Jeong, Jaehee; Premsankar, Gopika; Ghaddar, Bissan; Tarkoma, Sasu

A robust optimization approach for placement of applications in edge computing considering latency uncertainty

Published in:
Omega (United Kingdom)

DOI:
10.1016/j.omega.2024.103064

Published: 01/07/2024

Document Version
Publisher's PDF, also known as Version of record

Omega 126 (2024) 103064

Contents lists available at ScienceDirect

Omega

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

A robust optimization approach for placement of applications in edge computing considering latency uncertainty☆

Jaehee Jeong [a,c], Gopika Premsankar [b], Bissan Ghaddar [c,d,*], Sasu Tarkoma [e]

[a] Department of Management Sciences, University of Waterloo, Canada
[b] Department of Information and Communications Engineering, Aalto University, Finland
[c] Ivey Business School of London, Canada
[d] Department of Technology, Management and Economics, Technical University of Denmark, Denmark
[e] Department of Computer Science, University of Helsinki, Finland

ARTICLE INFO

Keywords:
Telecommunication networks
Uncertainty in network latency
Robust optimization
Edge computing
Application placement

ABSTRACT

Edge computing brings computing and storage resources close to end-users to support new applications and services that require low network latency. It is currently used in a wide range of industries, from industrial automation and augmented reality, to smart cities and connected vehicles, where low latency, data privacy, and real-time processing are critical requirements. The latency of accessing applications in edge computing must be consistently below a threshold of a few tens of milliseconds to maintain an acceptable experience for end-users. However, the latency between users and applications can vary considerably depending on the network load and mode of wireless access. An application provider must be able to guarantee that requests are served in a timely manner by their application instances hosted in the edge despite such latency variations. This article focuses on the placement and traffic allocation problem faced by application providers in determining where to place application instances on edge nodes such that requests are served within a certain deadline. It proposes novel formulations based on robust optimization to provide optimal plans that protect against latency variations in a configurable number of network links. The robust formulations are based on two different types of polyhedral uncertainty sets that offer different levels of protection against variations in latency. Extensive simulations show that our robust models are able to keep the number of chosen edge nodes low while reducing the number of latency violations as compared to a deterministic optimization model that only considers the average latency of network links.

1. Introduction

Cloud computing has revolutionized IT infrastructure and service delivery, wherein cloud providers provide access to a shared pool of computing, analytics, storage and networking resources to other companies [1]. These shared resources are housed in large data centers typically located far away from users [2]. Application providers – such as Netflix, Pinterest, Twitter, etc. – rent computing and storage resources from the cloud provider[1] to host their applications [1,3]. Applications hosted in the cloud enjoy many benefits such as high reliability and rapid provisioning of resources with minimal effort [1].

A cloud-based application can be made accessible to end-users over the Internet. For example, let us consider a simple video analytics application hosted in the cloud – requests (e.g., to detect an object in a video stream) are sent to the application instance, where the requests are served (e.g., video streams are processed using computer vision algorithms) and responses are sent back to the users (e.g., a bounding box is added around the identified object). A significant drawback of having cloud data centers located only in a few geographic regions is that the latency of accessing their resources is high when end-users are far away from the data centers [2].

Accordingly, a new transformative paradigm, *edge computing* [2,4–7], that complements traditional cloud computing has emerged. In edge computing, computing and storage resources are brought close to the end-users in the form of small data centers, called *edge nodes*, which

☆ Area: Production Management, Scheduling and Logistics. This paper was processed by Associate Editor Furini.

* Corresponding author.
E-mail addresses: jh5290@kaist.ac.kr (J. Jeong), gopika.premsankar@aalto.fi (G. Premsankar), bghaddar@ivey.ca, ghaddar@dtu.dk (B. Ghaddar), sasu.tarkoma@helsinki.fi (S. Tarkoma).

[1] Applications that rely on large cloud providers are listed on each provider's page: Amazon Web Services (https://aws.amazon.com/ec2/customers/), Google Cloud (https://cloud.google.com/customers), and Microsoft Azure (https://azure.microsoft.com/en-us/resources/customer-stories/).

https://doi.org/10.1016/j.omega.2024.103064
Received 26 May 2023; Accepted 19 February 2024
Available online 21 February 2024

are widely distributed within populated areas. It is similar to cloud computing in that application providers rent resources from a shared pool of resources. Although edge nodes (ENs) typically have a smaller capacity than data centers for cloud computing, ENs have sufficient capacity to host and run multiple applications. By moving computing resources closer to the user, a higher communication latency to distant cloud data centers is avoided. An edge computing platform [5] provides a software environment that enables application providers to run their applications on ENs closer to users. Examples of applications that benefit most from the lower latency are video analytics, augmented reality, gaming, autonomous drones, robotics, and connected cars [8]. In such applications, the latency between sending a request and receiving a response must be within a few tens of milliseconds [2,4] in order to maintain a good user experience [9]. Additionally, processing data at the edge reduces the amount of data sent to the cloud, which can lead to cost savings, especially in situations with costly data transmission.

Similar to cloud computing, application providers rent resources from the shared pool of resources to host their application. Thus, application providers must make decisions on which edge nodes to *place* (i.e., run) their application. Once the application is placed, user requests must be sent to the appropriate edge node, resulting in a *traffic allocation* decision of where user requests are served. However, this decision-making problem is more challenging in edge computing than in the cloud. A recent empirical study of a live edge computing platform shows that placement and traffic allocation decisions made by application providers are not very efficient, resulting in low utilization of computing resources on certain edge nodes [5]. This indicates that the problem of placing edge-based applications and allocating requests is difficult and requires new policies to be designed as compared to the cloud [5]. This is mainly due to the strict latency deadlines for requests served by edge-based applications. The latency between users and applications can vary considerably depending on network conditions (e.g., instantaneous load), mode of wireless access (e.g., WiFi, LTE or 5G), processing load and variations in the network paths [5,10–13]. An application provider must be able to guarantee that requests are served in a timely manner by their application instances hosted in the edge despite such latency variations. This is especially required for interactive applications such as gaming (augmented reality and virtual reality) wherein the user experience is heavily dependent on the responsiveness of the application [9].

This article focuses on the application placement and traffic allocation problem faced by application providers in determining where to place application instances on edge nodes such that requests are served within a certain deadline. This is a fundamental problem that arises when application providers leverage edge computing platforms to host their applications [5]. Formulating this decision-making problem from the perspective of an application provider is an under-studied problem in the literature. One relevant paper is by Bülbül et al. [1] that addresses the placement of application instances for cloud computing taking into account the uncertainties in demand and price of the rented resources. There are several challenges in choosing appropriate edge nodes. First, the choice of edge nodes depends on the demand (i.e., number of incoming requests) expected to be served by the application instances as well as the latency between the edge nodes and the source of requests. Deploying an application on multiple edge nodes can help ensure that latency deadlines are met as requests can always be served from the edge node nearest to the originating request. However, this low latency comes at a high cost to the application provider. Keeping multiple instances of the application running at all times is costly and inefficient, as the computing resources are reserved even when there are not many incoming requests. Second, it is challenging to consistently serve requests with low latency when the latency between users and edge nodes can vary considerably. A mechanism that only optimizes the placement and traffic allocation based on the average or median latency of network links may result in many missed deadlines when the network latency increases. A simple and conservative approach would be to optimize the placement for the worst (highest) expected latency between users and edge nodes. However, this would result in a high cost as many edge nodes are chosen to guarantee deadlines are met, also resulting in potentially underutilized resources. Thus, an efficient placement and allocation mechanism must take the uncertainty in network latency into account when placing application instances on the edge nodes. Optimal placement and demand allocation for applications in edge computing have been studied [14,15]; however, only a few consider solutions that account for uncertainties in making placement decisions. Among those that do, the focus is on uncertainty in traffic or user demand [16–18], the number of edge node (infrastructure) failures [17,19], or the number of malicious flows on network links [20].

The main contribution of this paper is twofold: to the best of our knowledge, this work is the first to consider the uncertainty in network latency when making placement and traffic allocation decisions in edge computing. We formulate a mathematical optimization model from the perspective of an edge application provider that reduces its cost while still meeting strict latency thresholds despite variations in the network latency. To deal with the latency uncertainty, we propose a novel solution based on robust optimization to devise placement and allocation plans for application instances on edge nodes that account for the variation in the network latencies between users and edge nodes. Robust optimization is well-suited for including uncertainties in optimization models [21,22], as it does not require a probability distribution for the uncertain parameter (latency in our case). Instead, it only needs an uncertainty set which is the set of possible realizations of uncertain parameters. Robust optimization takes advantage of the ease of modeling uncertainties compared to other approaches, such as stochastic optimization, so it has been studied in many fields for the past two decades [23–27]. Based on robust optimization, our solution protects against latency variations in a configurable number of network links. We present robust formulations based on two different types of polyhedral uncertainty sets that offer different levels of protection against variations in latency. Additionally, we propose efficient methods to solve the robust optimization models based on mixed integer linear programming. Our solutions are designed such that the placement and traffic allocation decisions are done for the next time slot, where each time slot is 10 min. We evaluate the obtained solutions through extensive Monte Carlo simulations. The results show that our robust models are able to keep the number of chosen edge nodes low while reducing the number of latency violations (by up to 99%) as compared to an optimization model that only considers the average latency of network links. The robust models are able to keep the latency of requests consistently low, with a low value for the maximum duration by which the latency of any single request exceeds the deadline.

The rest of the article is organized as follows. Section 2 describes the literature on optimal placement of applications in edge computing. Section 3 describes the system model, introduces the problem of placing applications and allocating requests in edge computing, and describes the robust formulation considering uncertainties in the network latency. Section 4 describes our method for solving the robust optimization model. Section 5 details the computational results obtained when evaluating the robust models, and Section 6 provides concluding remarks.

2. Literature review

The optimal placement of applications and allocation of requests for applications in edge computing has been studied extensively, with different objectives of minimizing latency [30,31], reducing energy [30,32,33], or increasing user throughput [34]. However, the proposed solutions do not consider uncertainties in any parameters when choosing the locations for hosting edge-based applications. Only a few articles

Table 1
Existing literature on edge computing that include uncertainty.

Work	Application	Source of uncertainty	Structure of uncertainty	Methodology
Yu et al. [19]	Placing delay-sensitive applications in edge computing	Failure of the network links	A reliability parameter between 0 and 1	Fully polynomial-time approximation schemes, randomized algorithm
Badri et al. [16]	Placing applications in edge computing	Mobility of end-users	Proposed scenario generation model	Multi-stage stochastic programming approach
Nguyen et al. [18]	Placing applications in edge computing	Application demand	Budgeted uncertainty set	Two-stage robust optimization approach
Bülbül et al. [1]	Placing applications in cloud computing	Demand & price of the rented resources	Proposed scenario tree	Multi-stage stochastic programming approach
Qu et al. [28]	Traffic allocation and placement in edge computing	Failure of the network links or edge nodes	Cardinality constraint	Set function optimization approach
Li et al. [29]	Application placement and request routing	Demand & rate of incoming requests	Zipf distributions	Approximated dynamic optimization approach
Cheng et al. [17]	Placing applications in edge computing	Resource demand & edge node failures	Budgeted uncertainty set	Two-stage robust optimization approach
This work	Placing applications in edge computing	Network latency	Budgeted/variable-budgeted uncertainty set	Robust MILP

have considered uncertainties when optimizing the placement of applications and allocation of requests, which we review next. Nguyen et al. [18] consider the problem of placing applications in edge computing under uncertainties in the demand for the application. They propose a two-stage robust optimization framework to choose the edge nodes for hosting application instances and to decide the amount of computing resources required at each node. The framework proposed by Nguyen et al. [18] is extended by Cheng et al. [17] by including uncertainties in the resilience of edge nodes, i.e., edge nodes may experience failures during which they cannot serve any requests. The placement decisions are made taking into account uncertainties in both the incoming demand and resilience of the edge nodes [17]. A two-stage robust optimization framework is proposed to solve this problem in an iterative manner. Bülbül et al. [1] address the placement of application instances for cloud computing taking into account the uncertainties in demand and price of the rented resources. The authors propose various multi-stage stochastic programming formulations, such as scenario-based or chance-constrained models, and compare them. Yu et al. [19] consider the placement of applications for delay-sensitive applications in edge computing under uncertainties in the failure of the network links. In particular, the data lost on any network link due to failures must be limited to a certain ratio. The authors propose heuristics based on approximation and randomization schemes to solve this problem efficiently for different scenarios. Qu et al. [28] also consider the failure of network links or edge nodes in devising a traffic allocation and placement algorithm for edge computing. The goal is to minimize the latency of serving requests while being robust to a certain number of failures. The problem is solved through linear approximation and submodular approximation approaches, that are shown to obtain solutions that are close to the optimal in small network instances. Badri et al. [16] consider the placement of applications in edge computing under uncertainties in the mobility of end-users. The goal is to devise a placement scheme that reduces the energy consumption of edge nodes while still keeping the latency experienced by users low even if they move to a different location. The authors propose a solution based on multi-stage stochastic optimization, which requires the generation of a large number of scenarios to obtain a good solution. To solve this problem, the authors devise a greedy algorithm based on the sample average approximation method to obtain solutions. Li et al. [29] consider the placement of applications under uncertainties in the demand (or rate of incoming requests). The goal is to minimize the latency of serving requests while satisfying long-term budget constraints for placing applications across different time slots. The problem is solved through an approximated dynamic optimization approach that solves the problem for individual time slots which are solved efficiently through a rounding-based approach. None of the articles described above consider the optimal placement of applications under uncertainties in network latency. However, this is a fundamental problem that arises when application providers leverage public edge computing platforms to host applications that require both low latency and high utilization of rented resources [5]. The end-to-end latency in accessing services in the edge or cloud varies due to several reasons, including, variability in the time taken to transfer data both towards the server and to receive a response (for instance, the wireless network used to access the network may be congested), queuing delays at different network hops, and energy management policies of the networking equipment (for example, certain devices may need to be moved from an inactive to active state) [9]. While the variability in the demand for services can be predicted with reasonable accuracy [5], predicting variations in the latency is more challenging, and thus, it is important to devise optimization models that account for variations in network latency between users and edge nodes.

Table 1 summarizes the previous literature considering uncertainties for the placement of applications and allocation of requests. It compares our paper to the different papers in regard to the source of uncertainty and the proposed solution approaches. Our paper proposes a robust optimization model to place applications in edge computing such that the latency of serving requests is low even when the network latency varies.

3. Problem formulation

3.1. Problem description

We consider the problem where an application provider must decide where to place their application instances[2] in an edge computing platform. The edge computing platform provides multiple edge nodes ($\mathcal{J} = \{1, 2, \ldots |\mathcal{J}|\}$) that can be used to host application instances. The demand for applications comprises user requests that must be processed by the application instance(s). The application provider rents C_j computing resources from edge nodes to host their application instances. The goal of the application provider is to choose the minimum number of edge nodes such that user requests are satisfied with

[2] The application instance could be virtual machines, containers, or functions, depending on the particular edge computing platform. Our model is agnostic to the specific type of virtualization used.

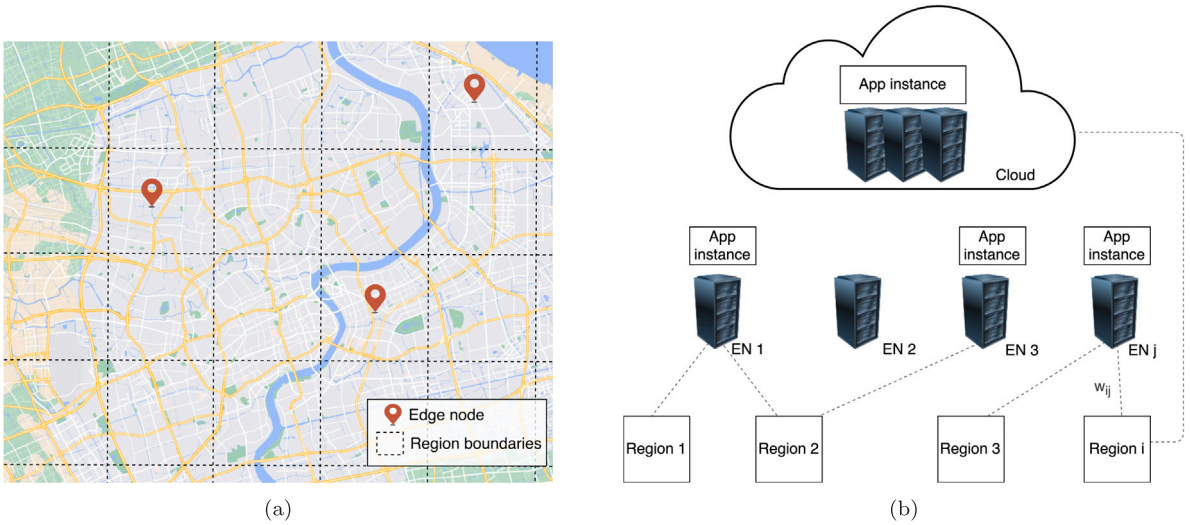

Fig. 1. (a) Example scenario showing region boundaries and locations of edge nodes. (b) Overview of system model indicating application instances, regions and assignment of demand from regions to edge nodes and the cloud.

Table 2
Summary of notation in the optimization problem.

Sym.	Description
$\mathcal{J}$	Set of edge nodes
$\mathcal{N}$	Set of edge nodes and cloud
$\mathcal{I}$	Set of regions from which demand originates
d_i	Demand in terms of number of requests for application from region i
c^{fixed}	Fixed cost to set up an application on EN j
c^{oper}	Operational cost (in terms of energy) to run tasks on an EN j
C_j	Capacity (in terms of CPU units) available for application instance on EN j
L	Threshold or deadline for average latency (in ms)
R	Minimum number of ENs on which application instance must be placed
υ	CPU units required to run application
κ	Threshold for maximum CPU utilization (in percent)
$\bar{t}_{i,j}$	Average latency between region i and EN j
x_j	Binary variable for whether application instance is loaded on EN j
$w_{i,j}$	Continuous variables indicating the # of requests from region i allocated to node j

a latency below a configured threshold. Users send requests to the application over wireless networks (typically, mobile or WiFi). The demand (user requests) for edge computing applications exhibits a strong geo-sensitive nature [5], and accordingly the demand is modeled as originating from different geographic regions, $\mathcal{I}$. The application provider does not know the precise locations of the edge nodes, but only has information about the average latency $t_{i,j}$ between the regions ($i \in \mathcal{I}$) to edge nodes ($j \in \mathcal{J}$). The application provider must decide which edge nodes are used to host their applications. The application is assumed to be always hosted in the cloud with sufficient computing resources to handle user requests. In addition to choosing the edge nodes, the allocation of traffic from each region to the appropriate edge node or cloud is also required. Accordingly, the set of compute nodes $\mathcal{N}$ includes all ENs ($j \in \mathcal{J}$) and the cloud (with index 0), i.e., $\mathcal{N} = \{0, 1, 2, \ldots, |\mathcal{J}|\}$. The number of requests assigned from each region to the edge node or cloud is indicated by $w_{i,j}$. Table 2 summarizes the notation used for this problem. Fig. 1(a) shows a sample scenario where the regions from which the demand originates are marked with dotted lines, and the edge nodes available for hosting application instances are indicated with location pins. Note that the figure shows regular grid-shaped regions for ease of representation but these regions may be irregularly shaped without impacting the proposed approach. Fig. 1(b) shows the relationship between regions, edge nodes and the cloud.

3.2. Deterministic optimization problem

The optimization problem aims to find an assignment of application instances to edge nodes (x_j) and an assignment of requests ($w_{i,j}$) from each region i to node j. The objective is to reduce the cost of running the application. Accordingly, there is a fixed cost c^{fixed} for running an application instance on an edge node, and operational costs (c^{oper}) depending on the utilization of the application instance ($\frac{\upsilon \cdot w_{i,j}}{C_j}$). Finally, the objective function includes a penalty (h) for requests that are sent to the cloud.

$$\min_{x,w} \quad \sum_{j \in \mathcal{J}} c^{fixed} x_j + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c^{oper} \cdot \frac{\upsilon \cdot w_{i,j}}{C_j} + \sum_{i \in \mathcal{I}} h \cdot w_{i,0} \tag{1}$$

The constraints are as follows.

1. *Traffic allocation constraints*. All the demand must be assigned to an EN or to the cloud.

$$\sum_{j \in \mathcal{N}} w_{i,j} = d_i, \quad \forall i \in \mathcal{I} \tag{2}$$

2. *Capacity constraint*. The application provider must specify the number of CPU units [35,36], C_j, that are reserved for running the application instance at each edge node. Accordingly, υ indicates the number of CPU units required to support a certain unit of demand (e.g. a target number of requests per second). This value can be obtained through benchmarking or profiling tools that evaluate the performance of application code under different workloads to obtain an estimate of CPU resources required to meet a target rate of incoming requests [37,38]. C_j denotes the number of CPU units rented in each edge node. The maximum utilization is restricted to a certain percent, κ, of

the total available CPU units to avoid any unpredictable impact on latency when the application instances are run at very high utilization [39]. Also, this constraint ensures that the demand for the application can be assigned to EN j only if the application is loaded.

$$v \sum_{i \in \mathcal{I}} w_{i,j} \leq \kappa C_j x_j, \quad \forall j \in \mathcal{J} \tag{3}$$

3. *Latency constraint*. Next, the average latency for served requests must be below a certain threshold, L. Accordingly, we consider the average latency, $\bar{t}_{i,j}$, between region i and compute node j in formulating the latency constraint. The average latency $\bar{t}_{i,j}$ is obtained from historical data. The requests assigned to each compute node ($w_{i,j}$) are multiplied by the corresponding average latency $\bar{t}_{i,j}$ and then divided by the total demand ($\sum_{i \in \mathcal{I}} d_i$) to obtain the average latency.

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{N}} \bar{t}_{i,j} w_{i,j} \leq \sum_{i \in \mathcal{I}} d_i \cdot L, \tag{4}$$

4. *Resilience constraint*. Each application instance must be hosted on at least R edge nodes for resilience. This is required so that the application provider can reliably serve the demand even if an edge node fails.

$$\sum_{j \in \mathcal{J}} x_j \geq R \tag{5}$$

5. *Constraints on variables*. Finally, the following constraints specify valid ranges for the variables.

$$w_{i,j} \geq 0, \quad \forall i \in \mathcal{I}, j \in \mathcal{N} \tag{6}$$

$$x_j \in \{0,1\}, \quad \forall j \in \mathcal{J} \tag{7}$$

The proposed optimization problem aims to minimize the cost of running the application while meeting the capacity constraint (Eq. (3)) and a latency threshold for user requests (Eq. (4)). The constraints depend on two input parameters — the expected demand (d_i) and the average latency between regions and edge nodes ($\bar{t}_{i,j}$). As these input parameters can vary at different times of the day, we anticipate that the application provider must solve the problem repeatedly for different time slots. The demand for edge-based applications follows end-users' daily activity patterns, which exhibit strong seasonality [5]. Accordingly, we consider that the demand can be accurately predicted ahead of time. Recent studies have shown that the workload (i.e., demand) characteristics remain stable for tens of minutes [40], and thus the optimization problem can be solved for time slots of that duration. The placement decisions from the obtained solution can be implemented near-instantaneously as the application resources are virtualized; the traffic allocation decisions can be implemented as routing table updates, which are also typically updated every few minutes (five minutes in [41]). In [1], the authors deal with demand uncertainty using multi-stage stochastic programming formulations. On the other hand, the other input parameter to the optimization problem, the latency of network links, varies often and with a significant magnitude [11,12,42]. Optimizing the placement and traffic allocation based on average latency alone may result in several missed deadlines for user requests due to variations in the latency. We focus on devising a placement and allocation plan for edge-based applications that incorporates the uncertainty in network latency, described next.

3.3. Robust optimization model

Dealing with uncertainty in optimization models has been widely studied in other research fields. The main approaches to incorporate uncertainty are stochastic optimization [43], chance-constrained optimization [44], and robust optimization [45]. Stochastic and chance-constrained optimization models assume that the probability distribution of uncertain parameters is known. A stochastic optimization model minimizes the expected objective value while satisfying the given constraints for individual scenarios that can be generated from the known probability distribution for the uncertain parameter. A chance-constrained optimization model enforces the constraint in a probabilistic manner. Each constraint is allowed to be violated with a predefined probability level. For both approaches, the probability distribution is critical to the quality of the solution. However, robust optimization handles the uncertainty using an uncertainty set. It assumes that the realization of uncertain parameters lies in a predefined set. In our considered problem, the latency is highly variable depending on the network load, network conditions and the mode of network access [5,10–12]; thus determining its probability distribution is difficult. Moreover, if the probability distribution is estimated incorrectly, a chance-constrained or stochastic optimization model may provide a sub-optimal solution. Thus, we choose robust optimization to incorporate uncertainties in network latency in our placement and allocation problem as it does not require a probability distribution for the network latency. Even if the latency is highly variable, its realization can be featured as a range. Thus, we here assume that latencies are realized in a predefined range. Robust optimization models typically consider two types of uncertainty sets — ellipsoidal uncertainty sets [22] and cardinality-constrained uncertainty sets [21]. Since we do not consider the correlation between latency realization, we adopt the cardinality-constrained uncertainty set [45]. A robust optimization model with this uncertainty set is usually tractable. This feature can be an advantage for the robust placement and allocation problem that has to be frequently solved at different times of the day.

With this assumption, we propose the robust formulation as follows. Since latency exists only in the latency constraint (4), we only need to consider robust constraints of the latency constraint. Let us assume that latency $t_{i,j}$ is uncertain, where the expected value is $\bar{t}_{i,j}$ and its realization lies within the interval $[\bar{t}_{i,j} - \hat{t}_{i,j}, \bar{t}_{i,j} + \hat{t}_{i,j}]$. Then, the realized latency value $t_{i,j}$ can be written as $t_{i,j} = \bar{t}_{i,j} + \hat{t}_{i,j}\zeta_{i,j}$, where $\zeta_{i,j}$ denotes the amount of latency variation from the mean value. We define $\boldsymbol{\zeta} := (\zeta_{i,j},\ i \in \mathcal{I}, j \in \mathcal{J})$ as the latency realization scenario for all links (i,j), and $\mathcal{U}$ to be the set of all possible latency realization scenarios. The robust constraint of (4) can be written as follows:

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{N}} \bar{t}_{i,j} w_{i,j} + \beta(\boldsymbol{w}) \leq \sum_{i \in \mathcal{I}} d_i L, \tag{8}$$

where the protection function $\beta(\boldsymbol{w})$ is defined as follows:

$$\beta(\boldsymbol{w}) = \max_{\zeta} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{N}} w_{i,j} \hat{t}_{i,j} \zeta_{i,j} \tag{9a}$$

$$\text{s.t.} \quad \boldsymbol{\zeta} \in \mathcal{U}. \tag{9b}$$

The protection function (9) represents the maximum latency sum which exceeds the expected latency for a given $\boldsymbol{w}$. Thus, constraint (8) can be immunized from all possible latency realizations within a given uncertainty set $\mathcal{U}$.

We note that there could be some trade-off between number of edge nodes and the risk of exceeding the latency deadline. This trade-off depends on how the uncertainty set $\mathcal{U}$ is constructed. In our work, we present two uncertainty sets to provide insight to the decision-maker (i.e., the application provider) on how they define the uncertainty set for their specific scenario.

First, the most famous polyhedral uncertainty set is the cardinality-constrained uncertainty set [21]. This uncertainty set considers all scenarios allowing up to Δ variated uncertain parameters. Here, Δ is called the *budget of uncertainty* for the set $\mathcal{U}$. In our model, the robust latency constraint is protected against all scenarios wherein up to Δ links have a higher realized value than the expected value (at most $\bar{t}_{i,j} + \hat{t}_{i,j}$). The cardinality-constrained uncertainty set is written as follows:

$$\mathcal{U}^{\Delta} := \left\{ \boldsymbol{\zeta} \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{N}|} \mid -1 \leq \zeta_{i,j} \leq 1 \qquad \forall i \in \mathcal{I}, j \in \mathcal{N}, \right.$$

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}|\zeta_{i,j}| \le \Delta\Big\}. \tag{10}$$

The parameter Δ controls the robustness of the solution. With a higher value of Δ, the corresponding robust optimal solution will be feasible against more scenarios. But, it can be over-conservative. Therefore, the decision-maker must decide the appropriate value of the Δ that balances cost and robustness. In [21], the authors introduce a probability bound for the violation of a robust constraint, which is represented as a function of the uncertainty budget and the number of uncertain parameters. Therefore, we can use it as a guideline for choosing the value of the uncertainty budget. Following Bertsimas and Sim [21], we can write the probability that the robust latency constraint is violated as follows:

$$P\left(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} t_{i,j}w_{i,j} > \sum_{i\in\mathcal{I}} d_i L\right) \le 1-\Phi\left(\frac{\Delta-1}{\sqrt{N^*}}\right), \tag{11}$$

where $t_{i,j} = \bar{t}_{i,j} + \hat{t}_{i,j}\zeta_{i,j}$ is the realized latency value of link (i,j), N^* is the total number of communication arcs between regions and compute entities ($N^* = |\mathcal{I}| \times |\mathcal{N}|$), and

$$\Phi(\theta) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\theta}\exp\left(-\frac{y^2}{2}\right)dy$$

is the cumulative distribution function of a standard normal distribution. The following uncertainty budget guarantees a 95% probability that the robust latency constraint is satisfied against all scenarios in the uncertainty set:

$$\Delta = 1 + 1.645\times\sqrt{N^*}, \tag{12}$$

where $\Phi(1.645) \approx 0.95$.

We note that the value of Δ in (12) only depends on the probability bound we want to guarantee, and N^* is always the same in the set $\mathcal{U}^{\Delta}$. However, the uncertain parameters in our model are the latency between regions and edge nodes chosen to host the application, so the status of *chosen* ENs can change the number of uncertain parameters. If the number of chosen ENs is small, the uncertainty budget (12) with N^* will provide a higher probability bound guarantee than we want. In other words, the cardinality uncertainty set $\mathcal{U}^{\Delta}$ can sometimes be over-conservative. Thus, we consider an additional uncertainty set, called *variable budgeted uncertainty set* [46], which is still a polyhedral uncertainty set. The variable budgeted uncertainty set defines the value of the budget of uncertainty as a function of decision variables and allows the size of the uncertainty set to vary depending on the decision variables. Thus, it can provide a less-conservative robust optimal solution in some cases. The variable budgeted uncertainty set is represented as follows:

$$\mathcal{U}(\boldsymbol{w}) := \Big\{\ \boldsymbol{\zeta}\in\mathbb{R}^{|\mathcal{I}|\times|\mathcal{N}|}\ \Big|\ -1\le\zeta_{i,j}\le 1 \qquad \forall i\in\mathcal{I}, j\in\mathcal{N},$$
$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}|\zeta_{i,j}| \le \gamma^0 + \gamma\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} y_{i,j}\ \Big\}, \tag{13}$$

where $y_{i,j}$ is a binary variable which is 1 if $w_{i,j}$ is greater than 0, and 0 otherwise. We note that the value of the uncertainty budget is changed as the value of $\boldsymbol{w}$ is changed. Depending on the value of $\boldsymbol{w}$, the size of the set $\mathcal{U}(\boldsymbol{w})$ can be smaller than the set $\mathcal{U}^{\Delta}$ while guaranteeing the same probability bound. The variable budgeted uncertainty set can guarantee the probability bound when γ_0 and γ are well defined [46]. Let $\Delta^p(\tilde{N})$ denote the value of the uncertainty budget, which guarantees a p probability at $\tilde{N}$. Poss [46] showed that if an affine function $\gamma_0+\gamma\tilde{N}$ is always larger than $\Delta^p(\tilde{N})$ for all $\tilde{N}\in\mathbb{R}$, the variable budgeted uncertainty set guarantees the probability bound of p. For instance, let assume that $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} y_{i,k} = \tilde{N}$. For a 95% probability that the robust latency constraint is satisfied, a tangent line to $1+1.645\times\sqrt{N}$ at $\tilde{N}$ can be a candidate of an affine function $\gamma^0+\gamma\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} y_{i,j}$. For more details, we refer the reader to Poss [46].

4. Solving the robust optimization model

The robust latency constraint (8) can be rewritten as follows.

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\bar{t}_{i,j}w_{i,j} + \max_{\zeta\in\mathcal{U}}\left\{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\hat{t}_{i,j}w_{i,j}\zeta_{i,j}\right\} \le \sum_{i\in\mathcal{I}} d_i\cdot L. \tag{14}$$

The constraint (14) is nonlinear, and we cannot handle this nonlinear constraint directly. However, it can be reformulated as a linear constraint by taking a dual of it. For both uncertainty sets (10) and (13), we present the reformulation of the robust latency constraint (14).

First, we note that the worst-case realization of latency always occurs when the realized latency has a higher value than the nominal value. Thus, we can re-write the bounds of uncertain parameter $\zeta_{i,j}$ as $0\le\zeta_{i,j}\le 1,\ \forall i\in\mathcal{I}, j\in\mathcal{N}$ for both uncertainty sets (10) and (13). With the cardinality-constrained uncertainty set (10), the protection function $\beta(\boldsymbol{w})$ is equivalent to as follows:

$$\beta(\boldsymbol{w}) = \max \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} w_{i,j}\hat{t}_{i,j}\zeta_{i,j} \tag{15a}$$
$$\text{s.t.} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\zeta_{i,j} \le \Delta \tag{15b}$$
$$0\le\zeta_{i,j}\le 1, \quad \forall i\in\mathcal{I}, j\in\mathcal{N}. \tag{15c}$$

The dual of the problem (15) is written as follows:

$$\min\ \Delta\eta^C + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\pi^C_{i,j} \tag{16a}$$
$$\text{s.t.}\ \ \eta^C_s + \pi^C_{i,j} \ge \hat{t}_{i,j}w_{i,j}, \quad \forall i,j\in\mathcal{N} \tag{16b}$$
$$\eta^C \ge 0 \tag{16c}$$
$$\pi^C_{i,j}\ge 0, \quad \forall i,j\in\mathcal{N}, \tag{16d}$$

where η^C and $\pi^C_{i,j}$ are dual variables associated to constraints (15b) and (15c), respectively. Therefore, the robust latency constraint with the set (10) can be reformulated to the following constraints:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\left(\bar{t}_{i,j}w_{i,j}+\pi^C_{i,j}\right) + \Delta\eta^C \le \sum_{i\in\mathcal{I}} d_i\cdot L \tag{17a}$$
$$\eta^C + \pi^C_{i,j} \ge \hat{t}_{i,j}w_{i,j}, \quad \forall i,j\in\mathcal{N} \tag{17b}$$
$$\eta^C \ge 0 \tag{17c}$$
$$\pi^C_{i,j}\ge 0, \quad \forall i,j\in\mathcal{N}, \tag{17d}$$

Similarly, the robust latency constraint with the set (13) can be reformulated as follows:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}}\left(\bar{t}_{i,j}w_{i,j}+\pi^V_{i,j}\right) + \left(\gamma_0+\gamma\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{N}} y_{i,j}\right)\eta^V \le \sum_{i\in\mathcal{I}} d_i\cdot L \tag{18a}$$
$$\eta^V + \pi^V_{i,j} \ge \hat{t}_{i,j}w_{i,j}, \quad \forall i,j\in\mathcal{N} \tag{18b}$$
$$w_{i,j} \le M_3 y_{i,j}, \quad \forall i,j\in\mathcal{N} \tag{18c}$$
$$\eta^V \ge 0 \tag{18d}$$
$$\pi^V_{i,j}\ge 0, \quad \forall i,j\in\mathcal{N} \tag{18e}$$
$$y_{i,j}\in\{0,1\}, \quad \forall i,j\in\mathcal{N} \tag{18f}$$

where η^V and $\pi^V_{i,j}$ are dual variables associated to constraints in the set (13). The constraint (18c) enforces that $y_{i,j}$ takes 1 if $w_{i,j}$ is bigger than 0. Unfortunately, the constraint (18a) is still nonlinear because there are bilinear terms $y_{i,j}\eta^V$. These bilinear terms can be linearized using the reformulation linearization technique (RLT) [47]. With the RLT, the bilinear term $y_{i,j}\eta^V$ is substituted for $\mu_{i,j}$, and the following constraints are added:

$$\mu_{i,j} - \eta^V \ge -M_4(1-y_{i,j}), \quad \forall i,j\in\mathcal{N} \tag{19a}$$
$$\mu_{i,j} \le M_4 y_{i,j}, \quad \forall i,j\in\mathcal{N} \tag{19b}$$
$$\mu_{i,j} \le \eta^V, \quad \forall i,j\in\mathcal{N} \tag{19c}$$

$$\mu_{i,j} \geq 0, \quad \forall i, j \in \mathcal{N} \tag{19d}$$

where M_4 is a big-M.

The dualization of the robust latency constraint introduces additional variables and constraints, increasing the size of the model. For the cardinality-constrained uncertainty set (10), the dualization requires $(|\mathcal{I}| \times |\mathcal{N}| + 1)$ additional variables and $|\mathcal{I}| \times |\mathcal{N}|$ additional constraints. For the variable budgeted uncertainty set (13), $|\mathcal{I}| \times |\mathcal{N}|$ additional binary variables are needed for $y_{i,j}$. Also, we need dualization and linearization to get a linear reformulation. The number of additional variables and constraints for the dualization is the same as the cardinality-constrained uncertainty set. But the linearization requires $|\mathcal{I}| \times |\mathcal{N}|$ additional variables and $3 \times |\mathcal{I}| \times |\mathcal{N}|$ additional constraints. We note that the robust model with the variable budgeted uncertainty set can easily face computational difficulties. We evaluate different sizes of network instances in the next section to demonstrate the scalability of the solutions.

5. Evaluation

This section presents the results from evaluating our robust models in different types of test instances. In edge computing, the demand for an application can be predicted with high accuracy due to the stronger seasonality as compared to cloud-based applications [5]. Thus, our test instances comprise scenarios where the application provider determines a placement strategy for the next 10 min based on the known (predicted) demand. Our focus is on the robustness of the optimization models to variations in latency, and thus, we do not evaluate forecasting methods for demand. The goal of the computational experiments is to evaluate whether our robust models are able to keep the number of chosen edge nodes low and average latency lower than a configured threshold, when there is variation in the network latency. Another important consideration is whether the robust solutions are obtained within a reasonable time for different types of test instances so that the placement decisions can be promptly implemented for the next 10-min time slot.

Generation of test instances. The network parameters are chosen to be representative of real-world networks, resulting in the following test instances: small networks (S1–S10) with 10 regions and 5 edge nodes, medium networks (M1–M10) with 18 regions and 10 edge nodes, and large networks (L1–L10) with 25 regions and 10 edge nodes. The number of edge nodes is restricted to 10, following an analysis of datasets from a real-world public edge computing platform that showed that the maximum number of edge nodes (or sites) in a city was 11 [5]. In each test instance, the demand from each region is drawn from a uniform distribution with minimum and maximum values as indicated in Table 3. This corresponds to the number of user requests in a time slot of 10 min. The computational capacity C_j rented from each edge node is chosen in steps of 1024 CPU units, as is custom in large cloud computing platforms [35]. In each test instance, the average latency between network elements is chosen such that a region is close to at least two ENs (with an average latency drawn from a uniform distribution of between 10 to 15 ms). The average latency between regions to remaining (more distant) edge nodes is drawn from a uniform distribution between 20 to 30 ms. The two closest edge nodes are randomly chosen with uniform probability from the complete list of edge nodes. The latency to the cloud is typically over 100 ms [8] and accordingly, the average latency is drawn from a uniform distribution between 120 to 150 ms. Note that the uniform distribution is used to draw the average latency values in the test instances, whereas later in the simulations, we use a different probability distribution to draw different realizations of the latency values (see *Simulation setup and performance metrics*). Finally, the application parameters are set as follows. The computing requirement, υ, is set to 1024 CPU units to support a demand of 100 requests per second, and the latency threshold L is set to 20 ms. To understand the impact of the resilience requirement R, its value is set to 1 for the first six test instances, and set to between 2 to 4 for the remaining test instances.

Table 3
Parameters used to generate the test instances.

Instance	Min demand	Max demand	Compute capacity (C_j)	Resilience (R)
S1	1 000	5 000	1024	1
S2	10 000	15 000	1024	1
S3	10 000	15 000	3072	1
S4	15 000	20 000	1024	1
S5	15 000	20 000	3072	1
S6	15 000	20 000	5120	1
S7	1 000	5 000	1024	2
S8	10 000	15 000	3072	2
S9	15 000	20 000	5120	2
S10	15 000	20 000	3072	3
M1	1 000	5 000	1024	1
M2	10 000	15 000	1024	1
M3	10 000	15 000	3072	1
M4	15 000	20 000	1024	1
M5	15 000	20 000	3072	1
M6	15 000	20 000	5120	1
M7	1 000	5 000	1024	3
M8	1 000	5 000	1024	4
M9	15 000	20 000	5120	3
M10	15 000	20 000	3072	4
L1	1 000	5 000	1024	1
L2	10 000	15 000	1024	1
L3	10 000	15 000	3072	1
L4	15 000	20 000	2048	1
L5	15 000	20 000	3072	1
L6	15 000	20 000	5120	1
L7	1 000	5 000	1024	3
L8	1 000	5 000	1024	4
L9	15 000	20 000	5120	3
L10	15 000	20 000	5120	4

Evaluated models. We evaluate the deterministic model (D), robust model with the cardinality-constrained uncertainty set (called robust fixed, RF), and robust model with the variable budgeted uncertainty set (called robust variable, RV) proposed in Sections 3.2 and 3.3. Additionally, we evaluate the deterministic model (D) with more conservative estimates of the latency $t_{i,j}$ in constraint (4). Specifically, each $t_{i,j}$ is set to $\bar{t}_{i,j} + \sqrt{0.5\bar{t}_{i,j}}$. The D model with constraint (4) updated with these values results in conservative solutions that are protected from variations in latency. This model is denoted as deterministic-conservative model (DC) and is similar to the robust formulation presented by Soyster [48].

All optimization models are evaluated with the costs in the objective function set as follows: c^{oper} as 350, c^{fixed} as 10 and h as 1. The latency threshold L is set to 20 ms, and κ is set to 70% [41]. For the value of the uncertainty budget for RF and RV, we consider a 95% probability bound for the robust latency constraint. In other words, the robust latency constraint should be satisfied at least 95% for all scenarios in the uncertainty set. For RF, Δ is set to $1 + 1.645 \times \sqrt{N^*}$, where $N^* = |\mathcal{I}| \times \sum_{j \in \mathcal{J}} x_j$. For RV, γ_0 and γ in the function $\gamma_0 + \gamma \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{N}} y_{i,j}$ are obtained as the coefficients of the line tangent to $1 + 1.645 \times \sqrt{N}$ at $\tilde{N}$, which is the expected number of ENs instead of *all* ENs. This function reduces the conservatism of the solution of RV as a fewer number of links are protected against latency variations, dependent on the number of edge nodes that are expected to be opened; the expected number of ENs is simply calculated based on the computational requirements alone as $\lceil \frac{\upsilon \cdot \sum_i d_i}{0.7 C_j} \rceil$. All optimization problems are solved with CPLEX (version 22.1.1) with default settings through its Python API on a MacBook Air with an M1 processor and 16 GB RAM running macOS Ventura version 13.6.1. The source code for the optimization and evaluation are available at https://github.com/gpremsan/robust-edge-placement-latency-uncertainty/.

Simulation setup and performance metrics. The four different models – D, DC, RF and RV – are first compared in terms of the number of edge nodes on which the application instances are placed, the value

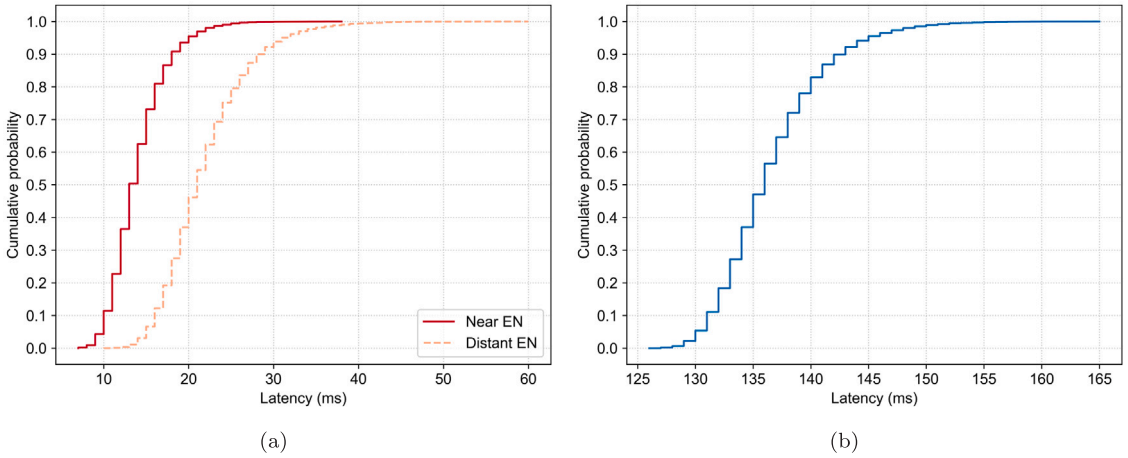

Fig. 2. CDF of values sampled from a GEV distribution for latency between a representative region and (a) near and distance edge nodes, and (b) the cloud.

of the objective function (Eq. (1)) and CPU time (in seconds) taken to obtain a solution. A lower number is better for all evaluated metrics. Next, the robustness of the solutions is evaluated through Monte Carlo simulations, wherein we evaluate each model's solution under different realizations of latency. The latency values are drawn from a generalized extreme value (GEV) distribution that is found to be the best fit for modeling the long tail of latency values seen in practical edge computing environments [11]. In our evaluation, we consider a type I GEV distribution with the *location* parameter set to the average latency ($\bar{t}_{i,j}$) in the optimization problems, and the *scale* parameter (that controls the spread in the sampled values) to half the average latency for the network links between regions and the cloud, and 0.2 times the location value for the links between ENs and regions. Fig. 2 shows the CDF of the sampled latency values from an example region to a nearby EN, distant EN and cloud, with the values drawn from the described GEV distribution. The figure shows that although the median latency is low, there is a long tail of latency values as is typical in such networks [11,12]. In each Monte Carlo simulation, we evaluate the number of latency violations (i.e., the number of times the average latency exceeds the configured threshold, expressed as a percentage of the 10,000 Monte Carlo sample runs), the average duration of the latency violations and the maximum duration of the latency violations.

All reported results are the average of 10 randomly generated instances. Specifically, for each instance, we generate 10 separate networks with randomly-generated network layouts (to determine the edge nodes closest to each region), demand and latency values. We run Monte Carlo simulations for each network with 10,000 samples from the GEV distribution, before reporting the average and standard deviation of the performance metrics for each instance. Note that we report the weighted average and weighted standard deviation for the average duration of the latency violations, as the number of violations varies across the iterations. Specifically, the weighted average latency duration is calculated by weighting each sample with the number of latency violations in the particular iteration. The maximum duration of latency violations is the maximum across all iterations for each instance.

5.1. Costs comparison and analysis

In this section, we first evaluate the solutions obtained by the four methods, in terms of the value of the objective function, the time taken to obtain a solution, and the number of edge nodes on which the application instances are placed. Table 4 reports the results for the evaluated metrics. First, we focus on the number of edge nodes chosen by each method. The table shows that on average the number of chosen ENs is lowest for D, with up to one or two more ENs chosen by the other

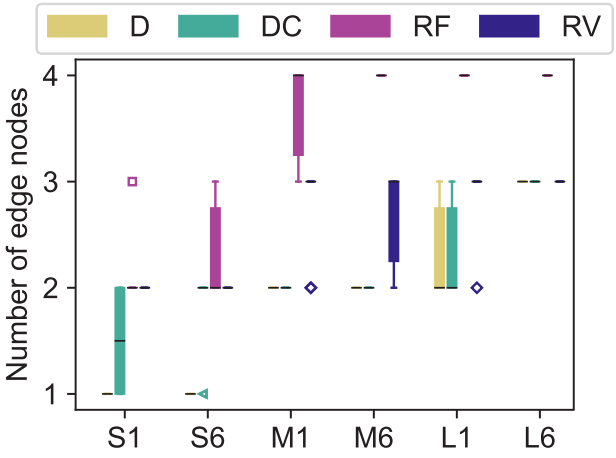

Fig. 3. Box plot of the number of edge nodes chosen by each method for two small, medium and large test instances.

methods. This is expected as D only considers the average latency $\bar{t}_{i,j}$ between the regions and the compute nodes, and thus, optimistically assumes that the latency threshold can be met. DC chooses up to one more edge node in a few test instances (16 out of the 300 networks) as it has a more conservative estimate of the average latency, resulting in more conservative solutions. On the other hand, when comparing the solutions of the robust models, RF tends to choose more edge nodes than RV. However, across all iterations, RF chooses a maximum of two more edge nodes than D and a maximum of one more edge node than DC solution, whereas RV chooses only up to one more edge node than D. The number of edge nodes chosen by the methods is exactly the same in 156 out of the 300 tested networks. This includes instance types 7–10 where the resilience (R) is set to a high value, wherein the number of edge nodes is exactly the same across all iterations, with R edge nodes chosen in each case. Fig. 3 presents a visual summary of the number of chosen edge nodes for a subset of the test instances from each category. The figure clearly shows that the number of edge nodes varies between 1–2 for different iterations. However, even when the number of chosen edge nodes are similar, the robustness of the solutions varies as we will investigate in the next section.

Next, Fig. 4 shows the utilization of individual ENs across all iterations for small, medium and large instances. The figure shows that D keeps the median utilization level at a higher level than other methods. DC is able to keep the utilization levels similar to D in the medium and large instances, whereas the median utilization for DC is 8% lower

Table 4
Summary of the number of chosen edge nodes, objective function value and solution time per test instance.

Instance	Total cost				Number of chosen ENs				CPU time (s)			
	D	DC	RF	RV	D	DC	RF	RV	D	DC	RF	RV
S1	193.37 ± 19.93	198.37 ± 22.07	204.36 ± 22.00	203.37 ± 19.93	1.00 ± 0.00	1.50 ± 0.50	2.10 ± 0.30	2.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.00
S2	775.63 ± 29.21	775.63 ± 29.21	775.63 ± 29.21	775.63 ± 29.21	3.60 ± 0.49	3.60 ± 0.49	3.60 ± 0.49	3.60 ± 0.49	0.03 ± 0.04	0.03 ± 0.04	0.02 ± 0.03	0.05 ± 0.03
S3	262.54 ± 12.75	265.54 ± 9.80	269.54 ± 11.43	266.54 ± 8.31	1.60 ± 0.49	1.90 ± 0.30	2.30 ± 0.46	2.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.01	0.02 ± 0.00
S4	1081.30 ± 24.92	1081.30 ± 24.92	1081.30 ± 24.92	1081.30 ± 24.92	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
S5	363.77 ± 8.31	363.77 ± 8.31	366.77 ± 11.43	363.77 ± 8.31	2.00 ± 0.00	2.00 ± 0.00	2.30 ± 0.46	2.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.00
S6	216.26 ± 4.98	224.26 ± 6.63	229.26 ± 8.38	226.26 ± 4.98	1.00 ± 0.00	1.80 ± 0.40	2.30 ± 0.46	2.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.02 ± 0.01	0.02 ± 0.00
S7	203.37 ± 19.93	203.37 ± 19.93	204.37 ± 22.00	203.37 ± 19.93	2.00 ± 0.00	2.00 ± 0.00	2.10 ± 0.30	2.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
S8	266.54 ± 8.31	266.54 ± 8.31	269.54 ± 11.43	266.54 ± 8.31	2.00 ± 0.00	2.00 ± 0.00	2.30 ± 0.46	2.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.01 ± 0.00
S9	226.26 ± 4.98	226.26 ± 4.98	229.26 ± 8.38	226.26 ± 4.98	2.00 ± 0.00	2.00 ± 0.00	2.30 ± 0.46	2.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.02 ± 0.01
S10	373.77 ± 8.31	373.77 ± 8.31	373.77 ± 8.31	373.77 ± 8.31	3.00 ± 0.00	3.00 ± 0.00	3.00 ± 0.00	3.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
M1	335.55 ± 18.96	335.55 ± 18.96	352.55 ± 21.47	343.55 ± 20.83	2.00 ± 0.00	2.00 ± 0.00	3.70 ± 0.46	2.80 ± 0.40	0.01 ± 0.00	0.01 ± 0.00	0.05 ± 0.01	0.24 ± 0.16
M2	1373.20 ± 23.70	1373.20 ± 23.70	1373.20 ± 23.70	1373.20 ± 23.70	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	0.04 ± 0.01	0.04 ± 0.01	0.14 ± 0.01	0.58 ± 0.16
M3	457.73 ± 7.90	457.73 ± 7.90	477.73 ± 7.90	464.73 ± 9.53	2.00 ± 0.00	2.00 ± 0.00	4.00 ± 0.00	2.70 ± 0.46	0.01 ± 0.00	0.01 ± 0.00	0.06 ± 0.01	0.18 ± 0.09
M4	1918.20 ± 23.70	1918.20 ± 23.70	1918.20 ± 23.70	1918.20 ± 23.70	8.00 ± 0.00	8.00 ± 0.00	8.00 ± 0.00	8.00 ± 0.00	0.04 ± 0.01	0.04 ± 0.02	0.07 ± 0.02	0.14 ± 0.07
M5	642.73 ± 7.90	642.73 ± 7.90	652.73 ± 7.90	642.73 ± 7.90	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.03 ± 0.01	0.04 ± 0.01	0.06 ± 0.01	0.49 ± 0.17
M6	387.64 ± 4.74	387.64 ± 4.74	407.64 ± 4.74	394.64 ± 6.93	2.00 ± 0.00	2.00 ± 0.00	4.00 ± 0.00	2.70 ± 0.46	0.02 ± 0.01	0.01 ± 0.00	0.06 ± 0.01	0.22 ± 0.07
M7	345.55 ± 18.96	345.55 ± 18.96	352.55 ± 21.47	345.55 ± 18.96	3.00 ± 0.00	3.00 ± 0.00	3.70 ± 0.46	3.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.04 ± 0.02	0.07 ± 0.03
M8	355.55 ± 18.96	355.55 ± 18.96	355.55 ± 18.96	355.55 ± 18.96	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.02 ± 0.01	0.04 ± 0.01
M9	397.64 ± 4.74	397.64 ± 4.74	407.64 ± 4.74	397.64 ± 4.74	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.05 ± 0.01	0.07 ± 0.03
M10	407.64 ± 4.74	407.64 ± 4.74	407.64 ± 4.74	407.64 ± 4.74	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.02 ± 0.01	0.04 ± 0.01
L1	478.18 ± 33.68	478.18 ± 33.68	495.18 ± 29.33	484.18 ± 29.82	2.30 ± 0.46	2.30 ± 0.46	4.00 ± 0.00	2.90 ± 0.30	0.03 ± 0.02	0.02 ± 0.01	0.09 ± 0.02	0.59 ± 0.59
L2	1925.04 ± 36.66	1925.04 ± 36.66	1925.04 ± 36.66	1925.04 ± 36.66	8.00 ± 0.00	8.00 ± 0.00	8.00 ± 0.00	8.00 ± 0.00	0.04 ± 0.01	0.04 ± 0.01	0.08 ± 0.02	0.19 ± 0.04
L3	645.01 ± 12.22	645.01 ± 12.22	655.01 ± 12.22	645.01 ± 12.22	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.04 ± 0.02	0.04 ± 0.01	0.09 ± 0.01	0.88 ± 0.22
L4	1347.10 ± 18.33	1347.10 ± 18.33	1347.10 ± 18.33	1347.10 ± 18.33	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	0.05 ± 0.00	0.05 ± 0.01	0.22 ± 0.02	0.78 ± 0.13
L5	898.07 ± 12.22	898.07 ± 12.22	898.07 ± 12.22	898.07 ± 12.22	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	0.05 ± 0.01	0.06 ± 0.01	0.17 ± 0.04	1.80 ± 0.79
L6	544.84 ± 7.33	544.84 ± 7.33	554.84 ± 7.33	544.84 ± 7.33	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.03 ± 0.00	0.03 ± 0.01	0.10 ± 0.02	0.59 ± 0.25
L7	485.18 ± 29.33	485.18 ± 29.33	495.18 ± 29.33	485.18 ± 29.33	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.02 ± 0.01	0.01 ± 0.01	0.11 ± 0.03	0.14 ± 0.12
L8	495.18 ± 29.33	495.18 ± 29.33	495.18 ± 29.33	495.18 ± 29.33	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.01	0.05 ± 0.01
L9	544.84 ± 7.33	544.84 ± 7.33	554.84 ± 7.33	544.84 ± 7.33	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.12 ± 0.03	0.14 ± 0.05
L10	554.84 ± 7.33	554.84 ± 7.33	554.84 ± 7.33	554.84 ± 7.33	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	4.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.02 ± 0.01	0.08 ± 0.03

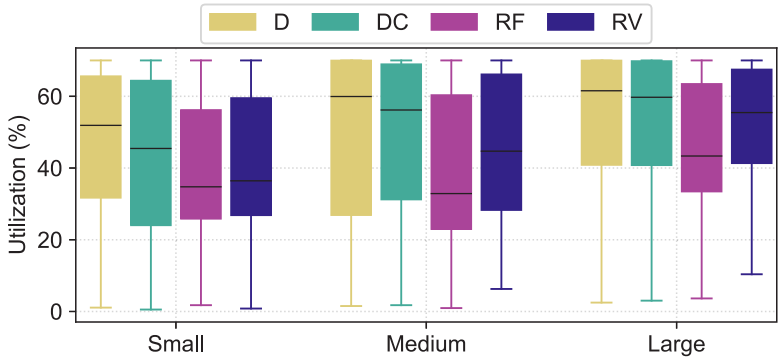

Fig. 4. Box plot of utilization of ENs for small, medium and large test instances.

than the median utilization of D in the small instances. The robust solutions, RF and RV, are able to keep the utilization values reasonably high despite choosing more ENs to meet the latency deadlines (see next section). All methods result in sometimes running ENs with a low utilization; this is mainly due to the additional ENs chosen to meet the strict latency deadline as well as the high resilience requirement in instances 7–10 of each category. Note that the utilization is restricted to 70% of the capacity when solving the optimization models, and thus, the maximum utilization never exceeds this value.

Finally, all solutions are obtained within a short time (see Table 4), with a maximum of 15.68 s for RV and a solution time of below 1 s for the vast majority of the test instances (248 out of 300 networks). The solution time for other methods (D, DC and RF) is below 1 s for all tested networks. Thus, the proposed robust models can easily obtain solutions well in time for the 10-min time slots used in this paper. Moreover, the short solution time implies that the robust models can be used to make placement and allocation decisions for shorter time slots as well.

5.2. Robustness of solutions

We investigate the robustness of the solutions through Monte Carlo simulations. Table 5 reports the number of latency violations observed in the simulations, i.e., the number of times the average latency of requests served is above the latency threshold (L) of 20 ms. The results show that the robust solutions are consistently able to maintain a low number of latency violations, with the fewest number of violations for RF. There are no latency violations with RF for the medium and large networks. In fact, on average, the number of requests that experience a latency greater than the set threshold is close to zero with RF and below 20 with RV. This is much lower than the thousands of requests that exceed the latency threshold with both D and DC. One reason for this is that RF tends to choose more edge nodes than the other methods. However, the number of latency violations is lower for the robust solutions even in cases where the number of edge nodes chosen by the deterministic methods is the same. Thus, in addition to the actual choice of edge nodes, the number of requests assigned to different edge nodes also plays a crucial role in keeping the average latency low.

Table 5
Summary of the number of latency violations, average and maximum duration of the latency violations per test instance.

Instance	Number of latency violations (%)				Average duration of latency violations (ms)				Maximum duration of latency violations (ms)			
	D	DC	RF	RV	D	DC	RF	RV	D	DC	RF	RV
S1	42.72 ± 19.55	11.03 ± 6.17	0.0 ± 0.00	0.03 ± 0.03	1.55 ± 0.37	1.00 ± 0.09	0.38 ± 0.35	0.55 ± 0.25	10.77	8.76	1.31	2.26
S2	32.63 ± 33.24	16.76 ± 2.32	0.0 ± 0.00	0.00 ± 0.00	1.78 ± 0.51	0.94 ± 0.04	0.37 ± 0.44	0.39 ± 0.54	10.04	7.99	0.68	1.01
S3	36.70 ± 23.61	11.12 ± 8.06	0.0 ± 0.00	0.01 ± 0.02	1.48 ± 0.44	0.93 ± 0.09	0.20 ± 0.18	0.64 ± 0.38	10.47	6.39	0.33	2.86
S4	45.35 ± 31.29	11.47 ± 7.38	0.0 ± 0.00	0.00 ± 0.00	1.70 ± 0.43	0.91 ± 0.07	0.09 ± 0.00	0.16 ± 0.17	9.78	6.89	0.09	0.28
S5	41.10 ± 36.00	14.46 ± 7.20	0.0 ± 0.01	0.03 ± 0.03	1.89 ± 0.44	0.95 ± 0.07	0.31 ± 0.15	0.59 ± 0.30	10.02	7.34	0.63	2.97
S6	42.07 ± 21.67	10.56 ± 7.12	0.0 ± 0.00	0.03 ± 0.03	1.46 ± 0.26	0.92 ± 0.09	0.33 ± 0.35	0.60 ± 0.31	9.87	6.08	0.71	2.06
S7	43.72 ± 33.61	15.74 ± 5.40	0.0 ± 0.00	0.03 ± 0.07	1.91 ± 0.39	1.04 ± 0.09	0.64 ± 0.32	0.63 ± 0.23	13.76	8.74	1.31	3.57
S8	41.98 ± 34.03	9.50 ± 7.12	0.0 ± 0.00	0.02 ± 0.03	1.81 ± 0.38	0.94 ± 0.06	0.74 ± 0.00	0.67 ± 0.22	10.86	8.63	0.74	2.24
S9	47.12 ± 35.87	10.41 ± 7.55	0.0 ± 0.00	0.07 ± 0.10	1.89 ± 0.31	0.95 ± 0.05	0.69 ± 0.00	0.63 ± 0.14	12.42	8.51	0.69	3.74
S10	34.06 ± 30.49	12.45 ± 5.93	0.0 ± 0.00	0.01 ± 0.01	1.65 ± 0.42	0.92 ± 0.05	0.46 ± 0.00	0.80 ± 0.43	9.83	7.83	0.46	2.46
M1	87.90 ± 5.96	10.53 ± 2.82	0.0 ± 0.00	0.04 ± 0.03	1.87 ± 0.19	0.68 ± 0.06	0.00 ± 0.00	0.35 ± 0.18	9.17	4.68	0.00	1.66
M2	51.54 ± 37.08	6.85 ± 5.14	0.0 ± 0.00	0.00 ± 0.00	1.56 ± 0.41	0.60 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	9.97	4.17	0.00	0.00
M3	86.79 ± 11.63	9.32 ± 3.64	0.0 ± 0.00	0.13 ± 0.11	1.81 ± 0.28	0.61 ± 0.03	0.00 ± 0.00	0.33 ± 0.06	8.94	5.26	0.00	2.17
M4	50.40 ± 34.47	7.85 ± 4.30	0.0 ± 0.00	0.00 ± 0.00	1.44 ± 0.40	0.58 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	7.04	4.17	0.00	0.00
M5	80.54 ± 17.04	10.95 ± 0.88	0.0 ± 0.00	0.08 ± 0.10	1.73 ± 0.38	0.63 ± 0.03	0.00 ± 0.00	0.31 ± 0.08	9.65	4.31	0.00	1.68
M6	83.47 ± 17.26	9.92 ± 2.60	0.0 ± 0.00	0.18 ± 0.16	1.81 ± 0.32	0.61 ± 0.04	0.00 ± 0.00	0.33 ± 0.06	7.62	4.66	0.00	1.86
M7	84.70 ± 13.92	12.83 ± 1.30	0.0 ± 0.00	0.02 ± 0.03	1.89 ± 0.29	0.71 ± 0.04	0.00 ± 0.00	0.30 ± 0.13	9.36	6.19	0.00	1.03
M8	82.38 ± 19.83	12.07 ± 2.87	0.0 ± 0.00	0.00 ± 0.01	1.90 ± 0.29	0.69 ± 0.03	0.00 ± 0.00	0.51 ± 0.16	9.85	6.09	0.00	0.89
M9	75.87 ± 20.00	11.82 ± 1.24	0.0 ± 0.00	0.04 ± 0.06	1.65 ± 0.39	0.64 ± 0.02	0.00 ± 0.00	0.42 ± 0.07	8.51	5.10	0.00	1.62
M10	75.89 ± 19.49	11.11 ± 2.96	0.0 ± 0.00	0.01 ± 0.03	1.63 ± 0.37	0.64 ± 0.03	0.00 ± 0.00	0.36 ± 0.09	8.24	4.21	0.00	1.46
L1	87.29 ± 18.65	8.89 ± 3.33	0.0 ± 0.00	0.10 ± 0.09	1.76 ± 0.26	0.54 ± 0.04	0.00 ± 0.00	0.29 ± 0.08	7.89	3.64	0.00	1.6
L2	60.93 ± 25.74	3.37 ± 2.97	0.0 ± 0.00	0.00 ± 0.00	1.20 ± 0.38	0.47 ± 0.04	0.05 ± 0.00	0.00 ± 0.00	6.66	3.31	0.07	0.00
L3	80.90 ± 18.77	7.00 ± 2.71	0.0 ± 0.00	0.08 ± 0.13	1.55 ± 0.39	0.49 ± 0.02	0.00 ± 0.00	0.31 ± 0.05	7.32	3.21	0.00	1.13
L4	71.36 ± 27.29	8.27 ± 2.66	0.0 ± 0.00	0.00 ± 0.00	1.42 ± 0.39	0.51 ± 0.02	0.14 ± 0.00	0.00 ± 0.00	7.06	3.24	0.14	0.00
L5	63.00 ± 31.95	6.53 ± 3.24	0.0 ± 0.00	0.00 ± 0.00	1.43 ± 0.44	0.49 ± 0.03	0.00 ± 0.00	0.23 ± 0.00	6.77	4.15	0.00	0.38
L6	88.33 ± 13.26	4.71 ± 3.82	0.0 ± 0.00	0.24 ± 0.34	1.70 ± 0.34	0.49 ± 0.04	0.00 ± 0.00	0.30 ± 0.06	6.65	4.10	0.00	2.36
L7	81.12 ± 27.45	9.56 ± 2.49	0.0 ± 0.00	0.01 ± 0.02	1.67 ± 0.26	0.54 ± 0.03	0.00 ± 0.00	0.19 ± 0.07	7.00	3.82	0.00	0.53
L8	56.21 ± 25.41	8.51 ± 3.26	0.0 ± 0.00	0.00 ± 0.01	1.17 ± 0.27	0.56 ± 0.02	0.00 ± 0.00	0.16 ± 0.06	5.94	3.94	0.00	0.31
L9	76.34 ± 24.00	8.63 ± 1.03	0.0 ± 0.00	0.01 ± 0.01	1.54 ± 0.41	0.51 ± 0.02	0.00 ± 0.00	0.21 ± 0.12	6.95	3.57	0.00	0.71
L10	58.23 ± 31.42	7.55 ± 2.88	0.0 ± 0.00	0.00 ± 0.01	1.33 ± 0.45	0.50 ± 0.03	0.00 ± 0.00	0.61 ± 0.12	7.01	3.71	0.00	1.90

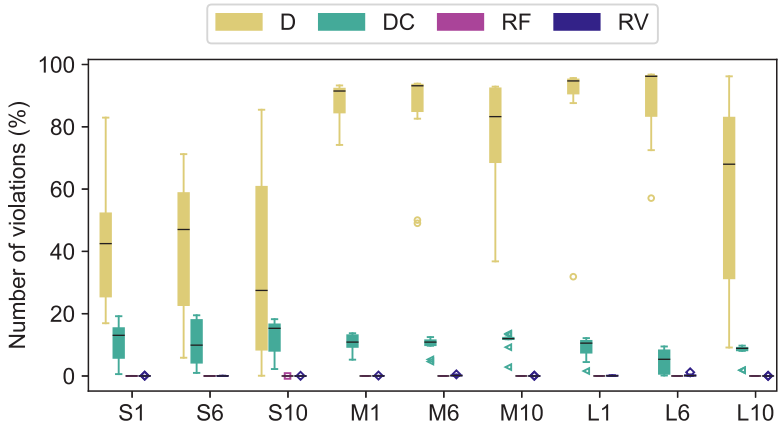

Fig. 5. Box plot of the number of latency violations (as a percentage of the number of iterations in the simulations) for three small, medium and large test instances.

Fig. 5 shows the number of latency violations for three small, medium and large instances types across the ten different instances. Here, the networks in instance type 10 have the same number of edge nodes for all four methods. The figure clearly shows that the number of latency violations is higher for both D and DC. We note that RV is able to keep the number of latency violations lower than D, despite choosing the same number of edge nodes in the majority of the test instances (248 out of 300 networks). The robust solutions tend to assign the demand to multiple edge nodes to be able to withstand variations in the latency on certain network links. On the other hand, D and DC optimistically assign the demand from regions to a single edge node, and thus, cannot meet the latency thresholds when the latency on that link varies. This is despite optimizing with a more conservative estimate of network latency in DC. The large standard deviation of the results for D and DC indicates that these solutions can sometimes keep the number of latency violations low. However, this is instance- and iteration-dependent. In contrast, the robust models (RF and RV) are able to keep the latency violations in all iterations and instances low, as indicated by the low spread in the results.

Next, we examine the duration of the latency violations. Table 5 shows that on average, the latency is violated by 1 to 2 ms for the deterministic models, whereas the value is below 1 ms for the robust solutions. Although the average duration seems low for DC and D, the standard deviation of the results indicates that D exhibits large variations in the duration of the latency violations, especially for medium and large instances. This is also indicated by the maximum duration by which the latency is violated, which are higher for the deterministic models than the robust models. To examine this more closely, Fig. 6 shows the duration of latency violations across all Monte Carlo simulations for a subset of networks (three small, medium and large test instance types). The figure shows that the robust solutions are consistently able to keep the number of latency violations low, with a low median value. In contrast, both DC and D have several cases where the latency exceeds the threshold by more than 5 ms, with a

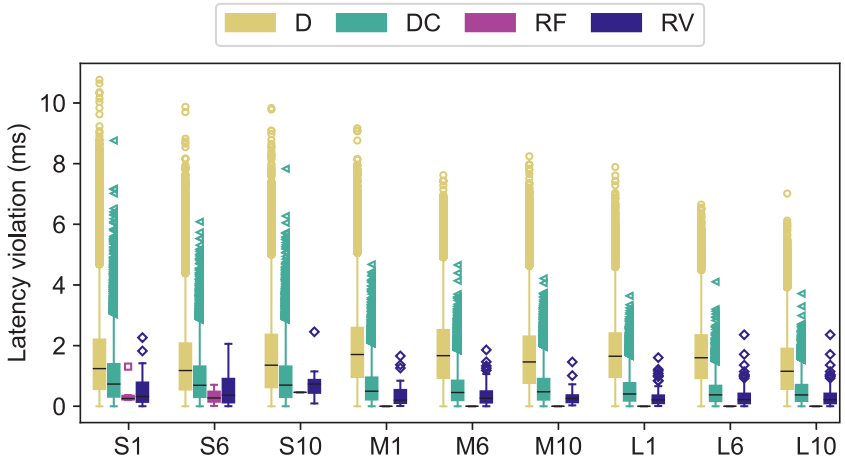

Fig. 6. Box plot of duration of latency violations for three small, medium and large test instances.

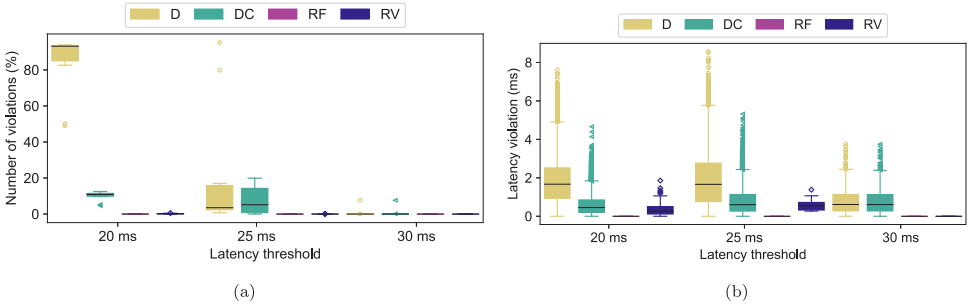

Fig. 7. Box plot of (a) number of latency violations (as a percentage of number of iterations in the simulations) and (b) duration of latency violations for M6 test instance under three different latency thresholds.

maximum of 8.76 ms and 13.76 ms respectively. These violations in latency are very detrimental to the application provider. In contrast, the robust solutions are able to keep the duration of the latency violations consistently low, with a maximum duration of 4.16 ms with RV and 1.31 ms with RF.

5.3. Sensitivity analysis

We present a sensitivity analysis of the deterministic and robust models' solutions to the following parameters: (i) the latency threshold of L which restricts the average latency for served requests in Eqs. (4) and (14), and (ii) the computing capacity threshold in Eq. (3) which restricts the maximum utilization of edge nodes.

Varying the latency threshold. We increase the latency threshold (L) from 20 ms to 25 ms and 30 ms, and evaluate all test instances using the same methodology as before. As expected, increasing the latency threshold results in fewer latency violations for all methods. Similar to the previous results, the number of latency violations and duration of the maximum violation are lower for the robust solutions. The number of chosen edge nodes are similar to that with the lower latency threshold in all cases. Due to space constraints, we only report the results for a single network (M6) with three different latency thresholds of 20 ms, 25 ms and 30 ms. A similar trend (as discussed next) is observed in all other test instances. Fig. 7(a) shows that the number of latency violations drop when the latency threshold increases. When the latency threshold is 30 ms, all methods show very few latency violations, with up to 20% with D or DC. RV and RF are able to keep the number of latency violations lower than the deterministic solutions in all iterations. Fig. 7(b) shows that the duration of the latency violations also reduces when the latency threshold is increased. Again, RV and RF are able to keep the duration of the latency violations very low. In all simulations, two ENs are chosen by all four methods, and thus, the objective function values are the same (not presented in the figure).

Varying the computing capacity threshold. So far we restricted the utilization of edge nodes (κ) to 70%, as is standard practice in data centers for cloud computing [39,41]. In this section, we experiment with varying κ between 60% to 80% (in increments of 5%) to understand the impact of this threshold on the results. In many test instances, the number of chosen ENs and latency violations are similar to the previous results with κ set to 70%. This is due to two reasons. First, in these test instances, the ENs were not fully utilized (i.e., their utilization was not close to 70%) and thus, increasing or decreasing this threshold by up to 10% did not impact the results. Second, the resilience constraints also mandate that a certain number of ENs are chosen, in which case, again there is not much impact of changing the utilization threshold.

Thus, we focus on large instances as they represent scenarios where the edge nodes are almost fully utilized in the original results. Due to space limitations, we present the results for L4 and L6 instances. Table 6 presents the results from the sensitivity analysis by varying the utilization threshold κ. We see that the number of chosen ENs increase as the utilization threshold is reduced; this is because more ENs are required to satisfy the incoming demand while still meeting latency and capacity constraints. The number of latency violations and duration of latency violations are similar across different utilization thresholds, with RV and RF consistently better than the deterministic solutions. The latency constraint in optimization problem captures only the network latency and does not account for the processing time on the EN (which is a function of the utilization of the EN) – this is left as future work where we will examine the impact of increasing the utilization of ENs on the overall latency.

5.4. Summary and discussion

The results presented in this paper show that our robust models, RF and RV, are able to devise placement and traffic allocation strategies for

Table 6
Summary of results for L4 and L6 test instances with different thresholds for the maximum computing capacity.

κ	% of latency violations				Average duration of latency violations (ms)				Max. duration of latency violations (ms)				Number of chosen ENs			
	D	DC	RF	RV	D	DC	RF	RV	D	DC	RF	RV	D	DC	RF	RV
L4 test instance																
60%	62.36 ± 34.34	6.58 ± 3.48	0.0 ± 0.00	0.0 ± 0.00	1.54 ± 0.47	0.51 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	7.22	3.71	0.00	0.00	7.00 ± 0.00	7.00 ± 0.00	7.00 ± 0.00	7.00 ± 0.00
65%	67.09 ± 30.57	7.75 ± 2.15	0.0 ± 0.00	0.0 ± 0.00	1.36 ± 0.33	0.49 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	6.25	3.65	0.00	0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00
70%	71.36 ± 27.29	8.27 ± 2.66	0.1 ± 0.30	0.0 ± 0.00	1.42 ± 0.39	0.51 ± 0.02	0.14 ± 0.00	0.00 ± 0.00	7.06	3.24	0.14	0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00	6.00 ± 0.00
75%	73.39 ± 32.48	7.74 ± 2.52	0.0 ± 0.00	0.0 ± 0.00	1.52 ± 0.29	0.50 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	6.33	4.36	0.00	0.00	5.20 ± 0.40	5.20 ± 0.40	5.20 ± 0.40	5.20 ± 0.40
80%	76.35 ± 22.87	6.78 ± 2.65	0.0 ± 0.00	0.0 ± 0.00	1.52 ± 0.43	0.50 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	7.16	3.92	0.00	0.00	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00	5.00 ± 0.00
L6 test instance																
60%	82.86 ± 23.57	7.54 ± 2.83	0.0 ± 0.00	0.06 ± 0.09	1.66 ± 0.33	0.49 ± 0.03	0.00 ± 0.00	0.28 ± 0.09	7.11	3.34	0.00	1.61	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00
65%	89.37 ± 10.12	5.75 ± 3.27	0.0 ± 0.00	0.05 ± 0.09	1.67 ± 0.33	0.49 ± 0.03	0.00 ± 0.00	0.22 ± 0.06	6.60	2.86	0.00	0.95	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00
70%	88.33 ± 13.26	4.71 ± 3.82	0.0 ± 0.00	0.24 ± 0.34	1.70 ± 0.34	0.49 ± 0.04	0.00 ± 0.00	0.30 ± 0.06	6.65	4.10	0.00	2.36	3.00 ± 0.00	3.00 ± 0.00	4.00 ± 0.00	3.00 ± 0.00
75%	92.59 ± 10.73	7.36 ± 2.66	0.0 ± 0.00	0.07 ± 0.07	1.81 ± 0.24	0.49 ± 0.03	0.00 ± 0.00	0.30 ± 0.10	7.27	3.99	0.00	1.49	2.20 ± 0.40	2.30 ± 0.46	4.00 ± 0.00	3.00 ± 0.00
80%	89.01 ± 14.44	8.27 ± 2.15	0.0 ± 0.00	0.12 ± 0.21	1.76 ± 0.32	0.49 ± 0.03	0.00 ± 0.00	0.28 ± 0.05	7.14	3.52	0.00	1.38	2.00 ± 0.00	2.00 ± 0.00	4.00 ± 0.00	2.90 ± 0.30

applications such that the latency of individual requests is consistently below the target latency threshold despite variations in the network latency. This is possible with a few number of edge nodes, often comparable with the deterministic models and only up to one or two edge nodes chosen in certain instances. Among the robust models, RF chooses up to one more edge node than RV in our evaluated instances. This allows RF to keep both the number of latency violations and maximum duration of the latency violation lower than RV. The choice of a particular robust model depends on the type of application and the application provider's tolerance for latency violations. For certain latency-sensitive applications, such as safety or mission-critical applications [12], the latency for each request originating from a particular region must be met with a latency close to the target threshold. For such applications, the latency for all requests must be served with a low latency and missing the deadline for even a single request can be catastrophic [12]. The RF method is a good candidate for determining edge placement strategies for such latency-sensitive applications. On the other hand, RV is suitable for less latency-sensitive applications where the latency may be violated for a few requests without severe consequences. For such applications, RV offers lower costs by choosing fewer number of edge nodes, while at the same time reducing the number of latency violations as compared to the deterministic models. Finally, the solution time for the robust models is low and thus, the models can be used to obtain placement and allocation decisions in an online manner for a 10-min time slot. We also experimented with a few larger instances to investigate the impact on solution time. For example, with 50 regions and 15 edge nodes, RF and RV take a CPU time of 2.10 ± 0.94 and 32.54 ± 19.47 seconds respectively. When the number of regions (50) and edge nodes (25) are large, the CPU time increases to 551.94 ± 629.28 and 3198.65 ± 5513.30, which puts it out of reach in planning allocations for 10-min slots. However, in such cases, it would be possible to reduce the size of the problem (and solution time) by dividing the problem into sub-problems with fewer regions and edge nodes (This is possible because the geographical location of certain edge nodes may put it out reach of certain regions). Also, we anticipate that even if the number of edge nodes increases, the number of regions in a city would be at the most 20 to 25.

6. Conclusion

This article considered a latency-aware placement of applications and traffic allocation to the applications in edge computing such that strict latency deadlines can be met even with variations in the network latency. We proposed two robust optimization models that offer different levels of protection against variations in the network links between regions and edge nodes. This is achieved through two different polyhedral uncertainty sets. We presented efficient methods to solve the robust optimization models based on dualization. Through extensive simulations, we showed that our proposed solutions are able to keep the number of chosen edge nodes low while keeping the latency violations consistently low. Apart from this, the solutions were able to also reduce the duration by which the latency deadlines are exceeded, consistently below 4 ms across all tested instances. In future work, we plan to consider the problem from the perspective of an edge platform provider that has to place multiple applications and provide guarantees for meeting different application-specific latency deadlines. We also plan to incorporate the processing time (on servers) in the latency constraint to account for the impact of running edge nodes with higher utilization on overall latency (and thus, user experience). The problem from an edge computing provider-perspective would include additional constraints regarding the utilization of memory and computational resources in the edge nodes. The complexity of the problem would grow quickly as the number of applications with different latency thresholds increases, resulting in computational difficulties that require new methods to efficiently obtain a solution robust to variations in the network latency. Another promising area for future work is to consider the uncertainty in demand, as the incoming number of requests can also experience variations in different time slots. Also, the multistage robust optimization approach could be another option for future work. Since we assume a single time slot in this work, the application provider must solve the problem repeatedly for different time slots. It is an optimization problem in which decisions are made sequentially. However, in the context of robust optimization, decisions in a time slot may be affected by decisions and uncertainty realizations in previous time slots. Thus, decisions depend on previous time slots but not future time slots. These relationships between decision variables in the multistage robust optimization approach bring enormous computational difficulty. Another direction of interest is to consider multi-stage stochastic programming formulations and chance-constrained models and compare them to the robust model.

CRediT authorship contribution statement

Jaehee Jeong: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization. **Gopika Premsankar:** Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Bissan Ghaddar:** Writing – original draft, Supervision, Methodology, Funding acquisition, Conceptualization. **Sasu Tarkoma:** Writing – review & editing, Supervision, Methodology, Funding acquisition.

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

Data will be made available on request.

Acknowledgments

The authors appreciate the funding support from Waterloo Institute for Sustainable Energy, Ontario, Canada. Bissan Ghaddar was supported by NSERC Discovery Grant 2017-04185. Gopika Premsankar was supported by the Research Council of Finland under Grant 338854. The authors would like to thank Ekaterina Khlebova for analyzing latency datasets as part of background study for this work. Additionally, they would like to thank the anonymous referees for their helpful and insightful suggestions for improving the paper.

References

[1] Bülbül K, Noyan N, Erol H. Multi-stage stochastic programming models for provisioning cloud computing resources. European J Oper Res 2021;288(3):886–901.
[2] Satyanarayanan M. The emergence of edge computing. Computer 2017;50(1):30–9.
[3] Basu S, Chakraborty S, Sharma M. Pricing cloud services—the impact of broadband quality. Omega 2015;50:96–114.
[4] Premsankar G, Di Francesco M, Taleb T. Edge computing for the Internet of Things: A case study. IEEE Internet Things J 2018;5(2):1275–84.
[5] Xu M, Fu Z, Ma X, Zhang L, Li Y, Qian F, Wang S, Li K, Yang J, Liu X. From cloud to edge: a first look at public edge platforms. In: Proceedings of the 21st ACM internet measurement conference. 2021, p. 37–53.
[6] Microsoft Azure. Azure private edge zone. 2022, Accessed on 18 Aug 2022, https://docs.microsoft.com/en-us/azure/private-multi-access-edge-compute-mec/overview.
[7] Amazon. AWS Wavelength. 2022, Accessed on 18 Aug 2022, https://aws.amazon.com/wavelength/.
[8] Lin L, Liao X, Jin H, Li P. Computation offloading toward edge computing. Proc IEEE 2019;107(8):1584–607.
[9] Dean J, Barroso LA. The tail at scale. Commun ACM 2013;56(2):74–80.
[10] Zheng Z, Zhang Y, Lyu MR. Investigating QoS of real-world web services. IEEE Trans Serv Comput 2012;7(1):32–9.

[11] Gorlatova M, Inaltekin H, Chiang M. Characterizing task completion latencies in multi-point multi-quality fog computing systems. Comput Netw 2020;181:107526.
[12] Iorio M, Risso F, Casetti C. When latency matters: measurements and lessons learned. ACM SIGCOMM Comput Commun Rev 2021;51(4):2–13.
[13] Ljubić I, Mouaci A, Perrot N, Gourdin É. Benders decomposition for a node-capacitated virtual network function placement and routing problem. Comput Oper Res 2021;130:105227.
[14] Ceselli A, Fiore M, Premoli M, Secci S. Optimized assignment patterns in mobile edge cloud networks. Comput Oper Res 2019;106:246–59.
[15] Salaht FA, Desprez F, Lebre A. An overview of service placement problem in fog and edge computing. ACM Comput Surv 2020;53(3):1–35.
[16] Badri H, Bahreini T, Grosu D, Yang K. Energy-aware application placement in mobile edge computing: a stochastic optimization approach. IEEE Trans Parallel Distrib Syst 2019;31(4):909–22.
[17] Cheng J, Nguyen DT, Bhargava VK. Resilient edge service placement under demand and node failure uncertainties. IEEE Trans. Netw. Serv. Manag. 2024;21(1):558–73. http://dx.doi.org/10.1109/TNSM.2023.3290137.
[18] Nguyen DT, Nguyen HT, Trieu N, Bhargava VK. Two-stage robust edge service placement and sizing under demand uncertainty. IEEE Internet Things J 2021;1.
[19] Yu R, Xue G, Zhang X. Provisioning QoS-aware and robust applications in Internet of Things: A network perspective. IEEE/ACM Trans Netw 2019;27(5):1931–44.
[20] Gicquel C, Vanier S, Papadimitriou A. Optimal deployment of virtual network functions for securing telecommunication networks against distributed denial of service attacks: a robust optimization approach. Comput Oper Res 2022;105890.
[21] Bertsimas D, Sim M. The price of robustness. Oper Res 2004;52(1):35–53.
[22] Ben-Tal A, El Ghaoui L, Nemirovski A. Robust optimization. Vol. 28, Princeton University Press; 2009.
[23] Gabrel V, Murat C, Thiele A. Recent advances in robust optimization: An overview. European J Oper Res 2014;235(3):471–83.
[24] Pelletier S, Jabali O, Laporte G. The electric vehicle routing problem with energy consumption uncertainty. Transp Res B 2019;126:225–55.
[25] Lotfi R, Kheiri K, Sadeghi A, Babaee Tirkolaee E. An extended robust mathematical model to project the course of COVID-19 epidemic in Iran. Ann Oper Res 2022;1–25.
[26] Han B, Zhang Y, Wang S, Park Y. The efficient and stable planning for interrupted supply chain with dual-sourcing strategy: a robust optimization approach considering decision maker's risk attitude. Omega 2023;115:102775.
[27] Wang X, Jiang R, Qi M. A robust optimization problem for drone-based equitable pandemic vaccine distribution with uncertain supply. Omega 2023;119:102872.
[28] Qu Y, Dai H, Wu F, Lu D, Dong C, Tang S, Chen G. Robust offloading scheduling for mobile edge computing. IEEE Trans Mob Comput 2022;21(7):2581–95.
[29] Li R, Zhou Z, Zhang X, Chen X. Joint application placement and request routing optimization for dynamic edge computing service management. IEEE Trans Parallel Distrib Syst 2022;33(12):4581–96.
[30] Sun Y, Zhou S, Xu J. EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE J Sel Areas Commun 2017;35(11):2637–46.
[31] Gao B, Zhou Z, Liu F, Xu F. Winning at the starting line: Joint network selection and service placement for mobile edge computing. In: IEEE conference on computer communications 2019. IEEE; 2019, p. 1459–67.
[32] Aydın N, Muter İ, Birbil Şİ. Multi-objective temporal bin packing problem: An application in cloud computing. Comput Oper Res 2020;121:104959.
[33] Premsankar G, Ghaddar B. Energy-efficient service placement for latency-sensitive applications in edge computing. IEEE Internet Things J 2022;9(18):17926–37.
[34] Farhadi V, Mehmeti F, He T, La Porta TF, Khamfroush H, Wang S, Chan KS, Poularakis K. Service placement and request scheduling for data-intensive applications in edge clouds. IEEE/ACM Trans Netw 2021;29(2):779–92.
[35] Amazon AWS. What do I need to know about CPU allocation in Amazon ECS?. 2022, Accessed on 09 Aug 2022, https://aws.amazon.com/premiumsupport/knowledge-center/ecs-cpu-allocation/.
[36] Kubernetes. Resource Management for Pods and Containers: CPU resource units. 2022, Accessed on 09 Aug 2022, https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#meaning-of-cpu.
[37] Stillwell ML, Vivien F, Casanova H. Virtual machine resource allocation for service hosting on heterogeneous distributed platforms. In: IEEE 26th International Parallel and Distributed Processing Symposium. IEEE; 2012, p. 786–97.
[38] Delnat W, Truyen E, Rafique A, Van Landuyt D, Joosen W. K8-scalar: a workbench to compare autoscalers for container-orchestrated database clusters. In: Proceedings of the 13th International Conference on software engineering for adaptive and self-managing systems. 2018, p. 33–9.
[39] Jindal A, Podolskiy V, Gerndt M. Performance modeling for cloud microservice applications. In: Proceedings of the 2019 ACM/SPEC international conference on performance engineering. 2019, p. 25–32.
[40] Zhang H, Tang Y, Khandelwal A, Stoica I. SHEPHERD: Serving DNNs in the Wild. In: 20th USENIX Symposium on Networked Systems Design and Implementation. 2023, p. 787–808.
[41] Chou D, Xu T, Veeraraghavan K, Newell A, Margulis S, Xiao L, Ruiz PM, Meza J, Ha K, Padmanabha S, et al. Taiji: managing global user traffic for large-scale Internet services at the edge. In: Proceedings of the 27th ACM symposium on operating systems principles. 2019, p. 430–46.
[42] Høiland-Jørgensen T, Ahlgren B, Hurtig P, Brunstrom A. Measuring latency variation in the Internet. In: Proceedings of the 12th international conference on emerging networking experiments and technologies. 2016, p. 473–80.
[43] Heyman DP, Sobel MJ. Stochastic models in operations research: stochastic optimization. Vol. 2, Courier Corporation; 2004.
[44] Uryasev SP. Probabilistic constrained optimization: methodology and applications. Vol. 49, Springer Science & Business Media; 2000.
[45] Bertsimas D, Brown DB, Caramanis C. Theory and applications of robust optimization. SIAM Rev 2011;53(3):464–501.
[46] Poss M. Robust combinatorial optimization with variable budgeted uncertainty. 4OR 2013;11(1):75–92.
[47] Adams WP, Sherali HD. Linearization strategies for a class of zero-one mixed integer programming problems. Oper Res 1990;38(2):217–26.
[48] Soyster AL. Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper Res 1973;21(5):1154–7.