignment problem. To do that, a node-link approach was assumed, since it has been shown to be more practical when utilizing the optimization results to configure network devices [26]. More specifically, the optimization results must be transformed into paths together with their respective output interfaces, and then transferred to the SDN controller which updates flow tables in the switches. Using the node-link notation presented in [26] this technical issue is easier to handle.

The used notation, meaning of the symbols, etc., related to the formulated optimization problem are sketched in Table 1.

The constraints defining the feasible solutions of our optimization problem are defined below. For clarity, we define various groups of the constraints. This way, we start with the **network constraints**. The are given below:

$$\sum_e A_{ev} x_{ed} - \sum_e B_{ev} x_{ed} = \begin{cases} H_d & \text{if } v = S_d \\ 0 & \text{if } v \neq S_d, v \neq T_d \\ -H_d & \text{if } v = T_d \end{cases}$$

$$v = 1, 2, \ldots, V \quad d = 1, 2, \ldots, D \tag{1}$$

$$x_e = \sum_d x_{ed} \qquad e = 1, 2, \ldots, E \tag{2}$$

$$x_e \le C_e \tau_e \qquad e = 1, 2, \ldots, E \tag{3}$$

$$x_{ed} \le M \epsilon_{ed} \qquad e = 1, 2, \ldots, E \quad d = 1, 2, \ldots, D \tag{4}$$

$$\sum_e A_{ev} \epsilon_{ed} \le 1 \qquad v = 1, 2, \ldots, V \quad d = 1, 2, \ldots, D \tag{5}$$

Next, **computing constraints** are defined as follows:

$$\sum_r \sum_d j_{vrd} \le p_v \qquad v = 1, 2, \ldots, V \tag{6}$$

$$j_{vrd} \le M \sum_e B_{ev} x_{ed} \qquad v = 1, 2, \ldots, V \quad v \neq S_d \quad r = 1, 2, \ldots, R$$

$$d = 1, 2, \ldots, D \tag{7}$$

$$G_r^{\min} \le g_{rd} \le G_r^{\max} \qquad r = 1, 2, \ldots, R \quad d = 1, 2, \ldots, D \tag{8}$$

$$\sum_v o_{vrd} = 1 \qquad r = 1, 2, \ldots, R \quad d = 1, 2, \ldots, D \tag{9}$$

$$o_{vrd} j_{vrd} \ge o_{vrd} g_{rd} \qquad v = 1, 2, \ldots, V \quad r = 1, 2, \ldots, R \quad d = 1, 2, \ldots, D \tag{10}$$

Table 1
Notation used in relation to the optimization task.

Index	**Meaning of the index**	**Constant**	**Meaning of the constant**
$v = 1, 2, \dots, V$	Network nodes	Γ_{ek}	The intercept of the kth linear segment approximating a convex function of link delay vs. link traffic load on link e
$d = 1, 2, \dots, D$	Demands (flows) between nodes pairs (note that a demand from v_1 to v_2 is not the same as a demand from v_2 to v_1 and, in general, we can even distinguish among different demands defined for the same pair of nodes)	Ξ_e	fixed unit cost of energy for link e (we assume a linear dependence between traffic volume on link e and a resulting energy cost)
$e = 1, 2, \dots, E$	Network links (corresponding to interfaces that can be switched on or off)	Ψ_v	fixed unit cost of energy for computing resources associated with node v (we assume a linear dependence between computing load and a resulting energy cost)
$k = 1, 2, \dots, K$	Linear segments approximating a convex function representing link delay vs. traffic volume	Δ_e	Energy cost of switching on link e
$r = 1, 2, \dots, R$	Computing services comprised in each demand	Λ_v	Energy cost of switching on computing resources associated with node v
Constant	**Meaning of the constant**	F_n	Normalization coefficient scaling overall energy consumption of the infrastructure
M	Big-M (a sufficiently large number)	F_β	Normalization coefficient scaling overall delay violation for all the demands satisfied in the network
A_{ev}	= 1 if link e starts in node v; 0, otherwise	Variable	Meaning of the variable
B_{ev}	= 1 if link e finishes in node v; 0, otherwise	$x_{ed} \geq 0$	**Continuous**: volume of a flow for demand d on link e
C_e	Capacity of link e	x_e	**Continuous**: total capacity allocated on link e to flows
S_d	Source node of demand d	$g_{rd} \geq 0$	**Continuous**: processing power consumed by service r of demand d
T_d	Destination node of demand d	$j_{vrd} \geq 0$	**Continuous**: processing power consumed in computing node v satisfying service r of demand d
H_d	Predicted volume of the flow demand d	ζ_e	**Continuous**: delay experienced on link e
$\Omega_d^{\max}$	Latency requirements of demand d (if a particular demand can only be satisfied with higher latency than the required one, a latency violation occurs)	θ_{vrd}	**Continuous**: delay introduced by processing service r of demand d in computing node v
P_v	Processing power available in a computing node associated with network node v (the same indices apply to computing and network nodes, as computing resources can be associated with any network node; for the sake of simplicity, the computing node associated with network node v will be further referred to as computing node v)	$\beta_d \geq 0$	**Continuous**: overall (introduced by both network transmission and computing processing) latency experienced by demand d
Π_v	= 1 if computing node associated with network node v is a DC (a special case of a computing node that cannot be turned off and has virtually unlimited computing resources); 0, otherwise	$\beta_d^{\text{violation}} \geq 0$	**Continuous**: violation of latency requirements of demand d
$G_r^{\min}$	Minimum processing power that has to be assigned in a computing node to handle service r	n	**Continuous**: total amount of energy used in the cloud network, comprising network and computing resources
$G_r^{\max}$	Maximum processing power that has to be assigned in a computing node to handle service r	ϵ_{ed}	**Binary**: = 1 if link e is a part of a path satisfying demand d; = 0, otherwise
$\Theta_r^{\min}$	Minimum latency with which computing service r can be satisfied	o_{vrd}	**Binary**: = 1 if computing service r of demand d is realized in computing node v; = 0, otherwise
$\Theta_r^{\max}$	Maximum latency with which computing service r can be satisfied	o_{vd}	**Binary**: = 1 if any computing service of demand d is realized in computing node v; = 0, otherwise
Φ_{ek}	The slope of the kth linear segment approximating a convex function of link delay vs. link traffic load on link e	τ_e ρ_v	**Binary**: = 1 if link e should be switched (to transmit any data); = 0, otherwise **Binary**: = 1 if computing node v should be switched on (to handle any service); = 0, otherwise

The two **additional computing constraints valid solely for the first optimization model** read as given below:

$$o_{vd} \geq o_{vrd} \qquad v = 1, 2, \dots, V \quad d = 1, 2, \dots, D \quad r = 1, 2, \dots, R \tag{11}$$

$$\sum_r o_{vrd} = R o_{vd} \qquad v = 1, 2, \dots, V \quad d = 1, 2, \dots, D \tag{12}$$

The **latency constraints** are defined as follows:

$$\zeta_e \geq \Phi_{ek} x_e + \Gamma_{ek} \qquad e = 1, 2, \dots, E \quad k = 1, 2, \dots, K \tag{13}$$

$$\theta_{vrd} = \frac{\Theta_r^{\max} - \Theta_r^{\min}}{G_r^{\min} - G_r^{\max}} j_{vrd} + \frac{\Theta_r^{\min} G_r^{\min} - \Theta_r^{\max} G_r^{\max}}{G_r^{\min} - G_r^{\max}} o_{vrd}$$

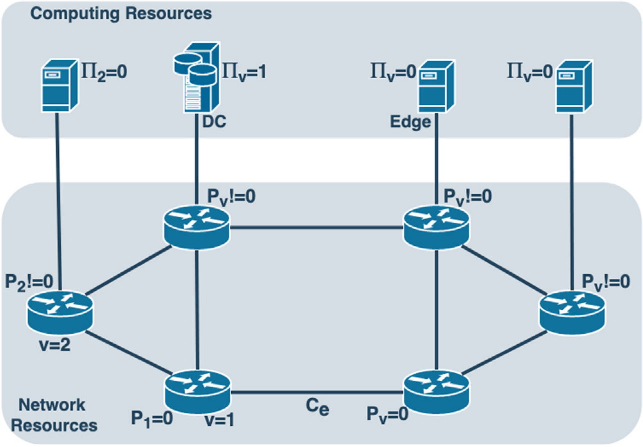

Fig. 1. Illustration of the architecture.

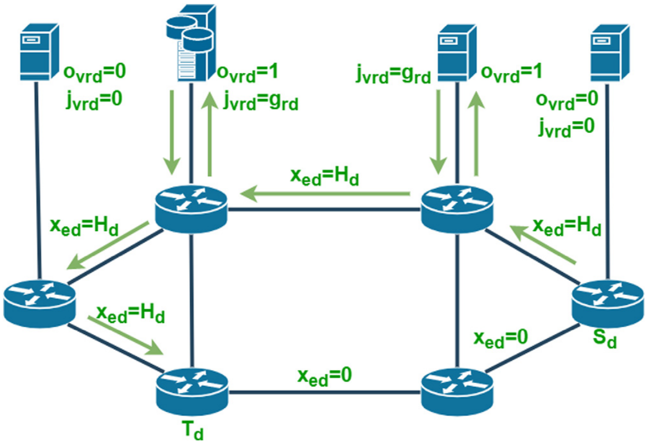

Fig. 2. Example of demand provisioning.

$$v = 1,2,\dots,V \quad r = 1,2,\dots,R \quad d = 1,2,\dots,D \tag{14}$$

$$\beta_d = \sum_{vr} \theta_{vrd} + \sum_{e} \epsilon_{ed}\zeta_e \qquad d = 1,2,\dots,D \tag{15}$$

$$\beta_d^{\text{violation}} \geq \beta_d - \Omega_d^{\max} \qquad d = 1,2,\dots,D \tag{16}$$

Finally, there are also **energy constraints** given:

$$\sum_d \sum_r j_{vrd} + \Pi_v \leq M\rho_v \qquad v = 1,2,\dots,V \tag{17}$$

$$n = \sum_e (\Delta_e\tau_e + \Xi_e x_e) + \sum_v (\Lambda_v\rho_v + \Psi_v \sum_d \sum_r j_{vrd}) \tag{18}$$

Finally, the **optimization goal function** is given as:

$$\textbf{minimize} \quad \frac{1}{F_n} n + \frac{1}{F_\beta} \sum_d \beta_d^{\text{violation}} \tag{19}$$

Supplementary to the formal problem definition, Fig. 1 illustrates the architecture, indexing of network nodes, different types of computing resources and link's capacity.

The cloud network is composed of computing and networking resources, modeled as follows. The network part is a unidirectional graph with V nodes and E edges with capacity limitations applicable to the links (C_e). The computing resources can be associated with any network node and are provisioned in two forms: Data Centers (Π_v = 1, with virtually unlimited capacity) and edge resources limited in terms of capacity (P_v). Each demand d originates in its source node (S_d) and terminates in its destination node (T_d). To satisfy a demand, sufficient network capacity should be reserved to carry the demand's volume (H_d) between S_d and T_d. Additionally, R computing services are associated with each demand d. Thus, to satisfy the demand in the computing domain, sufficient computing resources (range between $G_r^{\min}$ and $G_r^{\max}$) must be provisioned to process each computing service $r = 1,2,\dots,R$. Processing can be conducted in computing resources associated with any network node traversed by the path of the demand (including source and destination nodes).

Each demand d, has a latency threshold ($\Omega_d^{\max}$) that must be met not to violate the requirements. Both network transmission and processing in computing nodes introduce additional latency according to the models explained below.

To make notation of our optimization model readable, we grouped constraints into network, computing, latency, and energy categories. The model will be explained according to those categories.

Network constraints model resource allocation to satisfy demands in the transmission dimension. Eq. (1) represents basic constraints, responsible for enforcing flow conservation in intermediate nodes and satisfying demands in the source and destination nodes. Eq. (2) is used to determine the sum of all traffic flows in the network. Eq. (3) ensures that links' capacity is not exceeded by the demands, while Eq. (4) sets a binary variable indicating if link e is traversed by the path satisfying demand d. The fact that flows cannot be bifurcated is ensured by Eq. (5).

Computing constraints model how the computing part of each demand is satisfied. Eq. (6) verifies that computing resources in each computing node are enough to satisfy all the demands assigned to be processed in the node. Eq. (7) ensures that computing resources associated with network node v can be consumed only if a path satisfying demand d traverses node v. To satisfy any demand, each service r comprised by that demand must be also satisfied by assigning computing resources within the range specific for that service (Eq. (8)). As the service is the smallest part of the computing demand it has to be entirely processed in a single computing node. This property is ensured by Eq. (9). Eq. (10) ensures that computing node v satisfying service r of demand d (i.e., $o_{vrd} = 1$) reserves a proper amount of computing resources (namely, equation $j_{vrd} = g_{rd}$ is enforced). Note that we multiply binary and continuous variables; therefore, linearization is performed (as this can be done with well-known linearization techniques, we skip the details here).

Additional computing constraints in Eqs. (11) and (12) are valid only for the first optimization model. The former equation is auxiliary and ensures that realizing any of services r of demand d in node v is equivalent to realizing demand d in node v. The latter equation enforces that all services $r = 1,2,\dots,R$ of the same demand d are realized in the same computing node v. Those constraints are ignored in case of the second optimization model which assumes that services $r = 1,2,\dots,R$ of demand d can be processed in different computing nodes. Fig. 2 illustrates how demand can be satisfied in the modeled architecture. The path between source and destination is presented pointing that requested bandwidth is assigned solely on consecutive links of that path. Furthermore, computing resources are assigned in two first network nodes traversed by the path. The former one is an edge node while the latter is DC node.

Latency constraints reflect the fact that latency is modeled as originating independently from two different sources: transmission through the network and processing in computing nodes. Regarding the network delay we assume delays experienced by traffic is dependent on traffic load on the links according to a monotonously increasing function. We decided to use a model based on the M/M/1 queue to obtain delay from traffic load on a link, with respect to its capacity, as described in Chapter 4.3.2 of [26]. According to this model, the delay function can be approximated with a series of linear segments (the so-called 'piecewise linear approximation'). Since the delay function is a convex, increasing function, and we aim at minimizing the delay values in the optimization goal, the convexity of the problem is maintained.

Therefore, we can use a standard linearization procedure presented in the mentioned chapter (eqs. (4.3.11)-(4.3.12) and (7.1.10)-(7.1.11) in [26]) without necessity of applying non-continuous variables. The linearization is performed according to Eq. (13).

Processing latency is separately modeled by Eq. (14), following [27]. We assume that computing resources assigned to satisfy service r may vary between $G_r^{\min}$ and $G_r^{\max}$, and thus, it is reasonable to assume that the more resources within range $\left[G_r^{\min}, G_r^{\max}\right]$ are assigned, the lower latency will be introduced. Therefore, the computing latency varies in corresponding range between $\Theta_r^{\min}$ and $\Theta_r^{\max}$. In Eq. (14) variable j_{vrd} ensures this linear relationship between assigned computing resources and resulting latency [27]. Additionally, binary variable o_{vrd} introduces minimum latency ($\Theta_r^{\min}$) even if maximum applicable computing resources are assigned (i.e., when $j_{vrd} = G_r^{\max}$).

Eq. (15) calculates the overall latency experienced by demand d as a sum of the two previously explained contributions. Note that also in this case we multiply binary and continuous variables, so further linearization is performed. Finally, Eq. (16) finds the latency violation for each demand. The value $\sum_d \beta_d^{\text{violation}}$ expressing the summarized delay violation of all demands satisfied by the infrastructure is directly applied in the optimization goal.

Similar to latency ones, energy constraints follow an energy consumption model where energy consumption is due to both: network links and computing nodes.

As for the network energy cost, we model energy consumption using a non-proportional function comprising two components. The first one reflects the baseline energy costs to keep a link active ($\Delta_e \tau_e$), while the second one represents variable energy cost of transferring specific amounts of traffic through the link ($\Xi_e x_e$). The fact that the energy cost of switching link on is taken into account prevents us from using a pure continuous linear modeling of this aspect. We decided to use the mentioned energy cost model as:

- the presented model is simple, yet realistic, since it is frequently reported as being appropriate for current SDN hardware [28] and distributed systems [29];
- a simpler model, where energy cost characteristics is purely proportional, does not reflect the behavior of current network hardware;
- other, more complex, energy cost characteristics can also be considered: if the traffic-dependent energy consumption increases in a convex way, then we could linearize it as we did to model delays. If it increases in a concave way; we should linearize it with the use of binary variables, which introduce additional hindrances to the optimization and further increase the complexity of the problem.

Eq. (3) ensures that each link transmitting any amount of traffic is switched on. As for energy consumption of computing infrastructure, we model it in different ways for DCs and edge resources. The energy consumption model for an edge node is analogous to the network link energy model with the initial cost of switching on a node ($\Lambda_v \rho_v$) and varying consumption proportional to the amount of utilized computing resources ($\Psi_v \sum_d \sum_r j_{vrd}$). In case of DCs, we neglect the constant part of equation and take into account only the component linearly dependent from the assigned computing power. Such an approach is reasonable, as edge resources are usually composed of small processing units, like for example, single server dedicated to offer its resources for computing tasks. Such a server can be easily switched off or hibernated when it is not occupied by any particular computing demand. On the other hand, DCs are large-scale facilities with virtually unlimited computing resources being provided to numerous customers and for different purposes. Thus, it would be non-realistic to assume switching-off whole DC while the contribution of any single demand to the constant part is negligible. However, linear part remains relevant as any additional load increases energy consumption. This assumption is modeled in Eq. (17), used to enforce that variable ρ_v (indicating that the computing resources in node v are active) is set to 1. Finally, Eq. (18) ensures that overall energy consumption is a sum of energy consumed in the network (first summation) and computing parts (second summation). One must note that variable n expressing overall energy consumption is directly applied to the optimization goal.

In our problem, we use a multi-objective optimization. It is a well-known problem that there is no single universal method for dealing with multiple criteria simultaneously. Out of the viable options described in [30] (weighted sum method, weighted min–max method, weighted global criterion method, lexicographic method, bounded objective function method, and goal programming) we select the weighted sum method due to its simplicity and an intuitive character, and since other methods have more disadvantages from the viewpoint of our particular optimization problem.

Two criteria are included in the optimization goal with different normalization coefficients F_i, where i represents n (overall energy consumption of infrastructure) and $\sum_d \beta_d^{\text{violation}}$ (summary delay violation for all the demands satisfied in the network). As a result, the optimization goal takes the form expressed by Eq. (19). While minimizing overall energy consumption is intuitive, the latency part requires some kind of clarification. The most straightforward approach is to consider latency requirements as a constraint in the model. Such an approach may lead to infeasibilities in situations where providing service in a best effort is more convenient. On the other hand, the optimization goal can be formulated to directly minimize the latency. This approach may lead to unnecessary energy and resource consumption in scenarios when sufficient latency for particular demands may be ensured at lower cost. Therefore, minimizing overall violation of latency requirements brings the most expected results which are to ensure satisfactory quality of service without any unnecessary costs in terms of energy or resources.

4. Heuristic

The problem formulated in Section 3 is $\mathcal{NP}$-complete due to the numerous binary variables modeling non-linear dependencies in multi-objective optimization problem, and some heuristics should be proposed to effectively find a sub-optimal solution.

The formulated optimization models address static optimization scenarios only. Thus, heuristic proposed in this section follow analogous assumptions and constraints. The aim is to allocate network and computing resources assigning the complete set of demands in the network. Demands are listed according to the decreasing order of their network demand volumes (H_d). The rationale is to handle the most network demanding flows first.

The proposed heuristic is inspired by the concept of shortest paths with some modifications of weights assigned to the edges. Thus, whenever stated that the shortest path is find or established, it means that weights are respected by an algorithm. The default weight for each link is equal to 100. In case when there are many equivalent paths the algorithms arbitrarily select one of them.

To make the description more compact, in the following we refer computing resources of a network node as ‘resources in the node’ and network nodes associated with a DC are denoted as ‘DC nodes’, while all other nodes are denoted as ‘edge nodes’.

General heuristic workflow. The schema our heuristic follows in relation to each demand d is given below (the details of each step are explained in the consecutive paragraphs):

1. create an auxiliary graph on the basis of the network topology,
2. modify weights assigned to the network edges in a way that fully reflects the corresponding optimization problem and aim,
3. find the path between the demand source (S_d) and destination (T_d) using one of the three proposed modes with two of them introducing some preference for paths traversing data centers,

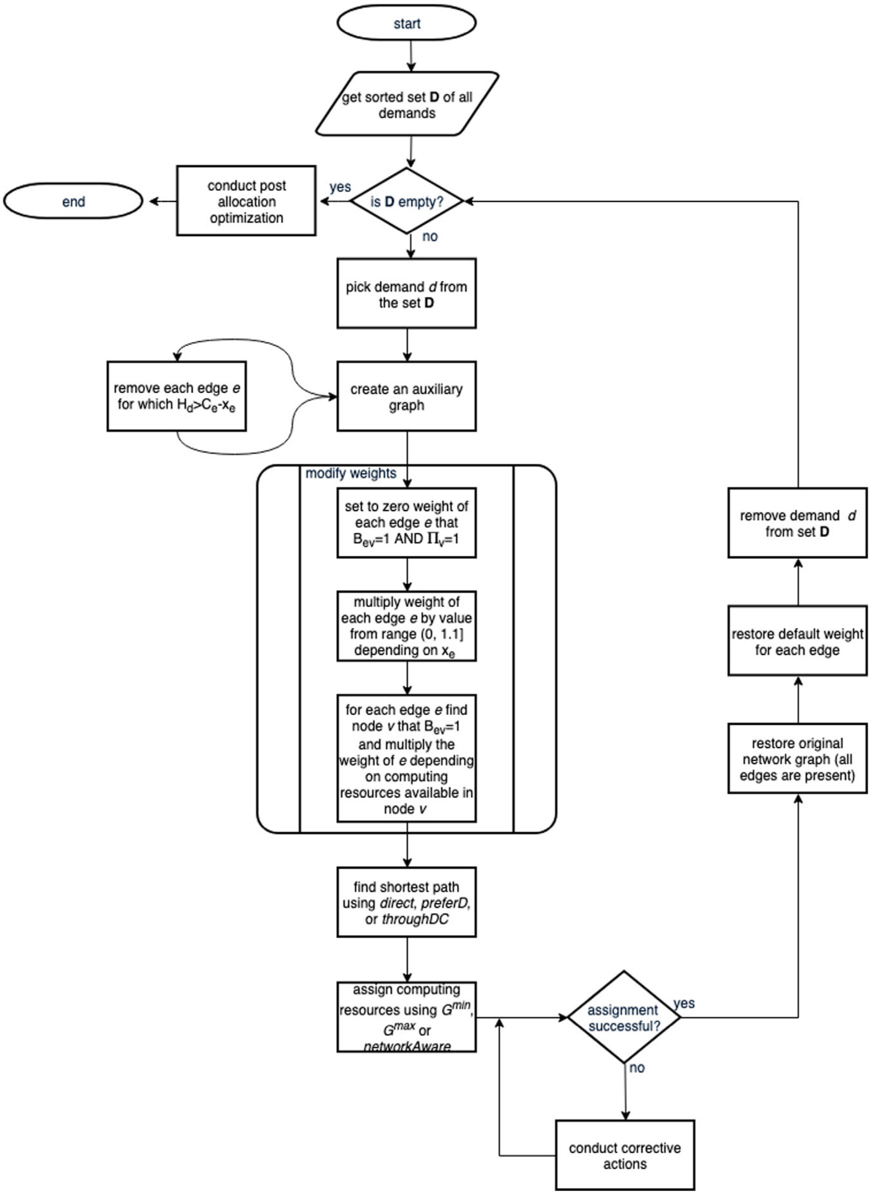
start
get sorted set **D** of all demands
is **D** empty?
yes
no
conduct post allocation optimization
end
pick demand *d* from the set **D**
create an auxiliary graph
remove each edge *e* for which H_d>C_e-x_e
modify weights
set to zero weight of each edge *e* that B_{ev}=1 AND Π_v=1
multiply weight of each edge *e* by value from range (0, 1.1] depending on x_e
for each edge *e* find node *v* that B_{ev}=1 and multiply the weight of *e* depending on computing resources available in node *v*
find shortest path using *direct*, *preferD*, or *throughDC*
assign computing resources using G^{min}, G^{max} or *networkAware*
assignment successful?
yes
no
conduct corrective actions
restore original network graph (all edges are present)
restore default weight for each edge
remove demand *d* from set **D**

Fig. 3. Flowchart diagram of the proposed heuristic.

4. assign computing resources to handle computing tasks of the demand using one of the three proposed sub-algorithms, where especially one is designed to balance the trade-off between resource/energy consumption and achieved latency,
5. if computing resources are not sufficient to handle the demand on the shortest path, perform corrective actions taking advantage of virtually unlimited resources in DC and possibly allow for partial service degradation in an incremental way.

Furthermore, once all of the demands are handled, an additional optimization step is conducted to further improve the results in terms of both energy consumption and overall latency violation. Despite using the well-known approach that is based on the shortest path algorithm performed on an auxiliary graph, the proposed heuristic comprises numerous novel aspects briefly mentioned in each bullet and carefully explained below. General workflow of the heuristic is presented in Fig. 3.

Creation of an auxiliary graph. For each demand, we consider a separate auxiliary graph composed by removing from the graph all the edges that are not able to handle the request network load (H_d) due to the insufficient unoccupied link capacity ($C_e - x_e$).

Modification of the weights. Three modifiers are applied to the default weights of edges. First, the weight is zeroed for all the links adjacent to DC nodes. The rationale is to increase the chance that the shortest path will be directed through the DC nodes having virtually unlimited computing resources.

Second, the weight of each link is multiplied by a factor proportional to the delay that this link introduces. The factor varies from zero (exclusively for minimal traffic) to 1.1 (inclusively for overloaded links): $(0, 1.1]$. If a particular link is not handling any traffic, its weight remains unchanged which means it is relatively high (not reduced is equal to 100). This mechanism reduces the number of links that are switched on in the network and helps decreasing energy consumption.

The last modifier reflects the state of the computing resources in the node to which a particular link is directed. Namely, the weight of the link is multiplied by a factor proportional to the utilization of computing resources of the link's destination node. As a consequence, least loaded edge computing nodes are preferable. This mechanism directly increases the chances that sufficient computing resources will be available on the shortest path. Simultaneously, if computing resources are switched off, the weight is not reduced in order to limit the energy consumption imposed by switching on computing resources.

One needs to note that these modifiers operate as separate mechanisms introduced in our heuristic in the particular order defined above. It is the result of a comprehensive analysis which is not reported in the paper for the sake of brevity. The aim is to fully reflect the corresponding optimization problem making this stage a novel contribution.

Finding the shortest path. The heuristic can be configured to search for the shortest path in these three different manners: `direct`, `preferDC`, and `throughDC`. The `direct` approach finds the shortest path between the demand's source and destination using Dijkstra algorithm run on the auxiliary graph [31]. It is the most intuitive approach, taking into account only weights previously assigned to the edges.

The `preferDC` approach is a mechanism that analyzes all equivalent shortest paths established with respect to the same rules as in case of the `direct` method. Namely, from the set of those shortest paths it selects any one that traverses a DC node.

The `throughDC` finds the shortest path between source and destination, but under the constraint that the path traverses the DC node. The aim is to place all the computing tasks in DCs and switch off all edge computing resources. It is expected to limit the energy consumption related to the computing infrastructure, but it will led to an increased utilization of network resources.

The rationale of those various configuration options is to introduce the possibility to adjust the heuristic based on different optimization aims and investigate the impact on the final result. The fact that different levels of preference for paths traversing data centers are introduced justifies the novelty of this procedure.

Assignment of computing resources. Computing resources requirement, for each demand, varies in a predefined range limited by lower bound G^{min} and upper bound G^{max} (following the model proposed in [27]). Therefore, the first step is to determine the amount of resources to be assigned. The proposed heuristic is configurable in this aspect and three possible independent modes are available. The two extreme approaches are to assign G^{min} or G^{max} resources to the nodes where we decide to place them. The rationale of the former one (further referenced as G^{min}) is to achieve the lowest possible computing resources utilization and energy consumption. On the other hand, the latter approach (further referenced as G^{max}) minimizes latency introduced by the computing.

A more sophisticated approach (denoted as `networkAware`) is proposed in our heuristic to balance the trade-off between resource/energy consumption and achieved latency. The `networkAware` mode takes advantage of the fact that a path satisfying network requirements is already known. Thus, the algorithm may estimate the latency originating from the network by multiplying the number of links in the path and average latency determined by the link's latency model. After that, the latency threshold Ω_d^{max} for demand d is decreased by the estimated transmission latency. The rationale is to estimate how fast services must be processed to meet latency requirements. The resulting time is then translated to the amount of computing resources to be assigned, again with the usage of computing latency model. One must note that in case when latency requirements for the computing part are too strict, then G^{max} amount of resources will be assigned to minimize the value of latency violation $\beta_d^{\text{violation}}$. Once the amount of computing resources is determined, the assignment phase begins. If a path presumed to handle the request traverses a network node associated with DC, then all the services are handled in that DC. This assumption takes advantage of the fact that DCs have virtually unlimited computing resources, and thus, cannot get congested contrary to edge nodes which are limited. If any DC node is available on the path, then computing resources are assigned in the first node that, jointly, has enough computing resources and some part of its resources is already occupied (that is, the computing resources are on). If such a node does not exists, then the later constrained is ignored and the only requirement is to offer enough computing power (further corrective actions are described below). The rationale is to limit the number of edge nodes that have to be turned on and to achieve reduction in terms of energy consumption.

Table 2
Explanation of abbreviations denoting heuristic configurations.

Approach to finding shortest path	Computing resource assignment	Abbreviation

`direct`	G^{min}
	G^{max}
	`networkAware`
		`directMIN`
		`directMAX`
		`directNA`

`preferDC`	G^{min}
	G^{max}
	`networkAware`
		`preferDCMIN`
		`preferDCMAX`
		`preferDCNA`

To summarize this step: computing resources are assigned according to the described procedure and assuming one out of three different approaches to define the expected amount of resources to be assigned. In particular, the `networkAware` approach should be considered as an innovative contribution. All of those approaches can be combined with the three configuration options proposed for path selection. As a result, nine combinations are available and become the subject of the study performed in this paper as described in Section 5.2 and summarized in Table 2.

Corrective actions. If computing resources are not sufficient to handle the request on the presumed path, a following procedure is applied. All the resources temporarily assigned to the demand are released and the new shortest path traversing a DC node is established between the demand's source (S_d) and destination (T_d). All the services are handled in the DC then. If it is not possible to find a path traversing any DC, the algorithm again tries to utilize the shortest path but with the relaxed constraints for computing resources. Namely, the algorithm iteratively decreases the amount of computing resources to be assigned by 10% in each round until there are enough computing resources available on the path.

Post-allocation optimization. To further optimize the energy consumption (n) and limit overall latency violation $\sum_d \beta_d^{\text{violation}}$, two additional actions are performed once the allocation process is finished for all the demands. The first one finds all demands d that achieved a better level of latency than required (Ω_d^{max}) and tries to decrease amount of computing resources allocated to handle those demands. The process stops before the latency requirements for particular demand are violated (i.e., just when $\beta_d^{\text{violation}} = 0$). The rationale is to utilize as few computing resources as possible and still meet latency requirements.

Second action is to find all demands d that exceed Ω_d^{max} threshold and, if possible, increase the amount of assigned computing resources to reduce that violation. The process stops just after the latency requirements for particular demand are met ($\beta_d^{\text{violation}} = 0$). The rationale is to eliminate any latency violations if only unassigned computing resources are available in nodes satisfying computing demands that exceed the threshold.

Reference approach. In addition to the proposed heuristic with different configurations we formulated also a baseline approach that can be considered as a benchmark for heuristic. In this case, demands are processed in random order (as the one given in the SNDLib library [32]). To handle request d, the shortest path between the source (S_d) and destination (T_d) is found under assumption that weights are equal to one for each network edge. Then, computing resources are assigned in the first node v on the path that has the sufficient amount of these resources to satisfy at least one of the services r comprised by the demand. The consecutive computing services related to this demand are satisfied according to the same schema. Such assignment of computing resources is valid no matter if this shortest path traverses a DC. The requirements for computing resources are always minimal (equal to $G_r^{\min}$ for all $r = 1, 2, \ldots, R$). If there are no enough computing resources to handle the request on the presumed path, all the resources temporarily assigned to the demand are released and the new shortest path traversing a DC node is established between S_d and T_d. All services are handled in the DC then. Any additional optimization actions are performed after the resources have been assigned. To sum up, the reference approach considered in our studies is, therefore, a simple shortest path based approach which lacks all of the improvements proposed in our algorithm.

5. Validation

To validate our study we employed realistic assumptions that are described further in this section. Afterwards, numerical results are presented and carefully analyzed.

5.1. Assumptions

Resource dimensioning. Network resources are described by the bandwidth available on each network link and expressed in Mbps. To model realistically computing resources we consider that the three different types of resources, i.e., CPU, RAM and storage, are proportionally scaled, and thus, are usually proportionally utilized.

However, each type of computing resources can still be expressed in different units. For example, in case of computing, available clock cycles (e.g. [4,33,34]), number of cores (e.g. [17,35,36]), or virtual units (e.g. [11,37]) can be considered.

In our work, based on the conducted research, we have decided to express network resources with a single value representing throughput, while computing resources are represented by a number of available cores. Particular absolute values are not critical because it is more important to properly scale both types of resources in relation to traffic demands. We follow assumptions taken in works considering 5G scenarios, e.g., [11,17,34]. Thus, we assumed that link capacity is equal to 100 units, data centers with virtually unlimited capacity, and each edge node with 2^x number of cores, where $x = 5, 6, 7, 8$.

Demand model. Those heterogeneous resources are consumed by corresponding parts of demands arriving to the infrastructure. Volume of network demands between each pair of nodes is obtained from the SNDlib project [32] and further scaled separately for each considered topology. Nevertheless, it is important to properly adjust requirements on computing resources. Authors of [17] provided a mapping between bandwidth required by different VNFCs and CPU core usage for VNFs comprised by those chains. A similar approach was taken in [37], but only a single type of generic service chain was considered. Ref. [38] is the third recent source of the mapping between throughput and CPU requirements of particular applications. Based on these three works, we assume that each demand in our model will comprise 3 generic services ($R = 3$). Furthermore, $G_r^{\min}$ is equal to 1 CPU for each service r of demand d. In the same time, $G_r^{\max}$ is equal to 2, 4 and 16 for $r = 1$, $r = 2$ and $r = 3$, respectively. For the first model assuming that all three services must be satisfied in the same computing node, the minimum and maximum assigned computing resources are equal to 3 ($\sum_r G_r^{\min}$, sum of $G_r^{\min}$ for three services) and 22 ($\sum_r G_r^{\max}$, sum of $G_r^{\max}$ for three services), respectively.

Each demand in our models is subject to a latency threshold ($\Omega_d^{\max}$). Several works propose thresholds for various services. Ref. [27] considers conversational services, streaming services, and background services assuming that latency threshold equal to 150, 300, and 600 ms, respectively. Another taxonomy was proposed in: [35] and [23]. Namely, web, video streaming, VoIP, and gaming services have latency requirements equal to 500, 100, 100, and 60 ms, respectively. More demanding services are considered in [17], where 1 ms of latency is a threshold for augmented reality and smart factory services, and 5 ms of latency must be ensured for massive IoT service. Finally, authors of [39] focus on SDN-based industrial IoT applications with fog computing. The considered services are: process monitoring, environmental monitoring, fault diagnosis, product testing, and inventory management while corresponding latency requirements for those services are equal to 10, 50, 20, 50, and 80 ms. As we are considering modern 5G services (some recent examples of services can be found in [7]) offered through WAN, we have decided to assume latency threshold equal to 20 ms. The rationale is to assume a restrictive value due to the nature of 5G-based services; however, the threshold cannot be as demanding as those proposed in [17] because the services are offered over long distances.

Latency model. Latency originates transmission through the network and processing in computing nodes as explained in Section 3. However, it is important to estimate both values in a realistic way to verify if any of the two contributions is not a dominant contributor to the overall latency. For network links, latency value varies between 0 and 11 ms depending on network load and is consistent with corresponding values present in the literature (e.g., in [27]). Similarly, we considered different approaches to the latency originating from processing performed in computing nodes. Authors of [15] and [12] simply assume that for generic VNF constant delay is equal to 10 and 0.5 ms, respectively. A more sophisticated approach was proposed in [40] and [27], where execution times of VNFs are randomly selected from range [10, 20] ms and vary between 10 and 30 ms in function of utilization of a computing node, respectively.

To ensure realistic assumptions regarding the minimum and maximum computing latency ($\Theta_r^{\min}$ and $\Theta_r^{\max}$, respectively), we have scaled values present in the literature, respecting the fact that each demand in our model will comprise 3 generic services ($R = 3$). As a result, $\Theta_r^{\min}$ is equal to 0.5 for each service while $\Theta_r^{\max}$ is equal to 3, 6, and 60 for $r = 1$, $r = 2$ and $r = 3$, respectively. One needs to note, that for the first model that assumes that all three services must be satisfied in the same computing node, the minimum ($\sum_r \Theta_r^{\min}$) and maximum ($\sum_r \Theta_r^{\max}$) latency in a node satisfying the demand is equal to 1.5 and 69, respectively.

Energy consumption model. In our model, energy is also consumed by both transmission devices and computing nodes. Models for energy consumption were carefully explained in Section 3. The relation between the cost of activating a link and the linear coefficient must be carefully evaluated. According to [41] and [42], a router port consumes 20 and 1000 W of energy, respectively. However, in both cases the load-dependent part is not considered. On the other hand, authors in [29] showed that energy consumption of an idle 1 Gbps port is equal to 180 W, while a port working with the full rate consumes almost 200 W. Similar values are presented in [43], with a switch consuming 245 W when idle and 300 W when all of its ports operate under full load. Considerations presented in [44] are more comprehensive in regards to both switches and routers. A switch is assumed to consume 760 W when totally idle while switching any of links adds 5 W to this value. Simultaneously, for a router's port energy consumption is expressed as 14.5 W/Gbps without considering any constant part. We tried to find values which are most relevant in the 5G context and generic enough to be suited to the partially conflicting values reported in literature. Consequently, we assume that the energy cost of switching port on equals 180 W, while every single Mbps of load adds 0.02 W to that value, reaching 200 W under maximum utilization.

Concerning energy consumption of the computing infrastructure, in works [41,43,44], and [45], authors reported various ranges of server's energy consumption as a function of load. For an idle server the proposed values are 150, 121, 325, and 200 W, while the peak load energy consumption is equal to 300, 750, 380, and 328 W, respectively. Also in this case, our aim was to assume some generic values consistent with the reported values, that could also fit in the 5G context. So, we assumed that each edge computing node consumes 150 W if switched on, while each demand introduces energy consumption equal to $5 * g_{rd}$ (where g_{rd} is an amount of processing power consumed by service r of demand d).

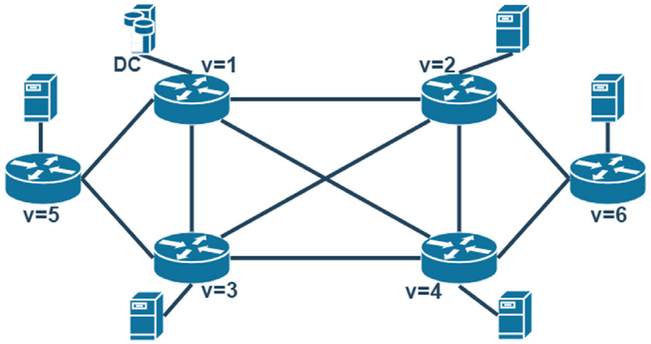

Fig. 4. The *n6* topology.

Topologies and placement of DCs. The results are presented for four different network topologies from the SNDlib project [32]: `polska`, `nobel-us`, `nobel-eu`, and `germany50`. The topologies consist of 12, 14, 28 and 50 nodes; 18, 21, 41, and 88 links; and 66, 91, 378 and 662 demands, respectively. Additionally, one small regular network topology referred as *n6* (see Fig. 4) is also analyzed as a simple case to validate correctness of the solution using the optimization model.

Furthermore, as locations of DCs in these networks have significant impact on results also some realistic locations must be ensured. In the `polska` topology, nodes `Warsaw` and `Poznan` are selected as gateways supporting Internet Exchange Points (IXPs) for the whole country, and most of DCs are placed in those cities. In the `nobel-us` topology, `Pittsburgh` and `Palo-Alto` nodes host DCs as proposed in [46]. For `nobel-eu`, we follow work [47], where nodes `Frankfurt` and `Paris` are suggested to host DCs. Finally, in the `germany50` topology, real life conditions force us to locate DCs in `Frankfurt` and `Berlin` as both cities are widely known to host the biggest DCs in the country. In the *n6* node with index $v = 1$ hosts DC (as shown in Fig. 4) to ensure reasonable connectivity (node's degree equal to 4) to the computing facility.

Optimization goal scaling. Finally, we have also introduced some scaling into the optimization goal. Namely, the overall energy consumption (n) was additionally divided by 20 ($F_n = 20$) while summary delay violation remained unchanged ($F_\beta = 1$). The rationale is to obtain approximately equal impact of both factors on the final goal value for all the network architectures. We also conducted studies over different proportions of both components of optimization goal, but those considerations are not reported.

5.2. Numerical results

To present our results, we applied the following approach. In the first step, we draw some general conclusions regarding the efficiency of heuristics under different configurations. The next step is to investigate some specific cases for particular networks and optimization models. A combination of network and optimization model is referred as scenario, while *m1* and *m2* abbreviations denote first and second optimization models. Furthermore, we compare the various configurations of the heuristic. Whenever we mention one of optimization models or problems in the context of heuristic performance, we mean that heuristic solves the problem constrained according to assumptions of model.

In Table 2, abbreviations denoting the different configurations of the heuristic are summarized for easy reference. Six different abbreviations are introduced to combine two approaches for finding shortest path with three ways of assigning computing resources (as explained in Section 4). Note that, whenever the `throughDC` approach to shortest path finding process is applied, the results are independent of the computing resource assignment algorithm. The reason is that in this configuration services are always deployed in the DCs, and thus the post-allocation optimization has full freedom to modify the amount of assigned resources and, no matter which starting point is selected, the resulting assignment is always the same. Therefore, this approach can be simply denoted as `throughDC`.

5.2.1. General observations

Values of optimization goal achieved by different configurations are presented in Table 3 to provide a general overview of the results. The first observation is that any kind of heuristic significantly improves results when compared to the `reference` approach. This has several reasons. First of all, the overall energy consumption is much higher as most of the links and edge computing resources are turned on, as the `reference` approach does not try to accommodate traffic on links that are already switched on. Latency also suffers because the `reference` approach assumes assignment of computing resources with the minimal value ($G_r^{\min}$) for each service and any post-allocation procedures are applicable. The final conclusion regarding the `reference` approach is that the bigger is the topology, the more significant is the difference between the `reference` approach and various configurations of heuristic. This observation simply results from the fact that in bigger networks statistical multiplexing can bring more advantages to heuristic aimed at accumulating network and computing load on selected links and computing nodes.

The second general conclusion regards the `throughDC` configuration, which – as already explained – is independent of the way computing resources are assigned. This configuration constantly provides improvements in comparison to the `reference` approach; however, efficiency of this configuration is the worst among all the options. As all services are deployed in DCs, the post-allocation optimization ensures that computing resources are either sufficient to eliminate latency violation or they are assigned $G^{\max}$ to satisfy the computing demand. It turns out that any edge location must be turned on what brings reduction in energy consumption. However, both the mentioned advantages come at the cost of sub-optimal routing aimed at ensuring that each path between S_d and T_d traverses a DC node. Then, significant increase can be observed in: network resource utilization (indirectly related to the average path length), overall resource utilization, and also maximum resource utilization over all network links. Those negative consequences predominate, thus making `throughDC` inefficient.

Another general observation relates to the comparison of the `direct` and `preferDC` approaches as regards the way they find the shortest paths. In most of the cases, both methods combined with the same algorithm for assigning computing resources provide the same results. Differences may occur only when following three phenomena jointly appear: (1) there is more than one shortest path between S_d and T_d; (2) at least one of those paths traverses a DC node and (3) this particular node is not selected by the `direct` method (it depends on implementation of the shortest path algorithm).

To be more precise, the `preferDC` approach provides better results only when solving first optimization problem in the case of *nobel-eu* and *germany50* networks, and with combination with the `networkAware` resource assignment strategy. In those two cases, network resources close to the DCs are sufficient to handle the additional traffic routed through the DCs due to the preference of those computing facilities. Only in these cases, the advantages (i.e., full flexibility of computing resources assignment) resulting from placing services in DCs overcome potential shortcomings of network performance degradation. Only `networkAware` is able to achieve potential gains as it aims at rationally assigning computing resources at edge locations; thus, this approach limits the situations when corrective actions will direct additional traffic to DCs increasing probability of network congestion. The reason why `preferDC` is less relevant when solving second optimization problem comes from the fact that ability to spread services over computing nodes makes it possible to more effectively utilize edge locations over shortest paths not traversing DC nodes.

On the other hand, the `direct` algorithm brings advantages in numerous scenarios, e.g., solving second optimization problem used on

Table 3
Goal function values for different configurations of heuristic in all studied networks.

Sim. scenario	`reference`	`directMIN`	`directMAX`	`directNA`	`preferDCMIN`	`preferDCMAX`	`preferDCNA`	`throughDC`
n6 m1	1752.12	286.00	266.59	265.38	286.00	266.59	265.38	305.38
n6 m2	1752.12	263.49	267.45	263.49	263.49	267.45	263.49	301.75
polska m1	3800.30	728.18	688.60	672.88	728.18	688.60	672.88	841.18
polska m2	3800.30	676.52	708.46	676.52	676.52	708.46	685.33	838.07
nobel-us m1	5028.43	716.41	755.21	755.21	716.41	755.21	755.21	773.07
nobel-us m2	5028.43	687.31	691.27	687.31	687.31	691.27	687.31	760.38
nobel-eu m1	20075.73	3320.92	3475.23	3521.79	3320.92	3475.23	3475.23	4366.50
nobel-eu m2	20075.73	3157.96	3293.69	3157.96	3157.96	3293.69	3293.69	4338.63
germany50 m1	35104.49	4552.27	5983.59	8281.90	6255.37	5983.59	5983.59	8208.25
germany50 m2	35104.49	4339.28	4421.58	4339.28	6014.49	5347.09	5347.09	8172.04

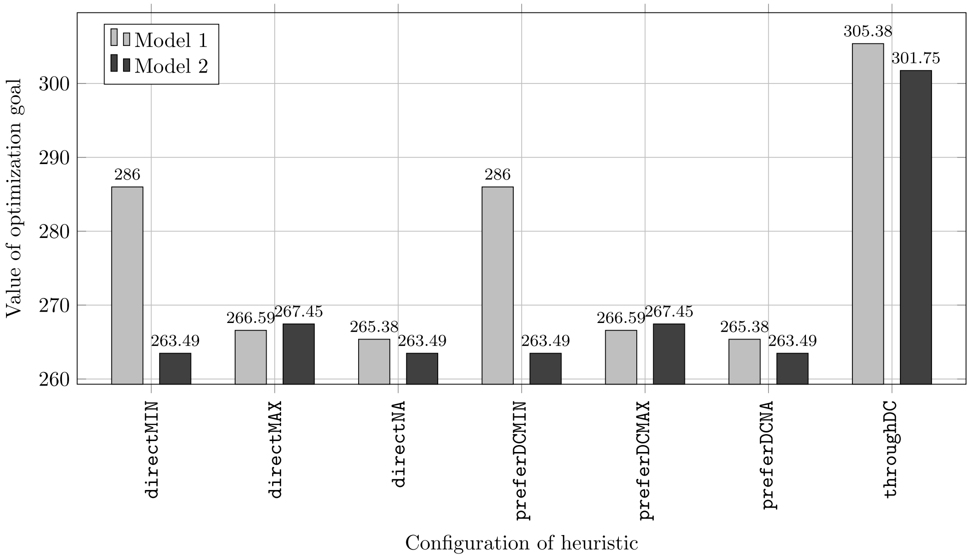

Fig. 5. Optimization goal in function of different heuristic configurations in network *n6*.

networks *polska* and *nobel-eu* when combined with the `networkAware` algorithm; and when solving second optimization problem applied to the *germany50* network, no matter which algorithm is exercised for the computing domain; and when solving first optimization problem used on *germany50* when combined with the $G^{\min}$ algorithm for the computing domain. In all of those cases, network resources around DC nodes are a potential bottleneck. Therefore, it is more reasonable to distribute traffic over various shortest paths instead of accumulating load in the areas adjacent to the DC. In other words, without perfect dimensioning of resources, potential advantages in the computing domain are predominated by negative consequences in the network domain. This effect is especially noticeable in the *germany50* network, where differences between configurations are most significant. This network is the biggest in terms of a number of nodes, demands and links, while a number of large-scale DCs remains at the level of two. Those facts lead to the following observations. The more demands exist in the network, the more likely congestion near DCs will occur; and, the longer are the paths, the more efficient and flexible edge locations can be utilized to meet latency requirements. Those conclusions are especially important as the *germany50* network is characterized with a topology common for MAN networks being the best target for 5G slicing.

The last observation regards the comparison of effectiveness of solving optimization problems in some of the networks. The conclusions are that, in general, solving the second optimization problem provides better results in most cases (marginal exceptions for the *n6* and *polska* networks will be further explained during a topology-specific analysis). The advantage of solving second problem is more significant for algorithms that assign computing resources in a more greedy way ($G^{\max}$ and `networkAware`). Finally, solving of second optimization problem is most advantageous for the *germany50* network which represents a topology typical for MAN networks, especially important in the context of 5G deployments.

5.2.2. *Topology-specific analysis*

The aim of the following analysis is to provide some insights for particular topologies. Thus, general conclusions that are not directly negated remain valid, and are only further extended by additional observations.

Fig. 5 presents values of the optimization goal for the *n6* network. As this network is small, the number of shortest paths available between each pair of nodes is strongly limited. Therefore, both `direct` and `preferDC` approaches provide the same results. Two other interesting observations concern the comparison solving effectiveness of optimization models. Firstly, for `directMIN` and `preferDCMIN`, solving the second problem provides a significant improvement, as ability to adjust an assignment of computing resources with a service granularity can be much more efficient comparing to the assignment of the whole demand. Second, small deterioration of results can be observed for the `directMAX` and `preferDCMAX` configurations when the solution of second optimization problem is compared with the first one. The reason for this phenomenon is that the first model enforces deployment of all services comprised by a single demand in one computing node; thus, a larger number of edge locations is activated when compared to the second optimization model. As a consequence, post-allocation optimization may allocate more resources and limit latency violations. However, significant improvements in latency come at the cost of higher energy consumption due to the increased number of edge locations being turned on and due to larger amount of utilized computing resources. Both factors are present in the optimization model. They balance each

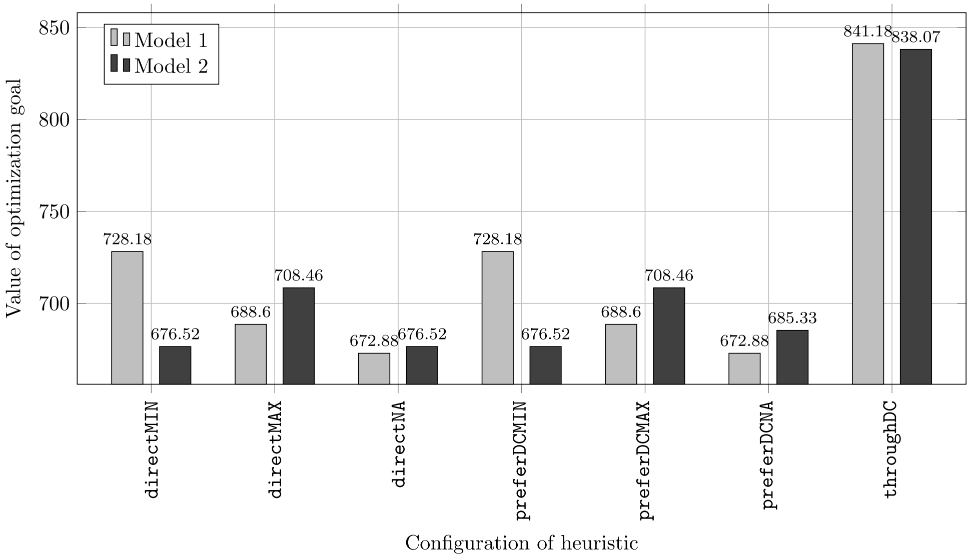

Fig. 6. Optimization goal in function of different heuristic configurations in network *polska*.

other making the final value of the optimization goal only slightly different.

The results obtained for network *polska* are visualized in Fig. 6. Note that differences between results obtained by various configurations are not significant when normalized by the absolute values of the optimization goal. However, even these differences suggest some interesting conclusions. First of all, when solving the first problem it can be observed that networkAware dominates $G^{\max}$, which is superior to the $G^{\min}$ approach combined with any shortest path algorithms. A high value of the optimization goal achieved when the $G^{\min}$ approach is applied results from a considerable level of latency violation. It occurs because during allocation with minimal possible computing resources a fewer number of edge locations must be switched on. Thus, post-allocation optimization is not able to increase the amount of assigned computing resources to the threshold that may limit the violation. On the other hand, $G^{\max}$ enforces much more edge locations to be turned on, and thus the latency violation can be minimized but at the cost of additional energy consumption. The networkAware approach is able to find a reasonable trade-off between those two criteria. It is possible due to the fact that in the *polska* topology the number of DCs in comparison to the number of nodes and demands is quite high. Therefore, latency in computing domain is a bottleneck that should be reasonably overcome. An interesting observation is that such conclusions are not valid when solving the second optimization problem. The directMIN, directNA, preferDCMIN, and preferDCNA approaches can more effectively assign computing resources during post-allocation procedures with a service granularity ensured by assumptions of the second optimization model. On the other hand, despite turning on more edge locations, directMAX and preferDCMAX are not able to take advantage of statistical multiplexing to effectively utilize those computing resources, and simultaneously, to direct less traffic to the DCs where virtually unlimited amount of computing resources can be used to minimize latency violations. The conclusion is that in case of small topologies, where a number of demands is also relatively small, some additional planning tools can be useful to effectively utilize resources being turned on.

The last counter-intuitive result that can be observed in the *polska* topology regards the fact that solving the first optimization problem provides better results for the directMAX, directNA, preferDCMAX, and preferDCNA configurations. The reasons are exactly the same as in case of the *n6* topology. Namely, the first model enforces a deployment of all services comprised by a single demand in one computing node, and thus, a larger number of edge locations is turned on when compared to the second optimization model. As a consequence, post-allocation optimization may allocate more resources and limit latency violation. However, also in case of this topology, significant improvements in latency come at the cost of higher energy consumption due to the increased number of edge locations being turned on and a bigger amount of utilized computing resources. Both factors are present in the optimization task and balance each other — hence, only slightly different final value of the optimization goal.

Fig. 7 presents results collected in the *nobel-us* network. The first interesting observation relates to the fact that in case of solving the first problem, most of the configurations do not provide much better results than the throughDC approach. It is a consequence of the fact that DCs are associated with nodes that are often traversed by shortest paths, no matter which shortest path algorithm was applied. It further implies that post-allocation optimization assigns computing resources almost equal to $G^{\max}$, no matter if the $G^{\max}$ or networkAware approach was applied. The directMIN and preferDCMIN strategies are able to significantly minimize the optimization goal in case of solving the first problem. Both configuration assign minimum possible computing resources needed to handle requests, and finally, less edge locations must be turned on. Moreover, it is less likely that corrective actions will direct a request to the DC which is not on the shortest path. In consequence, both energy consumption and network resource utilization are reduced. This positive effect is slightly obscured by the fact that latency originating from computing is higher, but still, such an approach remains advantageous.

Analyzing the results of solving the second optimization problem we can observe that the aforementioned location of DCs still ensures the same behavior of the direct and preferDC approaches. Secondly, advantage of all other configurations against the throughDC approach is much more significant. Thanks to the ability of the second optimization model to separately deploy different services comprised by a single demand, edge resources can be utilized more effectively. This phenomenon leads to less severe latency violations and additionally limits the corrective actions delegating computing tasks to DCs that do not reside on the shortest paths. We can conclude that these approaches are applicable for networks such as *nobel-us*, that is to infrastructures in which DCs are located in nodes with a high betweenness centrality parameter. This value denotes the ratio of shortest paths traversing a particular node to the total number of shortest paths existing in the network.

The *nobel-eu* network is the first significantly bigger topology being considered, but still the ratio of DC to edge nodes remains at the level

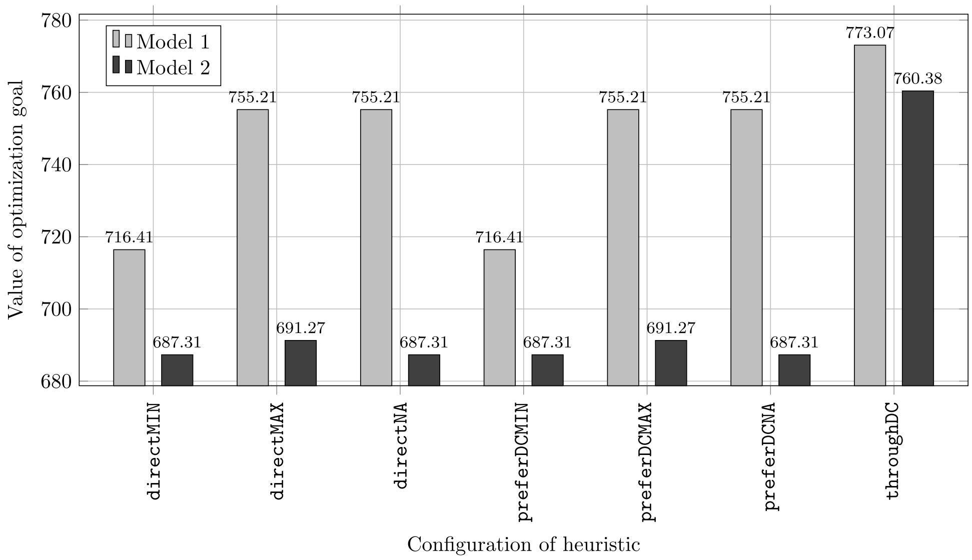

Fig. 7. Optimization goal in function of different heuristic configurations in network *nobel-us*.

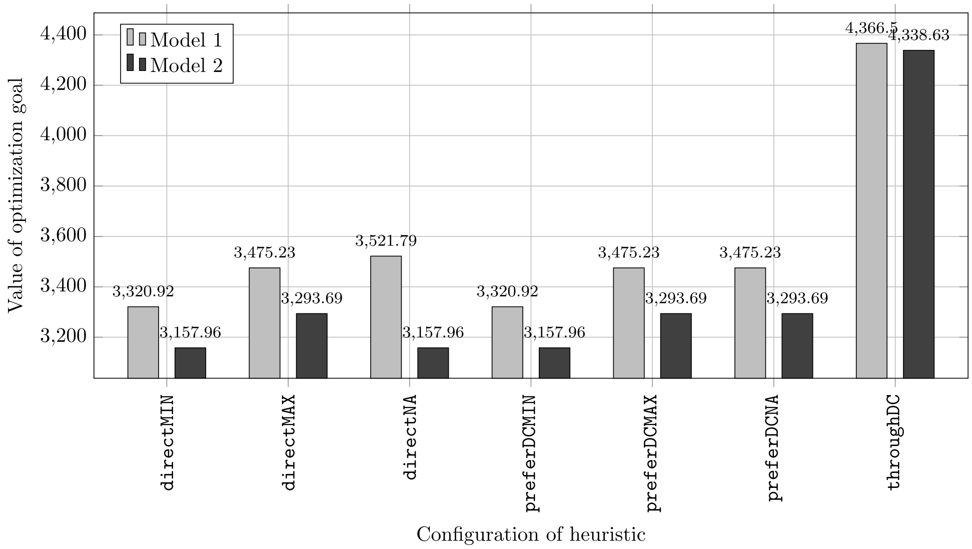

Fig. 8. Optimization goal in function of different heuristic configurations in network *nobel-eu*.

typical for WAN infrastructures. Results are presented in Fig. 8. One needs to note that, due to high absolute values of the optimization goal, most of the differences between approaches are not significant. It means that effectiveness is ensured by most of the configurations when solving both optimization problems, this directly stems from the fact that DCs can be easily accessed and are likely to be traversed by shortest paths between numerous pairs of nodes. Thus, the `preferDC` option only slightly changes results when compared to the `direct` method, and simultaneously, different computing resource assignment methods deploy services with quite high computing power taking advantage of virtually unlimited capacity of DCs. This further leads to similar levels of energy consumption and latency in the computing domain. This effect is a bit weaker when solving the first optimization problem; however, in this case, results obtained by different computing resource allocation mechanisms are similar to the ones related to utilization of network resources. Such a effect occurs as heuristic for the first model less efficiently utilizes edge locations and corrective actions are more likely to occur (by directing traffic to DCs).

Secondly, when solving both optimization problems, the $G^{\min}$ approach is more efficient than two other options. The reason is that during the allocation phase the $G^{\min}$ approach turns on a smaller number of edge locations and less frequently forces corrective actions which direct demands to DC. Therefore, energy consumption is reduced in both domains, and so is latency. These advantages come at the cost of increased latency in the computing domain since less computing resources can be assigned during post-allocation optimization. Still, the $G^{\min}$ approach dominates other methods.

Finally, a counter-intuitive result can be observed when comparing `directNA` with `preferDCNA` simultaneously when solving both optimization problems. Namely, the preference for DCs slightly improves results obtained when solving the first problem, as only a bit higher network resource utilization and energy consumption is dominated by improvement in the latency violation aspect. On the other hand, in the case of solving the second optimization problem the preference for DCs results in a significant increase of average path length and consequently, in an increase on the amount of computing resources assigned to handle requests. Both disadvantages imply a decreased gain related to the latency in the computing domain and they finally result in a worst (i.e., higher) value of the optimization goal.

The German network (*germany50*) is the last considered topology, results are presented in Fig. 9. It is especially important as this network

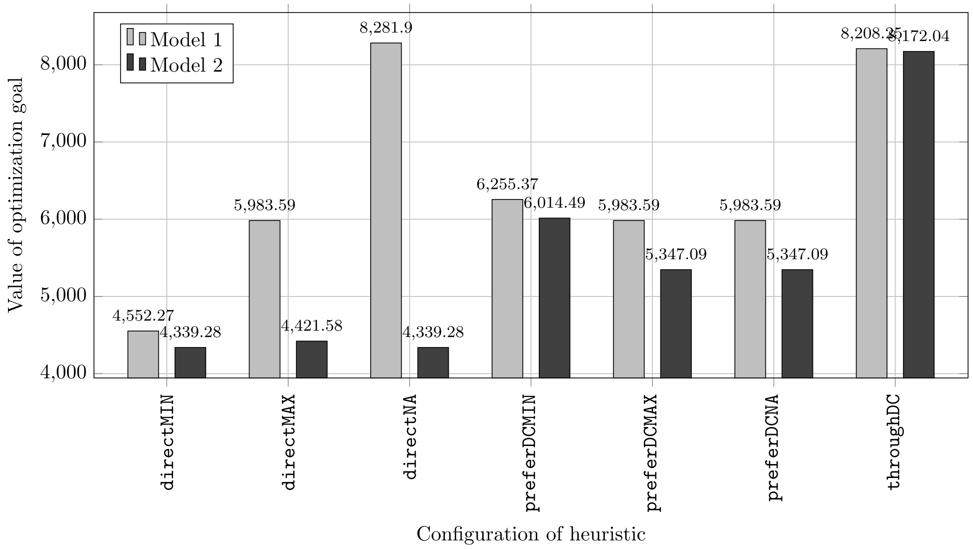

Fig. 9. Optimization goal in function of different heuristic configurations in network *germany50*.

is a representative of MAN networks with a significantly higher number of nodes and demands when compared to the number of DCs. Such scenario is especially relevant when considering 5G networks as it is the most intuitive ecosystem for deployment of slices.

Let us start by analyzing the solution of the first optimization problem and the `direct` shortest path algorithm combined with various computing resource assignment methods. The $G^{\min}$ approach provides significantly better results as it engages less edge nodes, and simultaneously, is less likely to cause corrective actions directing demands to significantly more distant DCs. On the other hand, both the $G^{\max}$ and `networkAware` approaches try to assign much more computing resources to limit latency originated in the computing domain. As a side effect, edge nodes become overloaded and demands must be handled in DCs. Therefore, energy consumption and network utilization are increased, which further causes increase in the overall latency. The `networkAware` configuration causes even more problems, as it underestimates the expected load on several links by further diminishing the latency. One needs to note that the effect of such an underestimation provides even worse results than the `throughDC` approach (which from the very beginning directs all the traffic to DCs).

On the other hand, the preference for DCs imposed by the `preferDC` approach performs better when combined with `networkAware`, as it eliminates the underestimation explained above. Simultaneously, this preference diminishes the output performance of `preferDCMIN` when compared to `directMIN` because of the change imposed in routing causes by engagement of edge nodes that cannot be utilized as effectively as in case of the `direct` shortest path mechanism.

When analyzing results obtained for the second optimization model, we can observe superior results when the `direct` shortest path mechanism is combined with any algorithm used for assignment of computing resources. Ability to deploy services comprised in a single demand on different computing nodes creates an opportunity to effectively reduce the number of edge locations involved by the $G^{\max}$ and `networkAware` approaches. Furthermore, post-allocation optimization is able to ensure desired latency in the computing domain more efficiently.

At the same time, the `preferDC` approach performs better in the `preferDCMAX` and `preferDCNA` configurations involving slightly more edge locations being able to handle more services with minimal cost in terms of energy and network resource utilization. However, `preferDC` approach compared to `networkAware`, causes a slight increase in average path length and network resource utilization.

Table 4
Summary of parameters for CPLEX calculations for different networks.

Scenario	Solution found?	Stop reason	Execution time [h]	Achieved solution gap [%]
n6 m1	Yes	Optimum found	10.1	0.01
n6 m2	Yes	Optimum found	43.8	0.01
polska m1	Yes	RAM limit achieved	16.8	9.32
polska m2	Yes	RAM limit achieved	29.7	10.42
nobel-us m1	Yes	Time limit achieved	48	14.29
nobel-us m2	Yes	Time limit achieved	48	17.13
nobel-eu m1	Yes	Time limit achieved	48	12.31
nobel-eu m2	Yes	Time limit achieved	48	13.56
germany50 m1	**No**	Time limit achieved	48	N/A
germany50 m2	**No**	Time limit achieved	48	N/A

Finally, the most important conclusion regards the comparison between solutions for the optimization models. Solution of the second model constantly provides results superior to the ones obtained with the first model, as concerns corresponding configurations of heuristic. Ability to deploy different services of the same demand in various computing nodes brings the most significant advantages, when the `direct` shortest path mechanism is enabled. In those configurations effective utilization of edge locations can bring the most significant improvements by reducing a number of situations when corrective actions directing demands to DC are needed. As a result, network resource utilization is decreased, along with energy consumption and overall latency. In case of the `directNA` configuration, solution of the second model reduces a value of the optimization goal by more than 45%. Those observations prove that a proper modeling of 5G slices is especially important in MAN-like architectures which are most common for these purposes.

5.2.3. Comparison with near-optimal solutions

Prior to presenting the exact ILP results obtained through CPLEX software, a couple of observations must be pointed out. Calculation parameters are summarized in Table 4. Due to the complexity of the optimization models, execution time was limited to 48 h, as longer calculations did not bring significantly better results in terms of the achieved solution gap (i.e., optimization gap between the relaxed and best integer solutions). The calculations were conducted in parallel on three servers in our local data center, each equipped with 24 GB of RAM memory and 12 CPUs with clock frequencies equal to 2.67, 2.8 or 2.93 GHz, depending on the machine, and on the node of Prometheus

Table 5
Ratios between the best and the worst configurations of heuristic and comparison to solutions provided by CPLEX.

scenario	The best configuration(s)	The worst configuration(s)	Best/worst ratio	Best/CPLEX ratio	Worst/CPLEX ratio
n6 m1	directNA, preferDCNA	throughDC	0.87	1.11	1.27
n6 m2	directMIN, directNA, preferDCMIN, preferDCNA	throughDC	0.87	1.16	1.33
polska m1	directNA, preferDCNA	throughDC	0.80	1.33	1.66
polska m2	directMIN, directNA, preferDCMIN	throughDC	0.81	1.41	1.75
nobel-us m1	directMIN, preferDCMIN	throughDC	0.93	1.17	1.26
nobel-us m2	directMIN, directNA, preferDCNA, preferDCMIN	throughDC	0.90	1.16	1.28
nobel-eu m1	directMIN, preferDCMIN	throughDC	0.76	1.57	2.06
nobel-eu m2	directMIN, directNA, preferDCMIN	throughDC	0.73	1.59	2.19
germany50 m1	directMIN	directNA	0.55	–	–
germany50 m2	directMIN, directNA	throughDC	0.53	–	–

supercomputer equipped with 128 GB of RAM memory and 24 CPUs each of clock frequency equal to 2.5 GHz. Local machines were using CPLEX in version 12.5 while on supercomputer node the most recent version 12.10 was used. The best achieved results from all the machines are presented in Table 4.

For the *n6* network we were able to achieve an optimal solution while for *nobel-us* and *nobel-eu* 48 h long calculations enabled achieving the gap between 12.31 and 17.13%. In the case of *polska* network even a huge amount of RAM available on a supercomputer node was not sufficient to finalize computing, however, achieved solution gap is at an acceptable level. Furthermore, for *germany50* CPLEX was not able to find a solution and estimate the gap even after 48 h of calculations. One must note that execution times are at orders of magnitude higher when compared with heuristics providing solutions for any topology within single seconds.

Let us now investigate the relation between the optimization goal value achieved by the best and the worst configurations of the heuristic and to compare those results with the ones provided by CPLEX. In Table 5, we enumerate consecutive configurations of heuristic (columns are related to various network topologies) by providing the best and the worst results (more than one configuration means that all mentioned options provided exactly the same result). Furthermore, the ratio between the best and worst values of the optimization goal is provided, as well as the relation between the best/worst results obtained by the heuristic and the value of optimization goal achieved by CPLEX (two last columns).

The following conclusions can be drawn based on the presented results. When solving the second optimization problem, which is most relevant to the 5G network slices, the directMIN and directNA configurations always achieve the best result. Simultaneously, the preferDC shortest path algorithm performs very well in all networks, except for *germany50*, especially when combined with the $G^{\min}$ approach and occasionally with networkAware. This means that it is reasonable to prefer DCs only in infrastructures where number of DCs is sufficient and those DCs are associated with network nodes of high degree and high betweenness centrality. This is usually not the case for MAN networks, as represented by *germany50* topology.

On the other hand, the $G^{\min}$ resource assignment method does not perform well when solving the first optimization problem applied to relatively small networks (*n6* and *polska*). In the same time, the $G^{\max}$ approach does not provide the best results in any of the networks. It means that a blind assignment of maximum computing resources is not efficient and limitations intelligently applied by the networkAware approach are useful and may lead to heuristic solutions closer to the optimal.

As mentioned before, the *germany50* topology is adequate for 5G slices applications being considered in this work. Thus, it is even more important to note, that in this network, the best results are obtained when solving the second optimization model along with directMIN and directNA (analogously to other networks). At the same time, the directNA configuration brings the worst result when solving the first optimization problem applied to the same network. This scenario (the first optimization model for the *germany50* topology) is a perfect example of a mismatch between a model (not suited to the 5G slices and NFV deployments) and an infrastructure (representing typical environment for 5G slices deployment). It is the only scenario, for which throughDC is not the worst configuration and which is replaced by the directNA, for all other scenarios, provides results closest to the optimal.

Furthermore, the following conclusions may be drawn when considering three ratios presented in Table 5. The less complex is the topology, the closer to the optimal solution are the results achieved by both the best and the worst configuration. This phenomenon is quite intuitive, as in more complex networks the optimal solution may be extremely sophisticated and only static optimization methods are able to get close to it, contrary to the heuristic being based on a kind of modification of the shortest path algorithm. Finally, for more complex networks, a smaller number of configurations is able to achieve result being equally close to the optimal solution. At the same time, the difference between the best and the worst result is bigger. However, an optimized placement of DCs may be significantly better for any configuration of the heuristic, as can be observed in case of the *nobel-us* topology.

In Table 6, more detailed results are provided for each network topology, optimization model and three solving methods: results obtained by CPLEX, and the best, and the worst, configuration of heuristic. In addition to an optimization goal value, in consecutive columns, the following metrics are presented: overall energy consumption in both domains, overall delay violation, overall computing resources assigned, ratio of activated edge locations to the total number of edge nodes, average length of all network paths, and the amount of network resources utilized to satisfy network demands.

CPLEX always returns the best results due to its ability to effectively utilize network resources. It is reflected by the fact that an average hop number of paths satisfying demands is significantly increased when compared to the shortest path routing and the best heuristic configuration. At the same time, overall network resource utilization is significantly lower. This is possible as demands of the highest requirements on network resources (H_d) are routed through the shortest paths while less demanding requests are intentionally routed through the less congested network areas.

As a result of such an optimization CPLEX rarely turns on any edge location, and this significantly decreases overall energy consumption. All computing services can be successfully handled in DCs without imposing additional network-originated latency thanks to the fact that network traffic is evenly distributed over the less congested network. It is confirmed by the delay violation equal to zero, while the assigned computing power remains lower than in case of heuristic.

Therefore, even the best configuration of heuristic provides worse results for most of the factors. It is especially important in case of energy consumption and delay violation, as both factors have a direct impact on the optimization goal value. The only parameter improved by the heuristic when compared to the CPLEX method is average length of path. It confirms the fact that heuristic is based on a modified shortest path algorithm with additional respect of existence of the computing domain. However, even with a lower average path length,

Table 6
Detailed results for different solution methods applied to both optimization models in all studied networks.

Scenario	*Solving method*	optimization goal	Energy consumption	Delay violation	Computing power	Edge usage	Avg. hops	Network usage
n6 m1	CPLEX	239.95	4798.99	0.00	560.61	0.00	2.40	797.05
	Best heuristic	265.38	5307.68	0.00	566.22	0.40	1.83	830.00
	Worst heuristic	305.38	5981.63	6.30	616.03	0.00	2.43	1074.00
n6 m2	CPLEX	226.82	4536.37	0.00	508.09	0.00	2.23	797.02
	Best heuristic	263.49	5036.87	11.60	542.05	0.20	1.83	830.00
	Worst heuristic	301.75	5908.87	6.30	601.48	0.00	2.43	1074.00
polska m1	CPLEX	506.38	10127.55	0.00	1228.31	0.09	3.24	1301.18
	Best heuristic	672.88	11897.60	78.00	1263.64	0.64	2.94	1470.50
	Worst heuristic	841.18	11816.87	250.33	1420.02	0.00	3.94	1837.90
polska m2	CPLEX	479.12	9582.41	0.00	1119.40	0.00	3.42	1270.46
	Best heuristic	676.52	11107.29	121.16	1165.49	0.45	3.00	1491.60
	Worst heuristic	838.07	11754.70	250.33	1407.59	0.00	3.94	1837.90
nobel-us m1	CPLEX	613.85	12277.00	0.00	1587.57	0.00	4.81	958.89
	Best heuristic	716.41	13989.22	16.95	1707.53	0.42	3.57	1078.70
	Worst heuristic	773.07	13908.56	77.64	1768.98	0.00	3.92	1182.16
nobel-us m2	CPLEX	591.77	11835.40	0.00	1463.01	0.08	3.76	1020.24
	Best heuristic	687.31	13671.49	3.73	1643.98	0.42	3.57	1078.70
	Worst heuristic	760.38	13654.80	77.64	1718.23	0.00	3.92	1182.16
nobel-eu m1	CPLEX	2117.69	42353.70	0.00	6664.90	0.00	5.63	1462.87
	Best heuristic	3320.92	44511.83	1095.33	6490.27	0.50	4.99	1524.00
	Worst heuristic	4366.50	49886.26	1872.19	7917.71	0.00	6.03	1886.80
nobel-eu m2	CPLEX	1984.03	39680.60	0.00	5986.38	0.04	5.59	1432.82
	Best heuristic	3157.96	43751.26	970.40	6338.16	0.50	4.99	1524.00
	Worst heuristic	4338.63	49328.80	1872.19	7806.21	0.00	6.03	1886.80
germany50 m1	CPLEX	–	–	–	–	–	–	–
	Best heuristic	4552.27	74763.48	814.10	10857.46	0.44	6.69	2309.60
	Worst heuristic	8281.90	88620.81	3850.86	13033.46	0.75	7.70	2674.80
germany50 m2	CPLEX	–	–	–	–	–	–	–
	Best heuristic	4339.28	73620.99	658.23	10628.96	0.44	6.69	2309.60
	Worst heuristic	8172.04	85732.20	3885.43	13893.16	0.00	8.41	3320.00

the overall consumption of network resources remains higher in case of heuristic which sequentially assigns resources for consecutive demands. In general, the scarce utilization of edge resources is possible due to low energy cost of network infrastructure compared to computing, it also depends on source and destination of nodes and latency constraints.

On the other hand, when analyzing results achieved by the worst configuration of heuristic, one should note that in most cases any edge location is turned on. However, such a blind deployment of computing services solely in DCs comes at the cost of significantly higher network resource utilization, what further imposes higher requirements on computing resources in order to decrease the overall delay. Even in this case, delay requirements are often violated and energy consumption is increased.

Another set of conclusions explaining a couple of further counter-intuitive results that can be observed based on the results presented in Table 6. In the three cases, the CPLEX solver engages a single edge node to handle services. It is because a potential penalty for directing more traffic to DCs is higher than the cost of turning on edge location which is further fully utilized. In all three cases the best configuration follows the preference of edge utilization as the percentage of edge resources being tuned on is highest among all the scenarios. There are also three cases when the worst configuration of heuristic achieves lower energy consumption than the best one. In those cases, heuristic had to intentionally increase an average path length and edge nodes utilization to distribute the load among more resources with the aim to minimize overall delay violation. Finally, there is only a single situation when the worst heuristic outperforms the best result in the context of delay violations. It happens in a very small network, where network latency is not an important contributor to the overall goal value. At the same time, limitations on computing resources in an edge location engaged by the best heuristic resulted in an additional computing delay.

The last aspect to be studied is comparison of both optimization models applied to different networks. All detailed results achieved by different solving methods are, in general, worse in case of the first optimization model. It is a result of increased flexibility ensured by the second model. Due to the potential of statistical multiplexing, less computing resources are needed. It is expressed by lower edge usage and assigned computing power, what further decreases energy consumption without introducing additional delay violations. However, interesting results can be observed. In case of the *polska* network when solving the first optimization problem, the best heuristic ensures a lower delay violation. The reasons have been already explained during detailed studies of this topology, and additional parameters presented in Table 6 only confirm that more edge locations engaged by the first optimization problem create an opportunity to assign more computing resources during post-allocation optimization. The trade-off of higher energy consumption makes the final optimization goal value of solution for both models almost equal to each other with a minimal advantage of the solution of first optimization model. Finally, in case of the *nobel-us* network, the network usage parameter is deteriorated when solving the second optimization compared to the first one. It happens despite the fact that in the same time average path length is decreased. It is because that in order to minimize optimization goal value, demands of higher network requirements have been handled by the longer paths. This counter-intuitive result is an indirect consequence of deployment of DCs in association with network nodes characterized with large values of betweenness centrality and degree.

6. Conclusions

This paper addresses the problem of satisfying requirements of modern services deployed in heterogeneous cloud networks comprising both cloud and edge computing nodes interconnected through an SDN network. The optimization task is a complex multi-objective ILP problem covering latency, energy and network performance indicators. The 5G network is considered as an example of a cloud network, where slices, defined as a set of virtual resources, are provisioned to satisfy demands. Furthermore, the relevant models of latency and energy consumption are formulated based on a comprehensive review of the *state-of-the-art*. Latency and energy requirements are taken from relevant use cases recently studied for 5G networks. As the defined optimization tasks are difficult, or even impossible, to solve for large-scale network instances, a configurable heuristic is also proposed.

We performed our numerical evaluations in various network topologies and several contexts. Namely, different configurations of the heuristic were analyzed. It was shown that the heuristic is able to effectively provide satisfactory results for large-scale problems and both optimization models. This way, we are able to justify that the proposed approach is well suited. Efficiency of the proposed heuristic is significantly better when compared to the reference approach based on shortest path algorithm. Moreover, the importance of matching between optimization model and heuristic configuration to the 5G ecosystem properties was assessed utilizing the topology that represents MAN-like infrastructure being typical target for slice deployments. Additionally, the performance of the proposed heuristic was compared against the near-optimal solutions. For all the experiments, realistic assumptions and configuration values are taken based on the comprehensive review of the available literature.

To briefly summarize the most important conclusions, four different areas are addressed in the following paragraphs.

6.1. Infrastructure-related conclusions

The first important remark concerns the fact that any preference of big data centers to deploy computing services is reasonable only in infrastructures with carefully dimensioned and balanced network and computing resources. Additionally, network resources in the close proximity to the DCs must be sufficient to handle excess traffic. Only if these requirements are met, the advantages of full flexibility of computing resource assignment in DCs overcome potential shortcomings stemming from the network performance degradation. Otherwise, it is more reasonable to distribute traffic over various shortest paths. The attractiveness of preference towards big computing facilities can be strengthened when those DCs are properly placed in the network, for example, in conjunction with network nodes of high degree and high betweenness centrality.

Comparing the performance of different solving methods as a function of infrastructure architecture, the following conclusions can be drawn. The less complex is the topology, the closer to the optimal solution are the results achieved by any configuration of the heuristic. This phenomenon directly follows the fact that in more complex networks only static optimization methods are able to get close to the optimal solution, which is significantly different from the most intuitive solutions. Simultaneously, for more complex networks, the difference between results achieved by various configurations of the heuristic is more significant. Therefore, to conduct valuable research it is required to analyze architectures typical for 5G slicing and NFV deployments.

6.2. Conclusions regarding configuration of the heuristic

One of the most important observation relates to the general advantage of the `networkAware` approach used to assign computing resources. This approach aims at reasonably assigning computing resources at edge locations; thus it limits the situations when corrective actions will direct additional traffic to DCs and, as a consequence, increase probability of congestion in the network domain. The `networkAware` approach also allows to take advantage of the second optimization model's properties making it more suitable to 5G networking.

Simultaneously, it is not reasonable to blindly direct services to DCs as performed in the case of the `throughDC` shortest path algorithm. As already pointed, even a slight preference of DCs should be carefully considered and applied only in selected scenarios. Therefore, the `direct` shortest path algorithm is the most universal applicable in majority of scenarios, including the most desirable infrastructures with optimally deployed DCs.

In conclusion, the `direct` shortest path algorithm combined with the `networkAware` approach used for assignment of computing resources (denoted in the paper as `directNA`) is expected to provide the best results in most scenarios. In some selected scenarios, very promising results can also be achieved by the `directMIN`, `preferDCNA`, and `preferDCMIN` configurations. It is also important to note that the most promising `directNA` configuration performs poorly in the case of a mismatch between the optimization model and basic assumptions regarding the 5G infrastructure. This issue will be address in the two consecutive sections below.

6.3. Comparison of the formulated optimization models

The second optimization model provides better results (lower value optimization goal) in most cases, with only marginal exceptions. Furthermore, the achieved detailed quality indicators are in general inferior in case of the first optimization model. Especially, lower edge usage and assigned computing power can be observed in case of second model. It further decreases energy consumption without introducing additional delay violations. This predominance results from increased flexibility ensured by the second model and, related to it, statistical multiplexing.

Furthermore, the second optimization problem, which is most relevant to the 5G network slices, takes the most significant advantages of the `direct` shortest path algorithm combined with the `networkAware` approach used for assignment of computing resources. Moreover, superiority of second model can be especially observed in case of infrastructures relevant to the 5G slices (in our work, represented by the `germany50` topology).

6.4. 5G-specific conclusions

Finally, conclusions regarding the 5G slices and VNF deployments are summarized here. The German network (*germany50*) is the topology that represents MAN-like infrastructures being especially relevant when considering 5G networks as the most intuitive ecosystem for deployment of slices. One of the properties of such an infrastructure is related to the significantly higher number of nodes and demands when compared to the number of DCs. Due to that it is more likely to create congestions near DCs. Additionally, MAN networks are usually more complex and average paths are longer in comparison to other infrastructures. Thus, it is possible to more effectively and flexibly utilize edge locations to meet latency requirements. Furthermore, location and a total number of DCs in MAN networks are usually not satisfying due to the practical constraints. Therefore, any preference of DCs in case of 5G slices should be carefully investigated before the service and VNF deployment in order not to deteriorate the performance of the infrastructure. That is why the `directMIN` and `directNA` configurations perform best in the *germany50* topology. However, for the latter this happens only in case of solving the second optimization model. All of those properties result in the most demanding character of MAN networks, and – in consequence – the resulting differences between the results achieved by different solution methods are most significant.

The fact that the `directNA` configuration, which outperforms other configurations in different scenarios, brings the worst result for the first optimization model applied to the MAN infrastructure demonstrates that a proper modeling of 5G slices is especially important in MAN-like architectures. Namely, the first optimization model applied to the *germany50* topology is a perfect example of a mismatch between a model (not tailored for the 5G slices) and an infrastructure (representing a typical environment for 5G slices deployment).

Despite the fact that it was possible to draw conclusions summarized above, several possible avenues for future work can be envisioned.

First of all, our optimization model can be extended by considering more sophisticated relations between services (e.g. services varying from demand to demand, or network requirements changing after satisfying particular services). Second, we so far we have assumed that both optimization criteria are, approximately, equally important for a network or service provider. However, there is a potential for more detailed studies on multi-objective aspect of the problem, e.g., different weights of both factors can be used, Pareto-based investigations can be performed, etc. Finally, both the optimization model and the proposed heuristic can be studied in a more sophisticated infrastructure comprising both wide area and metro area networks interconnected in selected locations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Piotr Borylo: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Visualization, Project administration, Funding acquisition. **Massimo Tornatore:** Conceptualization, Methodology, Writing - review & editing, Supervision. **Piotr Jaglarz:** Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft. **Nashid Shahriar:** Conceptualization, Writing - review & editing. **Piotr Chołda:** Methodology, Writing - review & editing, Supervision. **Raouf Boutaba:** Conceptualization, Supervision.

Acknowledgment

This work was funded by the National Science Centre, Poland under project no. 2018/02/X/ST7/00810 and supported in part (computing power) by PLGrid Infrastructure, Poland.

References

[1] CISCO, Cisco Visual Networking Index Global Mobile Data Forecast, 2017–2022, Cisco White Paper, 2019, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf.

[2] 3GPP, System architecture for the 5G System (5GS), Technical Specification 23.501, 2017.

[3] Q. Zhang, F. Liu, C. Zeng, Adaptive interference-aware VNF placement for service-customized 5G network slices, in: 2019 IEEE Conference on Computer Communications IEEE INFOCOM'19, April 2019, pp. 2449–2457.

[4] J. Zhang, X. Hu, Z. Ning, E.C.. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks, IEEE Internet Things J. 5 (4) (2018) 2633–2645.

[5] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, A. Jajszczyk, Energy-aware fog and cloud interplay supported by wide area software defined networking, in: 2016 IEEE International Conference on Communications ICC'16, 2016.

[6] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei, M.R. Khosravi, Efficient resource management and workload allocation in fog-cloud computing paradigm in IoT using learning classifier systems, Comput. Commun. 153 (2020) 217–228.

[7] Q. Ye, W. Zhuang, X. Li, J. Rao, End-to-end delay modeling for embedded VNF chains in 5G core networks, IEEE Internet Things J. 6 (1) (2019) 692–704.

[8] G. Garcia-Aviles, M. Gramaglia, P. Serrano, F. Gringoli, S. Fuente-Pascual, I.L. Pavon, Experimenting with open source tools to deploy a multi-service and multi-slice mobile network, Comput. Commun. 150 (2020) 1–12.

[9] N. Huin, M. Rifai, F. Giroire, D.L. Pacheco, G. Urvoy-Keller, J. Moulierac, Bringing energy aware routing closer to reality with SDN hybrid networks, IEEE Trans. Green Commun. Netw. 2 (4) (2018) 1128–1139.

[10] M. Dias de Assunção, R. Carpa, L. Lefèvre, O. Glück, P. Boryło, A. Lasoń, A. Szymański, M. Rzepka, Designing and building SDN testbeds for energy-aware traffic engineering services, Photonic Netw. Commun. 34 (3) (2017) 396–410.

[11] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, P. Castoldi, Latency-aware composition of Virtual Functions in 5G, in: 2015 IEEE Conference on Network Softwarization NetSoft'15, 2015.

[12] P. Vizarreta, M. Condoluci, C.M. Machuca, T. Mahmoodi, W. Kellerer, QoS-driven function placement reducing expenditures in NFV deployments, in: 2017 IEEE International Conference on Communications ICC'17, 2017.

[13] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, N. Pissinou, Joint host-network optimization for energy-efficient data center networking, in: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013, pp. 623–634.

[14] K. Zheng, X. Wang, L. Li, X. Wang, Joint power optimization of data center network and servers with correlation analysis, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 2598–2606.

[15] A. Varasteh, M. De Andrade, C.M. Machuca, L. Wosinska, W. Kellerer, Power-aware virtual network function placement and routing using an abstraction technique, in: 2018 IEEE Global Communications Conference GLOBECOM'18, 2018.

[16] M.-T. Thai, Y.-D. Lin, Y.-C. Lai, Joint server and network optimization toward load-balanced service chaining, Int. J. Commun. Syst. 31 (10) (2018).

[17] L. Askari, F. Musumeci, M. Tornatore, Latency-aware traffic grooming for dynamic service chaining in metro networks, in: 2019 IEEE International Conference on Communications ICC'19, 2019.

[18] T. Nguyen, A. Girard, C. Rosenberg, S. Fdida, Routing via functions in virtual networks: The curse of choices, IEEE/ACM Trans. Netw. 27 (3) (2019) 1192–1205.

[19] G. Li, B. Feng, H. Zhou, Y. Zhang, K. Sood, S. Yu, Adaptive service function chaining mappings in 5G using deep Q-learning, Comput. Commun. 152 (2020) 305–315.

[20] S. Taeb, N. Shahriar, S.R. Chowdhury, M. Tornatore, R. Boutaba, J. Mitra, M. Hemmati, Virtual network embedding with path-based latency guarantees in elastic optical networks, in: 2019 IEEE 27th International Conference on Network Protocols, ICNP, 2019.

[21] P. Lindberg, J. Leingang, D. Lysaker, S.U. Khan, J. Li, Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems, J. Supercomput. 59 (1) (2012) 323–360.

[22] Q.D. La, M.V. Ngo, T.Q. Dinh, T.Q. Quek, H. Shin, Enabling intelligence in fog computing to achieve energy and latency reduction, Digit. Commun. Netw. 5 (1) (2019) 3–9.

[23] M. Savi, M. Tornatore, G. Verticale, Impact of processing costs on service chain placement in network functions virtualization, in: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network NFV-SDN'15, 2015, pp. 191–197.

[24] C. Pham, N.H. Tran, S. Ren, W. Saad, C.S. Hong, Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach, IEEE Trans. Serv. Comput. 13 (1) (2020) 172–185.

[25] P. Boryło, P. Chołda, J. Domżal, P. Jaglarz, P. Jurkiewicz, A. Lasoń, M. Niemiec, M. Rzepka, G. Rzym, R. Wójcik, SDNRoute: Integrated system supporting routing in software defined networks, in: 2017 19th International Conference on Transparent Optical Networks ICTON'17, Girona, Spain, 2017, pp. 1–4.

[26] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[27] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, R. Boutaba, Delay-aware VNF placement and chaining based on a flexible resource allocation approach, in: 2017 13th International Conference on Network and Service Management CNSM'17, 2017.

[28] F. Kaup, S. Melnikowitsch, D. Hausheer, Measuring and modeling the power consumption of OpenFlow switches, in: 10th International Conference on Network and Service Management and Workshops CNSM'14, 2014, pp. 181–186.

[29] Y. Yang, M. Xu, D. Wang, S. Li, A hop-by-hop routing mechanism for green internet, IEEE Trans. Parallel Distrib. Syst. 27 (1) (2016) 2–16.

[30] J.R. Arora, Introduction to Optimum Design, Elsevier, Oxford, UK, 2012.

[31] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271.

[32] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0—Survivable network design library, NET 55 (3) (2010) 276–286.

[33] W. Fang, M. Lu, X. Liu, L. Gong, Z. Zhu, Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections, IEEE/OSA J. Opt. Commun. Networking 7 (4) (2015) 314–324.

[34] J. Zhang, W. Xia, F. Yan, L. Shen, Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing, IEEE Access 6 (2018) 19324–19337.

[35] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, A. Pattavina, Protection strategies for virtual network functions placement and service chains provisioning, Networks 70 (4) (2017) 373–387.

[36] D. Harutyunyan, N. Shahriar, R. Boutaba, R. Riggio, Latency-aware service function chain placement in 5G mobile networks, in: 2019 IEEE Conference on Network Softwarization NetSoft'19, 2019, pp. 133–141.

[37] M. Zeng, W. Fang, Z. Zhu, Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks, J. Lightwave Technol. 34 (14) (2016) 3330–3341.

[38] A. Gupta, B. Jaumard, M. Tornatore, B. Mukherjee, A scalable approach for service chain mapping with multiple SC instances in a wide-area network, IEEE J. Sel. Areas Commun. 36 (3) (2018) 529–541.

[39] J. Wang, D. Li, Adaptive computing optimization in software-defined network-based industrial internet of things with fog computing, Sensors 18 (2018) 2509.

[40] L. Qu, C. Assi, K. Shaban, Delay-aware scheduling and resource optimization with network function virtualization, IEEE Trans. Commun. 64 (9) (2016) 3746–3758.

[41] F. Larumbe, B. Sanso, A tabu search algorithm for the location of data centers and software components in green cloud computing networks, IEEE Trans. Cloud Comput. 1 (1) (2013) 22–35.

[42] X. Dong, T. El-Gorashi, J.M.H. Elmirghani, Green IP over WDM networks with data centers, J. Lightwave Technol. 29 (12) (2011) 1861–1880.

[43] P. Ruiu, A. Bianco, C. Fiandrino, P. Giaccone, D. Kliazovich, Power comparison of cloud data center architectures, in: 2016 IEEE International Conference on Communications ICC'16, 2016.

[44] S. Yang, P. Wieder, R. Yahyapour, X. Fu, Energy-aware provisioning in optical cloud networks, Comput. Netw. 118 (C) (2017) 78–95.

[45] C. Canali, R. Lancellotti, M. Shojafar, A computation- and network-aware energy optimization model for virtual machines allocation, in: 7th International Conference on Cloud Computing and Services Science CLOSER'17, CLOSER 2017, 2017, pp. 71–81.

[46] Y. Wu, M. Tornatore, C.U. Martel, B. Mukherjee, Green and low-risk content placement in optical content delivery networks, in: 2016 IEEE International Conference on Communications ICC'16, 2016.

[47] K. Walkowiak, A. Kasprzak, M. Klinkowski, Dynamic routing of anycast and unicast traffic in elastic optical networks, in: 2014 IEEE International Conference on Communications ICC'14, 2014, pp. 3313–3318.