Black-box optimization for anticipated baseband-function placement in 5G networks

Zorello, Ligia Maria Moreira; Bliek, Laurens; Troia, Sebastian; Maier, Guido; Verwer, Sicco

DOI
10.1016/j.comnet.2024.110384

Publication date
2024

Document Version
Final published version

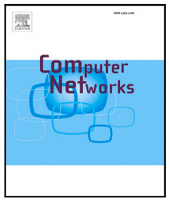
Published in
Computer Networks

Computer Networks 245 (2024) 110384

Contents lists available at ScienceDirect

Computer Networks

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



Black-box optimization for anticipated baseband-function placement in 5G networks

Ligia Maria Moreira Zorello [a], Laurens Bliek [b], Sebastian Troia [a,*], Guido Maier [a], Sicco Verwer [c]

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
[b] School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[c] Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

Keywords:
5G RAN
Black-box optimization
Intelligent optimization
Surrogate-based optimization

ABSTRACT

In the context of the ever-evolving 5G landscape, where network management and control are paramount, a new Radio Access Network (RAN) as emerged. This innovative RAN offers a revolutionary approach by enabling the flexible distribution of baseband functions across various nodes, all tailored to meet the ever-shifting demands of both system requirements and user traffic patterns. As users move within the network, the need to anticipate and strategically position these baseband functions becomes crucial for seamless network operation. Traditionally, this challenge has been tackled through a two-step process: first, forecasting traffic patterns, and then optimizing resource allocation accordingly. However, this approach falls short in guaranteeing an efficient placement when actual traffic demands surge onto the network. It often leads to resource overbooking, constraint violations, and excessive power consumption, putting strain on the network's capabilities. In this paper, we introduce a novel framework based on a black-box optimization approach. This tool empowers prediction algorithms not just with historical traffic data but also with insights from optimization outcomes. The goal is to minimize a loss function related to power consumption and constraint violation: this ensures a predicted placement that is feasible and whose power is close to optimal. This approach ensures that the predicted placement is both feasible and power-efficient, bridging the gap between theoretical prediction and practical implementation. Remarkably, our proposed method, while potentially sacrificing some degree of traffic prediction accuracy, outperforms the conventional two-step approach by delivering a more efficient baseband function placement.

1. Introduction

The fifth generation of mobile networks (5G) is a new approach to address the strict requirements of various applications and enhance their overall performance. It was necessary to improve the Decentralized Radio Access Network (DRAN) of 4G, as it would result in an inefficient and costly architecture for mobile operators at scale. 4G introduced the Centralized RAN (C-RAN) to separate the Remote Radio Head (RRH) and Baseband Unit (BBU) and centralize and virtualize the BBUs in central offices. This design improves network management and control, processing scalability, and reduces overall power consumption by better utilizing computing resources. However, several issues still need to be addressed to enable its widespread deployment. Combining baseband functions in a single central node imposes strict latency requirements between sites and also requires high traffic capacity over the fronthaul links connecting the RRHs to their BBU [1].

To address these issues and strike a balance between power consumption, processing, and bandwidth capacity, some baseband functions can be virtualized and distributed across local offices in the network. The 3rd Generation Partnership Project (3GPP) has proposed functional splits that indicate the degree of centralization of baseband functions [2]. In this setup, the RRH remains at the antenna site and the BBU functions are implemented as Virtual Network Functions (VNFs) in Distributed Unit (DU) and Centralized Unit (CU). The DUs deploy lower-layer network functions in the access network, and the CUs deploy the remaining functions over cloud-enabled nodes in the metro network.

Network operators must efficiently place CUs in the network to ensure correct operation and reduce operational costs [3–5]. By deploying CU functions as VNFs, operators can modify the placement based on network requirements. However, these requirements are not static, particularly in urban areas where user movement impacts traffic

* Corresponding author.
E-mail address: sebastian.troia@polimi.it (S. Troia).

https://doi.org/10.1016/j.comnet.2024.110384
Received 22 November 2023; Received in revised form 25 February 2024; Accepted 30 March 2024
Available online 4 April 2024

volume [6,7]. Operators must place and move CU VNFs to follow traffic patterns and ensure quality of service while minimizing power consumption. The efficient placement of VNFs is a new optimization problem with significant effort in the literature to solve it. However, aspects related to costs and functional split constraints have not been fully considered. Operators must minimize operational costs while ensuring service requirements are met. Power models that consider different network aspects are cumbersome to use. Our previous works [8,9] shows that network power consumption plays a key role in NFV infrastructure cost. Therefore, we study optimal VNF placement to minimize node and network power consumption while ensuring compliance with split and service latency requirements. We consider CU placement while assuming DUs are fixed in the access network.

The placement of baseband VNFs is significantly impacted by network traffic, which fluctuates daily due to user movement in urban areas. Consequently, network operators may need to relocate these VNFs across different nodes to adapt to changing traffic patterns and comply with split and service requirements. To ensure correct network operations and avoid service disruption, the placement of CUs must be planned in advance. While machine learning algorithms can predict future traffic patterns, their accuracy does not immediately translate into efficient optimization. Machine learning algorithms can aid this anticipated optimization by leveraging the predictable daily traffic variations in urban areas caused by inhabitants' and commuters' regular displacement. These algorithms provide accurate traffic forecasts and enhance the applicability of optimization frameworks in real-time scenarios. Several works available in the literature combine these predictive algorithms with the optimization problem in a two-step approach to enable resource allocation anticipation [6,7,10–12]. However, as demonstrated in our previous work [8], unforeseen events can cause disruptions in these patterns, rendering traditional techniques that use machine learning to predict traffic and subsequently optimize placement incapable of providing solutions that ensure feasibility in real-time. In fact, these methods do not guarantee that actual traffic can be accommodated by the predicted placement. Traditional traffic prediction algorithms train models to minimize prediction error, but applying the uncertainty of the prediction to the optimization model may lead to poor decisions [13].

This paper presents a novel training method of prediction models based on the outcome of an optimization problem that depends on the forecast value. This tool ensures operators that the predicted placement is feasible and efficient when applying the real traffic. The proposed framework is based on a black-box method. It allows training the prediction algorithm not just with the traffic historical data, but also including the feedback from the optimization outcomes into the training process. For this, we model the optimal placement of CU VNFs in a metro-network to minimize the system power consumption. This paper uses a Mixed Integer Linear Programming (MILP) and a heuristic to solve this problem. During the training phase, the black-box algorithm iteratively updates the weights of the traffic prediction according to a loss function, which is obtained by applying the real traffic to the predicted CU VNF placement. It computes a penalty for constraint violations and the difference between the resulting power consumption with respect to the oracle available during the training phase.

The remainder of the paper is organized as follows. Section 2 describes the related works. Sections 3 and 4 detail the proposed framework, including the MILP formulation and the black-box-based framework to train the traffic prediction algorithm. Section 5 explains the simulation environment and describes the results. Section 6 concludes the paper.

2. Related works

2.1. Resource allocation in 5G RAN

Several works are available in the literature to perform resource allocation in 5G RAN. Al-Quzaweeni et al. [3] minimized the power consumption when allocating baseband functions and routing users' traffic considering both processing and network components. For this purpose, they modeled the placement of VNFs related to the baseband and core functions over different nodes of an optical network. They demonstrated that the virtualization of baseband functions can substantially reduce the system power consumption. Tinini et al. [14] optimized the placement and migration of baseband units over cloud and fog nodes in a virtual passive optical network to minimize the consumed power. They proposed a dynamic algorithm that decides whether to migrate the virtual BBUs according to the traffic variation, and then calls a mathematical program for the optimal placement. They showed that their algorithm reduces power consumption and blocking probability with respect to other solutions. Singh et al. [15] modeled an integer quadratic problem that jointly decides the split option to indicate the functions hosted by the CU, and the assignment of the CU and the DU over the network node. Their results showed that distributed processing is usually more efficient than centralized scenarios. These works provide solutions to place different baseband functions over network nodes; however, they fail to evaluate the functional split latency, which is an important requirement as it determines the network efficiency and quality of service.

Klinkowski et al. proposed in [16] a DU allocation and fronthaul and midhaul flow routing guaranteeing the quality of service for all the service requests. They formulated a MILP optimization to minimize the number of DUs and the service latency. They explained that the greater is the number of demands generated, the higher is overall flow latency. In addition, they demonstrated that larger networks require more DUs to ensure that the latency requirements are respected. Gupta et al. [17] optimized the placement of CUs based on energy consumption and number of handovers considering different functional splits. They placed CUs with the goal of reducing the energy consumption according to the number of active CUs, to the number of handovers and to the additional energy consumed by the migration of DUs. Their results revealed that their framework improves the system energy efficiency and the quality of service by reducing the number of handovers and of DU relocation. These works provide more complete formulations to evaluate the functional splits requirements; nevertheless, their power model considers only the node-related components. Our previous work [9] formulated a CU VNF placement problem considering different split options. It proposed a MILP model with the goal to minimize the node and the network power consumption ensuring that split-related constraints are respected. That work showed that the network component of the power consumption plays a key role in the placement of these functions. We further extended that work in [5] placing VNFs related both to CUs and DUs. That work formulates a more complete and flexible model to allocate resources and ensure latency and processing constraints related to the radio network units. Due to the complexity of the MILP model, we also developed a heuristic algorithm that was capable of reaching similar results.

All these works aimed at proposing models to optimize the VNF placement in 5G RAN assuming that the network conditions are known. However, as previously explained, the traffic presents significant variations during the day, making it necessary to recompute the placement multiple times a day. Because of the long RAN reconfiguration time, these algorithms cannot be immediately deployed in real-time systems. This paper presents an anticipatory optimization model based on the algorithms presented in [5].

2.2. Anticipatory resource allocation

The integration of machine learning has been shown to be handy in improving the dynamicity and applicability of optimization to real-time solutions. Several papers about traffic prediction then optimization (two-step approach) have been published in the past years. Zhang et al. [7] used machine learning techniques to cope with network

resource utilization and handover by predicting users' traffic and mobility. They presented some use cases showing the usefulness of such techniques in calculating the users traffic based on the spatial dependence of the traffic to their trajectories.

Bega et al. [18] described a tool to anticipate the allocation of resources. Their approach based on three-dimensional convolutional neural networks learns spatio-temporal features and forecasts the load of different services. The prediction is then integrated into a cognitive management framework to perform different management tasks in the network. They demonstrated through simulations the advantages in automating management and orchestration operations in networks. Rago et al. proposed in [10] an anticipatory allocation of edge resources based on predicted spatio-temporal user information. They used a convolutional long short-term memory algorithm to predict the user distribution in the cells and their service demands. This information was integrated with an optimization framework, which uses the predictions to allocate resources.

In the context of 5G RAN, Pelekanou et al. [11] deployed machine learning algorithms to anticipate the resource provisioning. It first forecasts the traffic using long short-term memory. Then, it selects the baseband functions to be deployed in a single centralized data center using neural networks. The later is trained based on an ILP model that minimizes the costs related to the node and to the links. Their results showed similar performance between their approach and the use of ILP model with real traffic. Yu et al. [12] proposed an algorithm to allocate, migrate and scale 5G RAN slices to enable a dynamic allocation. They developed a traffic-prediction-based optimization to minimize the slice degradation and the traffic to be migrated. Their algorithm first predicts the traffic using a multiple polynomial regression algorithm. Then, an algorithm takes this information as input to dynamically adjust and migrate RAN slices. They proved that properly migrating the slices reduces their degradation and that predicting the traffic helps deciding when to perform the slice migration. Chen et al. [19] assigned baseband units to the remote radio heads to improve cost efficiency and reduce the handover between different units based on historic data. Their approach initially predicts the network traffic volume and the number of expected handovers, and then maps baseband units to the remote radio heads using a greedy algorithm.

In a preliminary work [20], we proposed a two-step optimization to enable operators to plan the placement of CUs and the most suitable functional split. The output of the traffic prediction mechanism was used to determine the optimal placement. This approach enables operators to plan their network configuration beforehand; however, they should consider a margin of capacity to ensure feasibility. Based on these results, we proposed in [8] the placement of CU functions using multi-task algorithm, which predicts the expected and the quantile traffic. It applies the mean predicted traffic in the MILP objective function and the quantile prediction in the constraints. Hence, it estimates the costs and enforces the capacity limits to ensure the compliance to the constraints.

The works presented in this subsection consider the maximization of traffic prediction accuracy, *i.e.* the training goal is to minimize the error between the predicted and the real traffic values. This information is then used in different optimization algorithms to determine the placement ahead of time. Therefore, they do not consider the result of the resource allocation algorithm to which the predicted traffic is applied. Alternatively, this paper presents a novel approach that trains the predictive algorithm according to the outcomes of the CU placement, as explained in the next subsection.

2.3. Training predictive algorithms based on optimization

Finding a prediction model that gives good results when used jointly with a mathematical programming is a complex optimization problem. Verwer et al. [21] combined machine learning and optimization algorithms by encoding regression models into a mathematical program. Their approach maps LASSO and regression trees as part of the constraints of the optimization problem so that the prediction and the optimization are solved together by the solver. They showed that this white-box approach outperforms traditional black-box best-fit search. Wilder et al. [22] presented an end-to-end decision-focused learning framework to train neural networks based on optimization problems. Because the loss function in combinatorial problems is discrete, hence, discontinuous, they proposed a framework capable of propagating the gradients throughout the optimization using continuous relaxation. They demonstrated that, in traditional training methods, the error distribution is not aligned with the underlying optimization. Alternatively, their approach focuses on the aspects that are more important for the optimization problem, such that it can take better decisions. Following this work, Konishi and Fukunaga [23] used gradient boosting algorithms for the end-to-end learning. They fitted the algorithm to predict uncertain optimization parameters according to the final results.

Elmachtoub et al. [13] proposed the Smart Predict then Optimize (SPO) technique. They developed a loss function to train machine learning algorithms based on the outcome of linear-objective optimization formulations. Instead of measuring the quality of the prediction based on the prediction error as in traditional predict then optimize approaches, they modeled a cost function associated to the optimization problem. They demonstrated that their framework can be applied to many different optimization models with convex constraints. Moreover, they showed that the SPO loss performs better than traditional predict then optimize approaches. Mandi et al. [24] took advantage of the SPO technique to solve combinatorial optimization problems. Because of the complex nature of these problems, they proposed continuous relaxations of the combinatorial problem and used transfer learning to accelerate the training. They showed that the use of relaxation methods enables the SPO loss to be applied to different scenarios, including large-scale instances.

Surrogate-based optimization methods such as Bayesian optimization may also be used as they are designed for expensive optimization problems [25]. Yan et al. [26] proposed a loss function to guide the training of prediction models. They transformed the hard constraints of linear and quadratic problems to soft constraints that were incorporated into surrogate models to obtain the gradient to train the machine learning algorithms. They showed that by using soft constraints in surrogate models, their approach outperforms the traditional two-step predict and then optimize in different scenarios.

These works showed that a closer integration of machine learning techniques and the optimization model can improve the final optimization goal in many different scenarios. In the context of 5G RAN, there are no works available in the literature that explore the integration of predictive algorithms and optimization problems such that the results of the latter are used to train the former. Moreover, the works described in this section consider that the uncertain parameter, *i.e.* the value that is predicted, is applied uniquely in the objective function [13,22–24] or consider only soft constraints [26]. Instead, our framework evaluates the impact of the uncertainty applied both to the objective function and to the constraints of the optimization problem to determine the cost function.

2.4. Paper contribution

The configuration of 5G RAN should be performed several times a day to cope with the varying demand. Although widely used in the literature, the two-step approach that consists of predicting the traffic and then computing the resource allocation does not guarantee the best placement when applying it to the real traffic. It may lead to higher power consumption and constraint violations. The main contribution of this paper is a novel approach to train a predictive algorithm exploiting the outcomes of an optimization problem solved for the training traffic. To the best of our knowledge, this method has not been evaluated in the 5G RAN context and the use of black-box optimization tools was

not explored to train predictive algorithms based on the outcomes of the resource allocation.

The approach proposed in this paper trains a polynomial regression algorithm responsible for determining the traffic used in the MILP model, including the linear objective function and several optimization constraints. Polynomial regression models are widely used in the literature to design traffic prediction models. Despite its low complexity, this category of algorithms was shown to provide accurate prediction results [27]. Therefore, this work uses a black-box algorithm to determine the weights of the polynomial regression. It uses a black-box optimization tool called HyperOpt [28], which is a variant of Bayesian optimization that can deal with a large number of variables and problem variants.

3. Optimization of CU VNF placement

The optimization model consists of the placement of CU VNFs with the goal of minimizing the overall power consumption subject to the split and system constraints. The DUs are deployed into given nodes of the access network. While the gateway is unique and stationed within a high-performance cloud computing node, seamlessly connected to the network's core infrastructure. The CUs are placed over the metro optical network nodes considering that the baseband functions are separated using split 2 [2].

3.1. System description

We model the network as a graph $G = (N, E)$, where N and E are the set of nodes and virtual links. Each node is equipped with η_n servers and has a total computing capacity of C_n. A single node is used as gateway. The server power consumption depends on the node utilization, *i.e.* P_{idle} whenever active up to a maximum of P_{max} at full capacity. The nodes are connected through high-capacity fiber links. The set of virtual links E refers to the wavelengths available for the network fibers. Therefore, the capacity of each virtual link C_e is limited to the wavelength bandwidth of 100 Gbit/s. The nodes are equipped with a set of transponders, consuming P_{tx} each. A set of demands D representing the aggregated traffic λ_d^{DU} from the DUs connected to each node of the network except for the gateway. Each demand must pass through all radio-network units $U = \{DU, CU, GW\}$, *i.e.* it starts at a DU, then it is processed by an optimally placed CU, and it finally arrives to the gateway. Table 1 summarizes the parameters and the variables used in the following sections.

The baseband processing load depends on the split option. Considering split 2, the processing load at the CU is [5,29]:

$$\pi = \sigma \left(n_a^2 + 3 \cdot n_a + \frac{M_b \cdot C_r \cdot L_m}{3} \right) \cdot \frac{R}{10}, \tag{1}$$

where n_a is the number of antennas, M_b is the modulation, C_r is the coding rate, L_m is the number of MIMO layers, and R is the number of resource blocks per user. The scaling factor σ is the ratio of the computational complexity of split 2 over the complexity of all baseband functions according to [30].

The latency must comply to the split 2 requirements to guarantee the correct operation of the network functions between the DU and the CU. The propagation and processing latency components. The propagation latency (τ_e) is the time for the signal to propagate from an end to the other end of the fiber links. It is computed as the fiber link length divided by the speed of the light in glass (~200,000 km/s). The processing delay (ρ_n) is based on the CPU frequency, CPU capacity, and CU processing capacity [31]. The processing latency of the baseband VNFs in the CU associated to split 2 at a particular node $n \in N$ is defined as:

$$\rho_n = \frac{\pi}{f_{CPU} \cdot C_n}, \tag{2}$$

where f_{CPU} is the CPU frequency.

Table 1
List of decision variables and parameters of the MILP

Parameters	
n_d	Node source of demand d
n_{gw}	Gateway node
P_{idle}	Server idle power consumption
P_{max}	Server maximum power consumption
P_{card}^{CU}	Power consumption of the interfaces of radio-network unit CU
P_{infra}	Power consumption of the infrastructure
P_{tx}	Power consumption of the optical transponder
C_n	Node processing capacity
η_n	Servers per node
C_e	Maximum capacity of each virtual link $e \in E$
π	Processing workload of split 2
ρ_n	Processing latency of split 2 at node n to process 1 Gbit/s
τ_e	Propagation delay over link e
λ_d^u	Predicted traffic of demand d from u to $u+1$
t^{max}	Maximum allowed latency of functional split 2
$\mathcal{M}$	Very large number
Decision Variables	
w_n	Binary variable taking 1 if node n is active
$x_{e,u}^d$	Binary variable taking 1 if virtual link e carries demand d from radio-network unit u to the following unit
$z_{n,u}^d$	Binary variable taking 1 if node n processes demand d as radio-network unit u
T_n^d	Total processing latency for demand d at node n

3.2. MILP model

This optimization is a simplification of the MILP proposed in our previous work [5]. Solving the MILP model is NP-hard and it takes several seconds to compute the result. In order to reduce the execution time, this paper assumes the placement of CU VNFs only, decreasing the complexity of the problem.

Objective function

The optimization formulation goal is to minimize the overall power consumption at a determined time slot:

$$\min P_h \tag{3}$$

$$P_h = P_{net}\left(x_{e,u}^d, w_n\right) + P_{node}\left(w_n, z_{n,u}^d, \lambda\right) \tag{4a}$$

$$P_{net} = \sum_{e \in E} 2 P_{tx} x_{e,u}^d + \sum_{n \in N} P_{card}^{CU} w_n \tag{4b}$$

$$P_{node} = \sum_{n \in N} \eta_n \left(P_{infra} w_n + P_{idle} w_n + \frac{(P_{max} - P_{idle})}{C_n} \sum_{d \in D} \lambda_d^{DU} \pi z_{n,CU}^d \right) \tag{4c}$$

It depends on two main components: network and node equipment. The network power consumption (Eq. (4b)) includes the transponders and the midhaul and backhaul (x-haul) interface cards. Transponders consume P_{tx} whenever active for optical-electrical-optical conversion. The x-haul interfaces are responsible for connecting the nodes hosting the CUs to the DUs and to the gateway. The second term (Eq. (4c)) computes the total node power. It includes the power used by the servers to process the traffic and the infrastructure power in the selected nodes. The node infrastructure power P_{infra} considers the components of the node site, *i.e.* cooling, light. The server component accounts for the power consumed whenever the server nodes process CU functions. Therefore, the servers consume a certain idle power when active and, as the traffic increases, the power increases linearly.

Constraints

The objective function is subject to the following constraints. Eq. (5) routes the demands from the DU to the gateway, ensuring that it passes through the node hosting the CU that processes this demand. It denotes the flow conservation to assign a demand to virtual links $x_{d,e}^u$. The

virtual links between two radio-network units assigned to a demand arriving at a certain node (E_n^+) is equal to the number of outgoing links (E_n^-), unless the node hosts the radio-network unit in which demand starts or ends.

$$\sum_{e\in E_n^+} x_{e,u}^d - \sum_{e\in E_n^-} x_{e,u}^d = \begin{cases} z_{n,u'}^d - z_{n,u}^d \text{ if } n \text{ can host } u, u' \\ -z_{n,u}^d \text{ if } n \text{ can host } u, \text{ not } u' \\ z_{n,u'}^k \text{ if } n \text{ can host } u', \text{ not } u \\ 0 \text{ if } n \text{ cannot host } u, u' \end{cases} \quad (5)$$

$$u' = u + 1, \forall n \in N, \forall u, u' \in U \setminus \{GW\}, \forall d \in D$$

Eq. (6) ensures that a single CU processes each demand. Eq. (7) enforces that each demand is processed by the DU located in its origin, and Eq. (8) determines that the gateway is the destination of all demands.

$$\sum_{n\in N} z_{n,CU}^d = 1, \quad \forall d \in D \quad (6)$$

$$z_{n,DU}^d = \begin{cases} 1, & \text{if } n = n_d \\ 0, & \text{otherwise} \end{cases}, \quad \forall n \in N, \forall d \in D \quad (7)$$

$$z_{n,GW}^d = \begin{cases} 1, & \text{if } n = n_{gw} \\ 0, & \text{otherwise} \end{cases}, \quad \forall n \in N, \forall d \in D \quad (8)$$

Eq. (9) activates the variable w_n whenever a node is used, *i.e.* if n hosts a CU.

$$\mathcal{M} w_n \geq \sum_{d\in D} z_{n,CU}^d \geq w_n, \quad \forall n \in N \quad (9)$$

Eq. (10) limits the total traffic carried by virtual link e to the link capacity. Eq. (11) restricts the processing workload of all demands processed in a node.

$$\sum_{d\in D} \sum_{u\in U\setminus GW} x_{e,u}^d \leq C_e, \quad \forall e \in E \quad (10)$$

$$\sum_{d\in D} \lambda_d^{DU} \pi z_{n,CU}^d \leq C_n, \quad \forall n \in N \quad (11)$$

Eq. (12) bounds the propagation delay on the links and processing time at CU to the split requirements, *i.e.* $t^{max} = 1.5$ ms. Eq. (13a) and (13b) determine the processing delay T_n^d.

$$\sum_{e\in E} \tau_e x_{e,DU}^d + \sum_{n\in N} T_n^d \leq t^{max}, \quad \forall d \in D \quad (12)$$

$$t^{max} z_{n,CU}^d \geq T_n^d \geq 0 \quad (13a)$$

$$\sum_{d'\in D} \lambda_d^{DU} \rho_n z_{n,CU}^{d'} \geq T_n^d \geq \sum_{d'\in D} \lambda_d^{DU} \rho_n z_{n,CU}^{d'} - t^{max}\left(1 - z_{n,CU}^d\right) \quad (13b)$$

3.3. Heuristic

As shown in [5] the MILP model is a NP-hard problem. Hence, to obtain a faster model to be integrated in the black-box approach described in Section 4, we utilize the heuristic algorithm presented in [5] to compute the CU placement and routing respecting the latency and capacity constraints. In brief, the algorithm computes the placement of CUs based on the centrality of the network nodes with respect to the gateway. The algorithm sorts the nodes based on the centrality weight, computed based on the betweeness of each node and the number of transponders needed between the source of demand d to n and from n to the gateway GW:

$$C_B(n_d, n) = \frac{\sum_{n'\in N} \frac{|\sigma(n', GW|n)|}{|\sigma(n', GW)|}}{2\left(|\sigma(n_d, n)| + |\sigma(n, GW)|\right)} \quad \forall k \in K, n \in N, \quad (14)$$

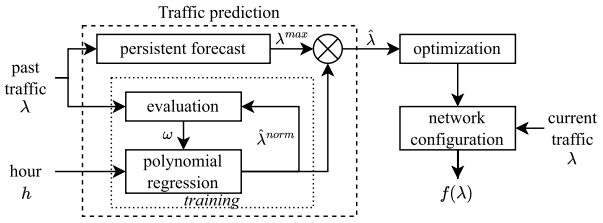


Fig. 1. Traditional two-step approach. The machine learning algorithm is trained based on the prediction error compared to the real value. The trained model forecasts the expected traffic $\hat{\lambda}$, which is then used by the MILP. The loss function $f(\lambda)$ can be computed only in the real time application. This approach does not guarantee constraint compliance nor minimum power.

where $\sigma(n', GW)$ is the number of shortest paths between nodes n' and GW, $\sigma(n', GW|n)$ is the number of shortest paths between nodes n' and GW passing through node n, n_d is the source of demand d. For each demand, the algorithm selects the node with highest C_B that satisfies all latency and capacity constraints. Then, it routes the demands applying Dijkstra algorithm to compute the shortest path. Please refer to [5] for more details regarding the heuristic algorithm.

4. Black-box optimization

This paper provides an anticipated planning of CU VNF placement ensuring compliance to all constraints in an automated way. The framework based on black-box techniques determines the weights of a polynomial regression according to the outcomes of a mathematical program.

4.1. Comparison of training methods

Unlike traditional optimization techniques, which rely upon a known mathematical model, black-box methods optimize a certain objective function based on data that may be noisy and without a well-defined model. In this type of problems, we query the value $f(x)$ of a certain point x, but the analytic form of the function $f(x)$ is not known. Surrogate models may be used to fit a model $m(f(x))$ based on past measurements x and $f(x)$ to compute other solutions x' [32]. $m(f(x))$ is a probabilistic model that map the solutions to a probability score $\mathcal{S}(f(x))$, which determines the efficiency of the solution. Bayesian optimization with Gaussian processes are inappropriate to solve problems in which thousands of iterations are needed because of its high computational complexity [33]. Therefore, for this application, we use HyperOpt [28], an open-source algorithm developed as a Python library that was designed for large-scale optimization models. Its surrogate model considers a Tree-structured Parzen Estimator (TPE) [34], which can manage large-scale problems well and it can also handle an extensive number of iterations. This technique is suitable for the problem this paper solves as there is no well-defined analytic function that relates the optimization outcomes to the predicted traffic. HyperOpt is a type of Bayesian optimization algorithm which uses expected improvement, for which Frazier et al. [35] indicated that rates of convergence are not known in general. Nevertheless, literature shows that these algorithms perform better than random search in all kinds of practical applications, with HyperOpt being among the most efficient Bayesian optimization algorithms that can find acceptable solutions quickly for a wide variety of tasks [36]. In Section 5.2, we verify whether HyperOpt can quickly find an acceptable solution on a simpler traffic prediction task.

Fig. 1 illustrates the traditional use of machine learning in optimization. Figs. 2 and 3 show the black-box-based optimization framework proposed in this paper described in detail in the following subsection.

As shown in the left-hand side, the traditional two-step approach trains the machine learning algorithm using past traffic information. It aims to maximize the prediction accuracy, *i.e.* the error of the predicted

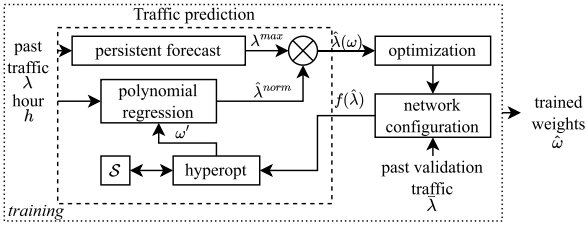


Fig. 2. Training of proposed black-box optimization approach. The machine learning algorithm is trained based on the loss function $f(\hat{\lambda})$, given by the optimization module and the past validation traffic, using a black-box algorithm (HyperOpt).

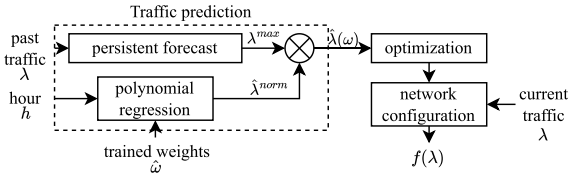


Fig. 3. Testing of proposed black-box optimization approach.
It ensures compliance with the constraints and minimizes the placement power.

traffic computed using a set of weights ω with respect to the real traffic. Then, the optimization uses the predicted traffic $\hat{\lambda}$ for a certain time slot to compute the CU placement and traffic routing. In particular, a polynomial regression algorithm predicts the normalized traffic of each node $\hat{\lambda}^{norm}(\omega)$ using a set of weights ω. The normalized traffic is then scaled using persistent forecasting, *i.e.* we use the maximum traffic observed in the previous day λ^{max} to determine the predicted traffic [8].

Alternatively, the proposed black-box optimization approach trains the machine learning algorithm based on the evaluation of the placement. In Fig. 2, the left block performs traffic prediction using the hour of the day h and the traffic of the previous days λ as input. We deploy the same prediction algorithm as previously described for the traditional two-step approach, *i.e.* computing the normalized traffic $\hat{\lambda}^{norm}$ using a polynomial regression and the scaling factor λ^{max} with persistent forecast. The expected traffic $\hat{\lambda}(\omega)$ is given as input to right block responsible for computing the CU VNF placement and routing (MILP or heuristic). Then, it applies the real validation traffic $\bar{\lambda}$ at this particular hour to determine the loss function $f(\bar{\lambda})$, representing the feedback of the number of constraints violated and how close the power consumption is to the oracle. We emphasize that the loss function f is computed by utilizing past validation traffic instead of relying on real-time traffic data as in the traditional two-step approach (see Fig. 1). The oracle is determined as the optimal placement using the real traffic. $f(\omega)$ is given as input to the HyperOpt algorithm to map it to a probability score to determine the next weight value ω' from a search space S.

Finally, after training the prediction algorithm, the final weights $\bar{\omega}$ can be used to predict the future placement as shown in Fig. 3. With this framework, it is enough to give the peak traffic of the past few days and the hour of the desired reconfiguration as input to run the optimization algorithms to determine the future network configuration. Section 4.2 details the black-box framework.

4.2. Black-box optimization for CU VNF placement

This section describes the black-box optimization framework based on HyperOpt [28]. The rationale behind this framework is that too tight prediction may cause constraint violation or excessive power when applying the real value, as we showed in our previous work [8]. In CU VNF placement, the underestimation of the expected traffic leads the optimization algorithm to use less nodes than necessary to accommodate the real traffic. Therefore, we observe violations of capacity and latency constraints. As inferred in [8], it is necessary to overestimate the traffic to ensure the compliance with all constraints. However, if the expected traffic is overestimated excessively, the model uses more nodes than necessary to host CU VNFs, increasing the power consumption.

The proposed approach calculates the polynomial regression to predict the traffic for each node based on the optimization result, *i.e.* verifying how many constraints were violated and the difference of power consumption with respect to the oracle. To speed up the learning phase, the algorithm limits the search for optimal parameters around the polynomial regression weights. For this, HyperOpt computes a parameter δ_b to be summed to the weights of the polynomial regression ω_n^b as Eq. (15) instead of calculating the polynomial regression weights ω_n for each node directly. Note that the initial polynomial regression weights ω_n^b are computed as in the traditional two-step approach, *i.e.* according to the traffic prediction error.

$$\hat{y}(x) = \sum_{b=0}^{B} (\omega_n^b + \delta_b) x^b, \tag{15}$$

where B is the polynomial regression degree. Algorithm 1 describes the pseudo-code of this framework.

Algorithm 1 Black-box optimization algorithm.

Input: Network graph $G(N, E)$, # evaluations $evals$, # startup evaluations $startup$, real traffic λ, hours H, degree of polynomial regression B
Output: Polynomial regression weights

```
1:  for all h ∈ H do
2:      Compute oracle placement by running the optimization model with real traffic λ (Section 3): x^d_{e,u}, w_n, z^d_{n,u}
3:      Get hourly power consumption P_h Eq. (4a)
4:  end for
5:  for all n ∈ N do
6:      Get polynomial regression weights ω^b_n of demand starting at node n and polynomial of degree B
7:  end for
8:  for i = 0; i ≤ evals; i++ do
9:      for all n ∈ N do
10:         for all b ∈ B do
11:             if i = 0 then
12:                 Start space δ_{n,b} ← 0
13:             else if i ≤ startup then
14:                 Select δ_{n,b} randomly from search space S
15:             else
16:                 Select δ_{n,b} from search space S using TPE based on previous computed costs f(δ)
17:             end if
18:         end for
19:     end for
20:     Get the optimization cost f(δ)_i using Algorithm 2
21: end for
```

The first step is to compute the oracle CU VNF placement, *i.e.* the placement with the real traffic (lines 1 to 4). For each hour in the set of hours H used for training, it runs the MILP or heuristic with the real traffic λ_d of the demands (line 2) to obtain the placement and routing variables $x^d_{e,u}$, w_n and $z^d_{n,u}$. Then, it gets the hourly oracle power consumption as Eq. (4a) (line 3). Next, it calculates the polynomial regression weights ω_n^b for each node n considering the traffic of demand d, which is received by n, over the period H (lines 5 to 7). After this, the HyperOpt algorithm starts (lines 8 to 21). We set $evals$ evaluations in which the algorithm searches for a solution. In the first iteration, we force the algorithm to use the polynomial regression traffic to compute the placement, *i.e.* we set δ to 0 for all nodes and weights (lines 11 and

12). Then, for the following *startup* jobs, HyperOpt randomly selects the parameter $\delta_{n,b}$ for each node n and for each polynomial regression weight b (lines 13 and 14). It picks each value from a probabilistic search space $\mathcal{S}$ of possible configurations. In this work, it is important that the search space is always positive to ensure that the algorithm will overestimate the predicted traffic. Nevertheless, if the traffic is too high, the placement algorithm will not find any feasible solution due to the system constraints. Therefore, the search space is set as the absolute value of a normal distribution with mean 0 and standard deviation of 0.167, such that 68.2% of the values is lower than 0.5:

$$\mathcal{S} = |\mathcal{N}(0, 0.167)|. \tag{16}$$

For the remaining iterations, HyperOpt uses TPE [34] to select other candidate values for δ (line 16). After selecting $\delta_{n,b}$ for each node $n \in N$ and for each weight $b \in B$, it calls Algorithm 2 (line 20) to obtain the loss of this selection.

Algorithm 2 Optimization cost

Input: Parameters δ, polynomial regression weights ω and degree B, nodes N, real traffic λ, hours H
Output: Cost $f(\delta)$
1: **for all** $n \in N$ **do**
2: **for all** $h \in H$ **do**
3: Get maximum traffic of previous day λ_d^{max} of demand d starting at node n (Eq. (17b))
4: Compute predicted traffic $\hat{\lambda}_{d,h}$ (Eq. (17c))
5: **end for**
6: **end for**
7: **for all** $h \in H$ **do**
8: Get CU VNF placement by running the optimization with predicted traffic $\hat{\lambda}$ (Section 3): $\hat{x}_{e,u}^d, \hat{w}_n, \hat{z}_{n,u}^d$
9: Compute hourly power consumption P_h with predicted traffic $\hat{\lambda}$ (Eq. (4a))
10: Get number of MILP constraint violations: c_h
11: **end for**
12: Compute cost $f(\delta) \leftarrow \frac{1}{H} \sum_{h \in H} \frac{|P_h - \hat{P}_h|}{P_h} + \sum_{h \in H} c_h$

To determine the loss function, it runs the MILP or the heuristic model with the predicted traffic and then verifies the result when applying the real traffic. It starts by computing the predicted traffic using the new weights (lines 1 to 6) as explained in Section 4.1. The traffic prediction first determines the hourly normalized traffic $\hat{\lambda}_{d,h}^{norm}$ based on the hour of the day as in Eq. (17a). Then, it obtains the scaling factor, *i.e.* maximum traffic of the previous day λ_d^{max} as in Eq. (17b). Finally, it computes the total hourly traffic $\hat{\lambda}_{d,h}(\delta)$ by multiplying the normalized value $\hat{\lambda}_{d,h}^{norm}(\delta)$ and the scaling factor λ_d^{max} (lines 3 and 4):

$$\hat{\lambda}_{d,h}^{norm}(\delta) = \sum_{b=0}^{B} h^b(\omega_n^b + \delta_{n,b}) \tag{17a}$$

$$\lambda_{d,h}^{max} = \max_{h' \in H'} \lambda_{d,h'} \tag{17b}$$

$$\hat{\lambda}_{d,h}(\delta) = \lambda_d^{max} \cdot \hat{\lambda}_{d,h}^{norm}(\delta) \tag{17c}$$

where $H' = \left[24(\lfloor h/24 \rfloor - 2) + 1, 24(\lfloor h/24 \rfloor - 1)\right]$.

After computing the predicted traffic, the algorithm computes the CU VNF placement and the loss function (lines 7 to 12). For each training hour, it runs the MILP or the heuristic algorithm using the predicted traffic $\hat{\lambda}_{d,h}$ to obtain the placement and routing variables $\hat{x}_{e,u}^d$, $\hat{w}_n$, $\hat{z}_{n,u}^d$ (line 8). Then, it gets the hourly power consumption $\hat{P}_h$ as in Eq. (4a) applying the real traffic to the predicted placement (line 9). Next, it calculates the constraint violation penalty c_h. This metric determines the number of times the predicted placement is not compliant with the model constraints when applying the real traffic (lines 10). Consequently, to ensure that the constraints are not violated and that the power consumption is as close as possible to the oracle, the algorithm computes the loss $f(\delta)$ associated to the δ parameters of this iteration as:

$$f(\delta) \leftarrow \frac{\frac{1}{|H|} \sum_{h \in H} |P_h - \hat{P}_h|}{\sum_{h \in H} P_h} + \sum_{h \in H} c_h \tag{18}$$

This work assumes that satisfying all constraints is more important than reducing the power consumption because the former have a direct impact on the quality of service and network operation. Therefore, the first term (power) is smaller than the second term (constraint violation) unless there is full constraint compliance. The loss $f(\delta)$ computed using Algorithm 2 in each evaluation step is then used by the TPE algorithm to determine δ of the following iterations.

After training the polynomial regression weights using Algorithms 1 and 2, they can be used to predict the traffic and compute the future CU VNF placement.

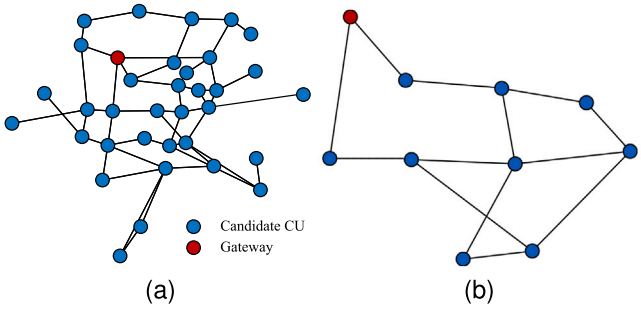


Fig. 4. Large (a) and small (b) network topology.

5. Simulation results and discussion

This section discusses the performance of the black-box-based optimization of CU VNF placement explained previously. First, it describes the simulation environment. Next, it assesses the performance of the HyperOpt algorithm in determining the polynomial regression weights. Then, it evaluates the black-box optimization compared to the traditional two-step approach highly used in the literature.

5.1. Simulation environment

Fig. 4 illustrates the network topology considered in the simulations. The large network topology (a) illustrates the topology inspired by a typical network covering a large metropolitan area from the Metro-Haul project [37]. It contains 34 nodes, among which one gateway and destination of all demands. The small topology (b) is a subset of the central nodes of the same topology, including the gateway. The remaining nodes that are used for computing and communication are equipped with one server of type Intel® Xeon® Gold 6134 with 8 cores, 3.7 GHz, and 537.6 GFLOPS of processing capacity [38]. The server consumes 130 W when idle, reaching a maximum power consumption of 870 W when operating at full capacity. The nodes have radio-network unit cards that consume 10 W for the midhaul, and 1 W for the backhaul [39]. The antennas are configured according to the specifications of [40], *i.e.* they use 64QAM modulation, 2 × 2 MIMO, coding scheme of 1 in the uplink, and a single user per transmission time interval sending 100 resource blocks of data. The nodes are connected through bidirectional fiber links, containing at most 80 wavelengths. The nodes are also equipped with a set of 100 Gbit/s transponders, each of which consumes 110.4 W [41] whenever active. Table 2 summarizes the simulation input parameters.

We obtained the mobile traffic per base station by merging the call detail record from the TIM Big Data Challenge dataset [42] to the base

Table 2
Input parameters used in the simulations.

Parameters	Description	Value
f_{CPU}	CPU frequency	3.7 GHz
C_n	Node processing capacity	537.6 GFLOPS
P_{idle}	Server idle power consumption	130 W
P_{max}	Server maximum power consumption	870 W
P^{CU}_{card}	X-haul interface cards power consumption	11 W
M_b	Modulation	64
n_a	Number of antennas	2
L_m	Number of MIMO layers	2
C_r	Code rate	1
R	Number of resource blocks	100
C_e	Wavelengths per fiber	80
P_{tx}	Transponder power consumption	110.4

stations position [43] through Voronoi tasselation. Next, we combined this dataset to the topology shown in Figure 4 and the maximum capacity of each node to generate per-node hourly traffic. For this, we created an algorithm that first associates the base stations to the nodes based on their estimated location using Voronoi tasselation. Then, the algorithm iteratively reassigns the base stations such that the ratio between the traffic of all base stations assigned to a node and the total traffic is as close as possible to the expected capacity of the node.

The dataset to train the polynomial regression contains the traffic of the last weekdays of November. The testing dataset consists of the following three weekdays. Moreover, the baseline for traffic prediction is polynomial regression of degree 4. We developed this framework using Python, and solved the MILP model with CPLEX.

5.2. Assessment of HyperOpt for traffic prediction

First, this section assesses the performance of the HyperOpt algorithm for finding good weights for the polynomial regression function and evaluate its convergence. The analytic function to compute polynomial regression is well known and the solution is optimal. Thus, it is possible to verify if HyperOpt makes mistakes in determining the weights, by comparing its predictive performance to a standard polynomial regression learning method. The validation of HyperOpt to predict the weights of a polynomial regression applies a search space with the uniform distribution:

$$S' = \{\omega_0 = \mathcal{U}(0,1), \omega_i = \mathcal{U}(-10,10)\}, \quad \forall i \in \{1,2,3,4\} \tag{19}$$

where $\omega_i, i \in \{0,1,2,3,4\}$ represent the polynomial regression weights. In addition, the HyperOpt loss function is the Mean Squared Error (MSE):

$$f(\delta)' = \frac{1}{H}\sum_{h\in H}\left(\lambda_h - \hat{\lambda}_h\right)^2 \tag{20}$$

where H is the set of training hours, λ_h is the hourly real traffic and $\hat{\lambda}_h$ is the hourly predicted traffic.

Fig. 5 depicts the variation of the loss function according to the weights selected by the HyperOpt algorithm at each iteration during the training phase. With Bayesian optimization algorithms such as HyperOpt, it is standard practice to consider only the best result up to a certain iteration in order to encourage exploration of the search space [28]. This best loss is plotted as well. Figs. 6 and 7 illustrate the real traffic (oracle) compared to the polynomial regression computed with traditional methods (standard regression) and with HyperOpt using the training and testing data.

These results show that after about 3000 iterations, HyperOpt finds a solution that is close to standard polynomial regression, with the MSE with respect to the oracle of 0.01948 versus 0.01041 of the standard polynomial regression. After 10,000 iterations, the MSE lowers to 0.01567. Using the computed weights with the testing dataset, the MSE of the HyperOpt traffic prediction is 0.01092 and the standard polynomial regression is 0.00519. These results show that, although HyperOpt does not reach the exact solution computed with the analytic polynomial regression method, it can find good predictors.

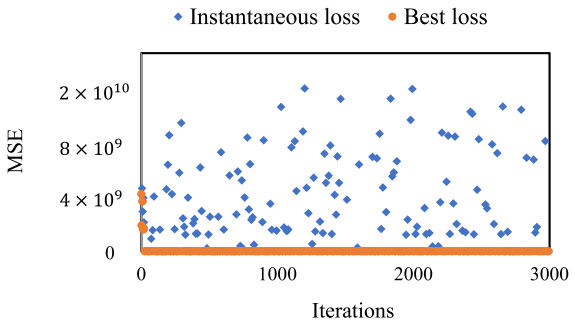


Fig. 5. MSE loss based on the result with the iteration values.

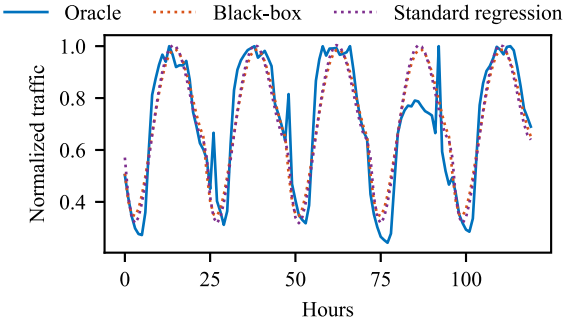


Fig. 6. Traffic predicted using the oracle, standard polynomial regression and HyperOpt during the training phase.

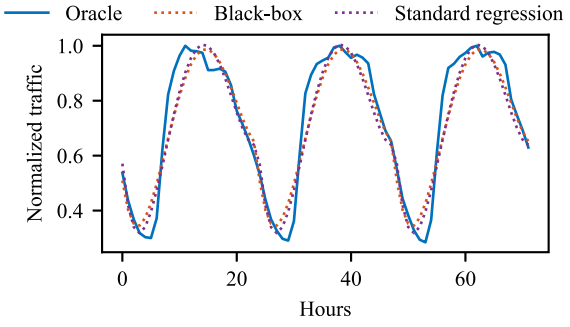


Fig. 7. Traffic predicted using the oracle, standard polynomial regression and HyperOpt during the testing phases.

5.3. Evaluation of the black-box-based optimization with the small topology

The advantage of using HyperOpt over standard polynomial regression is that it allows to optimize the weights using the optimization results instead of the traffic prediction error. This is not possible in standard polynomial regression because the relation between the regression weights and the outcome of the optimization is not analytically defined. The next step is to evaluate the quality of solutions found when performing such optimization. The parameters of the HyperOpt algorithm both with MILP and heuristic algorithms are $evals = 2000$ and $startup = 200$, meaning that it selects random values for $\delta_{n,b}$ from the search space in the first 200 iterations, and that the algorithm runs for 2000 iterations in total. To validate the black-box optimization results, this section shows the average results over 5 replications of the simulations using MILP and heuristic algorithms and their respective confidence intervals.

5.3.1. HyperOpt loss

The goal of the framework proposed in Section 4 is to apply the result of the optimization to determine the weights of the polynomial

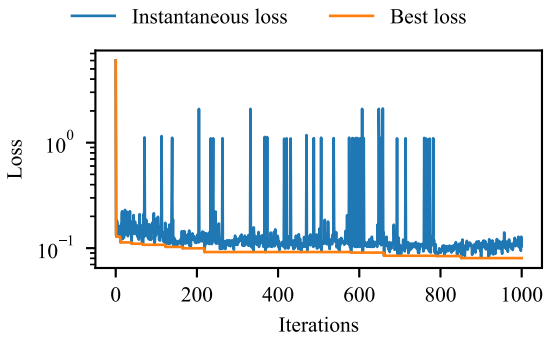


Fig. 8. Loss variation over the iterations.

Table 3
R^2 score and MSE of black-box and polynomial-regression predicted traffic with training set for the small topology.

Node	Standard regression		Black-box with MILP		Black-box with heuristic	
	MSE	R^2	MSE	R^2	MSE	R^2
1	5.37	0.72	40.56	0.98	12.23	0.84
2	8.99	0.67	43.89	−0.27	34.58	0.99
3	14.49	0.88	76.9	−0.45	182.22	−5.27
4	0.65	0.9	48.62	0.7	117.44	−1.46
5	22.05	0.23	114.44	−3.42	111.74	0.35
6	1.05	0.95	20.84	0.73	20.32	0.71
7	64.1	0.55	93.2	−2.01	112.41	−0.47
8	2.73	0.1	6.71	0.96	17.37	0.36
9	27.13	0.34	58.44	−7.71	110.27	0.33

regression models used for forecasting the traffic of each topology node. Therefore, HyperOpt gets the optimization loss associated to the former node-traffic prediction to compute a new set of weights. Fig. 8 depicts the variation in the loss function along the iterations. This graph shows that the best loss value (orange curve) obtained as in Eq. (18) is under 1 after few iterations. This result means that the black-box algorithm finds a feasible solution during the startup evaluations. Since solutions without constraint violation penalties has already been found, the main goal of HyperOpt after the startup phase is to reduce the power consumption. Therefore, it uses the TPE algorithm to improve the previous results and find a solution whose power consumption is as close as possible to the optimal solution. The final solution used in the remaining of this section was achieved at iteration #851. After that iteration, the best loss does not decrease any further. As mentioned in the previous subsection, it is standard practice to consider only the best result obtained by the HyperOpt algorithm.

5.3.2. Traffic prediction performance

The next step is to use the weights computed by HyperOpt to forecast the traffic of each topology node and then compute the CU VNF placement. Hereinafter, the terms *black-box with MILP* and *black-box with heuristic* refer to the results using the MILP algorithm and with the heuristic algorithm, respectively. Tables 3 and 4 compare the traffic predicted using the black-box optimization framework and the standard polynomial regression to the oracle using the training and the testing sets, respectively. It uses the coefficient of determination (R^2 score) computed as Eq. (21) and the MSE metrics. Figs. 9 and 9(b) show the total oracle and the traffic predicted with black-box and standard polynomial regression in training and testing phases, respectively.

$$R^2 = 1 - \frac{\sum(Y_i - \hat{Y}_i)^2}{\sum(Y_i - \frac{1}{n}\sum Y_i)^2} \tag{21}$$

Compared to the polynomial regression, the black-box optimization performs worse in terms of prediction accuracy. Overall, the R^2 scores are significantly lower and the MSE values are significantly higher in

Table 4
R^2 score and MSE of black-box and polynomial-regression predicted traffic with testing sets for the small topology.

Node	Standard regression		Black-box with MILP		Black-box with heuristic	
	MSE	R^2	MSE	R^2	MSE	R^2
1	1.27	0.93	26.81	0.15	5.03	0.81
2	2.18	0.9	16.99	0.61	12.31	0.69
3	13.91	0.88	57.54	0.64	145.19	0.11
4	0.42	0.94	48.64	−0.82	118.47	−0.66
5	4.83	0.73	56.6	−0.03	56.39	−0.03
6	0.92	0.95	16.93	0.46	15.75	0.43
7	11.09	0.9	27.18	0.82	34.89	0.74
8	0.76	0.51	2.52	0.41	9.66	−0.81
9	6	0.8	19.51	0.68	51.35	0.34

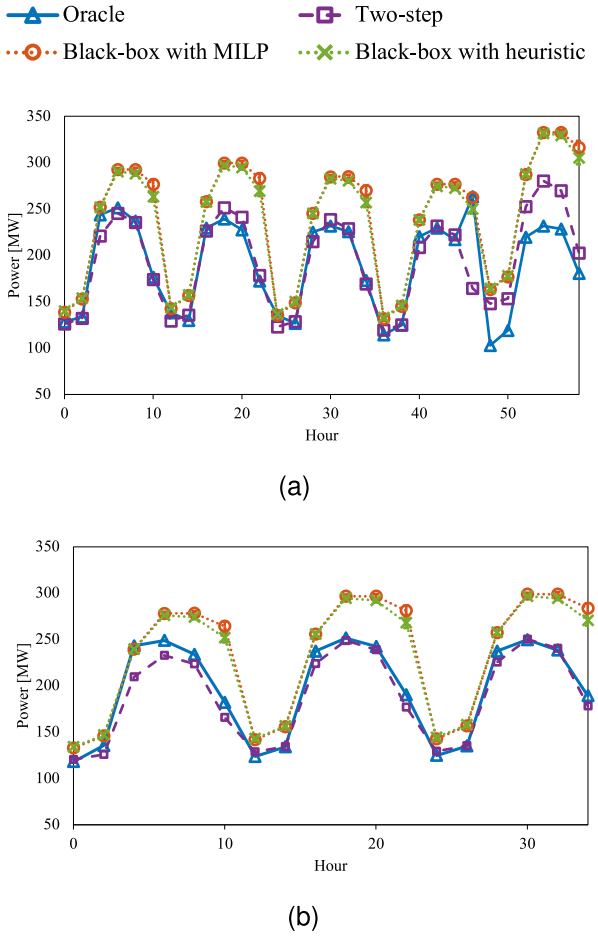


Fig. 9. Comparison of oracle, black-box, black-box with heuristic and regression traffic during training (a) and testing (b) phases using the small topology.

black-box with MILP and with heuristic, compared to the standard regression. In addition, several nodes present a negative coefficient of determination using the black-box optimization. This metric normally ranges between 0 and 1; however, models with very inaccurate predictions may have a negative R^2 score. In particular, nodes 5 present the worst results using black-box with MILP, with MSE and R^2 of 114.44 and −3.42, respectively. For the black-box with heuristic, node 3 has the worst performance with MSE of 182.22 and R^2 score of −5.27. These results indicate that the prediction at these nodes follow a very different pattern with respect to the real traffic. Overall, the total traffic over the network is also considerably impacted, as shown in the graphs of Figs. 9(a) and 9(b). The black-box traffic prediction represents the average over five simulations and the variance is lower than 1%. The

overall traffic is about 26% and 24% higher in the training phase using black-box with MILP and heuristic, and 19% and 18% in the testing phase. This overestimation of the traffic is due to the goal of the black-box optimization. Unlike Section 5.2, which aimed to optimize the traffic prediction, this section uses the black-box algorithm to optimize the placement of baseband functions, ignoring that the traffic prediction may be incorrect. Consequently, while detrimental for the prediction accuracy, the traffic overeestimation can lead to a more efficient placement. The following subsection shows how the black-box approach achieves this.

5.3.3. Optimization performance

Based on the traffic predicted using the weights computed with the black-box framework, this section verifies the performance of the CU VNF placement applying the black-box-based traffic prediction to the MILP model and to the heuristic algorithm. The first step is to compute the CU VNF placement and traffic routing using the predicted traffic. Then, apply the real traffic to verify the actual power consumption and the compliance with the constraints.

In order to evaluate the effectiveness of the black-box optimization, we compare the results with MILP and heuristic to the two-step approach in two scenarios, called hereinafter *two-step* and *two-step with buffer*. The *two-step* solution is the traditional predict and then optimize approach, in which the traffic estimated with the standard polynomial regression is used to compute the optimization. The *two-step with buffer* uses an artificial capacity buffer in the MILP model with standard polynomial regression to ensure feasibility in real time, *i.e.* it reduces the node processing capacity artificially in the MILP formulation until the predicted optimization result is compliant to all constraints in real time. Therefore, we rewrote the constraint in Eq. (11) as:

$$\sum_{d \in D} \lambda_d^{DU} \pi z_{n,CU}^d \leq C_n - buffer, \quad \forall n \in N \tag{22}$$

The minimum value to ensure that all constraints are respected in the training phase: $buffer = 35\%$.

Figs. 10(a) and 10(b) show the power consumption and constraint violation penalty obtained during the training phase. Figs. 10(c) and 10(d) present the same results for the testing phase. The oracle provides the best solution, consuming minimum amount of power and complying with all constraints; however, it uses traffic information that is not known in practice. The standard polynomial regression of the two-step approach can accurately predict the traffic, but it leads to constraint violations. By using an artificial buffer, the two-step with buffer provides a solution without constraint violation at the cost of higher power consumption. The black-box approach reduces the power consumption compared to the buffer whilst respecting all constraints.

Considering only the power consumption, the two-step without capacity buffer is the closest solution to the optimal, consuming in total about 930 kW less than the optimal with the training and test sets . This phenomenon happens because, in some time slots, the two-step solution consumes less power than the oracle. This reduction is a consequence of a misplacement of CU VNFs, which led the two-step solution to use less transponders or less nodes. Indeed, at all time slots with lower power consumption, the graphs in Figs. 10(b) and 10(d) show that there is a constraint violation penalty (slots 6, 34 and 46 for training and slots 6, 8, 10 and 16 for testing). Moreover, despite not impacting the power consumption, the two-step CU VNF placement based on the standard polynomial regression presents other time slots with capacity constraint violation. Consequently, the high prediction accuracy of the standard polynomial regression does not guarantee a feasible CU VNF placement with the real traffic.

As previously mentioned, we need to apply an artificial capacity buffer of 35% when optimizing the CU VNF placement using the standard polynomial regression to ensure that all constraints were respected during the training phase. As shown in the two-step with buffer curve of Fig. 10(d), this buffer also guarantees that the CU VNF placement is feasible in the testing phase. Nonetheless, this approach led to a high increase of 23.44% and 22.95% in power with respect to the oracle in the training and testing phases, respectively. This growth is driven by the number of nodes used, requiring at least one extra node at each time slot with respect to the oracle.

The black-box-based CU VNF placement obtains an intermediate solution in terms of power consumption using the MILP and the heuristic algorithms. Both solutions guarantee that all constraints are respected both during training and testing phases. Black-box with MILP consumes 31.34 MW and 21.32 MW more than the oracle in total during training and testing, respectively, representing less than 10% of increase. Black-box with heuristic presents a slightly higher power consumption with a difference of 50.99 MW in training and 27.54 MW in testing. With respect to the two-step with buffer, the black-box with MILP reduces the total power consumed during training and testing phases by more than 10% while the solution with the heuristic algorithm leads to a reduction of about 7% and 9%. The greater power is mostly caused by the number of nodes used during the instants with highest traffic. The different results obtained with the MILP and the heuristic algorithms are caused by the selection of different nodes, which leads to a greater number of used transponders; hence, the heuristic has higher power consumption.

HyperOpt is capable of training the traffic prediction depending on several factors from the model. By applying the prediction to the algorithms and, consequently, to the loss function, HyperOpt tries to overestimate the traffic generated at the different nodes. This traffic overestimation can also be seen as an artificial buffer calculated individually for each demand based on multiple optimization aspects. Consequently, the black-box optimization can learn a better buffer size than selecting a fixed value $buffer$ as shown in Eq. (22). Fig. 11 shows the theoretical capacity buffer applied by the black-box approach at each node compared to the two-step with static buffer. Please note that node #10 is not represented because it is the gateway. These graphs demonstrate that during the training and testing phases, the solution finds different node capacity buffer values. Instead of applying a static capacity buffer of 35% at all nodes and at all time slots, the traffic overestimation using the black-box framework enables a dynamic selection of the buffer. These results confirm that the proposed approach provides a solution that efficiently places the CU VNFs, with reduced power consumption compared to adding a fixed artificial buffer and respecting all constrains, because it was trained with the goal of finding an accurate placement model instead of an accurate prediction model.

The results presented in this section show that the use of the HyperOpt algorithm in the black-box optimization enables a more intelligent allocation of resources. The overestimation of traffic in some strategic nodes decreases the accuracy of the traffic prediction algorithm with respect to the standard regression. Nevertheless, it improves the final result: the allocation of resources during the actual deployment respects all system constraints without an excessive overestimation of resources as in the two-step with buffer approach.

5.4. Evaluation of the black-box-based optimization with the large topology

This section analyzes the black-box-based CU VNF placement with respect to the oracle and the two-step approach using the large topology. As previous explained, the complexity of the MILP algorithm lead to very high execution time; hence, the evaluation in this section considers only the use of the heuristic algorithm in all scenarios. Hereinafter, the term *oracle* refers to the heuristic CU VNF placement with the real traffic, and *two-step* and *two-step with buffer* are the heuristic using the standard regression prediction and the regression with buffer, respectively. To validate the black-box optimization results, this section shows the average results over 5 replications of the simulations and the confidence intervals.

Considering a polynomial regression algorithm of degree 4, the black-box optimization must compute a total of 170 parameters in

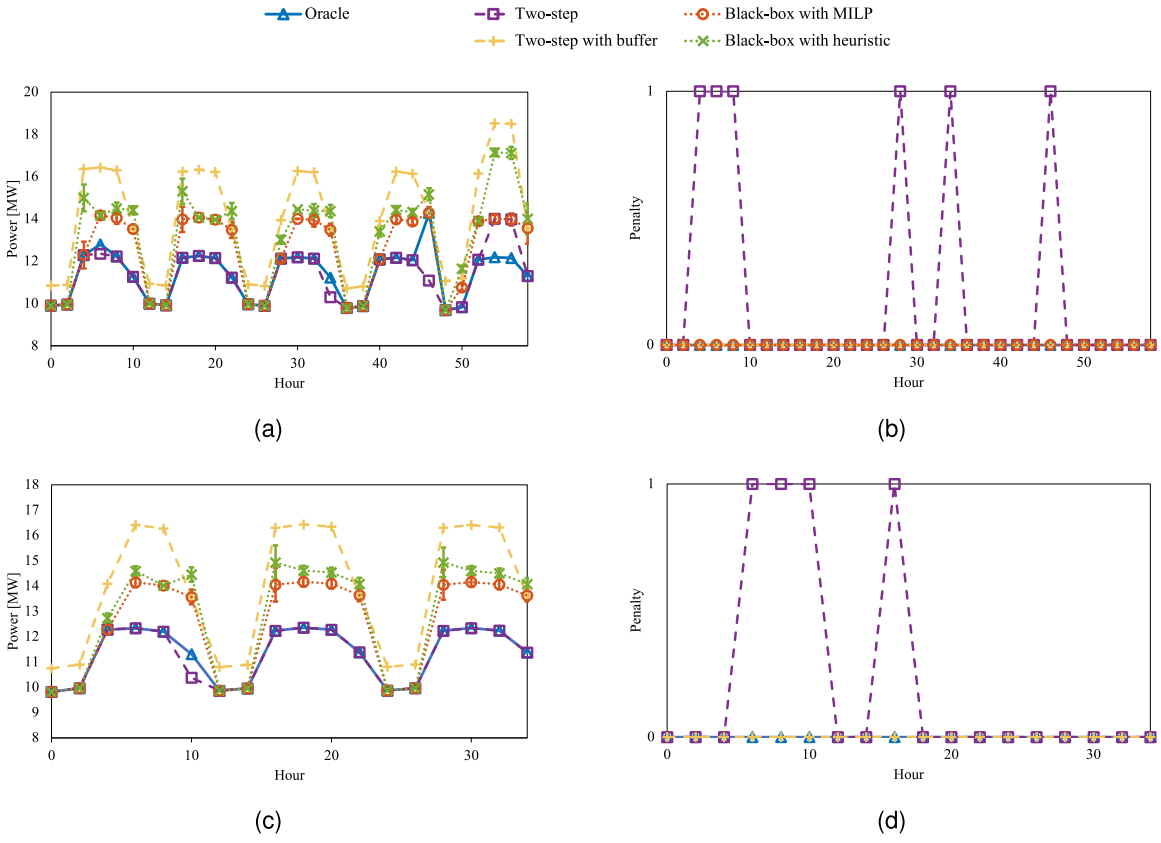


Fig. 10. Comparison of power consumption (a) (c) and constraint violation penalty (b) (d) of oracle, black-box, two-step and two-step with buffer during training (a) (b) and testing (c) (d) phase using the small topology.

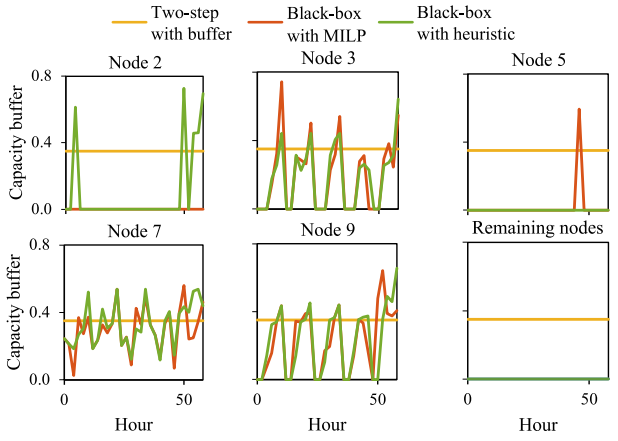


Fig. 11. Theoretical node capacity buffer at training phase using the proposed black-box approach and the two-step static buffer using the small topology.

this topology. Therefore, the setup of the HyperOpt algorithm for the training phase includes a higher number of evaluations: $evals = 2000$ and $startup = 200$. After running the black-box optimization for $eval$ iterations, the best loss computed as Eq. (18) was 7.2E−4, with standard deviation of 3.6E−5.

5.4.1. Traffic prediction performance

This subsection describes the accuracy of the traffic prediction using black-box and the standard polynomial regression with respect to the actual traffic. Tables 5 and 6 determine the MSE and R^2 score during training and testing, respectively. These tables show only the nodes with highest and lowest accuracy for visualization purposes.

Table 5
R^2 score and MSE of black-box and polynomial-regression predicted traffic with training set for the large topology.

Node	Standard regression		Black-box with heuristic	
	MSE	R^2	MSE	R^2
3	1.88	0.33	5.03	0.25
13	0.10	0.72	11.31	−4.07
19	0.71	0.55	23.00	−2.69
24	0.04	0.43	0.58	−0.84
26	64.10	0.55	149.15	0.32
30	6.69	0.33	161.95	−0.99
34	2.25	0.64	109.78	−1.49

Table 6
R^2 score and MSE of black-box and polynomial-regression predicted traffic with testing set for the large topology.

Node	Standard regression		Black-box with heuristic	
	MSE	R^2	MSE	R^2
1	2.51	0.46	3.59	0.6
3	0.75	0.6	2.21	0.57
13	0.07	0.81	12.08	−4.28
16	1.41	0.56	21.72	−0.91
19	0.14	0.89	17.47	−2.59
24	0.05	0.44	0.69	−0.88
30	2.29	0.62	129.13	−0.97
34	0.95	0.81	89.71	−1.35

Similar to the previous subsection, the black-box optimization presents significantly lower accuracy with respect to the standard regression model. All nodes have significantly worse MSE, which is

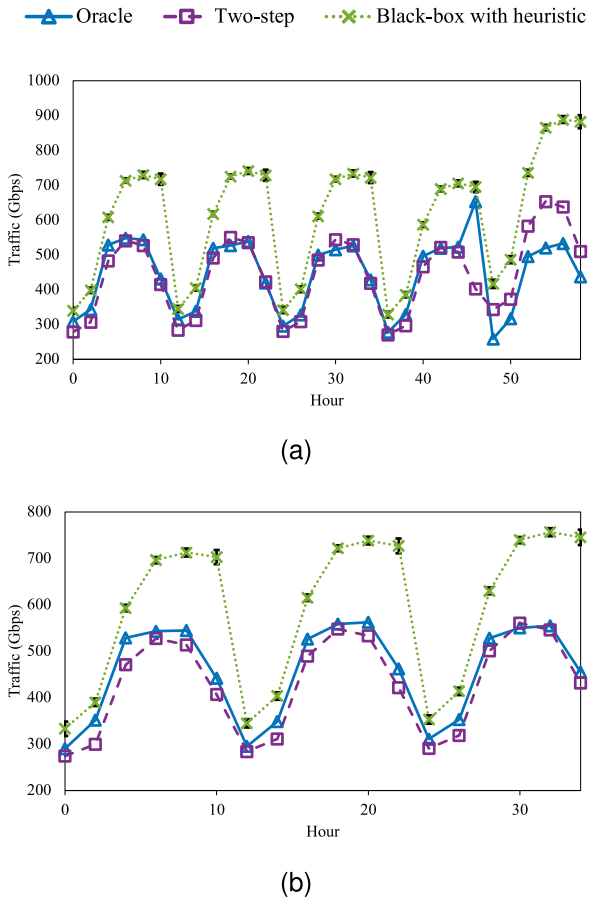


Fig. 12. Comparison of oracle, black-box and regression traffic during training (a) and testing (b) phases using the large topology.

related to an overestimation of the traffic, mostly in the peak hours. In addition, the R^2 score is overall lower than the standard polynomial regression, except for node 1 during testing phase. The low R^2 score indicates that the traffic curve follows very different pattern from the real traffic, in particular when it is negative. Figs. 12 illustrate this behavior. Figs. 12(a) and 12(b) present the training and testing traffic, respectively, of the actual traffic (oracle), standard regression (two-step) and HyperOpt prediction (black-box with heuristic) with its confidence level. The black-box traffic prediction is the average of five replications of the simulation and presents a variance of less than 1% for all timeslots. The standard regression used in the two-step approach follows fairly nicely the actual traffic. Instead, the black-box prediction overestimates the traffic in all time slots, with an average increase of 37% during training and 29% during testing.

5.4.2. Optimization performance

The traffic prediction described in the previous section was then fed to the heuristic algorithm to compute the CU VNF placement with the training and testing datasets using the trained weights. Figs. 13(a) and 13(b) show the power consumption and constraint violation penalty obtained during the training phase. Figs. 13(c) and 13(d) present the same results for the testing phase. As expected, the oracle consumes the least energy in average and complies with all constraints because it uses the real traffic. The two-step approach violate the constraints at several time slots both during training and testing. By using an artificial buffer of 40%, the two-step with buffer does not violate any constraints, but it has higher power consumption. The black-box is capable of complying to all constraints and also reduces the power consumption.

Table 7
Execution time in minutes with the small and large topology.

Topology	Algorithm	Total	Optimization	HyperOpt
Small	MILP	8206.5	8200	6.53
	Heuristic	14.5	9.47	5.03
Large	Heuristic	798	746.65	51.35

As observed with the small topology, the two-step has the most similar power consumption as the oracle, in average less than 1% higher. Nevertheless, it fails to respect the capacity constraints in several nodes at each time slot, reaching up to 4 violations with training and 5 with testing data. Thanks to the use of an artificial and static buffer of 40%, it is possible to ensure that all requirements are met at the cost of significantly higher power consumption (more than 18%). This result is due to the high number of active nodes, which increase both the processing and the network power consumption. The black-box approach also respects all constraints but at lower costs. Indeed, the consumption increases with respect to the oracle by 8% in training and testing, but it is lower than the two-step with buffer by about 6%. With respect to the multiple simulations for the black-box with heuristic, the power consumption variance represents on average 2.2% for each timeslot. It is worth noting that none of the training nor test replications violated any constraints.

This subsection confirms the capability of HyperOpt to train a traffic prediction algorithm tailored to the final application results. Instead of using a static underestimation of available resources as the two-step with buffer approach, the black-box solution provides an intelligent overestimation of traffic. Hence, it allows the optimization algorithm to correctly plan the CU VNF placement beforehand in an accurate and more efficient manner.

5.5. Analysis of the execution time

This section evaluates the execution time of the proposed approach. Table 7 shows the execution time with the small and with the large topology. *Optimization* presents the total time to run the optimization algorithms for all iterations. *HyperOpt* indicates the total time for HyperOpt to select the parameters. *Total* refers to the total training time, *i.e.* optimization and parameter selection.

The MILP model computes the CU VNF placement for all training time slots per iteration in approximately 490 s in average using the small 10-node topology. Instead, it takes about 600 ms to run the heuristic algorithm for all time slots per iteration. After calculating the CU VNF placement with the MILP or heuristic algorithms, HyperOpt selects the parameters for the following iteration in several milliseconds. In average, the computational time of each HyperOpt iteration is less than 400 ms, reaching a total HyperOpt execution time of 6.5 and 5 min using the MILP and heuristic algorithms, respectively.

In the large topology, the higher number of nodes lead to longer execution time. The heuristic algorithm computes the CU VNF placement for each time slot in approximately 1.5 s, resulting in more than 12 h to compute the placement for all 5000 iterations. With respect to the HyperOpt algorithm, as detailed in [28], the computational time increases quasi-linearly with the number of iterations. Hence, the greater number of iterations increases the average execution time of the algorithm to 2 s per iteration.

Based on these results, it is clear that the bottleneck is the evaluation of the MILP objective function. It may take several days to correctly train the model and hours to compute the CU VNF placement of an entire day. In the scenario in which a network operator manages a small topology, the solution with the MILP model may be appropriate. It is possible to train the algorithm every week to obtain more accurate placement for the following week. However, it does not scale for larger networks because such datasets would require more iterations for training and longer computation of the CU VNF placement.

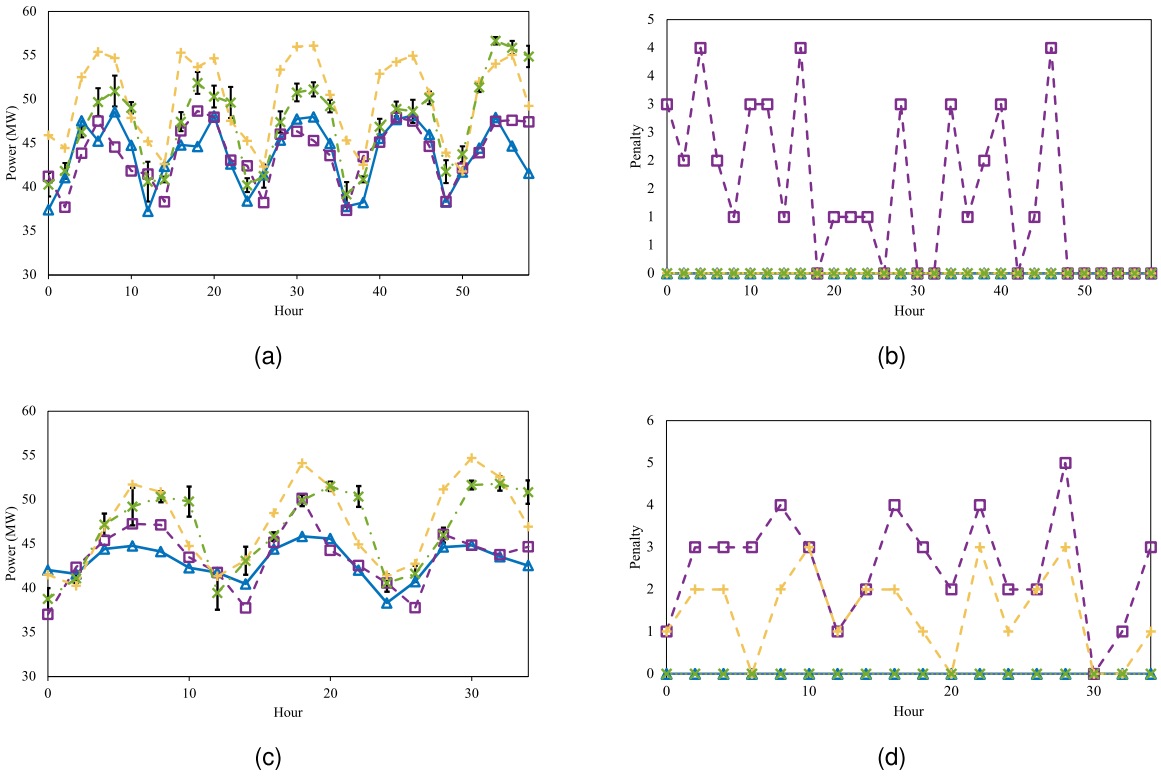


Fig. 13. Comparison of power consumption (a) (c) and constraint violation penalty (b) (d) of oracle, black-box, two-step and two-step with buffer during training (a) (b) and testing (c) (d) phase using the large topology.

In scenarios in which the network operators manage a topology with a greater number of nodes, the use of black-box with heuristic is more suitable. The training is complete in approximately 13 h using the traffic of a week as a training set. This corresponds to a few kWh at most used for computations,[1] while the resulting power consumption for one week is reduced by over 80 MWh on the test set, compared to the two-step with buffer approach. So the computational costs of the black-box with heuristic approach are negligible compared to the potential power savings. Moreover, the fast computation time also enables the use of larger datasets containing the information of more days to obtain more accurate models.

6. Final remarks

Traditionally, anticipatory optimization in 5G RAN first predicts traffic exploiting an accurate traffic prediction model and then performs the resource allocation. However, as shown in this paper, although machine learning algorithms can forecast traffic with little error, it does not ensure that the resource allocation is efficient when applying the actual traffic. Indeed, prediction accuracy itself generated violation of constraints when resources allocated on the basis of the predicted traffic are used to support the real demands. This paper describes a novel black-box-based optimization model to perform CU VNF placement. The goal is to exploit the outcomes of the optimization to train a polynomial regression model that performs traffic prediction. This model uses the HyperOpt algorithm to train the polynomial regression weights to reach the minimum cost, which is computed based on the power consumption and the penalty related to constraint violation. The results show that this approach can effectively place CU VNFs even if it significantly reduces the traffic prediction accuracy. The placement obtained during training and testing phases is feasible at all time slots, unlike the traditional two-step approach. Indeed, the goal of the black-box framework is to overestimate the traffic to guarantee the compliance with all constraints when the real traffic is applied. Moreover, it reduces the power consumption by more than 10% with respect to the two-step approach with an artificial capacity buffer. These results confirm that the proposed approach provides a more efficient CU VNF placement compared to traditional methods. The approach proposed in this paper is very promising as it can potentially be adapted for a multitude of applications and to train many machine learning algorithms: this aspect will be investigated in future works.

[1] Estimated using http://calculator.green-algorithms.org/.

Acronyms

3GPP	3rd Generation Partnership Project
BBU	Baseband Unit
CRAN	Centralized RAN
CU	Central Unit
DRAN	Decentralized Radio Access Network
DU	Distributed Unit
GW	Gateway
MILP	Mixed Integer Linear Programming
MIMO	Multiple-Input Multiple-Output
MSE	Mean Squared Error
RAN	Radio Access Network
RRU	Remote Radio Head
SPO	Smart Predict then Optimize
TPE	Tree-structured Parzen Estimator
VNF	Virtual Network Function

CRediT authorship contribution statement

Ligia Maria Moreira Zorello: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Laurens Bliek:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Sebastian Troia:** Writing – review & editing, Writing – original draft, Conceptualization. **Guido Maier:** Writing – review & editing, Writing – original draft, Conceptualization. **Sicco Verwer:** Writing – review & editing, Writing – original draft, Conceptualization.

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.

Data availability

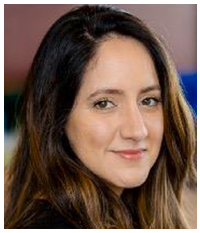
The data that has been used is confidential.

Acknowledgments

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

References

[1] L.M. Larsen, A. Checko, H.L. Christiansen, A survey of the functional splits proposed for 5G mobile crosshaul networks, IEEE Commun. Surv. Tutor. 21 (1) (2018) 146–172.
[2] 3GPP, Radio Access Architecture and Interfaces (Release 14), Technical Report 38.801, (38.801) 2017.
[3] A.N. Al-Quzweeni, A.Q. Lawey, T.E.H. Elgorashi, J.M.H. Elmirghani, Optimized energy aware 5G network function virtualization, IEEE Access 7 (2019) 44939–44958.
[4] Y. Xiao, J. Zhang, Y. Ji, Energy-efficient DU-CU deployment and lightpath provisioning for service-oriented 5G metro access/aggregation networks, J. Lightwave Technol. 39 (17) (2021) 5347–5361.
[5] L.M.M. Zorello, M. Sodano, S. Troia, G. Maier, Latency-aware baseband-function placement in 5G metro-access networks, IEEE Trans. Green Commun. Netw. (2022).
[6] R. Alvizu, S. Troia, G. Maier, A. Pattavina, Matheuristic with machine-learning-based prediction for software-defined mobile metro-core networks, J. Opt. Commun. Netw. 9 (9) (2017) D19–D30.
[7] H. Zhang, Y. Hua, C. Wang, R. Li, Z. Zhao, Deep learning based traffic and mobility prediction, in: Machine Learning for Future Wireless Communications, John Wiley and Sons, 2020, pp. 119–136.
[8] L.M.M. Zorello, L. Bliek, S. Troia, T. Guns, S. Verwer, G. Maier, Baseband-function placement with multi-task traffic prediction for 5G radio access networks, IEEE Trans. Netw. Serv. Manag. (2022).
[9] L.M.M. Zorello, S. Troia, M. Quagliotti, G. Maier, Power-aware optimization of baseband-function placement in cloud radio access networks, in: IFIP/IEEE International Conference on Optical Network Design and Modelling, 2020.
[10] A. Rago, G. Piro, G. Boggia, P. Dini, Anticipatory allocation of communication and compuational resources at the edge using spatio-temporal dynamics of mobile users, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4548–4562.
[11] A. Pelekanou, M. Anastasopoulos, A. Tzanakaki, D. Simeonidou, Provisioning of 5G services employing machine learning techniques, in: IEEE/IFIP International Conference on Optical Network Design and Modelling, 2018.
[12] H. Yu, F. Musumeci, J. Zhang, M. Tornatore, L. Bai, Y. Ji, Dynamic 5G RAN slice adjustment and migration based on traffic prediction in WDM metro-aggregation networks, J. Opt. Commun. Netw. 12 (12) (2020) 403–413.
[13] A.N. Elmachtoub, P. Grigas, Smart "predict, then optimize", Manage. Sci. 68 (2022) 9–26.
[14] R.I. Tinini, D.M. Batista, G.B. Figueiredo, M. Tornatore, B. Mukherjee, Energy-efficient baseband processing via vBBU migration in virtualized cloud-fog RAN, in: IEEE Global Communications Conference, 2020.
[15] R. Singh, C. Hasan, X. Foukas, M. Fiore, M. Marina, Y. Wang, Energy-efficient orchestration of metro-scale 5G radio access networks, in: IEEE International Conference on Computer Communications, 2021.
[16] M. Klinkowski, Optimization of latency-aware flow allocation in NGFI networks, Comput. Commun. 161 (2020) 344–359.
[17] H. Gupta, M. Sharma, A. Franklin, B.R. Tamma, Apt-RAN: A flexible split-based 5G RAN to minimize energy consumption and handovers, IEEE Trans. Netw. Serv. Manag. 17 (1) (2020) 473–487.
[18] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, X. Costa-Perez, DeepCog: optimizing resource provisioning in network slicing with AI-based capacity forecasting, IEEE J. Sel. Areas Commun. 38 (2) (2020) 361–376.
[19] L. Chen, T.M.T. Nguyen, D. Yang, M. Nogueira, C. Wang, D. Zhang, Data-driven C-RAN optimization exploiting traffic and mobility dynamics of mobile users, IEEE Trans. Mob. Comput. 20 (5) (2021) 1773–1788.
[20] L.M.M. Zorello, S. Troia, G. Maier, Machine-learning-aided resource allocation in 5G metro networks, in: CNIT Technical Report 07-Machine Learning and 5G/6G Networks: Interplay and Synergies, CNIT, 2021, pp. 113–125.
[21] S. Verwer, Y. Zhang, Q.C. Ye, Auction optimization using regression trees and linear models as integer programs, Artificial Intelligence 244 (2017) 368–395.
[22] B. Wilder, B. Dilkina, M. Tambe, Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization, in: AAAI Conference on Artificial Intelligence, 2019.
[23] T. Konishi, T. Fukunaga, End-to-end learning for prediction and optimization with gradient boosting, in: Machine Learning and Knowledge Discovery in Databases, 2021.
[24] J. Mandi, E. Demirovic, P.J. Stuckey, T. Guns, Smart predict-and-optimize for hard combinatorial optimization problems, in: AAAI Conference on Artificial Intelligence, 2020.
[25] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, N. de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE 104 (1) (2016) 148–175.
[26] K. Yan, J. Yan, C. Luo, L. Chen, Q. Lin, A surrogate objective framework for prediction+optimization with soft constraints, in: Conference on Neural Information Processing Systems, 2021.
[27] A. Boukerche, J. Wang, Machine learning-based traffic prediction models for intelligent transportation systems, Comput. Netw. 181 (2020).
[28] J. Bergstra, D. Yamins, D.D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: JMLR International Conference on Machine Learning, I-115–I-23, 2013.
[29] M. Shehata, A. Elbanna, F. Musumeci, M. Tornatore, Multiplexing gain and processing savings of 5G radio-access-network functional splits, IEEE Trans. Green Commun. Netw. 2 (4) (2018) 982–991.
[30] B. Debaillie, C. Desset, F. Louagie, A flexible and future-proof power model for cellular base stations, in: IEEE Vehicular Technology Conference, 2015.
[31] A.D. Domenico, Y. Liu, W. Yu, Optimal computational resource allocation and network slicing deployment in 5G hybrid C-RAN, in: IEEE International Conference on Communications, 2019.
[32] L. Bliek, S. Verwer, M. de Weerdt, Black-box combinatorial optimization using models with integer-valued minima, Ann. Math. Artif. Intell. 89 (2021) 639–653.
[33] L. Bliek, A. Guijt, S. Verwer, M. de Weerdt, Black-box mixed-variable optimisation using a surrogate model that satisfies integer constraints, in: Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, 2021, pp. 1851–1859.
[34] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems, vol. 24, 2011.
[35] P.I. Frazier, A tutorial on Bayesian optimization, 2018, arXiv preprint arXiv:1807.02811.
[36] L. Bliek, A. Guijt, R. Karlsson, S. Verwer, M. de Weerdt, Benchmarking surrogate-based optimisation algorithms on expensive black-box functions, Appl. Soft Comput. 147 (2023) 110744, http://dx.doi.org/10.1016/j.asoc.2023.110744, URL https://www.sciencedirect.com/science/article/pii/S1568494623007627.
[37] Metro-Haul, https://metro-haul.eu/.
[38] Intel, CTP metrics for Intel mocroprocessors - Intel Xeon scalable processors, 2019.
[39] L. Askari, F. Musumeci, L. Salerno, O. Ayoub, M. Tornatore, Dynamic DU/CU placement for 3-layer C-RANs in optical metro-access networks, in: International Conference on Transparent Optical Networks, 2020.
[40] Small Cell Forum, Small cell virtualization functional splits and use cases, 2016, 159.07.02.
[41] J.M.H. Elmirghani, T. Klein, K. Hinton, L. Nonde, A.Q. Lawey, T.E.H. El-Gorashi, M.O.I. Musa, X. Dong, GreenTouch GreenMeter core network energy-efficiency improvement measures and optimization, J. Opt. Commun. Netw. 10 (2) (2018) A250–A269.
[42] Telecom Italia, Big data challenge, 2019, https://dandelion.eu/datamine/open-big-data/.
[43] Unwire Labs, OpenCellid, http://opencellid.org/.



Ligia Maria Moreira Zorello received a M.Sc. in telecommunications engineering from Télécom Paris, France, in 2016, and a M.Sc. in computer engineering from Escola Politecnica da Universidade de São Paulo, Brasil, in 2018. She obtained her Ph.D. in information technology at Politecnico di Milano, Italy in 2022. Her Ph.D. research was on developing algorithms for 5G radio access networks baseband function placement. Her research interests are related to optimization in 5G radio access networks, network function virtualization, network slicing, and machine learning algorithms for communications.

Laurens Bliek obtained his M.Sc. in Applied Mathematics in 2014, and his Ph.D. in Systems and Control in 2019, both from Delft University of Technology The Netherlands His Ph.D. topic was on developing a learning-based automatic tuning algorithm for antenna control in microwave photonics. He worked as a postdoctoral researcher at the Algorithmics group of Delft University of Technology and is currently an Assistant Professor in the Information Systems group at the Department of Industrial Engineering and Innovation Sciences at Eindhoven University of Technology, the Netherlands. His main research interest is the combination of machine learning and optimization algorithms, especially in optimizing expensive cost functions.

Sebastian Troia is an Assistant Professor at the Department of Electronics, Information and Bioengineering at Politecnico di Milano (Italy) and a Fulbright Fellow at the Erik Jonsson School of Engineering and Computer Science at the University of Texas at Dallas (USA). In 2020, he earned his Ph.D. degree in Information Technology (Telecommunication area) cum laude from Politecnico di Milano. His current research interests are in the field of edge network softwarization and Machine Learning for communication networks. His work encompasses the development of intelligent control and orchestration plane architectures for SDN and SD-WAN in multi-layer (optical and IP) network scenarios. He has participated in various European Projects such as H2020 Metro-Haul, NGI Atlantic, and FP7 Marie Curie MobileCloud. Additionally, he served as an editor for the ITU Focus Group on Machine Learning for Future Networks including 5G (FG-ML5G). He has co-authored more than 50 publications in international journals, conferences and book chapters, with a focus on Machine-Learning for SDN, SD-WAN and NFV. He is a reviewer for several international journals, and a member of the Technical Program Committee for international conferences and workshops including IEEE ICC and IEEE GLOBECOM. He has been also a member of the Organizing Committee for IEEE HPSR 2019, DRCN 2020 and 2021, IEEE NetSoft 2022 and NoF 2024. He is the co-organizer the Edge Network Softwarization (ENS) workshop, now in its third edition in conjunction with IEEE NetSoft 2024 in St. Louis, USA.

Guido Maier received his Laurea degree in Electronic Engineering at Politecnico di Milano (Italy) in 1995 and his Ph.D. degree in Telecommunication Engineering at the same university in 2000. Until February 2006 he has been researcher at CoreCom (research consortium supported by Pirelli in Milan, Italy), where he achieved the position of Head of the Optical Networking Laboratory. On March 2006 he joined the Politecnico di Milano as Assistant Professor. In 2015 he became Associate Professor. His main areas of interest are: optical network modeling, design and optimization; SDN orchestration and control-plane architectures; SD-WAN and NFV. He is author of more than 150 papers in the area of Networking published in international journals and conference proceedings (hindex 25) and 6 patents. He is currently involved in industrial and European research projects. In 2016 he co-founded the start-up SWAN networks, spin-off of Politecnico di Milano. He is editor of the journal Optical Switching and Routing, General Chair of DRCN 2020, DRCN 2021 and NetSoft 2022, guest editor of a special issue of the IEEE Open Journal of the Communications Society and TPC member in many international conferences. He is a Senior Member of the IEEE Communications Society.

Sicco Verwer is currently an Associate professor at Delft University of Technology in machine learning for cybersecurity. He is the head of the TU Delft Cyber Analytics Lab where he works on understandable AI for intrusion detection and software understanding. His team won several AI challenges including ones on learning software models, automated reverse engineering, and adversarial machine learning. He received many grants and awards for his research including prestigious VENI and VIDI grants from NWO, and a test-of-time award from ECMLPKDD for his pioneering work on discrimination-free classification.