Stupar and Huljenic
Journal of Cloud Computing (2023) 12:23
https://doi.org/10.1186/s13677-023-00389-8

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Model-based cloud service deployment optimisation method for minimisation of application service operational cost

Ivana Stupar* and Darko Huljenic

Abstract

Many currently existing cloud cost optimisation solutions are aimed at cloud infrastructure providers, and they often deal only with specific types of application services. Unlike infrastructure providers, the providers of cloud applications are often left without a suitable cost optimisation solution, especially concerning the wide range of different application types. This paper presents an approach that aims to provide an optimisation solution for the providers of applications hosted in the cloud environments, applicable at the early phase of a cloud application lifecycle and for a wide range of application services. The focus of this research is the development of the method for identifying optimised service deployment option in available cloud environments based on the model of the service and its context, intending to minimise the operational cost of the cloud service while fulfilling the requirements defined by the service level agreement. A cloud application context metamodel is proposed that includes parameters related to both the application service and the cloud infrastructure relevant for the cost and quality of service. By using the proposed optimisation method, knowledge is gained about the effects of the cloud application context parameters on the service cost and quality of service, which is then used to determine the optimal service deployment option. The service models are validated using cloud applications deployed in laboratory conditions, and the optimisation method is validated using the simulations based on the proposed cloud application context metamodel. The experimental results based on two cloud application services demonstrate the ability of the proposed approach to provide relevant information about the impact of cloud application context parameters on service cost and quality of service and use this information for reducing service operational cost while preserving the acceptable service quality level. The results indicate the applicability and relevance of the proposed approach for cloud applications in the early service lifecycle phase since application providers can gain valuable insights regarding service deployment decision without acquiring extensive datasets for the analysis.

Keywords Cloud service, SaaS, IaaS, Cloud application, Operational cost, Cost optimisation method, Service model, Cloud application context, Quality of service, Cost management

Introduction

With the maturing of the cloud computing paradigm, many software service providers have workloads running in cloud environments [1]. A survey [2] on cloud computing trends among IT companies demonstrated that cloud computing costs account for a third of an organisation's overall IT spend. Survey results also indicated that 81% of surveyed organisations had at least one application or a part of their computing infrastructure in the cloud. Controlling cloud costs and taking full advantage of the purchased cloud resources were stated as some of the most common challenges organisations face. A survey [3] on the challenges related to cloud computing

*Correspondence:
Ivana Stupar
ivana.stupar@ericsson.com
Research Unit, Ericsson Nikola Tesla d.d., Zagreb, Croatia

paradigm adoption identified cost management as one of the most prominent issues reported by mature cloud users.

From the perspective of an application owner, deploying services to cloud environments often reduces costs by lowering capital expenses because application owners do not have to purchase the equipment necessary to host the application. Additionally, the hardware used for hosting applications on-premises is commonly dimensioned according to the application's peak load, making the available computing resources underutilised most of the time. By deploying application services to the cloud environments, application owners ensure that the allocation of IT resources needed for the application execution changes following the resource demand. As a result, application owners' financial commitments for on-premises hardware are significantly lowered, allowing them to invest in their core business. However, deciding on the best cloud service placement configuration is not simple and can significantly affect operational cost. Such placement decision includes the choice of a cloud infrastructure provider and virtual machine instances where the service or its components would be deployed. Cloud infrastructure providers charge for the computing resources they lend to their customers on demand using various pricing models [4]. Pricing models can significantly differ among many cloud infrastructure providers on the market, which further complicates the optimal choice. Finding an optimal service placement can be challenging for an application owner, not only because of various infrastructure providers and pricing models available on the market [5], but also due to the changing resource requirements caused by the dynamic service load. Often it is not trivial to determine which resources and in which amount are predominantly consumed by the application.

Since many variables affect the decision on the service placement configuration, in this paper, we examine an approach aiming at avoiding the suboptimal choice of application service deployment by using a model of the cloud application and its deployment environment to analyse and determine the parameters that affect the quality of service (QoS) and service operational cost. By conducting such an analysis, the parameters with the most significant effect on the given service operational cost and QoS can be identified. The knowledge about the parameters' influence can then be used to minimise service operational cost and fulfil service quality requirements. In order to create models that could be used for providing such knowledge, we define a cloud application context metamodel that contains both parameters related to the cloud application and its deployment environment. Based on the proposed metamodel, we create service models to predict how the application and cloud infrastructure parameters will affect the service performance and cost. The optimisation method presented in this paper uses the cloud application context metamodel and the knowledge about the effect that its parameters have on service operational cost and QoS to determine the optimal cloud service deployment configuration. The proposed optimisation method can be used by the application owners in the pre-deployment process of identifying the best service placement configuration so that unnecessary operational cost and additional redeployment expenses can be avoided.

With a significant number of various workflows executing in cloud environments, the proposed model-based approach can be used for the optimisation of a wide range of application types. In this paper, we use an example of two different application types to demonstrate how the knowledge about application services and their context could be used for optimising their deployment and minimising their operational cost. In our previous work [6], we already introduced the two applications used for the validation of our proposed solution. We also presented a detailed analysis of their resource consumption and ranking of the model parameters based on their influence on the service cost and QoS. In this paper, we summarise the results related to the predictive cost and QoS models presented in [6] and use them as input to the optimisation algorithm presented in this work. In this paper, our goal is to demonstrate the entire optimisation method, so in addition to the previously introduced QoS and cost models, we now present the cloud application context models for both use cases, describe the optimisation algorithm, and finally, we present the optimisation results and discuss the further work on the proposed solution.

The experimental results demonstrate that it is feasible to use service models and the proposed optimisation method to detect which parameters affect the service operational cost and QoS, as well as to use this knowledge in the decision process of determining optimised cloud service placement aimed at reducing the service operational cost. The remainder of this paper is structured as follows. Related work section brings an overview of the related work. Optimisation method section presents the proposed cloud application context metamodel and optimisation method. Use cases and analysis of application-related data section introduces two application services selected as use cases for optimisation, followed by Deployment optimisation method validation section, which brings the results of the validation of the proposed method. We conclude the paper in Conclusion section.

Related work

As the cloud is increasingly used in production environments, a considerable body of research is dealing with cost optimisation and SLA fulfilment in cloud environments, including different strategies for achieving the goal of cost minimisation. Some of the most common approaches include optimisation of load balancing [7] and service scaling algorithms [8], time scheduling of task invocation and performing [9, 10], as well as the strategy of optimising service placement [11, 12]. Various approaches to optimisation of cloud cost are using techniques from operations research and game theory [13], meta-heuristics [14, 15] with the focus on genetic and other evolutionary algorithms [9, 16]. Certain solutions focus on applying machine learning algorithms to the problem of cost and placement optimisation [12, 14, 17, 18], especially the approaches that aim at allocating resources dynamically in a cloud environment, thus often using machine learning techniques to realise adaptivity. As an example, Zhang et al. [12] present an architecture of a resource management system for cloud environments based on deep reinforcement learning. The proposed architecture consists of an intelligent resource manager that continually monitors resource utilisation and application QoS parameters, combines several optimising algorithms, and maps the application to the resources available in the second system component - resource pools.

In addition to achieving optimisation by using different techniques and strategies, the optimisation approaches also differ in terms of the workloads they focus on. Examples include scientific workflows [9, 19, 20], gaming [21], medical and healthcare [17, 22], education [23, 24], and big data [25, 26] workflows. In this paper, instead of focusing on a single type of workload, i.e. single application type, we provide a general approach based on service resource utilisation suitable for any application deployed as a SaaS service.

Research approaches in the area of cloud cost and placement optimisation are also divided by the perspective of stakeholders in the cloud ecosystem. As observed through literature review and several surveys [7, 14, 15, 27–29], many resource scheduling solutions and optimisation algorithms are intended for infrastructure service providers, dealing with the optimisation of resource allocation in terms of data centre resources [30–33]. Fewer solutions are provided for the optimisation of services for application providers. An example of service placement optimisation for application providers is presented in [11]. Although the presented approach considers the perspective of an application service provider, the solution is developed for the cloud resource market that offers negotiated prices and does not consider application-specific resource demands that might significantly affect the service cost.

Two studies closest to the modelling approach used as a part of the optimisation method presented in this paper are [34] and [35]. In [34], neural network and linear regression are used to predict CPU utilisation related to an e-commerce benchmark application using time series data. A similar approach is presented in [35], where a neural network is used to predict the execution time of observed tasks. However, this approach is focused on modelling a very specific task of building code in an online repository. The input variables of the presented model contain repository-specific information such as programming language. A model is created for each of the considered repositories, making this solution highly specific for the chosen use case. The authors of both works do not model the cost of the cloud service execution in the public cloud infrastructure. They also focus on predicting resource utilisation and do not examine which parameters affect the quality of service or service cost. In our research, we use interpretability techniques to identify the parameters with the most significant impact on the model output prediction.

Li et al. [11] develop a method and algorithms for cost-optimised cloud service placement in cloud environments subjected to dynamic pricing schemes. Although this approach is developed from the perspective of an application service provider and considers service cost, the solution is based on negotiated prices, and it is not considering application-specific resource demands that might significantly affect the service cost.

Several research attempts are considering solutions with application-specific models. In [36], the authors present a model and profile of application resource usage. This approach, however, was not based on applications deployed on virtualised infrastructure. Although the model parameters are related to processing power, storage, and network capacity, the resource model tracks some of the parameters that are, in most cases, not available to the cloud infrastructure users, nor does it include parameters more specific for cloud environments. Also, the primary purpose of the presented model is the prediction of the QoS parameters, while cost is not being considered, especially in terms of cloud-specific pricing models. The authors of [37] propose a web application model for a cloud data centre workload with the parameters of the user behaviour model derived from application analysis. The focus of this work was on creating dynamic user profile models in the context of a cloud environment to aid infrastructure utilisation optimisation. Such an approach provides helpful insight to infrastructure providers for capacity planning and reducing research and development costs.

As our goal is to provide a solution for a wide range of cloud application types, we also analysed the work related to the categorization of cloud application services. Several attempts have been made to categorise cloud application workloads, such as [38] and [39]. However, this job is most thoroughly covered in [40, 41], where groups of typical application workloads deployed in cloud environments are identified. Service categorisation in terms of service resource usage is essential for accurately meeting service resource requirements in a way that might minimise its cost. In the analysis performed in this paper, two application services are selected for the use cases based on the service categories identified in [41]. The categorisation of cloud application services presented in [41] is based on the Forward and Lethbridge taxonomy [42] which included nearly 200 application types grouped into four major categories and presents one of the most comprehensive analyses of application type classification.

Although the research that deals with cloud service cost optimisation is present in the literature, many of the available approaches do not consider multiple cost-affecting aspects of the service at the same time (e.g., deployment environment properties, service resource demand, user behaviour, and various pricing models offered by the cloud service providers). One of the aspects not included in the available research is the effect of the pricing model on the final service operational cost. Although the concept of pay-as-you-go means much more freedom and flexibility for customers, it usually means less flexibility for providers since it offers providers less possibility to predict, plan and evaluate their revenues accurately. For that reason, many providers offer alternative pricing models that often reduce the risk in their cost estimations and provide resources at lower prices to their users in case they choose packages with a predefined amount of resources and resource bundles. The market analysis [5] published at the end of 2013 brings an overview of the charging models of the Infrastructure as a Service (IaaS) cloud service delivery layer, including 53 IaaS providers in the analysis. Eventually, eight charging categories are formed based on the charging models offered by the IaaS providers, taking into account common features of the models. This analysis only deals with the IaaS charging models, and the comprehensive charging model for the cloud computing environment becomes even more complex when it includes PaaS and SaaS charging models. The current market is very heterogeneous regarding the pricing methods, and there are a lot of different offers varying considering the charged resources, charging unit, resource bundles, and predefined bundled offers [5, 43], etc. This fact additionally complicates the optimisation of the service cost in the cloud computing environment, especially when using the resources offered by multiple providers.

The literature review of the related research led to the conclusion that there is currently a lack of comprehensive solutions applicable in the early phases of the application lifecycle and to a wide range of cloud applications that would be able to optimise deployment configuration for a cloud application service while meeting the service QoS requirements and minimising its operational cost. In this paper, we present a model-based approach that enables the analysis of the impact that the parameters related to the application service and its deployment environment have on the service operational cost and QoS. The presented approach uses parameters that are not related to a specific application type and hence can be applied to various cloud application services, using the data that can be acquired before the application service reaches the operational phase so that the decision related to the application deployment strategy and infrastructure provider can be made on time.

Optimisation method

This section describes the proposed method for service deployment optimisation in the cloud environment with the goal of minimising service operational cost. The development of the optimisation method was guided by the following requirements:

- R1: The method should enable an analysis of interactions between application and infrastructure parameters which affect the application service cost and QoS.
- R2: The method should be applicable in the early phases of the application development with limited possibilities of acquiring large datasets for the analysis.
- R3: The method should be applicable to a wide range of cloud application services, not only to a specific application type.
- R4: The method should determine the deployment option for a cloud application service that will minimise the service operational cost while ensuring appropriate service quality, based on the conditions defined in the service level agreement.

Optimisation method overview

The optimisation method was developed based on a model of the cloud application service and its deployment environment. A model-based approach enables the analysis of model parameters interactions and their effect on service cost and QoS (R1). The chosen modelling techniques enable the applicability of the optimisation

method in the early phases of the application development using smaller data sets (R2). Also, using the model-based approach meant the possibility of applying the method to the cloud application services before they reach the production phase. The models created in the described method are based on the cloud application context metamodel, described later in this paper (Cloud application context metamodel section). The metamodel parameters were selected to be applied on any application service deployable in a cloud environment, enabling the generalisation of the proposed optimisation approach required by R3. The details of the metamodel development and the chosen parameters are described in [6] and Cloud application context metamodel section. The last requirement, R4, was tackled by the deployment optimisation algorithm, which provides optimised service placement decision based on the simulation of application service execution in given cloud deployment environments, using the model of cloud application and its context while taking into account the QoS requirements defined by the service level agreement.

The proposed method is illustrated in Fig. 1. In this paper, we use implemented cloud application services to validate the proposed method. However, in the case of very early phases of the application service lifecycle, such as planning, the optimisation method can also be applied for the services that are not yet implemented or prototyped. This can be a useful approach in estimating the service operation budget during project planning. When the application service is not yet developed, the estimation of the model-optimised deployment configuration can be obtained by using similar service data. Cloud application owners usually develop several applications of the same type for different customers, or they have an application that is being migrated to the cloud environment after previously being deployed on an on-premises infrastructure. In such cases, the method can be used based on the data of similar already implemented application services. This possibility is noted in the method diagram (Fig. 1) as the decision is made if the optimisation approach is applied to an already developed service using measured data or previously recorded data of a similar service.

The data collected for the optimisation method are specified by the parameters of the cloud application context metamodel (further described in Cloud application context metamodel section) that defines the cloud application service and its deployment environment. The obtained data are used to create the models of the service cost and QoS (QoS and cost models creation and parameter influence ranking section), and the cloud application context model (Implementation of the cloud application context model section). Both the QoS and cost models and the cloud application context model can be developed in parallel. The results of both method parts serve as an input to the deployment optimisation algorithm. The QoS and cost models of good prediction accuracy serve as an input for the analysis of the parameter influence on the output of the predictive models, i.e., service cost and the observed QoS metric. The parameter influence ranking provides a list of parameters ranked by their influence on the service cost and QoS. This list is used as the input for the deployment optimisation algorithm. The optimisation algorithm, described in Deployment optimisation algorithm section, uses the ranking to choose a service placement optimisation strategy in terms of a cloud infrastructure provider and an instance size. Another input to the optimisation algorithm is the cloud application context model. The model is used for the simulation of the application execution cost and SLA requirements fulfilment during the deployment optimisation and to validate the deployment setup proposed as the solution by the optimisation algorithm. Additionally, SLA requirements of the application service and the list of potential infrastructure providers are also provided as input to the optimisation algorithm by the optimisation method user. Additional explanations of the optimisation method parts are provided in the following sections.

Obtaining the data

Data is obtained either by performing measurements for an already implemented service to which the method is being applied or, in case there is no possibility to perform measurements, by using existing data of a similar service that can be used for an estimation of the optimised deployment option for a service that is not yet implemented. The measurement process applied to the chosen application service use cases is provided in detail in [6], where it is described together with the data analysis, which will also be briefly presented in Use cases and analysis of application-related data section of this paper. The obtained data includes metrics and parameters based on the cloud application context metamodel, defined in the next section.

Cloud application context metamodel

From the perspective of a cloud application service provider, crucial parts of the management of a cloud application in its operative state are the cost of running the application service in the cloud and making sure that it delivers the appropriate level of service quality. There are several aspects of the cloud application service and its deployment environment that affect the QoS and cost of application execution in a cloud environment; we refer to those as the cloud application context [6]. This section introduces the cloud application context metamodel

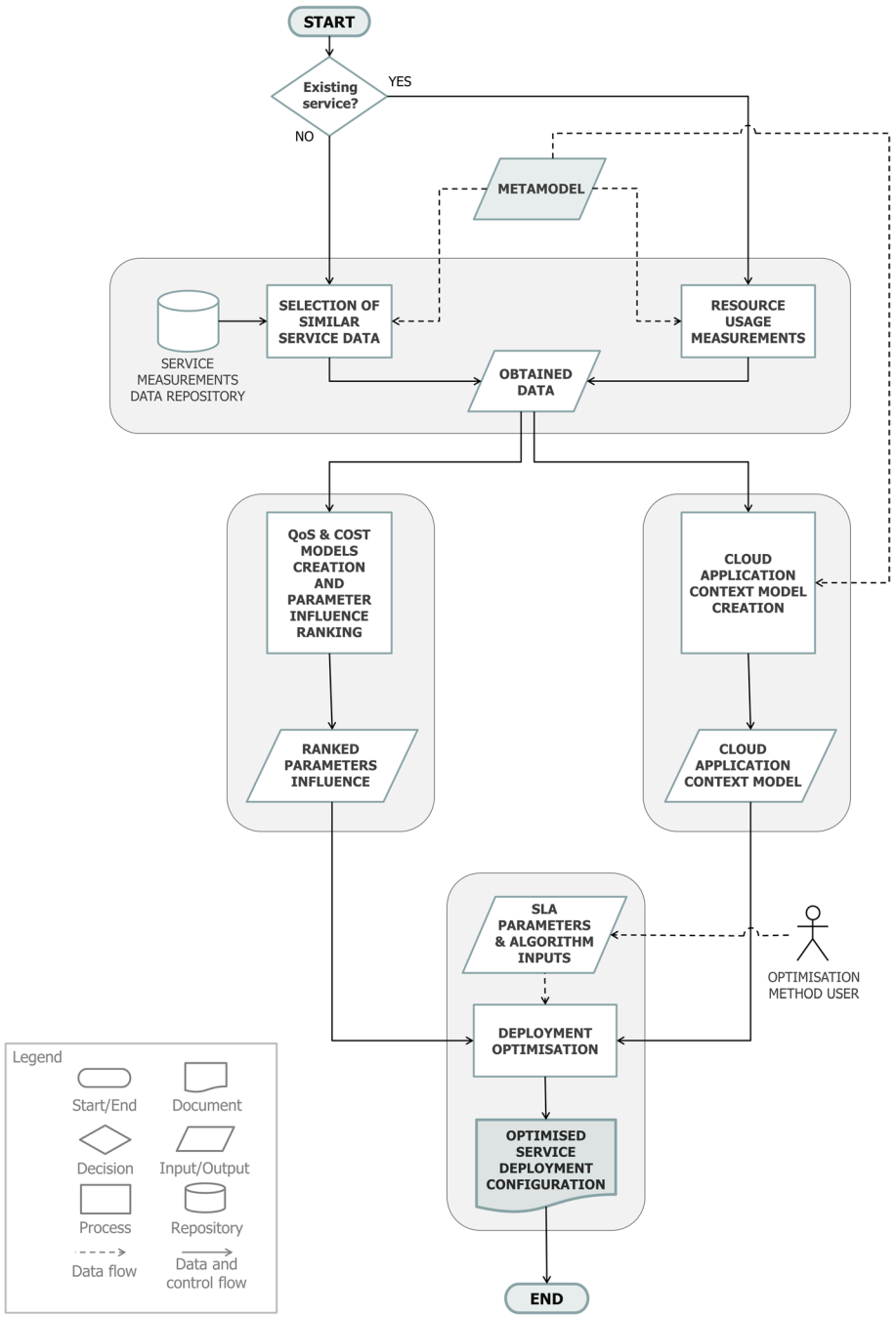

Fig. 1 Proposed optimisation method

and explains the method used for metamodel parameters selection.

To explain the relationship between the concepts of a model and a metamodel, we state the definition of a metamodel used in this paper - a metamodel is a model of models, it defines the structure and constraints for a group of models that share a common syntax and semantics [44]. In other words, a metamodel defines concepts and their relationships. Regarding the relationship between a model and a metamodel - a model is an instance of a metamodel if it is aligned with the structure that is defined by the metamodel. Following this relation, in the approach presented in this paper, the cloud application context metamodel is used for creating instances

of the cloud application service models. The proposed metamodel defines the parameters related to the cloud application and environment, as well as their structure and relationship.

Since one of the main goals of the optimisation method proposed in this paper is to observe the effect of various parameters that impact the execution of the cloud application on service cost and QoS, a metamodel was needed that would take into account several aspects of the cloud application service context. The challenge related to the creation of such metamodel is the selection of parameters that are relevant for determining the effects on cost and QoS, yet general enough to be applicable to the wide range of cloud applications. The appropriate selection of the metamodel parameters should result in the metamodel aligned with the defined requests for optimisation method development. The literature review demonstrated there is currently no model available which at the same time considers parameters related to cloud application resource usage, user-generated load, SLA requirements, properties of the application deployment environment, and cloud pricing models. To identify the relevant parameters for the description of the cloud application and its context, a survey was conducted that included:

- ontologies formally capturing the domain knowledge related to cloud infrastructure [45–50], application services [46, 47, 49], and cloud business models [51–53],
- cloud SLA monitoring ontology [54] and cloud monitoring metrics survey [55],
- properties defined by TOSCA (Topology and Orchestration Specification for Cloud Applications) [56], a formal language for the specification of cloud infrastructure and application services,
- parameters used for formal specification of cloud service requirements [41],
- parameters used in cloud infrastructure and application service simulators [57–61],
- metrics available through several monitoring tools for cloud resource usage [62, 63],
- survey of public cloud infrastructure providers and cloud pricing models [5], and
- books on the topic of cloud computing concepts and cloud pricing models [43, 64, 65].

The stated sources were chosen due to their comprehensive overview of the cloud applications and infrastructure (ontologies, surveys, specifications, books) or already demonstrated usability in a specific area of cloud application context (simulators, monitoring tools). The parameters included in the survey for defining cloud application context are related to the cloud application service itself and the cloud environment used for the service deployment, focusing on the parameters that affect the cloud service operational cost and quality of service. In the process of parameter selection, the following requirements were used:

- parameter should be applicable to any type of cloud application service,
- parameters related to infrastructure should be common for cloud infrastructure offers, i.e., applicable for all cloud infrastructure providers, and
- parameter values should be easily obtainable and understood by the application owner when considering infrastructure provider offers.

The selection process resulted in the parameters listed in Table 1. The table contains a short description of each parameter. Selected metamodel parameters (Table 1) and the relationship between them are illustrated in Fig. 2.

To provide a holistic view of the cloud application context and to enable analysis of interaction between the application and its deployment environment, we define two parameter domains. The *Application Domain* includes parameters specific to the application service itself, regardless of where it is deployed. The *Deployment Domain* consists of parameters related to the cloud infrastructure where the service will be deployed.

Parameters related to the application service, i.e., parameters from the Application Domain, include the application resource usage profile, which defines the amount of computing resources (CPU, RAM, storage and network usage) needed for the execution of the application service, or a specific service task that is being analysed, under a defined load. The load [66] is generated by the application user**s**, and it depends on the number of concurrent users that send requests to the cloud application, as well as the actions they are performing while using the application service, defined by the user type. User types are based on the actions that users perform and generate a certain load which affects the application's utilisation of resources. Further details on application user types, examples, and descriptions of user types analysed in validation use cases for the optimisation method presented in this paper can be found in [6]. Another application-related parameter is the Service Level Agreement (SLA), which defines the QoS requirements, commonly through a set of Service Level Objectives (SLOs) [66], that guarantee a level of service quality delivered to the application end users. An example of the SLO that could be a part of the application SLA is the response time of the

Table 1 Parameters of the cloud application context metamodel

Parameter Name	Description
Resource usage profile: CPU usage	CPU utilisation, number of used vCPU cores
Resource usage profile: RAM usage	RAM utilisation
Resource usage profile: Network usage	Amount of ingress and egress network traffic, network incoming and outgoing byte rate
Resource usage profile: Storage usage	Used storage capacity, read/write byte rates
Application users: Concurrent users	Number of concurrent users of the application service in a given moment, defined by the initial user number and the user number distribution
Application users: User Types	Various user types of the application service
Infrastructure provider	One or more infrastructure providers, owning the infrastructure resources that are used as a deployment environment of the application service
Resource properties: CPU properties	Properties related to CPU resources allocated to application service - vCPUs number, CPU frequency, etc.
Resource properties: RAM properties	Properties related to RAM resources allocated to application service - the amount of RAM
Resource properties: Network properties	Properties related to network resources allocated to application service - bandwidth, network ports throughput
Resource properties: Storage properties	Properties related to storage resources allocated to application service - HDD or SSD disk, local or network storage
Pricing model	Determines the way used resources are being charged, defined by infrastructure provider
Pricing model: Resource price	Price of a specific resource defined by infrastructure provider based on the resource properties and usage metric, or unit of time
Pricing model: Usage metric	Metric used by the infrastructure provider for billing (e.g., MB or GB can be used for measuring the consumption of network traffic).
Pricing model: Chargeable resources	Part of the pricing model defining which resource usage will be charged
Pricing model: Resource bundles	Defined groups of resources that are charged together, depending on the pricing model (e.g., RAM and CPU combination)
Data centre location	Geographical location of the data centre
Service Level Agreement	An agreement between the service provider and consumer defining service quality level through service level objectives
Service Level Objective	Part of the service level agreement

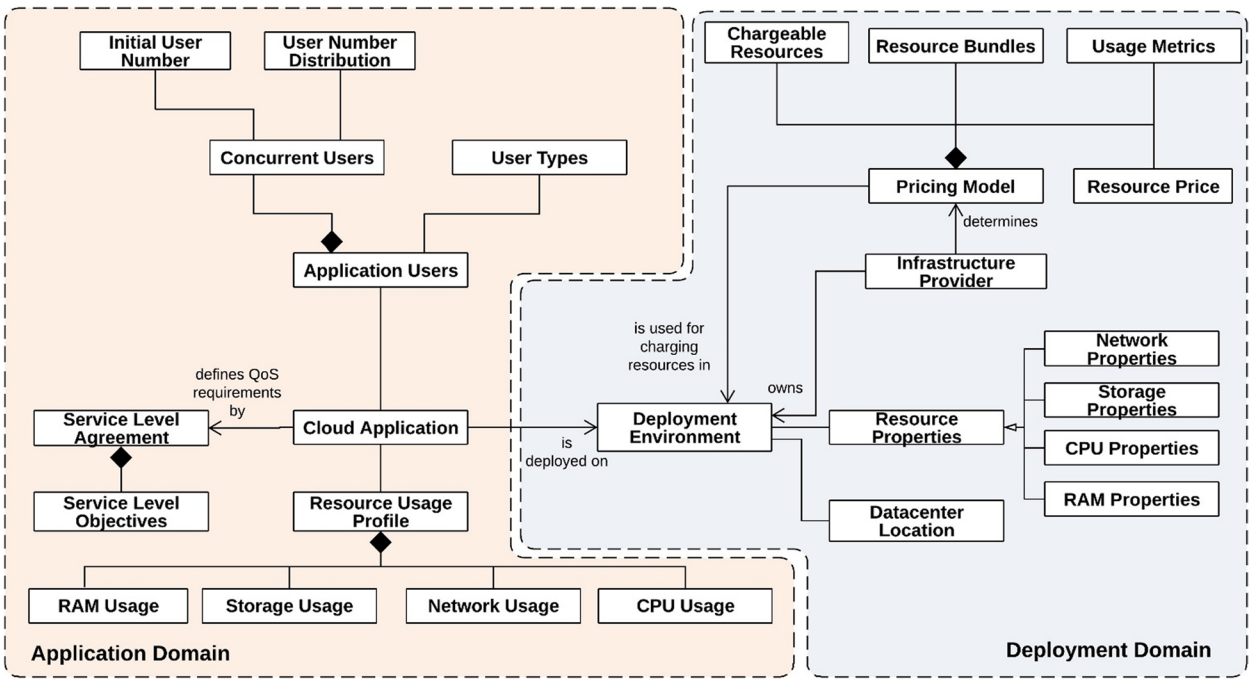

Fig. 2 Cloud application context metamodel

user request, which typically should not exceed a certain time in a defined percentage of total user requests made towards application service.

The remaining set of metamodel parameters is related to the infrastructure where the application will be hosted, i.e., its deployment environment (Fig. 2, Deployment Domain). The deployment environment chosen by the application provider is located in a specific geographic location and defines the amount and properties of each computing resource type. As an example, a cloud infrastructure provider might offer HDD and SSD storage, a different number of virtual CPU cores, etc. The amount of resources might be provided in predefined instances, or in the custom amounts requested by the application provider, who is the customer of the cloud infrastructure provider. Each deployment environment has one, or more (in the case of cloud federations), infrastructure providers. The infrastructure provider determines the pricing model according to which the used resources from the deployment environment will be charged to the users of the cloud infrastructure. The pricing model usually consists of prices of the resources, predefined resource bundles, the decision on which resources will be chargeable resources and how their consumption will be measured (i.e., specification of usage metrics). As an example, infrastructure providers might offer unlimited free ingress network traffic to attract application providers and cut their application onboarding cost, or offer limited network traffic for free on a monthly basis, etc. The offers of various infrastructure providers also differ in terms of charging granularity. The heterogeneity of pricing schemes and offers on the market makes the cloud infrastructure pricing models complex to evaluate in terms of their effect on the operational cost of the application, considering what might be a good option based on the application load and resource type utilisation.

In Fig. 3, we summarise how different parameters of the cloud application context interact and affect the application service cost and QoS. The execution cost of the cloud application service depends on the used pricing model, resource usage, and SLA requirements defined for the observed cloud application (since penalties for not fulfilling the SLOs specified by the SLA increase the service execution cost). The pricing model is defined through the choice of cloud deployment environment since the infrastructure provider that offers the resources in that environment specifies the pricing models. Chosen deployment environment also affects the realised QoS through the resource properties (e.g., number of vCPUs, amount of allocated RAM, etc.) defined by the deployment configuration. The application QoS is also affected by the application service resource usage, which is a result of the application resource usage profile and the load defined by the number of concurrent application users. Additionally, resource usage is directly affecting the resulting execution cost since the resources are charged on a per-use basis.

QoS and cost models creation and parameter influence ranking

To enable an analysis of the impact that the application service and cloud environment parameters have on the cost and QoS, models are created for the application service that the optimisation method is being applied to. The application cost and QoS prediction models created in this part of the method (Fig. 1) are used as input for the analysis of the parameter influence on the service cost and QoS.

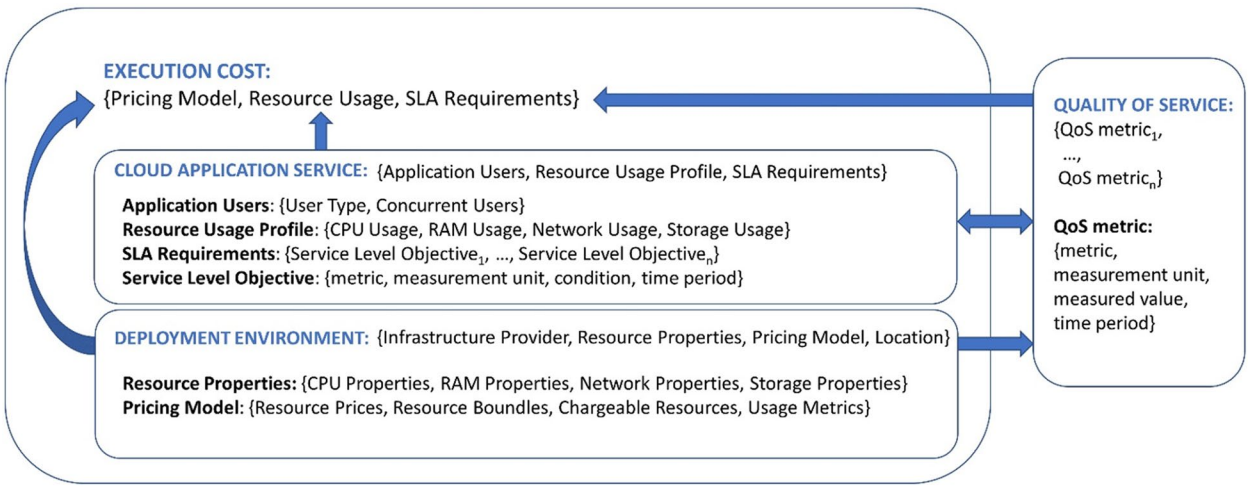

Fig. 3 Relations between cloud application context parameters, execution cost and QoS

To examine the possibility of predicting the cost and QoS of the cloud application service use cases, we implement models based on several regression techniques. We consider several statistical learning techniques used for the creation of cost and QoS models for the analysed cloud application services. The choice of algorithms was based on the size of the collected datasets, the predictive nature of the model, and the goal of enabling cloud application providers an easy approach for the evaluation of the best provider and service placement options that would require them to obtain only smaller experimental datasets. Since we want to explore the possibilities of predictive models, we performed analysis using regression techniques, including linear regression [67], Least Absolute Shrinkage and Selection Operator (Lasso) regression [68], Least Angle Regression (LARS) [69], Multivariate Adaptive Regression Splines (MARS) [70], and neural networks [67].

After obtaining QoS and cost models, we use the permutation importance method to inspect the importance of features used in the implemented prediction models. Permutation importance was introduced by Breiman in [71] where the method is described for random forest models, but the technique applies to regression as well. For the validation of the optimisation method proposed in this paper, we use permutation importance analysis on the regularised regression models [6]. Permutation importance is calculated after the model has been fitted. The permutations of single feature values in the validation data are then used to measure the effect of the permutations on the accuracy of predictions. The feature with the biggest impact on the predicted output data, i.e., the one that the model depends on extensively for predictions, will cause the greatest accuracy decline when permuted randomly. The output of the permutation importance method is the increase in prediction error when the single predictor values are permuted compared to the prediction error with all variables in their intact state. As the permutation importance measures how much each feature contributes to the model, during this part of the method, we examine the features with the highest positive permutation importance values that indicate the most significant effect on the variable predicted by the model.

The details on the implementation of the models using the stated modelling techniques and obtaining the permutation importance values can be found in [6], and the implemented models are presented in Implementation and validation of the cost and QoS models, parameter ranking section.

Implementation of the cloud application context model

In parallel to the cost and QoS predictive models, a model of the analysed application service is created (Fig. 1) based on the presented cloud application metamodel (Cloud application context metamodel section). The created model is used by the optimisation algorithm for the simulations of application execution under given load and deployment configurations to determine the deployment option that will minimise the service cost and fulfil the QoS requirements specified by the application SLOs.

CloudSim simulation toolkit [57] was used for the implementation of the models used by the optimisation algorithm. CloudSim is implemented in Java programming language with the intention of providing the possibility for simulator users to extend some of the tool's core features. The CloudSim layer, built on top of the core simulation engine, provides support for modelling and simulation of the main concepts of the cloud infrastructure - data centre, cloud orchestration, resource allocation, virtual machines and virtual machine management, as well as application service tasks (cloudlets). The User code layer is formed on top of the CloudSim layer. It uses the core concepts implemented in the CloudSim layer and exposes basic simulator entities related to the data centre host, virtual machine, and application service, which are used to specify simulation parameters and can be extended to simulate custom application configuration, workloads, scheduling policies, and provisioning techniques.

CloudSim, however, supports limited cost modelling, the prices of resources are defined by the host, and there is no support for custom pricing models. For the purpose of this research, the tool was extended to calculate the cost of using virtual machines for the defined pricing models based on the parameters marking the bundling of the resources, chargeable resources, and the choice of the pricing model itself, in order to be consistent with the cloud application context metamodel (Cloud application context metamodel section). To be able to use the simulator for the optimisation algorithm, support for the evaluation of the QoS metrics in terms of defined SLOs was introduced by calculating the given metric for the performed simulation. In the case of analysis performed for this paper, the average response time (cloudlet duration), and the percentage of response times longer than the given time threshold. Additionally, the simulator was extended to support the infrastructure provider parameter in accordance with the definition of infrastructure provider specified by the algorithm proposed in this paper as the input for the optimisation algorithm in Deployment optimisation algorithm section. The model parameters used for the simulation of application service

use cases and their validation can be found in Implementation and validation of the cloud application context models section.

Deployment optimisation algorithm

After the creation of application service and deployment environment models based on the cloud application metamodel, the models are used as input for the simulation-based optimisation algorithm. The overview of the main functional parts of the optimisation algorithm is depicted in Fig. 4. Besides the application service and cloud environment model parameters, the algorithm input is a list of cloud infrastructure providers (IP) specified as a tuple defined in (1). Each infrastructure provider is defined by the list of offered pricing models (PM), instance sizes (IS), geographical locations (L) of its data centres, and prices matrix containing the price of each instance size when charged by each pricing model at each data centre location. Parameters of an infrastructure provider are defined according to (2).

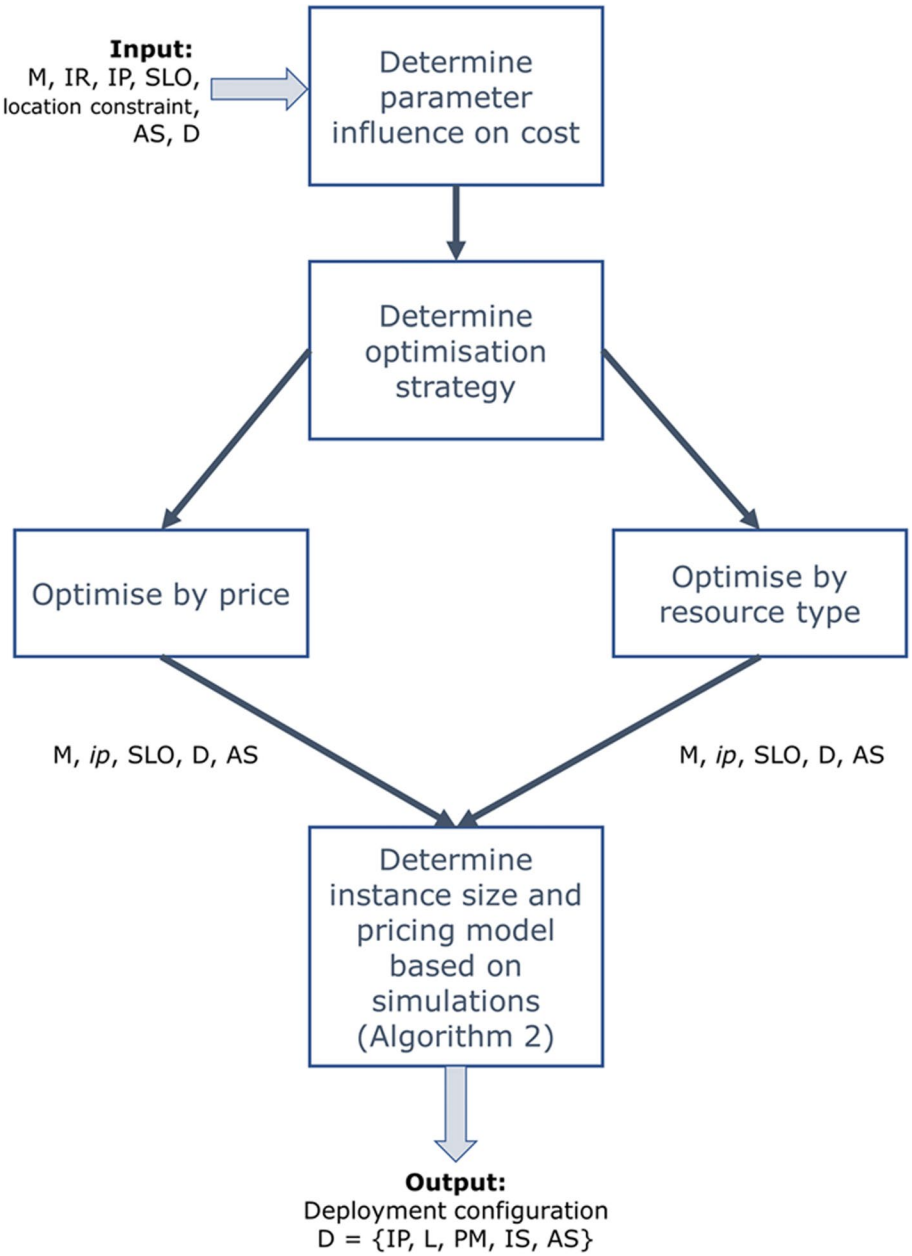

Fig. 4 Optimisation algorithm overview

$$IP = (IP_1, ..., IP_n) \tag{1}$$

$$IP_i = ((PM_1, ..., PM_j), (IS_1, \dots, IS_k), (L_1, \dots, L_m), (price\_matrix_{L_1}, ..., price\_matrix_{L_m})), i = 1, \dots, n \tag{2}$$

Each instance size offered by the infrastructure provider is defined according to (3), with the ordered tuple defining the amount of RAM in megabytes, the number of CPU cores, and storage in gigabytes. The instance sizes used in the infrastructure provider specification (2) are ordered based on the allocated resources, from smaller to larger resource amounts.

$$IS_i = (RAM, vCPUs, storage), i = 1, \dots k \tag{3}$$

Another input to the optimisation algorithm is the influence-based rank (IR) of the cloud application context parameters related to application service cost and QoS, defined according to (4) and ordered from the parameter with the highest importance rank to the parameters with the least importance.

$$IR = ((parameter_1, value_1), ..., (parameter_p, value_p)) \tag{4}$$

Additionally, to enable the assessment of the service quality of a deployment configuration, the algorithm input also includes the list of SLOs (5) that specify the QoS requirements for the application service. Each SLO (6) is defined by the QoS metric it relates to, the measurement unit of the metric value, a condition that should be met, and the time period that the SLO relates to (e.g., in the period of 24 hours, the average response time value should be less than 2 seconds).

$$SLO = (SLO_1, ..., SLO_s) \tag{5}$$

$$SLO_i = (QoS\ metric, measurement\ unit, condition, time\ period), i = 1, \dots, s \tag{6}$$

Application services deployed in the cloud environments can use the elastic properties of the cloud resources and can scale out automatically when more computing resources are needed to serve the current workload (based on a predefined resource utilisation threshold), as well as to scale in when the increased amount of resources is no longer needed. The scaling can be performed horizontally (e.g. by instantiating another virtual machine of the same size), or vertically (by increasing the resources allocated to the instance, i.e. upgrading the instance size). However, sometimes the service deployed in the cloud environment is not implemented in a way to fully utilise the autoscaling feature of the cloud environment, and in such cases, the autoscaling will not be performed. Hence, the preferred autoscaling option used in the algorithm can be defined via the autoscaling parameter AS_{input} (7). In case the AS_{input} is set to value *no*, the autoscaling will not be performed during the simulation, and in case it is set to *yes*, possible deployment configurations will include both options of horizontal and vertical scaling.

$$AS_{input} \in \{no, yes\} \tag{7}$$

As the last input parameter, the method user can specify the constraint related to the infrastructure provider data centre geographical location, which can in certain cases be important due to the nature of the application being deployed to the cloud environment, e.g. because of legal constraints related to the data. The location constraint is given via the numeric identifier of the data centre location, which corresponds to the numeric identifier of the data centre location used in the definition of the infrastructure provider (2). As an example, commonly available locations of data centres offered by cloud infrastructure service providers in the current IaaS market include the United States of America (USA) and locations within Europe, so the location L_1 could, for example, be assigned to the data centres in the USA, and L_2 could identify Europe as the data centre location. The location constraint marks the geographical location where the application owner wants to deploy the application service, and if there are no constraints in terms of geographic location, the location constraint parameter is set to value 0. Hence, the location constraint parameter is defined according to (8).

$$location\_constraint \in \{0, L_1, \dots, L_n\} \tag{8}$$

The deployment optimisation can be performed via five dimensions that are typically determined by the application owner during the deployment configuration selection:

- Infrastructure service provider, since it determines the offer of data centre locations, pricing models and instance sizes,
- Location of the data centre owned by the selected infrastructure provider,
- Pricing model,
- Instance size, and
- Decision on the autoscaling option used in the optimised deployment configuration

The output of the optimisation algorithm is an optimised deployment configuration which includes five optimisation dimensions mentioned above and is defined as:

$$D = \{IP, L, PM, IS, AS_{output}\} \tag{9}$$

$$AS_{output} \in \{no, horizontal, vertical\} \tag{10}$$

where IP is the selected infrastructure provider, L is the geographical location of the IP's data centre, PM is the chosen pricing model offered by the selected infrastructure provider, IS is the instance size that will be used for the application service deployment, and the AS is the decision on autoscaling, which is set to one of the options defined by (10). In case the application owner decided to deploy the application service without the possibility of autoscaling, the AS_{output} parameter of the optimised deployment configuration D will be set to 'no'. In case the application owner decided to optimise the service deployment including the possibility of autoscaling by providing 'yes' as the AS_{input} parameter value for the algorithm input (7), the AS_{output} value of the output deployment configuration will be set to 'horizontal' or 'vertical'(depending on the best autoscaling option determined by optimisation algorithm), or 'no' if there is no need for autoscaling in case of the given application service workload.

The goal of the optimisation algorithm is to determine the deployment configuration which minimises the cost of executing the application service in a cloud environment while enabling the fulfilment of SLOs specified for the application service. To determine such deployment configuration, the algorithm uses the list of parameters ranked by their influence on the application service cost and chooses a deployment optimisation strategy based on the parameter with the most significant impact. Parameter influence ranking is described in QoS and cost models creation and parameter influence ranking section, and its implementation on two use cases can be found in Implementation and validation of the cost and QoS models, paramter ranking section. Using this approach, the parameter that has the most influence on the service cost leads the strategy of infrastructure provider selection, as described later in this section, since it indicates how the most significant cost reduction can be achieved. Based on the optimisation strategy, two optimisation strategy groups are defined (Table 2).

The proposed algorithm (Fig. 4) is available as pseudocode (Algorithm 1) in its entirety in Appendix. Figure 4 contains an overview of the main functional parts of the algorithm. Algorithm inputs, defined in the previous section, include application service model parameters (M) needed to run the simulation (stated previously in Deployment optimisation algorithm section) using the CloudSim simulator (Implementation of the cloud application context model section), list of parameters ranked by their influence on service cost ($IRcost$) defined according to (4), list of infrastructure providers considered for deployment and defined according to (2), list of QoS requirements (SLO) (5), list of instance sizes considered for this analysis (IS) with each instance size defined according to (3), and the chosen autoscaling strategy (AS) (7). The proposed optimisation algorithm starts with identifying the ranked influence of cloud application context parameters on the service cost. The method *calculateIRCoefficients* calculates the influence rank coefficient (IR coefficient) for each parameter from the ranked parameter list which determines if there is one parameter among the received ranked parameters list that has a significantly stronger influence when compared to others, or if there are more parameters with similar influence. If there is a parameter in the list that has a permutation importance value higher than 90% of the first ranked parameter, the optimisation will be performed according to that parameter as well, and the optimised deployment configurations will be compared in the end to select the best one. This is done so that optimisation strategies based on parameters with similar influence values are considered. After the parameters on which the optimisation of deployment should be based are defined, the optimisation strategy is made based on the optimisation strategy group to which the most influential parameter affecting the application service cost belongs. For that purpose, two optimisation strategy groups are defined (Table 2). The *optimise by price* group includes the Provider parameter. In case the parameter most significantly affecting the cost of the application service is set to Provider, the cost is most affected by the prices offered by the infrastructure provider. In this case, the goal is to optimise the choice of the infrastructure provider based on the price rates provided for both the instances and

Table 2 Application service and cloud environment parameters according to the optimisation strategy groups

Optimisation strategy group	Parameter
Optimise by price	Provider
Optimise by resource type	Network: ingress traffic, egress traffic, network incoming byte rate, network outgoing byte rate CPU: CPU utilisation, number of vCPU cores; RAM: RAM utilisation, instance allocated RAM, storage: disk read rate, disk write rate, instance allocated storage, storage usage

network and to select the optimal instance size of the provider offering the most affordable prices. The second optimisation strategy group *optimise by resource type* includes parameters related to a specific resource type consumption. In cases when the parameters from this group related to the network have the most significant impact on the service execution cost, the application generates substantial quantities of network traffic, or predominantly uses network resources when compared to the other resource types. Since infrastructure providers charge the instances separately from the amount of generated network traffic (ingress and egress), the goal related to this parameter group is to optimise the choice of the infrastructure provider based on the prices offered for the network resources and perform instance rightsizing using the instance size offers of the selected provider. The parameters in this group include the utilisation and allocation parameters of other resource types as well. Parameters from this group that have the most significant impact on the service execution cost usually indicate which resource is utilised the most, or which resource is the one with the highest prices. Hence, the strategy related to this optimisation strategy group is to choose the infrastructure provider and pricing model that offers the greatest price-to-amount of resource ratio. The groups of resources in parameters belonging to this group are, besides network, CPU, RAM, and storage which correlate with the definition of resources allocated by the instance size used in this algorithm, defined by (3). After the optimisation parameter strategy has been defined based on the optimisation strategy group that the optimisation parameter belongs to, a method for the appropriate infrastructure provider optimisation is called (*optimiseByPrice* or *optimiseByResourceType*) which then initiates simulation-based instance rightsizing and pricing model selection through the *determineInstanceSizeAndPricingModel* method in Algorithm 2, which is also available entirely in Appendix of this paper.

The goal of the stated method is to identify the optimal instance size and pricing model, for a given infrastructure provider, selected based on the optimisation strategy, as described previously in this section. The inputs of the method include parameters of the application service and cloud environment model (M), infrastructure provider (*ip*) selected in the previous steps of the algorithm based on the parameter optimisation strategy group, list of QoS requirements defined via SLOs (SLO), deployment configuration with parts of the configuration set by the previous algorithm steps (D), and application provider's decision on autoscaling (AS). The algorithm uses method *simulate* (Appendix, Algorithm 2) to perform a simulation with the stated input parameters and returns the application execution cost based on the used instance size *is* and calculated according to pricing models offered by provider *ip*. The *simulate* method also returns the evaluation of the fulfilment of the SLOs specified for the application service (SLO) by comparing the results of the simulation with the QoS metric thresholds defined by the SLO parameter. The simulations are performed using the extended CloudSim tool, described in Implementation of the cloud application context model section.

Figure 5 displays one iteration of the method to provide a simplified overview. The method starts by selecting the smallest instance size from the set of instance sizes (IS) offered by the infrastructure provider *ip* to determine if this instance size would fulfil the QoS requirements when used without autoscaling. If this is the case, the deployment configuration parameters are set and returned since the QoS requirements are fulfilled by using the smallest instance size of the infrastructure provider optimised according to the prices determined by the optimisation strategy groups, and no autoscaling is needed, which makes this deployment configuration the one with the smallest application execution cost that fulfils the QoS requirements. If the QoS requirements are not met by the selected deployment configuration, the algorithm will check the application provider's autoscaling decision and try to determine the best autoscaling option by conducting a simulation using both horizontal and vertical autoscaling options and determine the one that will result with minimal execution cost and the fulfilled application service SLOs. In case neither of the autoscaling options enables the QoS requirements fulfilment, the larger instance size is selected for the evaluation. If the algorithm, by iterating through increasingly larger instance sizes offered by the given infrastructure provider, does not find the one which meets the QoS requirements, the next provider will be chosen from the list of infrastructure providers ordered by the conditions depending on the optimisation strategy group. This approach enables the selection of the infrastructure provider and deployment configuration, offering minimal prices for the resources that are needed for the application to achieve the specified QoS level for the given workload.

As can be noticed from the provided pseudocodes, the algorithm uses the ranked list of parameters that have the most impact on the application service cost and evaluates QoS fulfilment based on the simulations. The ranked list of parameters that have the most impact on the application service QoS is used to determine the resource that the autoscaling should be based on, i.e. determining if the utilisation of CPU, RAM, or network parameters should be used for the autoscaling. The example of using the parameter ranks based on their influence on application service QoS for the setting of the autoscaling parameters

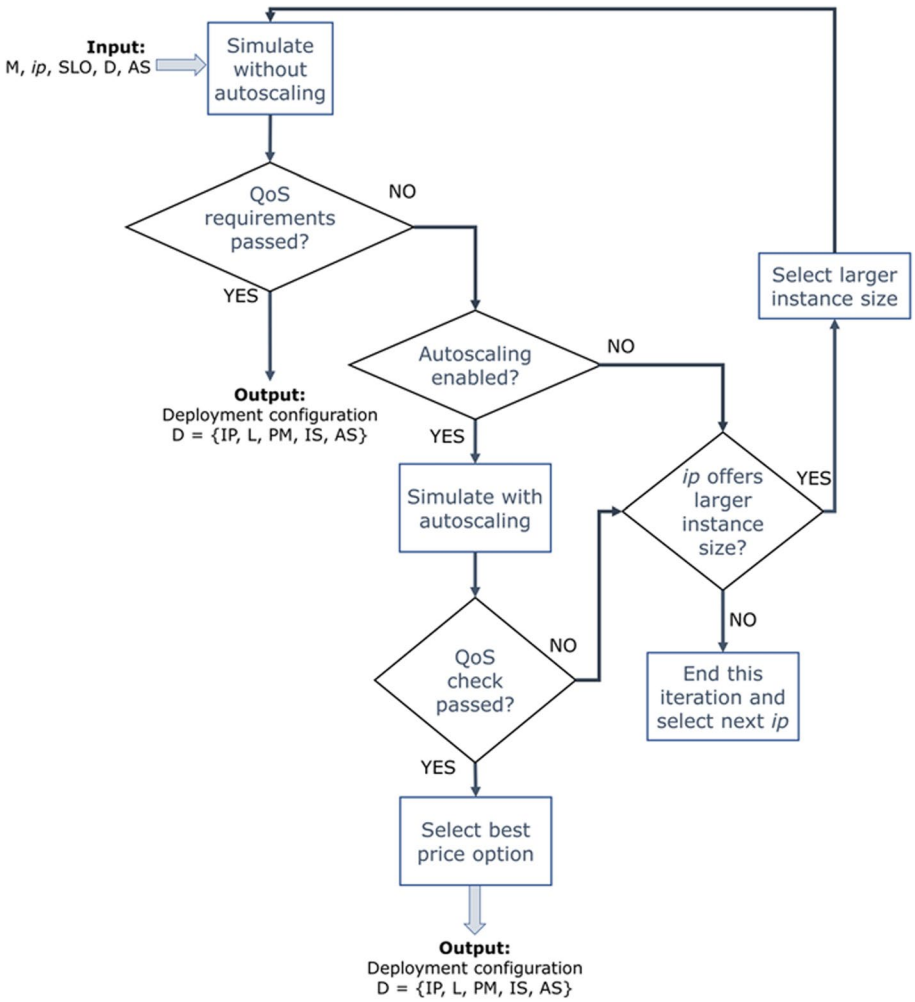

Fig. 5 Algorithm2 overview

is provided in Validation of the deployment optimisation algorithm section.

Use cases and analysis of application-related data

In this section, two application use cases selected for the method validation are presented, as well as a brief description of the measurements part of the method and the analysis of the gathered data. In this paper, we analyse two services of different application types, a medical record service and a video streaming service, to observe the differences in the resource utilisation during their execution, and to examine which parameters will affect the cost or QoS of each service to the most significant degree.

For the medical record system (MRS) service, we chose an open-source implementation of the MRS system [72]. The user type participating in this scenario is performing a query for retrieving a medical record of a patient. The implementation of our measurement scenario consists of a series of user requests for accessing the user interface of the MRS service, followed by providing user credentials and finally, sending the request for retrieving the medical record of a given patient from the medical record database. For our measurements, user requests were generated using a load generator deployed on machines outside of the testbed cloud environment. The response time was measured as the time needed for the user to perform the described sequence of requests and retrieve the patient's medical record, i.e., from the time of sending the first request to the time of receiving the patient data (Fig. 6).

As we want to examine the effect of service resource utilisation and infrastructure parameters on the service execution cost and QoS, we define the Service Level Objective (SLO) for the specification of the acceptable application QoS level for the MRS service. As an example of a QoS objective, we identify an SLO violation related to the percentage of user requests with a response time

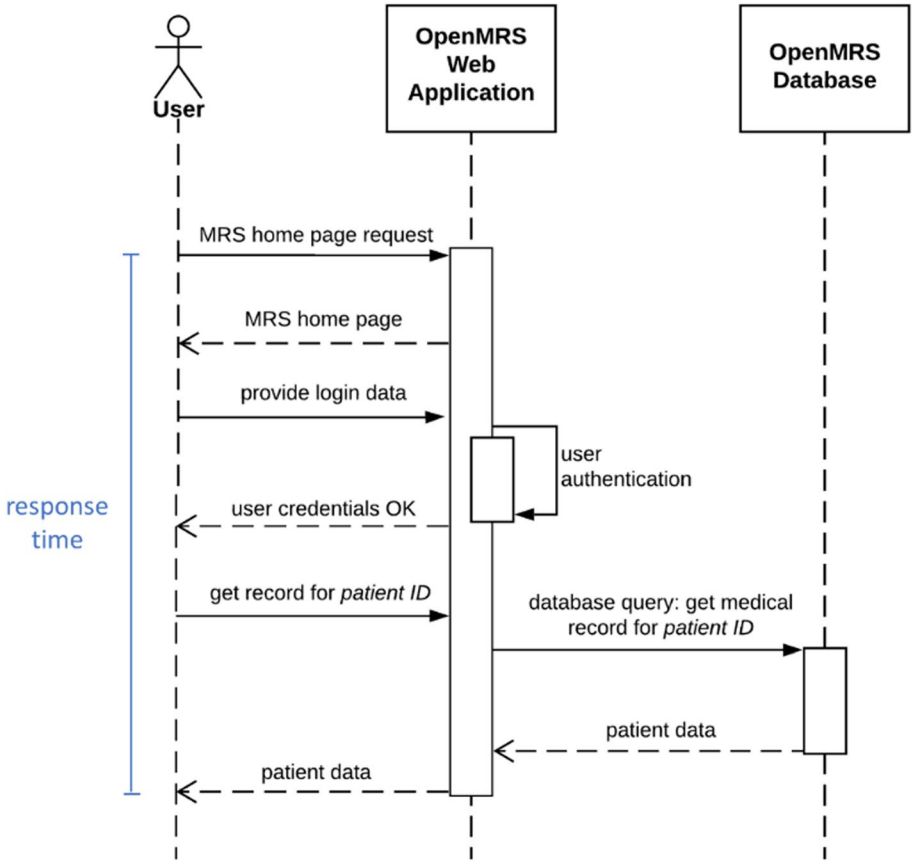

Fig. 6 UML sequence diagram of the MRS use case scenario

longer than 5 seconds. The number of such requests should not exceed 95% of total user requests.

As the second cloud application use case, we chose a video streaming service deployed in the cloud environment. For our measurements, we used Nimble Streamer video streaming server [73]. The user type participating in this scenario is based on the action sequence illustrated in Fig. 7, generating the requests for short video content. The response time in this context is the time needed for the transmission of the entire video file to the end user. Considering the deployment of the service in the cloud environment, the video server was deployed as a single component on an Ubuntu virtual machine. The client-side consists of the end users of a defined user type who generate the requests for the video content from outside of the cloud environment.

An example of a QoS metric we use for an SLO definition of a video streaming service is, similar to the MRS service, a response time of the user request, from the time of sending the request for the video file to the time when the last chunk of the video file is received by the user. For analysis purposes in the next sections, we observe if there are more than 95% of requests whose response time exceeds 10 seconds and define it as an SLO violation. To create cost and QoS models for the chosen cloud application use cases, we needed to collect the data that would allow us to observe the relation between service load, QoS (in this case, request response time), and average resources utilisation at the certain number of concurrent user requests and with a certain amount of resources allocated to a service. For that purpose, a cloud-based measurement environment was set up where we deployed both use case services and generated load using a load generator tool which also collected performance metrics, including request response times. As a measurement environment, we used a private cloud infrastructure deployed using OpenStack [74] platform. We provide detailed information on the measurement environment setup and used infrastructure used in [6].

Both use case services were deployed on the measurement environment using several instance sizes (Table 3) differing in resource quantities that were allocated to the services so we can observe the effect of instance sizing on the QoS and cost. User requests were generated using the JMeter load testing tool [75] that was installed on PCs outside the cloud environment.

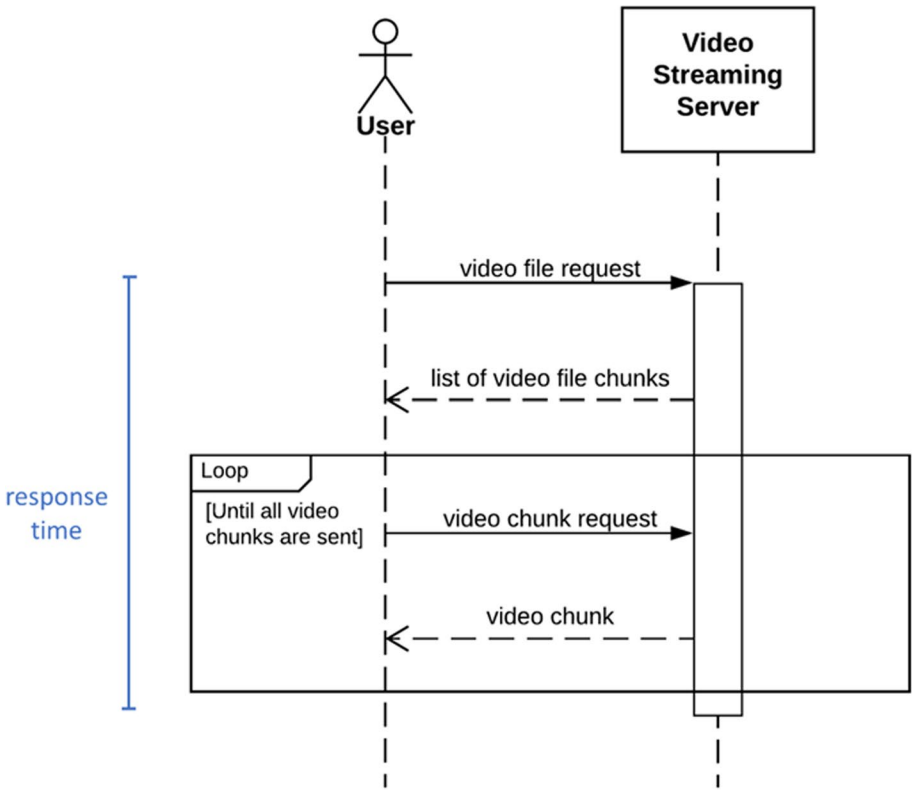

Fig. 7 UML sequence diagram of the video streaming use case scenario

Table 3 Resources quantity specified by instance types used for measurements

Instance type	RAM (MB)	CPU (virtual cores)	Storage (GB)
Small	2048	1	20
Medium	4096	2	40
Large	8192	4	80

We measured the consumption of resources defined by the parameters of the cloud application context metamodel (Table 4). Service resource usage data and parameters relevant for the quality of service, such as request response times, were collected each minute during the measurement periods of 15 minutes. Each 15-minute sample was gathered under constant load. The measurements were performed with the number of concurrent users increasing from one user up to 150, with the step of 10 concurrent users for each following sample. We recorded values of measured resource consumption for each sample. After performing measurements and collecting resource utilisation data, the cost of running both services for 24 hours was calculated [76] using publicly available price calculators of seven cloud infrastructure providers [77–83]. Table 4 brings mapping of measured parameters to the parameters of the cloud application context (initially presented in Table 1). More details on the measurement process and cost calculation can be found in [6].

After obtaining measurement data, we analysed the differences between the two selected use case applications in terms of resource utilisation. The detailed analysis of resource utilisation and the relationship between observed parameters can be found in [6]. In this paper, we briefly summarize the findings. When the two services are compared based on resource usage, it can be noticed that the MRS service utilises more CPU and RAM for the same number of concurrent users. The network ingress traffic amount is similar for both services since it in both cases consisted of simple HTTP requests. As expected, the video streaming service has significantly higher egress traffic compared to the MRS service, as its egress traffic is generated due to the streaming of a video file.

Deployment optimisation method validation

As we obtained the measurement data and performed the initial analysis, we proceeded with the validation of the optimisation method, i.e., the validation of its

Table 4 Mapping of measured data and proposed cloud application context metamodel parameters

Measured parameter name and unit	Related cloud application context parameter
Average CPU utilisation (%)	Resource usage profile: CPU usage
The number of virtual CPU cores	Resource usage profile: CPU usage
Network: average incoming byte rate (B/s)	Resource usage profile: network usage
Network: ingress traffic (GB)	Resource usage profile: network usage
Network: average outgoing byte rate (B/s)	Resource usage profile: network usage
Network: egress traffic (GB)	Resource usage profile: network usage
Average RAM used (MB)	Resource usage profile: RAM usage
Average disk read bytes rate (B/s)	Resource usage profile: storage usage
Average disk write bytes rate (B/s)	Resource usage profile: storage usage
Used storage (GB)	Resource usage profile: storage usage
Request response time (ms)	Service Level Objective
The number of parallel users	Application users: concurrent users
Instance vCPU cores	Resource properties: CPU Properties
Instance RAM (MB)	Resource properties: RAM Properties
Instance storage (GB)	Resource properties: Storage properties
$Provider_n$	Infrastructure Provider
Price (USD)	Pricing Model and Resource Price

components: prediction modes of the application service cost and QoS, service model based on the cloud application context metamodel, and model-based deployment optimisation algorithm.

Implementation and validation of the cost and QoS models, parameter ranking

To evaluate model accuracy, as well as to compare modes implemented using different machine learning methods, we used several metrics: Mean Absolute Error (MAE) [84], Mean Absolute Percentage Error (MAPE) [84], Root Mean Squared Error (RMSE) [85], and Coefficient of determination (R2) [67]. We implemented models for predicting the service cost and QoS for each analysed application service, using the cost and the request end-to-end response time as the predicted values. We applied a set of machine learning algorithms stated in QoS and cost models creation and parameters influence ranking section on both use case datasets to observe which algorithm will provide the best result and how much the results differ between the two use cases, as well as between dependent variables. We also analysed which features have the most impact on model output. As potential model features, we consider input variables presented in Table 4. The details of model implementation including the preparation of data, feature selection, regularisation parameters, machine learning environment setup, as well as evaluation of results are presented in [6].

In this paper, we will briefly report on the accuracy of the models and compare them based on the error estimation metrics reported in Table 5.

Table 5 Comparison of model accuracy. Best values for each model and metric are marked in bold print

Metric	Linear Regression	Lasso	LARS	MARS	ANN
response time, MRS					
MAE	243.39	476.62	**181.11**	181.47	221.05
MAPE (%)	9.33	19.82	5.79	**5.61**	6.24
RMSE	368.56	653.35	**263.47**	270.47	363.62
Adj. R2	0.9942	0.9817	0.9968	**0.9969**	0.9953
cost, MRS					
MAE	142.98	113.68	**99.00**	106.25	124.91
MAPE (%)	22.82	13.47	**6.33**	15.19	10.66
RMSE	270.99	257.48	212.86	**200.94**	202.71
Adj. R2	0.9963	0.9967	0.9977	**0.9980**	0.9979
response time, video streaming					
MAE	266.80	287.26	168.65	**104.87**	221.05
MAPE (%)	10.54	20.50	11.98	**5.09**	6.24
RMSE	495.25	401.45	226.01	**149.86**	363.62
Adj. R2	0.9588	0.9729	0.9914	**0.9962**	0.9867
cost, video streaming					
MAE	331.24	234.18	123.59	**72.77**	531.55
MAPE (%)	10.98	5.88	3.84	**2.02**	17.26
RMSE	463.54	333.45	209.41	**186.87**	719.94
Adj. R2	0.9982	0.9991	0.9996	**0.9997**	0.9750

It can be noticed that the best accuracy is in almost all cases achieved with the MARS models. The ANN cost models of both use cases have somewhat lower model accuracy, compared to the QoS models, which can be explained by the different prices and pricing models

offered by cloud infrastructure providers that affect service execution cost. Various pricing schemes could make it harder to produce patterns for cost prediction, especially when not using an extensive dataset. Contrary to the results presented in [35], the ANN models demonstrated less accurate results compared to the regularisation regression techniques. It is likely that the accuracy of the ANN models would improve with the larger sample size. However, the obtained results demonstrate the ability of the regularised regression techniques to produce accurate models even with a small sample size which would prove especially useful in the prototyping and early pre-production phases of the cloud service development.

In addition to model accuracy, we observe the importance of features used for predicting cost and response time for both use cases so we can gain knowledge about the factors that affect the service execution cost and QoS to the greatest extent. The permutation importance was calculated using the Python-based library Eli5 [86]. As a base for the permutation importance method, we chose models with the best accuracy results (Table 5). We report the results for MARS models since they demonstrated the best accuracy for both models of the two use cases. Weight values indicate the effect that each variable had on the prediction accuracy, i.e. higher positive values indicate the more significant impact of the feature on the predicted variable.

We first examine cost models for both services. The permutation importance ranks of the top four features for the model of video streaming service cost can be seen in Table 6. The feature with the highest importance for the video streaming cost model was the amount of egress network traffic, which can be explained by the heavy network load generated by transferring video files. Since the egress traffic is charged on a pay-per-use basis, its amount directly affects the overall service execution cost. The highest ranked feature in the case of the MRS cost model (Table 7) is also the amount of network egress traffic. However, the weight value of the network egress traffic is lower than the weight of the same feature in the video streaming cost model, indicating less impact on the execution cost amount due to the less generated egress traffic in the MRS service for the same load. The network egress feature is followed by variables marking a provider of the cloud infrastructure, similar to the video streaming service cost model with the service providers ordered differently. Such rank might indicate that the prices of the resources relevant for observed task execution offered by the provider with the highest weight value in the feature list resulted in the highest service execution cost. Although the ranks differ somewhat in the MRS and the video streaming service, Service Providers 1 and 4 in both cases have similar permutation importance weight values, and the Service Provider 3 weight value is lower for both services, demonstrating consistency regarding the effect that the prices and pricing models have on the overall service execution cost.

The rank of the features for both cost models can provide directions in terms of service placement optimisation with the goal of reducing the service cost. Based on the results for two observed services in our analysis, good options for cost reduction, in terms of service placement, would include cloud infrastructure providers that offer lower network egress prices, since this is the feature that had the most impact on the execution cost. Another option might be to optimise the content that the service sends over the network to reduce the overall amount of the generated egress traffic, if possible.

In addition to cost, we observe what features affect the response time the most. The permutation importance results for the QoS model of video streaming (Table 8) and MRS service (Table 9) demonstrate the same feature that has the most impact on the prediction of the response time - the number of parallel users. This is an expected result since the number of parallel user requests

Table 6 Permutation importance feature rank for the cost model of video streaming service

Weight	Feature
1.6223 ± 0.4371	NW Egress (GB)
0.9892 ± 0.4441	Service Provider 1
0.8785 ± 0.4269	Service Provider 4
0.6277 ± 0.2374	Service Provider 3

Table 7 Permutation importance feature rank for the cost model of MRS service

Weight	Feature
1.0311 ± 0.5252	NW Egress (GB)
0.9498 ± 0.5958	Service Provider 4
0.8550 ± 0.5181	Service Provider 1
0.6938 ± 0.3571	Service Provider 3

Table 8 Permutation importance feature rank for QoS model of video streaming service

Weight	Feature
2.2935 ± 0.7169	Users
0.0299 ± 0.0147	NW Egress (GB)
0.0001 ± 0.0001	Avg RAM used (MB)
0 ± 0.0000	Disk read bytes

Table 9 Permutation importance feature rank for QoS model of MRS service

Weight	Feature
2.0382 ± 0.3191	Users
0.8644 ± 0.1578	Avg RAM used (MB)
0.4996 ± 0.1399	Disk read bytes
0.4075 ± 0.1038	NW Egress (GB)

affects resource utilisation and the ability to process requests efficiently. Since we are interested in identifying a feature having a significant impact on response time in terms of service placement, we additionally observe features that ranked highest after the number of concurrent users. In the case of the video streaming service, the feature ranked second is network egress traffic, indicating that the streaming chunks of video data had an impact on the response times due to the limited bandwidth. Other features seem to have little or no effect on the response time of the video streaming service. After the number of concurrent user requests, the MRS service response times depended mainly on the used RAM, meaning that the processing of the user requests had a more significant impact on the MRS service response times than the bandwidth consumption, as compared with the video streaming service. This observation can be used by the application provider to identify the resources crucial for maintaining adequate service QoS level.

Implementation and validation of the cloud application context models

To validate the created models, we generated validation samples based on three workloads typical for applications hosted as a service in cloud environments. The three selected workload scenarios include rapid growth, dynamic load, and burst. The dynamic load validation scenario starts with a gradual increase in the number of parallel user requests, from one to 50. Subsequently, the number of concurrent users is decreased back to one. The rapid growth scenario covers the case of an increase of the simultaneous users from one to 150 parallel users within the duration of the sample. The burst validation scenario contains bursts in the number of concurrent user requests. The scenario starts with 30 parallel user requests, followed by several steep increases and decreases in the number of parallel requests with 10 as the lowest, and 120 as the highest number of concurrent requests in the sample.

The same measurement environment was used to gather the validation samples as the one used for service resource usage measurements, described in Use cases and analysis of application-related data section. The load for the validation scenarios was generated by the jMeter tool [75], and the validation samples were collected for the two application service use cases deployed on small, medium and large instance sizes (Table 3). To implement the models of the use case application services based on the cloud application context metamodel, the extended CloudSim simulator was used, described in Implementation of the cloud application context model section. The simulation setup includes specifying the parameters for cloud application service, defined by the data centre broker and cloudlets, and the cloud deployment environment, determined by hosts and virtual machines.

At the beginning of the simulation, a data centre object is instantiated, and the list of host objects is assigned to it. In CloudSim terminology, hosts represent physical servers used for application deployment, so we defined the list that contains two hosts representing the two servers we used in the measurement environment for the compute nodes. Each host has several parameters defining its resource capacities - the number of CPU cores and their frequency (MIPS), RAM (MB), storage (MB) and bandwidth (MBps). The frequencies of the servers' CPUs are mapped to MIPS ratings used by the CloudSim simulator according to [87]. Each server is, according to the used measurement environment modelled to have 10 GB/s network bandwidth. The model of the HP ProLiant BL460c Gen 8 server, used as one of the compute nodes in the measurement environment, was assigned 256 GB RAM and 1.2 TB storage capacity. The model of the other server used as a compute node in the testbed cloud environment, HP ProLiant DL 380 G6, was assigned 64 GB RAM and 1.9 TB storage capacity. Considering the virtual machines, we use models of the three instance sizes used in the analysis so far (Table 3). The small instance size was modelled by the following parameters: RAM size 2048 MB, 1 CPU core, storage size 20000 MB, and 5000 MB network bandwidth. For CPU MIPS values, 2000 MIPS computational capacity was set for all of the simulations of the MRS application service, since it was deployed on the HP ProLiant BL460c Gen 8 server, and 2930 MIPS was set for the simulations of the video streaming service, which was deployed on the HP ProLiant DL 380 G6 server.

The modelled load scenarios for both applications include dynamic load, rapid growth, and burst scenarios, introduced previously. The interarrival times between the generated requests were extracted from the collected workload scenario traces and were used by the data centre broker for instantiating cloudlets. As the broker initiates the creation of the cloudlet, it sets the cloudlet parameters which were determined based on the number of parallel cloudlets already executing on the virtual

machine and the utilisation of the virtual machine computing resources. For that purpose, a parallel CloudSim cloudlet scheduler developed by Antonescu et al. [88] was used together with resource utilisation information for the analysed load range. Additionally, the cloudlet input and output size parameters were defined according to the measured values of the user request size and the size of the response. The size of the cloudlet input was 12388 bytes, and the output was 97411 bytes for the MRS service. The video streaming service cloudlet had an input size of 8770 bytes, and the size of its output was 622918 bytes.

To evaluate the accuracy of models created for each application service use case, we calculate the mean absolute percentage error (MAPE) [84].

We also observe results via ECDF plots for the response time variable. In Figs. 8, 9, 10 and 11, we present ECDF plots for both use cases and several different instance sizes and validation scenarios. As can be observed from the ECDF plots, simulated values follow

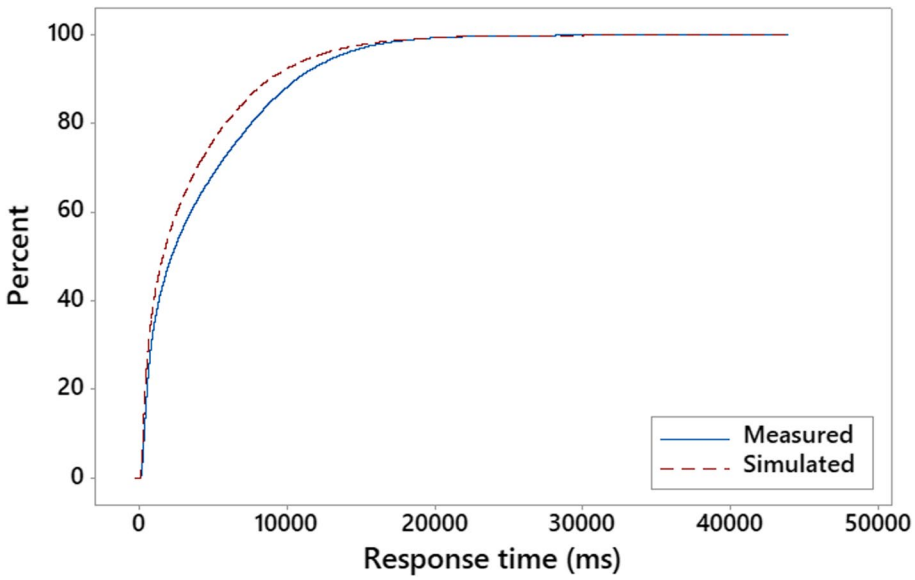

Fig. 8 ECDF plot of measured and simulated values of response time for burst validation scenario, MRS service, large instance

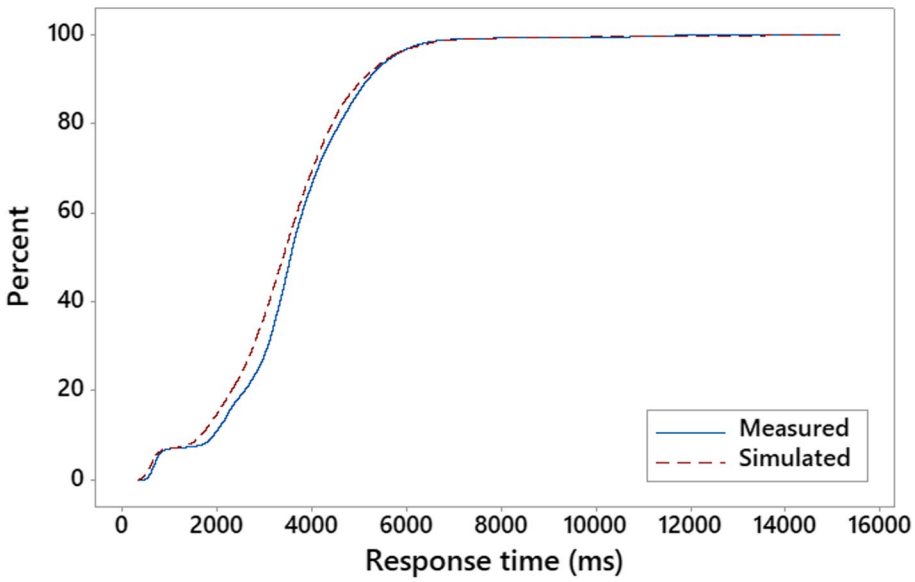

Fig. 9 ECDF plot of measured and simulated values of response time for rapid growth validation scenario, MRS service, medium instance

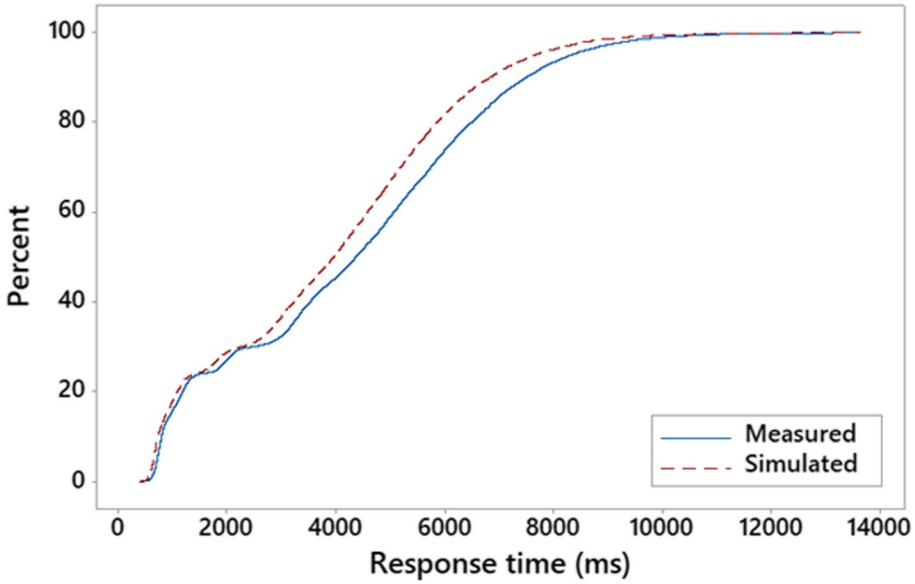

Fig. 10 ECDF plot of measured and simulated values of response time for rapid growth validation scenario, video service, large instance

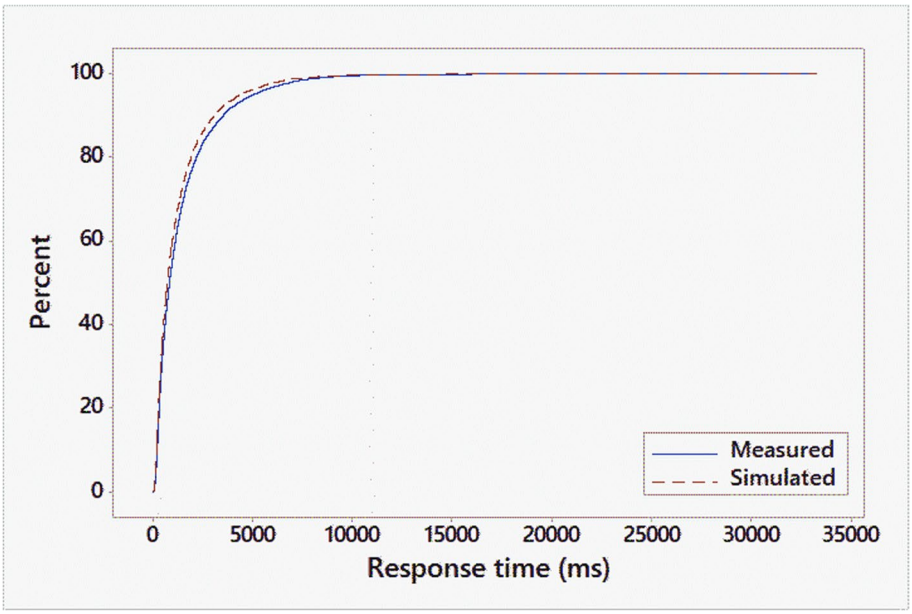

Fig. 11 ECDF plot of measured and simulated values of response time for dynamic load validation scenario, video service, medium instance

the trend of the measured sample response time values. However, occasionally the model overestimates the values of response times. The observed mean absolute percentage errors of the models for both services and all three load scenarios were in the range from 9,67% to 18.35%. By analysing the ECDF plots of the simulated and measured response time values, it can be observed that the models follow the distribution of the measured response times to a large extent, with the tendency of overestimating the response time values in the mid-range of the values. It can be noticed that the video service data generally has a wider spread of data when compared to the MRS service, which also resulted in somewhat larger mean average percentage errors. However, all of the presented models demonstrated good accuracy and they were used as input for the next step

of the optimisation method, a simulation-based deployment optimisation algorithm.

Validation of the deployment optimisation algorithm

In this section, a validation of the optimisation algorithm is described, as well as the scenarios used for the validation. For validation purposes, the optimisation algorithm was implemented in Java programming language within the extended CloudSim simulator to be able to perform a simulation of application service execution using various deployment configurations. As described previously in Deployment optimisation algorithm section, the optimisation algorithm requires several inputs: application service and cloud environment model parameters needed for the service simulation, metamodel parameters ranked by their influence on the cost and QoS of the application service, a list of cloud infrastructure providers together with the prices and pricing models they offer, QoS requirements defined in the SLA agreement and specified via SLOs for the observed application service, the decision on autoscaling strategy, and constraint related geographical location of the data centre used for application deployment.

To validate the optimisation algorithm, the algorithm inputs were defined according to the stated input list. We use models of the use case application services whose implementation and validation are described in Implementation and validation of the cloud application context models section, and the three workload scenarios described previously - dynamic load, rapid growth, and burst. The metamodel parameters ranked by their influence on the cost and QoS of the use case application services, obtained by the permutation importance method and provided in Implementation and validation of the cost and QoS models, parameter ranking section (Tables 6, 7, 8 and 9), are used as another input to the algorithm. The influence of the parameters on the service cost and QoS is defined via influence rank (IR) tuples for MRS (11, 12) and video streaming service (13, 14) for impact on cost and QoS, respectively. We use only the first four ranked parameters in the analysed IR tuples since the values of the permutation importance significantly decrease after the first four parameters in both services' QoS and cost influence ranks.

$$IR_{MRS\_cost} = ((NWegress, 1.03), (IP_4, 0.95), (IP_1, 0.85), (IP_3, 0.69)) \tag{11}$$

$$IR_{MRS\_QoS} = ((Users, 2.04), (AvgRAM, 0.86), (Disk\ readbytes, 0.5), (NWegress, 0.41)) \tag{12}$$

$$IR_{video\_cost} = ((NWegress, 1.62), (IP_1, 0.98), (IP_4, 0.88), (IP_3, 0.63)) \tag{13}$$

$$IR_{video\_QoS} = ((Users, 2.29), (NWegress, 0.03), (AvgRAM, 0.0), (Diskreadbytes, 0.0)) \tag{14}$$

As infrastructure providers considered in this analysis, we use the same seven infrastructure providers [77–83] that were used for the data analysis and the creation of the predictive models described in Implementation and validation of the cost and QoS models, parameter ranking section. The choice of stated providers was favourable for validation purposes since there is public information about their resource prices and pricing models. The same numerical identifiers used in the previous sections have been assigned to the chosen infrastructure providers, and the list of infrastructure providers used in this analysis is defined as:

$$IP = (IP_1, IP_2, IP_3, IP_4, IP_5, IP_6, IP_7) \tag{15}$$

Each of the infrastructure providers in the infrastructure providers (IP) tuple is defined according to the (2).

Due to the clarity of the presentation, we do not present parameters for each infrastructure provider in the tuple notation defined by (2). Instead, the specification of pricing models, prices, and data centre locations for each infrastructure provider used for the validation of the optimisation algorithm can be found in Table 10 and Table 11, and in the description that follows. Each of the seven providers chosen for this analysis has data centres available in two locations considered in this analysis - United States of America (USA), marked as L_1 in the IP location tuples (2), and Europe (EU), marked as L_2. In the validation scenarios, we use three pricing models. The first pricing model, predefined instance (marked as PM1), specifies the price of the instance size as a predefined bundle of CPU, RAM and storage resources. The second pricing model, unbundled instance (PM2), offers a separate price for each resource in an instance. This enables the cloud infrastructure user to specify a custom amount of resources in an instance since the instance size is not predefined by the infrastructure provider. However, the customer's ability to

Table 10 Pricing models offered by the infrastructure providers used in the analysis and locations of their data centres

Provider	Location	Pricing Models
IP_1	USA, EU	PM1, PM2, PM3
IP_2	USA, EU	PM1, PM2
IP_3	USA, EU	PM1, PM2
IP_4	USA, EU	PM1, PM2
IP_5	USA, EU	PM1, PM2, PM3
IP_6	USA, EU	PM1, PM2
IP_7	USA, EU	PM1, PM2

Table 11 Prices of the instance sizes and network transport (per hour, in USD) based on the choice of infrastructure provider, data centre location, and pricing model

	Pricing Model	IP_1	IP_2	IP_3		IP_4		IP_5		IP_6	IP_7	
		USA/EU	USA/EU	USA	EU	USA	EU	USA	EU	USA/EU	USA	EU
Instance size: small	PM1	0.0125	0.015	0.048	0.0555	0.09	0.11	0.039	0.0432	0.03	0.04	0.047
	PM2	0.014	0.0175	0.05	0.065	0.11	0.135	0.045	0.05	0.042	0.053	0.057
	PM3	0.0125	-	-	-	-	-	0.039	0.0432	-	-	-
Instance size: medium	PM1	0.025	0.03	0.096	0.111	0.14	0.17	0.078	0.0864	0.0638	0.083	0.1
	PM2	0.028	0.035	0.1	0.13	0.16	0.195	0.09	0.1	0.075	0.102	0.12
	PM3	0.025	-	-	-	-	-	0.078	0.0864	-	-	-
Instance size: large	PM1	0.1	0.06	0.192	0.222	0.23	0.29	0.157	0.173	0.1275	0.175	0.21
	PM2	0.16	0.07	0.2	0.26	0.25	0.35	0.185	0.2	0.15	0.23	0.25
	PM3	0.1	-	-	-	-	-	0.157	0.173	-	-	-
NW ingress (price/GB)	PM1	0.06	0.01	0	0	0	0	0.05	0.05	0	0	0
	PM2	0.06	0.01	0	0	0.12	0.12	0.05	0.05	0	0	0
	PM3	0.03	-	-	-	-	-	0.035	0.035	-	-	-
NW egress (price/GB)	PM1	0.06	0.01	0.09	0.09	0	0	0.05	0.05	0.0085	0.087	0.1
	PM2	0.06	0.01	0.09	0.09	0.12	0.12	0.05	0.05	0.0085	0.087	0.1
	PM3	0.07	-	-	-	-	-	0.065	0.065	-	-	-

customise the instance size is less convenient for the infrastructure provider, so the instances charged according to these pricing models usually have higher prices when compared to the predefined instance of the same size. In addition to instance sizes, the cloud consumer pays for the network transfer to the instance (NW ingress), as well as the outbound network transfer (NW egress). Certain providers offer the ingress traffic for free, and others might occasionally provide a lower price of ingress traffic to attract new customers, which is in our analysis marked as the third pricing model, discounted ingress (PM3), offered by two providers from the considered infrastructure providers set. The prices (Table 11) were taken from the publicly available information on the infrastructure providers' offer, available online [77–83]. As instance sizes, we use small, medium, and large instances whose resource quantities are defined in Table 3 and are defined in (16), (17), and (18), according to (3).

$$IS_{small} = (2048, 1, 20) \tag{16}$$

$$IS_{medium} = (4096, 2, 40) \tag{17}$$

$$IS_{large} = (8192, 4, 80) \tag{18}$$

Additionally, for the purpose of the algorithm validation, we define two SLOs for the specification of the QoS requirements. We base the choice of the used SLOs on the data that was gathered during the measurement process and define two SLOs based on the response time QoS metric. The SLOs used in this analysis are defined by (19) according to (5) and (6) where SLO_1 refers to the maximum percentage of user requests in the last 24 hours whose response times can exceed the threshold specified by the SLO for video streaming service (20) and MRS service (22). SLO_2 refers to the average response time value that should be less than 5 seconds on a daily basis for video streaming service (21) or 3 seconds for MRS service (23).

$$SLO = (SLO_1, SLO_2) \tag{19}$$

$$SLO_{1\_video} = (\text{response time, second, less than 95\% over 10 s, 24 hours}) \tag{20}$$

$$SLO_{2\_video} = (\text{response time average, second, less than 5 s, 24 hours}) \tag{21}$$

$$SLO_{1\_MRS} = (\text{response time, second, less than 95\% over 5s, 24 hours}) \tag{22}$$

$$SLO_{2\_MRS} = \text{(response time average, second, less than 3 s, 24 hours)} \quad (23)$$

For the simulation of autoscaling, the autoscaling parameter was determined for each service based on the ranked parameters and their influence on the application QoS. For both services, the most influential parameter affecting QoS was the number of parallel users, which is a reasonable result since this parameter directly affects the application load. However, since we want to determine the effect of resource utilisation on the application QoS, we consider the next parameter in the ranked list. For the MRS service, this was average RAM utilisation, and for the video streaming service, the second most influential parameter was network bandwidth. Based on the QoS impact parameter rank and the data measurements [6], we set the autoscaling conditions for the MRS service to either one of the following: allocated more than 2250 MB RAM, or average CPU utilisation higher than 97%. For the video streaming service, the autoscaling parameter was set to network incoming byte rate higher than 1.6 Mbps. Based on the defined inputs for the optimisation algorithm validation, we identify 256 possible deployment configurations for each service.

The performed simulations calculated the cost of the dynamic load, rapid growth, and burst workload scenario for the 24 hours, i.e. the daily execution cost for the MRS and video streaming services when executing the defined validation workloads. When all 256 possible deployment configurations for the algorithm input described in this section are considered, the mean price for the MRS service is 22.80 USD (st. dev. 13.35) for the dynamic load scenario, 24.50 USD (st. dev. 14.85) for the rapid growth scenario, and 25.98 USD (st. dev. 16.08) for the burst scenario. As for the video streaming service, the mean price for the dynamic load scenario is 62.43 USD (st. dev. 36.39) for the rapid growth scenario the mean price amounts to 65.38 USD (st. dev. 37.93), and for the burst scenario, the mean execution cost is 65.58 (st. dev. 38.01). The data demonstrated substantial spread (Fig. 12) due to the significant differences in the resource prices.

As the first validation scenario, we consider the dynamic load workload scenario for the MRS service. The minimal execution cost was determined for the following deployment configuration: D=IP$_6$, USA, PM1, no, small, in the amount of 2.16 USD. This is also the overall minimal cost for all of the 256 deployment configurations. The maximal price for this scenario was 46.76 for the configuration D=IP4, EU, PM2, horizontal, large.

The minimal execution cost for the MRS service and the rapid growth and burst scenario was determined for the following deployment configuration: D=IP$_6$, USA, PM1, horizontal, small, in the amount of 2.84 USD for the rapid growth scenario, and 3.01 USD for the burst scenario. The maximum price for this case was 52.90 USD for the rapid growth scenario, and

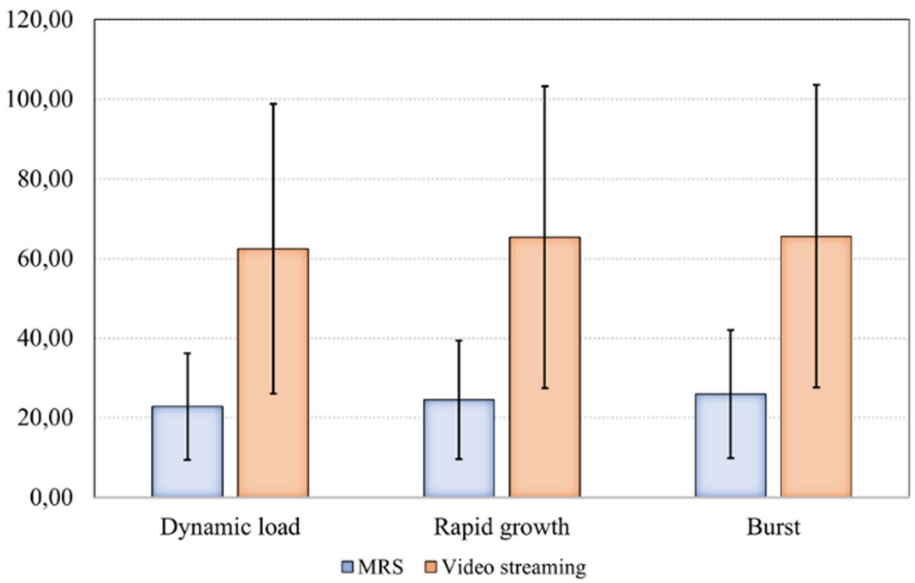

Fig. 12 The mean price (USD) and standard deviation for the application services and workload scenarios

55.54 USD for the burst scenario, for the same configuration D= IP_4, EU, PM2, horizontal, medium, which can be explained by the higher prices of the IP_4 provider and the somewhat increased resource prices of the pricing model PM2 due to the unbundled instance resources.

Similar to the MRS service, the minimal execution cost of the dynamic load scenario for the video service was determined for the following deployment configuration: D=IP_6, USA, PM1, no, small, in the amount of 8.86 USD, the overall minimal cost for all of the used deployment configurations. The maximal price for this scenario was 125.21 USD for the configuration D= IP_4, EU, PM2, horizontal, large.

The minimal execution cost for the rapid growth and burst scenarios for the video service was achieved for the following deployment configuration: D=IP_6, USA, PM1, horizontal, small, in the amount of 9.58 USD for the rapid growth scenario, and 9.68 USD for the burst scenario. Overall minimal prices were 9.07 USD and 9.11 USD for the rapid growth and the burst scenario, respectively. However, the deployment configurations used to achieve these prices did not fulfil the QoS requirements. The maximum cost was 136.81 USD for the rapid growth scenario, and 136.93 USD for the burst scenario, for the same configuration D=IP_4, EU, PM2, horizontal, large, due to the higher prices of the selected provider, increased resource prices of the pricing model PM2 and the larges instance size. The summarized information related to the execution cost of the application using the deployment configuration chosen by the algorithm ($Price_D$) and how they compare to the overall minimal price (i.e., the minimal price of all 256 deployment configurations, regardless of the fulfilment of the QoS requirements) and the maximal price for the given scenario are presented in Table 12. In addition to information about cost, Table 12 shows the values of QoS metrics for the deployment configurations chosen by the optimisation algorithm.

When comparing the price of the MRS service execution cost using the deployment configuration identified as optimal by the optimisation algorithm for the dynamic load scenario, the cost was reduced by 95.39% when compared to the maximal price of all possible deployment configurations for the analysed infrastructure providers, and it is 10.5 times cheaper than the average price for the MRS dynamic load scenario (22.80 USD). Similar reductions were achieved for the rapid growth scenario, where the decrease compared to the maximal price was 94.63%, and the burst scenario with a reduction of 94.58% compared to the maximal price of all deployment configurations. The deployment optimisation of the video streaming service resulted in a cost reduction of 92.93% compared to the maximal cost of the considered deployment configurations for the dynamic load scenario, 92.99% for the rapid growth, and 92.93% for the burst scenario.

Based on the used validation scenarios, the proposed simulation-based optimisation algorithm demonstrated the ability to find the deployment configuration with a minimal price which satisfies the QoS requirements. Further work related to the optimisation algorithm includes using more different application service types for algorithm validation, mainly focusing on the applications with dominant consumption of resource types not covered by the presented application service use cases.

Conclusion

Renting public cloud infrastructure and deploying applications in a cloud environment can undoubtedly cut down capital investments of the cloud application providers, but it can still be a very hard task for the application providers to determine what is the optimal service placement and how to choose the cloud infrastructure provider to minimise the service cost while maintaining the appropriate QoS levels.

In this paper, we propose an approach that allows SaaS providers to predict service execution cost and

Table 12 Comparison of prices, and the values of observed QoS metrics for the deployment configuration proposed by the algorithm for each validation scenario

Scenario	$Price_D$ (USD)	Overall Min price (USD)	Max price	Average response time (ms)	% of SLO violation
MRS: Dynamic load	2.16	2.16	46.76	2893.44	2.81
MRS: Rapid growth	2.84	2.09	52.90	2954.68	3.14
MRS: Burst	3.01	2.11	55.54	2981.17	3.38
Video: Dynamic load	8.86	8.86	125.21	1663.43	0.59
Video: Rapid growth	9.58	9.07	136.81	2279.81	1.1
Video: Burst	9.68	9.11	136.93	2010.27	0.53

observed QoS parameters for a cloud application service and to extract knowledge about the features of the cloud application context that affect the cost and QoS to the greatest extent. For our experiments, we used two applications as cloud service use cases - a medical record system, and a video streaming service. A set of features used to implement models of application service and its context is proposed. The features we use are chosen to be applicable to any application that can be deployed as a SaaS service, unlike many solutions available in the literature that implement service models using features specific to a particular type of application. In that way, the proposed approach can be applied to a broad spectrum of applications hosted in cloud environments.

In the proposed optimisation method, we use several statistical learning methods fit for the size of the acquired dataset - linear regression, regularisation regressions, and neural network. To evaluate models, we use several error metrics for the purpose of comparing the accuracy of implemented models. The results demonstrate the ability of regression models to accurately predict the cost and QoS parameters of applications deployed in the cloud, even when an extensive dataset is not available for analysis. We consider this especially relevant for services in the late development phase before they reach the operational phase of their lifecycle when there is no possibility to acquire large datasets and the decision on the production environment and cloud infrastructure providers has not yet been made. We further demonstrate the ability to identify the main contributors to the observed predicted variables, using the permutation importance method that can be applied even to the models implemented using sophisticated techniques such as neural networks. This property of the permutation importance method enables the applicability of our approach to a wide range of models, allowing the analysis of the effect of the cloud application context on the application's cost and QoS even in the case of complex cloud services. The created models of cloud application service and its context are used in the deployment optimisation algorithm which determines the optimised deployment configuration for the application service based on the parameters of the cloud application service context that have the highest impact on the application cost and QoS. The validation of the optimisation algorithm has proven the possibility of identifying the deployment configuration for the application service for a given set of possible deployment configurations and a given load scenario. The validation of the optimisation algorithm was performed using the real pricing offers from seven cloud infrastructure providers, which demonstrated that the operational cost of the application services was reduced by 90 to 95% when comparing the minimal price resulting from the deployment configuration proposed by the optimisation algorithm and the maximal price in all of the possible deployment configurations. The minimum price of the configurations which enabled the fulfilment of the application service QoS requirements was from 6.77 to 10.5 times cheaper than the average price for all of the analysed scenarios.

In addition to being able to identify the optimised deployment configuration which meets the QoS requirements of the service and minimises its operational cost, the proposed optimisation method demonstrated the ability to perform an analysis of the impact of parameters related to the application service and its context to the application service cost and QoS, to be applicable in the early phases of the application development with limited possibilities of acquiring large datasets for the analysis, and to be applicable to a wide range of cloud application services, not only on a specific application type.

Due to the stated properties of the proposed optimisation method, we conclude that it can be valuable to the cloud application service providers, especially in the context of a wide spectrum of various application types. The further steps in this line of research include extending the application of the optimisation method to the application services with dominant consumption of resource types not covered by the use cases presented and analysed in this paper, such as batch processing, or otherwise CPU or memory-intensive application types. In addition to the evaluation of the proposed optimisation approach using various cloud application types, another point of interest is to examine the application of the optimisation method when observing more complex QoS objectives relating to various aspects of QoS delivered to the application end users. Also, since the proposed method is model-based, it could easily be applied to services using concepts that extend the cloud computing paradigm, such as edge and fog computing.

Appendix

Input : M: application service and cloud environment model parameters
Input : IR_{cost}: parameters list ranked by influence on service cost
Input : IP: list of infrastructure providers
Input : SLO: list of QoS requirements defined via SLOs
Input : $location\_constraint$: location of data centre; 0 if there is no constraint, numeric identifier of selected deployment geographical location otherwise
Input : AS: application providers decision on autoscaling
Output: D: optimised deployment configuration

$group\_price\_optimisation = \{Provider\}$;
$D \leftarrow \emptyset$;
IR_coefficients_cost← calculateIRCoefficients(IR_{cost});
OPP← optimisationParameterPriorityList($IR\_coefficients\_cost$);
if $OPP.size > 1$ **then**
  $D\_list \leftarrow \emptyset$;
  foreach p *in* OPP **do**
    $d \leftarrow \emptyset$;
    if p *is in* $group\_price\_optimisation$ **then**
      $d \leftarrow optimiseByPrice(IP, location\_constraint, D, AS, SLO, M)$;
    else
      $d \leftarrow optimiseByResourceType(IP, p, D, AS, location\_constraint, SLO, M)$;
    end if
    $D\_list \leftarrow d$;
  end foreach
  $D \leftarrow$ *deployment configuration from* D_list *with minimal cost*;
else
  $p \leftarrow OPP[1]$;
  if p *is in* $instance\_price\_optimisation$ **then**
    $D \leftarrow optimiseByPrice(IP, location\_constraint, D, AS, SLO, M)$;
  else
    $D \leftarrow optimiseByResourceType(IP, p, D, AS, location\_constraint, SLO, M)$;
  end if
end if
calculateIRCoefficients (IR)
  $IR\_coefficients[1] \leftarrow 1$;
  foreach i *from 2 to* $IR.size$ **do**
    $IR\_coefficients[i] \leftarrow IR[i]/IR[1]$;
  end foreach
  return $IR\_coefficients$;
end
optimisationParameterPriorityList $(IR\_coefficients)$
  $opp \leftarrow \emptyset$;
  foreach i *from 1 to* $IR\_coefficients.size$ **do**
    if $IR\_coefficients[i] > 0.9$ **then**
      opp.add(IR_coefficients[i]);
    end if
  end foreach
  return opp;
end
optimiseByPrice $(IP, location_constraint, D, AS, SLO, M)$
  $IP\_PR \leftarrow$ *sorted list of IPs starting with the lowest instance and network price and datacentre available at location defined by the* $location\_constraint$ *parameter*;
  foreach ip *in* IP_PR **do**
    $D \leftarrow determineInstanceSizeAndPricingModel(ip, D, AS, SLO, M)$;
    if $ip \neq \emptyset$ **then**
      return;
    end if
  end foreach
  return D;
end
optimiseByResourceType $(IP, p, D, AS, location_constraint, SLO, M)$
  $IP\_RT \leftarrow$
  sorted list of IPs starting with the lowest price to resource amount ratio of the resource type defined by p *and data centre availableat* $location\_constraint$ *parameter*;
  foreach ip *in* IP_RT **do**
    $D \leftarrow determineInstanceSizeAndPricingModel(ip, D, AS, SLO, M)$;
    if $ip \neq \emptyset$ **then**
      return;
    end if
  end foreach
  return D;
end
determineInstanceSizeAndPricingModel (ip, D, AS, SLO, M)
  `// defined in Algorithm 2`
end

Algorithm 1 Deployment optimisation algorithm

```
Input  : M: application service and cloud environment model parameters
Input  : ip: infrastructure provider
Input  : SLO: list of QoS requirements defined via SLOs
Input  : D: deployment configuration with parts of the configuration set by the previous
         algorithm steps
Input  : AS: application providers decision on autoscaling
Output: D: optimised deployment configuration
determineInstanceSizeAndPricingModel (ip, D, AS, SLO, M)
    for is in ip.IS do
        as ← no;
        qos_pass, prices ← simulate(M, ip, is, as, SLO);
        if qos_pass then
            D.IS ← is;
            D.AS ← as;
            D.PM ← PM resulting in minimal price form prices;
            return;
        else
            if AS not equal no then
                as ← horizontal;
                qos_pass_h, prices_h ← simulate(M, ip, is, as, SLO);
                as ← vertical;
                qos_pass_v, prices_v ← simulate(M, ip, is, as, SLO);
                if qos_pass_h and qos_pass_v then
                    if min_price_h < min_price_v then
                        D.IS ← is;
                        D.AS ← horizontal;
                        D.PM ← PM resulting in min_price_h;
                        return;
                    else
                        D.AS ← vertical;
                        D.IS ← is;
                        D.PM ← PM resulting in min_price_v;
                        return;
                    end if
                end if
                if qos_pass_h then
                    D.AS ← horizontal;
                    D.IS ← is;
                    D.PM ← PM resulting with min_price_h;
                    return;
                end if
                if qos_pass_v then
                    D.AS ← vertical;
                    D.IS ← is;
                    D.PM ← PM resulting with min_price_v;
                    return;
                end if
            end if
        end if
    end for
    D.IP ← ∅;
    return D;
```

Algorithm 2 Identifying optimal instance size and pricing model

Acknowledgements
Not applicable.

Authors' contributions
This research paper is the result of both stated authors' efforts. Both authors were involved in conducting the research described in the paper. However, we state the certain part of the research that each author was more focused on. I.S. is the corresponding author, she proposed the described metamodel and method. She also designed the experimental set-up and use cases, and obtained data. D.H. is the main reviewer of the paper. He also participated in the metamodel and method design and contributed to the design of the proposed method validation. All authors have read and agreed to the published version of the manuscript.

Funding
This study was performed as a part of the employment of the corresponding author at the Research Unit of Ericsson Nikola Tesla d.d.

Availability of data and materials
The dataset supporting the conclusions of this article is available publicly in the Open Science Framework repository at https://osf.io/r9nzq/.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Both authors provide consent for publication.

Competing interests
The authors declare that they have no competing interests.

Received: 7 December 2021 Accepted: 3 January 2023
Published online: 18 February 2023

References

1. Buckley K (2018) By 2019, 60% of IT workloads will run in the cloud. 451 Research, Boston
2. IDG (2020) IDG cloud computing survey 2020. IDG Communications, Inc., Boston
3. Flexera (2021) Flexera state of the cloud report. Flexera, Illinois. Available online: https://resources.flexera.com/web/pdf/report-cm-state-of-the-cloud-2021.pdf
4. Mell P, Grance T et al (2011) The nist definition of cloud computing. NIST Special Publication National Institute of Standards and Technology, Gaithersburg
5. Rogers O, Fellows W (2013) The cloud pricing codex-2013. 451 Research LLC, Boston
6. Stupar I, Huljenić D (2019) Model-based extraction of knowledge about the effect of cloud application context on application service cost and quality of service. Sci Program 2019:1–19
7. Kaur A, Kaur B, Singh D (2017) Optimization techniques for resource provisioning and load balancing in cloud environment: a review. Int J Inf Eng Electron Bus 9(1):28–35
8. Antonescu AF, Braun T (2016) Simulation of sla-based vm-scaling algorithms for cloud-distributed applications. Futur Gener Comput Syst 54:260–273
9. Yu J, Buyya R (2006) Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Program 14(3, 4):217–230
10. Qi L, Yu J, Zhou Z (2017) An invocation cost optimization method for web services in cloud environment. Sci Program 2017:1–9
11. Li W, Svärd P, Tordsson J, Elmroth E (2013) Cost-optimal cloud service placement under dynamic pricing schemes. In: 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing. IEEE, Dresden, pp 187–194
12. Zhang Y, Yao J, Guan H (2017) Intelligent cloud resource management with deep reinforcement learning. IEEE Cloud Comput 4(6):60–69
13. Skourletopoulos G, Mavromoustakis CX, Mastorakis G, Sahalos JN, Batalla JM, Dobre C (2017) A game theoretic formulation of the technical debt management problem in cloud systems. In: 2017 14th International Conference on Telecommunications (ConTEL). IEEE, Zagreb, pp 7–12
14. Memeti S, Pllana S, Binotto A, Kołodziej J, Brandic I (2019) Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review. Computing 101(8):893–936
15. Madni SHH, Latiff M, Coulibaly Y, Abdulhamid SM (2016) An appraisal of meta-heuristic resource allocation techniques for iaas cloud. Indian J Sci Technol 9(4):1–14
16. Zhu Z, Zhang G, Li M, Liu X (2015) Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans Parallel Distrib Syst 27(5):1344–1357
17. Abdelaziz A, Elhoseny M, Salama AS, Riad A (2018) A machine learning model for improving healthcare services on cloud computing environment. Measurement 119:117–128
18. Xu Y, Yao J, Jacobsen HA, Guan H (2017) Cost-efficient negotiation over multiple resources with reinforcement learning. In: 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). IEEE, Barcelona, pp 1–6
19. Li Z, Ge J, Hu H, Song W, Hu H, Luo B (2015) Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. IEEE Trans Serv Comput 11(4):713–726
20. Rodriguez MA, Buyya R (2017) Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods. ACM Trans Auton Adapt Syst (TAAS) 12(2):1–22
21. Ahmad B, McClean S, Charles D, Parr G (2018) Analysis of energy saving technique in cloudsim using gaming workload. Cloud Comput 2018:143
22. Pakdel R, Herbert J (2016) Scalable cloud-based analysis framework for medical big-data. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol 2. IEEE, Atlanta, pp 647–652
23. Salánki Á, Kincses G, Gönczy L, Kocsis I (2017) Data analysis-based capacity planning of vcl clouds. Int J Cloud Comput 6(4):370–383
24. Koch F, Assunção MD, Cardonha C, Netto MA (2016) Optimising resource costs of cloud computing for education. Futur Gener Comput Syst 55:473–479
25. Toosi AN, Sinnott RO, Buyya R (2018) Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka. Futur Gener Comput Syst 79:765–775
26. Hauser CB, Domaschka J, Wesner S (2018) Predictability of resource intensive big data and hpc jobs in cloud data centres. In: 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, Lisbon, pp 358–365
27. Alkhanak EN, Lee SP, Khan SUR (2015) Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities. Futur Gener Comput Syst 50:3–21
28. Ranjan R, Benatallah B, Dustdar S, Papazoglou MP (2015) Cloud resource orchestration programming: overview, issues, and directions. IEEE Internet Comput 19(5):46–56
29. Fakhfakh F, Kacem HH, Kacem AH (2014) Workflow scheduling in cloud computing: a survey. In: 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations. IEEE, Ulm, pp 372–378
30. Xiong H, Filelis-Papadopoulos C, Dong D, Castañé GG, Morrison JP (2017) Towards a scalable and adaptable resource allocation framework in cloud environments. In: 2017 46th International Conference on Parallel Processing Workshops (ICPPW). IEEE, Bristol, pp 137–144
31. Maiyama KM, Kouvatsos D, Mohammed B, Kiran M, Kamala MA (2017) Performance modelling and analysis of an openstack iaas cloud computing platform. In: 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, Prague, pp 198–205
32. Chase J, Niyato D (2015) Joint optimization of resource provisioning in cloud computing. IEEE Trans Serv Comput 10(3):396–409
33. Li G, Xu S, Wu J, Ding H (2018) Resource scheduling based on improved spectral clustering algorithm in edge computing. Sci Program 2018:1–13
34. Islam S, Keung J, Lee K, Liu A (2012) Empirical prediction models for adaptive resource provisioning in the cloud. Futur Gener Comput Syst 28(1):155–162

35. Borkowski M, Schulte S, Hochreiner C (2016) Predicting cloud resource utilization. In: Proceedings of the 9th International Conference on Utility and Cloud Computing. IEEE, Shanghai, pp 37–42
36. Shimizu S, Rangaswami R, Duran-Limon HA, Corona-Perez M (2009) Platform-independent modeling and prediction of application resource usage characteristics. J Syst Softw 82(12):2117–2127
37. Magalhães D, Calheiros RN, Buyya R, Gomes DG (2015) Workload modeling for resource usage analysis and simulation in cloud computing. Comput Electr Eng 47:69–81
38. Kistowski JV, Herbst N, Kounev S, Groenda H, Stier C, Lehrig S (2017) Modeling and extracting load intensity profiles. ACM Trans Auton Adapt Syst (TAAS) 11(4):1–28
39. Shen S, Van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, Shenzhen, pp 465–474
40. Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P (2014) Cloud computing patterns: fundamentals to design, build, and manage cloud applications. Springer, Wien
41. Milenkoski A, Iosup A, Kounev S, Sachs K, Rygielski P, Ding J, Cirne W, Rosenberg F (2014) Cloud usage patterns: A formalism for description of cloud usage scenarios: A formalism for description of cloud usage scenarios. Technical Report: SPEC-RG (2013-001)
42. Forward A, Lethbridge TC (2008) A taxonomy of software types to facilitate search and evidence-based software engineering. In: Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: meeting of minds. ACM, Ontario, pp 179–191
43. Erl T, Puttini R, Mahmood Z (2013) Cloud computing: concepts, technology, & architecture. Pearson Education, Upper Saddle River
44. Mellor SJ, Scott K, Uhl A, Weise D (2004) MDA distilled: principles of model-driven architecture. Addison-Wesley Professional, Boston
45. Zhang M, Ranjan R, Haller A, Georgakopoulos D, Menzel M, Nepal S (2012) An ontology-based system for cloud infrastructure services' discovery. In: 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom). IEEE, Pittsburgh, pp 524–530
46. Rekik M, Boukadi K, Ben-Abdallah H (2015) Cloud description ontology for service discovery and selection. In: 2015 10th International Joint Conference on Software Technologies (ICSOFT), vol 1. IEEE, Colmar, pp 1–11
47. Di Martino B, Cretella G, Esposito A (2014) Towards a unified owl ontology of cloud vendors' appliances and services at paas and saas level. In: 2014 Eighth International Conference on Complex. Intelligent and Software Intensive Systems. IEEE, Birmingham, pp 570–575
48. Youseff L, Butrico M, Da Silva D (2008) Toward a unified ontology of cloud computing. In: 2008 Grid Computing Environments Workshop. IEEE, Austin, pp 1–10
49. Han T, Sim KM (2010) An ontology-enhanced cloud service discovery system. Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, pp 17–19
50. Moscato F, Aversa R, Di Martino B, Fortiş TF, Munteanu V (2011) An analysis of mosaic ontology for cloud resources annotation. In: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, Szczecin, pp 973–980
51. Weinhardt C, Anandasivam A, Blau B, Stößer J (2009) Business models in the service world. IT Prof 11(2):28–33
52. Tahamtan A, Beheshti SA, Anjomshoaa A, Tjoa AM (2012) A cloud repository and discovery framework based on a unified business and cloud service ontology. In: 2012 IEEE Eighth World Congress on Services. IEEE, Honolulu, pp 203–210
53. Karim B, Tan Q, Villar JR, de la Cal E (2017) Resource brokerage ontology for vendor-independent cloud service management. In: 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). IEEE, Chengdu, pp 466–472
54. Labidi T, Mtibaa A, Gaaloul W, Tata S, Gargouri F (2017) Cloud sla modeling and monitoring. In: 2017 IEEE International Conference on Services Computing (SCC). IEEE, Honolulu, pp 338–345
55. Aceto G, Botta A, De Donato W, Pescapè A (2013) Cloud monitoring: A survey. Comput Netw 57(9):2093–2115
56. Topology and orchestration specification for cloud applications (TOSCA). OASIS specification. (2021). www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca. Accessed Nov 2021
57. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50
58. Guzek M, Kliazovich D, Bouvry P (2013) A holistic model for resource representation in virtualized cloud computing data centers. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, vol 1. IEEE, Bristol, pp 590–598
59. Núñez A, Vázquez-Poletti JL, Caminero AC, Castañé GG, Carretero J, Llorente IM (2012) icancloud: A flexible and scalable cloud infrastructure simulator. J Grid Comput 10(1):185–209
60. Wickremasinghe B, Calheiros RN, Buyya R (2010) Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE international conference on advanced information networking and applications. IEEE, pp 446–452
61. Tian W, Zhao Y, Xu M, Zhong Y, Sun X (2013) A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. IEEE Trans Autom Sci Eng 12(1):153–161
62. Amazon cloudwatch. (2021). https://aws.amazon.com/cloudwatch/. Accessed 26 Oct 2021
63. Openstack telemetry service command-line client documentation. (2021). https://docs.openstack.org/mitaka/cli-reference/ceilometer.html. Accessed 5 Mar 2021
64. Furht B, Escalante A et al (2010) Handbook of cloud computing, vol 3. Springer, Boston
65. Weinman J (2012) Cloudonomics: The business value of cloud computing. Wiley, Hoboken
66. Becker S, Brataas G, Lehrig S (2017) Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications: The CloudScale Method. Springer, Cham
67. Kutner MH, Nachtsheim CJ, Neter J, Li W et al (2005) Applied linear statistical models. McGraw-Hill, New York
68. Hastie T, Tibshirani R, Wainwright M (2019) Statistical learning with sparsity: the lasso and generalizations. Chapman and Hall/CRC, Boca Raton
69. Efron B, Hastie T, Johnstone I, Tibshirani R (2004) Least angle regression. Ann Stat 32(2):407–499
70. Hastie T, Tibshirani R, Friedman JH (2017) The elements of statistical learning: Data mining, inference, and prediction. Springer Series in Statistics, New York
71. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
72. Wolfe BA, Mamlin BW, Biondich PG, Fraser HS, Jazayeri D, Allen C, Miranda J, Tierney WM (2006) The openmrs system: collaborating toward an open source emr for developing countries. In: AMIA Annual Symposium Proceedings, vol 2006. American Medical Informatics Association, Washington, pp 1146
73. Nimble streamer. (2021). https://wmspanel.com/nimble. Accessed 15 Oct 2021
74. Openstack. (2021). http://www.openstack.org. Accessed 15 Oct 2021
75. Apache jmeter. (2021). http://jmeter.apache.org/. Accessed 15 Oct 2021
76. Stupar I, Huljenić D (2017) Analyzing service resource usage profiles for optimization of cloud service execution cost. In: IEEE EUROCON 2017–17th International Conference on Smart Technologies. IEEE, Ohrid, pp 79–84
77. Centurylink price calculator. (2019). https://www.ctl.io/estimator/. Accessed 2 Aug 2019
78. Google price calculator. (2019). https://cloud.google.com/products/calculator/. Accessed 2 Aug 2019
79. Amazon price calculator. (2019). https://calculator.s3.amazonaws.com/index.html. Accessed 2 Aug 2019
80. Alibaba cloud price calculator. (2019). https://www.alibabacloud.com/pricing. Accessed 2 Aug 2019
81. Digital ocean price calculator. (2019). https://www.digitalocean.com/pricing/. Accessed 2 Aug 2019
82. Azure price calculator. (2019). https://azure.microsoft.com/en-us/pricing/calculator/. Accessed 2 Aug 2019
83. Oracle price calculator. (2019). https://cloud.oracle.com/en_US/cost-estimator. Accessed 2 Aug 2019

84. De Myttenaere A, Golden B, Le Grand B, Rossi F (2016) Mean absolute percentage error for regression models. Neurocomputing 192:38–48
85. Bendat JS, Piersol AG (2011) Random data: analysis and measurement procedures, vol 729. Wiley, Hoboken
86. Eli5 documentation. (2021). https://eli5.readthedocs.io/en/latest/. Accessed 15 Oct 2021
87. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput Pract Experience 24(13):1397–1420
88. Antonescu AF, Braun T (2014) Sla-driven simulation of multi-tenant scalable cloud-distributed enterprise information systems. In: International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing. Springer, Paris, pp 91–102

Publisher's Note