IEICE TRANS. INF. & SYST., VOL.E102–D, NO.5 MAY 2019

PAPER *Special Section on the Architectures, Protocols, and Applications for the Future Internet*

Multi-Target Classification Based Automatic Virtual Resource Allocation Scheme

Abu Hena Al MUKTADIR[†a)], ***Nonmember***, **Takaya MIYAZAWA**[†], ***Member***, **Pedro MARTINEZ-JULIA**[†], ***Nonmember***, **Hiroaki HARAI**[†], ***and*** **Ved P. KAFLE**[†], ***Members***

SUMMARY In this paper, we propose a method for automatic virtual resource allocation by using a multi-target classification-based scheme (MTCAS). In our method, an Infrastructure Provider (InP) bundles its CPU, memory, storage, and bandwidth resources as Network Elements (NEs) and categorizes them into several types in accordance to their function, capabilities, location, energy consumption, price, etc. MTCAS is used by the InP to optimally allocate a set of NEs to a Virtual Network Operator (VNO). Such NEs will be subject to some constraints, such as the avoidance of resource over-allocation and the satisfaction of multiple Quality of Service (QoS) metrics. In order to achieve a comparable or higher prediction accuracy by using less training time than the available ensemble-based multi-target classification (MTC) algorithms, we propose a majority-voting based ensemble algorithm (MVEN) for MTCAS. We numerically evaluate the performance of MTCAS by using the MVEN and available MTC algorithms with synthetic training datasets. The results indicate that the MVEN algorithm requires 70% less training time but achieves the same accuracy as the related ensemble based MTC algorithms. The results also demonstrate that increasing the amount of training data increases the efficacy of MTCAS, thus reducing CPU and memory allocation by about 33% and 51%, respectively.

key words: *multi-target classification, virtual resource allocation scheme, multiple QoS*

1. Introduction

The Future Internet is expected to support various types of services and applications (e.g. smart metering, video surveillance, connected vehicles, emergency rescue during disaster), each with diverse requirements (e.g. ultra-reliable connectivity, very low latency, massive number of devices, and quality of service (QoS) guaranteed through dynamic virtual resource control and adjustment). It would be based on network virtualization, software defined networking (SDN), and network slicing technologies [1]. Infrastructure provider (InP), virtual network operator (VNO), and application service provider (ASP) are the three stakeholders involved in the management of virtual resources [2].

The InP virtualizes its physical resources into different network elements (NEs) according to their functionalities and provides information about available NEs and their usage prices to the subscribing VNO(s). The NEs contain various combinations of CPU, storage, and bandwidth resources with associated values, and features like location and energy budget. For example, a NE consists of network software module installed in virtual machine (VM) with required CPU, storage, and bandwidth resources. NE can also be a shared physical modular infrastructure like WiFi access point characterized with its a location, cache storage, bandwidth, and transmission power requirements or a link bandwidth between two geographic locations. Hereafter, throughout this paper we use NE to represent 'virtual resource'. A VNO leases NEs from InP to provide networking platform with required QoS for the services offered by ASP.

Overcommitting and service level agreement (SLA) QoS violations are two common problems of NE allocation during peak demands [3]. Overcommitting refers to the situation where InP accepts new VNO requests, even though the available NEs are not sufficient to satisfy them. In order to accommodate new VNO requests InP usually adjusts the resources already allocated to incumbent VNOs, leading to the SLA QoS violation. InP loses revenue due to QoS violation penalty. Irrespective of peak demand situations, these two problems are particularly severe when the InP owns limited physical resources such as in a micro edge datacenter [4]. Customers of VNOs usually use only 10% of the leased NEs [5]. InP should incorporate feedbacks from VNO(s) to avoid over-allocation of NEs and maintain the reserve of resources for future requests, and thus these two problems can be solved to a large extent.

In this paper, we design the allocation of NEs that satisfy multiple QoS metrics for dynamic VNO requests for different amount of NEs by leveraging a proactive multi-target classification (MTC) based machine learning approach. The InP in our scenario owns limited physical resources and follows the VM allocation model of Amazon elastic cloud compute (EC2) [6] and Google compute engine (GCE) [7].

We propose an MTC based automatic NE allocation scheme (MTCAS) to be used by InP to predict and allocate a multiple QoS compliant set of NEs for each VNO request. MTCAS satisfies three objectives for InP (namely balancing the resource utilization, minimizing energy consumption, and avoiding over-allocation) by considering four QoS metrics for VNOs (provable QoS satisfaction guarantee, response time, location, and price). By employing MTCAS, InP is able to efficiently allocate resources of optimal locations for load-balancing and energy usage, as well as increase resource availability for future VNO requests by

Manuscript received August 23, 2018.
Manuscript revised December 11, 2018.
Manuscript publicized February 19, 2019.
[†]The authors are with National Institute of Information and Communications Technology (NICT), Koganei-shi, 184–8795 Japan.
a) E-mail: muktadir@nict.go.jp (Corresponding author)
DOI: 10.1587/transinf.2018NTP0016

avoiding potential over-allocation. Note that we consider balancing utilization of resources owned by InP in various places through the allocations by MTCAS. The prediction accuracy of MTCAS provably guarantees that the demanded QoS for a certain percentage of new requests will be satisfied by the allocated NEs. MTCAS also ensures that the VNOs get response for the requests within SLA compliant time, at desired location and price.

The remaining of this paper is organized as follows. We present a literature review of machine learning based NE management approaches and clarify our contributions in Sect. 2. Section 3 describes necessary background concepts to understand our proposal. Section 4 illustrates MTCAS with an example, discusses in detail the procedural steps of the proposed scheme, and presents an ensemble algorithm for MTCAS. Section 5 describes the preparation of two training datasets. Numerical evaluation results are discussed in Sect. 6. Finally, Sect. 7 concludes this paper and mentions about future directions.

2. Related Work and Contributions

InP employs either combinatorial optimization [8]–[11] or machine learning approaches [12]–[18] to solve the NE allocation problem without any feedback about the solutions from VNOs. Due to exponential computational time complexity, optimal solutions usually consider limited number of QoS constraints [19]. Therefore, many optimization-based prior works have tried to solve the NE allocation problem by considering one or two QoS constraints, such as computation time [8], resource consumption and load balancing [9], latency and reliability [10], and energy [11].

Machine learning based NE management have been addressed in several works [12]–[18]. Linear support vector machine (SVM) based classification of VNO requests is suggested in [12] to increase the acceptance rate of urgent requests requiring large amount of resources. Reinforcement learning (RL) agent with feedback is used in each substrate node and link to learn optimal policy to dynamically allocate resources to virtual nodes and links, which in turn improves VNO request acceptance and revenue [13]. A random forest (RF) algorithm-based method is presented in [14] to dynamically adjust the number of virtual network function (VNF) instances to minimize QoS violation and operation/leasing cost. VNF forwarding graph topology information is exploited by a neural network (NN) based algorithm to predict future resource metrics for each VNF component [15]. Ensemble prediction mechanisms are presented in [16] to turn off and assign VMs smartly to maintain SLA compliant QoS level. Autoregressive integrated moving average (ARIMA) based future resource demand prediction is adopted in [17] to improve consolidation and migration efficiency for the cloud servers employing local storage. The k-Nearest Neighbor (k-NN) algorithm is used for beam selection in the multiuser massive multiple-input multiple-output (MIMO) environment [18]. Interested readers are referred to [20]–[22] for a comprehensive use of above-mentioned machine learning algorithms in various applications.

Table 1 Comparison of MTCAS with related works.

Reference	Algorithm	Feedback	Classification
[12]	Linear SVM	×	Binary
[13]	RL	✓	×
[14]	RF	×	Multi-class
[15]	NN	×	×
[16]	Ensemble	×	×
[17]	Regression	×	×
[18]	k-NN	×	Multi-class
MTCAS	Classifier chain, Ensemble	✓	Multi-target

Our MTCAS supports feedback mechanism between InP and VNOs, and classification for resource allocation decisions. However, previous works [12]–[18] discussed earlier do not support both of them at the same time. There are several advantages of feedback mechanism for both InP and VNOs. Feedbacks from VNOs help InP to verify the correctness of resource allocation decisions and know about the satisfaction of VNOs, and according to these satisfaction measurements improve the training data, which is essential for machine learning algorithms. Feedback mechanism enables a VNO to reject an unacceptable offer (in terms of resource configuration and/or price) from InP, and request for a new offer. This also helps InP to avoid request rejection during peak demand situations, because in this case InP offers less amount of resources compared to the requested one and the VNO accepts it. Table 1 presents a comparison of related work [12]–[18] with MTCAS in terms of three criteria: employed machine learning *algorithm*, use of *feedback* among involved parties in decision process, and whether or not *classification* based allocation approach used.

Our motivation to adopt machine learning approach in this paper is explained in the following. Optimization focuses on obtaining exact solutions at the cost of increased computation complexity, which is acceptable for static situation. Machine learning approaches usually require less time to solve a problem than the conventional mathematical optimization approaches do. Therefore, they are better suitable for dynamic resource allocation decision making purposes, where both resource requests and availability change with time [18]. Particle swarm optimization [23], column generation [24], dynamic programming [25] methods have been used to overcome the exponential time issue of conventional optimization approaches. However, multiple available methods may introduce a new decision problem of deciding which solution to use at different resource availability situations. To avoid this problem, our objective is to propose a single solution, which we call "a general solution" for the dynamic ranges of VNO resource requests and provide a quick response (e.g., within 20 milliseconds [26]). Therefore, machine learning approaches are more favorable for our goal. However, poor prediction and absence of feedback from VNO to InP may result in resource over-allocation, under-utilization, or monetary losses for InP and VNOs [27]. Therefore, we also investigate the

ensemble approach with MTCAS to increase the prediction accuracy.

To the best of our knowledge there are no prior works on proactive NE management that consider the aforementioned three objectives for InP and four QoS metrics for VNOs. The contributions of this paper are three-fold.

- First, we propose MTCAS to be used by InP to allocate a multiple QoS compliant set of NEs to VNOs. MTCAS avoids resource over-allocation by incorporating VNOs feedback.
- Second, we propose a majority voting based ensemble algorithm (MVEN) for MTCAS.
- Third, we assess the performance of MTCAS by using MVEN and available MTC algorithms with synthetic training datasets. The proposed MVEN algorithm requires 70% less training time but achieves the same accuracy as the related ensemble based MTC algorithms. The results also demonstrate that increasing the amount of training data increases the efficacy of MTCAS, thus reducing CPU and memory allocation by about 33% and 51%, respectively.

3. Background Concepts

We discuss here the necessary background concepts related to MTC and ensemble algorithm.

3.1 MTC

An NE is described with a set of related features F. The NE is classified with a set of labels L determined as the most relevant features. Each label $l \in L$ can take a finite V number of values. Each feature $f \in F$ can be assigned with either a numeric or an alphanumeric value. There are four classification paradigms: binary ($|L| = 1$, $V = 2$), multi-class ($|L| = 1$, $V > 2$), multi-label ($|L| > 1$, $V = 2$), and multi-target ($|L| > 1$, $V \geq 2$) [30]. In MTCAS, the number of labels $|L|$ must be greater than or equal to the number of QoS metrics to be satisfied by InP for an NE request. Classifying an NE (e.g., a VM) instance with $|L| = 4$ labels (e.g., CPU, memory, storage, and NIC speed) and $V > 2$ values per label is an MTC problem. For example, the label related to NIC speed can have three values (e.g. 100Mbps, 1Gbps or 10Gbps), and the label related to CPU may represent the number of CPU cores (e.g. $1 \sim 9$).

3.2 Training, Testing, Request, and Prediction

Let InP creates N instances of a NE, which are classified with $|L|$ labels and V values per label. A collection of these classification information together with $|F|$ features per instance serves as the *training dataset* for MTC algorithms. The training dataset can be represented as a $N \times (|L| + |F|)$ dimensional matrix $\mathbf{T}_{N\times(|L|+|F|)}$, where each row represents an instance, first $|L|$ elements of each row indicates labels and remaining $|F|$ elements represent the features of that instance. Let M be a set of available MTC algorithms. k-fold cross-validation and train/test split are the two methods to train MTC algorithms $m \in M$ [30]. Cross-validation is preferred when N is small. Train/test split is suggested when N is large with known distribution function. Trained MTC algorithms, denoted as *classifiers*, are used to predict $|L|$ labels of a new NE request ('prediction') that meets $|L|$ QoS metrics. A VNO requests for a NE to InP in a prescribed form suitable for prediction by MTC algorithms. The prescribed form includes multiple mandatory (must mention the resource specifications) and optional (not necessary to mention) features. InP allocates the NE to VNO according to the prediction.

3.3 Metrics to Evaluate Prediction Accuracy

Exact matching (E_M) and Hamming score (H_S) are used metrics to determine prediction accuracy of the multi-target or multi-dimensional classification tasks [31]. E_M (label-set based global accuracy) and H_S (label-based mean accuracy) are defined by Eqs. (1) and (2), respectively [31].

$$E_M = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{y}'_i, \mathbf{y}_i) \tag{1}$$

$$H_S = \frac{1}{N|L|}\sum_{i=1}^{N}\sum_{l=1}^{|L|}\delta(y'_{il}, y_{il}) \tag{2}$$

where N is the total number of training instances indexed by i, and each instance has $|L|$ labels indexed by l, $\mathbf{y}'_i = \{y'_{il} : i = 1, 2, \cdots, N,$ and $l = 1, 2, \cdots, |L|\}$ is the vector of predicted labels and $\mathbf{y}_i = \{y_{il} : i = 1, 2, \cdots, N,$ and $l = 1, 2, \cdots, |L|\}$ is the vector of known labels. Note that y'_{il} and y_{il} is the predicted and known values, respectively, of the lth label of an instance i. In Eq. (1), $\delta(\mathbf{y}'_i, \mathbf{y}_i) = 1$ if $\mathbf{y}'_i = \mathbf{y}_i$, and otherwise $\delta(\mathbf{y}'_i, \mathbf{y}_i) = 0$. Similarly, in Eq. (2) $\delta(y'_{il}, y_{il}) = 1$ if $y'_{il} = y_{il}$, and otherwise $\delta(y'_{il}, y_{il}) = 0$. The desired prediction values of E_M and H_S are both 1.

3.4 MTC Methods and Algorithms

MTC problems are solved either by problem transformation methods (i.e. converting a multi-target problem into a set of binary classification problems and solving them by using binary classification algorithms) or by algorithm adaptation methods (i.e. adapting the algorithm to directly perform multi-target classification) [31]. Binary relevance (BR), classifier-chains (CC), and label powerset (LP) are the generally adopted problem transformation methods [30]. We explain the BR method in the following. Let the given multi-target dataset $\mathbf{T}_{N\times(|L|+|F|)}$ consists of two sub-matrices, i.e., $\mathbf{T}_{N\times(|L|+|F|)} = [\mathbf{A}_{N\times|L|} \mid \mathbf{B}_{N\times|F|}]$. Sub-matrices $\mathbf{A}_{N\times|L|}$ and $\mathbf{B}_{N\times|F|}$ represent $|L|$ labels and $|F|$ features for each of N instances. For each label $l \in L$ the BR method creates a matrix $\mathbf{T}^l_{N\times(1+|F|)} = [\mathbf{A}_{N\times 1} \mid \mathbf{B}_{N\times|F|}]$, where sub-matrix $\mathbf{A}_{N\times 1}$ represents the lth column of sub-matrix $\mathbf{A}_{N\times|L|}$. The BR method then independently learns the class (binary or multi-class)

of each label $l \in L$ by using the same algorithm and corresponding matrix $\mathbf{T}^{l}_{N\times(1+|F|)}$. The information about label correlations i.e. the probability of appearing two or more labels together is not preserved with BR method due to independent label learning, which sometimes may lead to poor prediction. In order to keep this correlation information, the CC method is presented where correlations are preserved by creating a chain of labels in the feature space. The label correlations in the LP method are conserved by considering all possible combinations of label combinations.

The available MTC algorithms $m \in M$ are implemented based on the BR, CC, and LP methods. Based on training time and prediction accuracy, one MTC algorithm $m \in M$ is selected by InP to use with MTCAS.

3.5 Ensemble Algorithm

Ensemble learning is the technique where multiple classifiers are strategically generated and combined to improve the classification or prediction performance of a poor classifier [32]. The motivation behind ensemble learning comes from the decision making process of our daily life events. For example, we buy a product by reading several reviews about it, or an employer hires a person based on several recommendations received with the application. Here each product review or recommendation acts as an expert classifier. The final decision is made by combining the decisions/opinions of individual classifiers to make a better decision [33]. Commonly used ensemble learning algorithms are adaboost, bagging, stacked generalization, mixture of experts, and voting based methods ([33] and the references therein).

4. The Proposal of MTCAS

We describe the flow of actions among ASP, InP, and VNOs, and explain how InP uses MTCAS to predict and allocate NEs with the help of Fig. 1 (Sect. 4.1). A simple example of MTCAS with the process of creating training dataset for VMs are explained in Fig. 2 (Sect. 4.2). In Sect. 4.4, we present the detail descriptions of the methodology used in MTCAS with the help of two algorithms. We then discuss the our proposed MVEN algorithm for MTCAS (Sect. 4.3).

Note that the number of labels $|L|$ for classifying an NE should be equal to or higher than the number of QoS parameters to be satisfied when this NE is requested. The number of features $|F|$ should be equal to or higher than the number of labels $|L|$, because labels are determined according to the features. Therefore, the number of QoS $\leq |L| \leq |F|$. The number of clusters b should be greater than or equal to 1, and it does not depend on the number of QoS, $|L|$, and $|F|$. The number of instances N for an NE should be large enough (e.g., a few hundreds) for preparing a reasonable training data. The InP is solely responsible to determining the numbers of $|L|$, $|F|$, b, and N. The number of QoS requirements and MTC algorithms $|M|$ are determined by InP according to the VNO requests and utilized machine learning software packages, respectively.

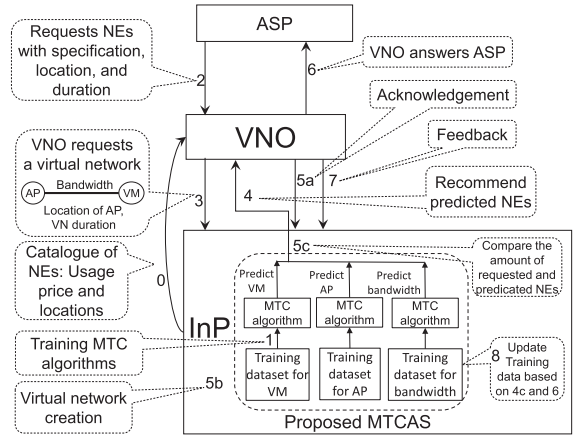


Fig. 1 The flow of actions among InP, VNO, and ASPs.

4.1 Flow of Actions

The InP provides a catalogue of NEs, which includes the types of NEs, their prices and locations, to the subscribing VNOs (step 0 of Fig. 1, where only one VNO is shown for simplicity). The catalogue of each NE containing detailed information, not disclosed to VNOs, serves as the training dataset. For each type of NE, MTCAS trains a suitable MTC algorithm $m \in M$, with the training datasets (step 1). An ASP requests for NEs with abstract resource specifications and intended usage duration (step 2). In response to the ASP's request, the VNO identifies the necessary configurations of requested NEs (VM, access point (AP), and link bandwidth) and submits a virtual network (VN) request in a specified format to InP with required QoS metrics, locations, and intended duration (step 3). InP uses trained MTC algorithms of step 1 to predict and recommend the VNO a set of resources suitable to meet the QoS requirements (step 4). Upon receiving an acknowledgment from the VNO (step 5a), InP allocates NEs to VNO to create a VN (step 5b). InP compares the configurations of requested and predicted NEs to determine whether there is an over-allocation or under-allocation (step 5c). VNO answers back to ASP (step 6). After the intended duration is over VNO releases the NEs and provides feedback about the usage of leased resources (step 7) to InP. InP updates the training datasets by using the information obtained from steps 5c and 7, and re-trains MTC algorithms (step 8).

In our considered scenario the VNO cannot exactly estimate resource requirements for the requested NEs for two reasons. First, ASPs usually over-estimate the abstract resource specifications for NEs. For example, it was reported that only 10% of leased computation resources are used [5]. Second, InP informs the VNO only the names and locations of NEs. However, the actual resource availability situation, which changes with time, is not disclosed to VNOs. Thus, the VNO cannot make exact estimation due to incorrect and incomplete information [39]. The VNO can be able to

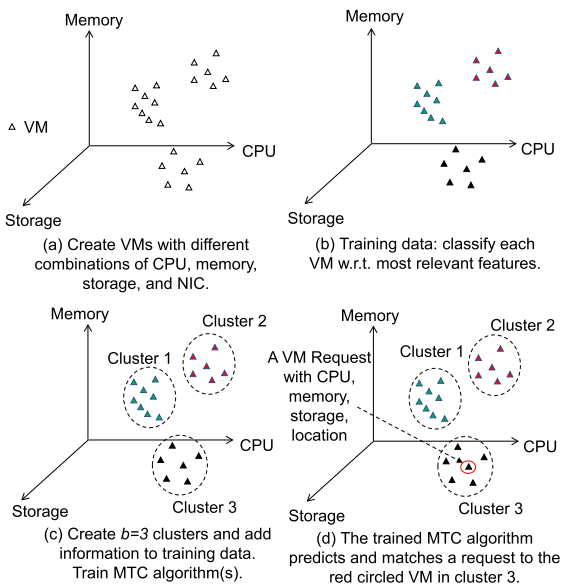


Fig. 2 An illustrative example of MTCAS.

compute and request for the exact amount of required resources, only if all the information exchanged among InP, VNO, ASP are made public [40]. On the other hand, InPs motivation for avoiding resource over-allocation is to increase revenue and improve resource utilization, where InP wants to satisfy as many VNO requests as possible with its limited physical resources.

4.2 An Illustrative Example of MTCAS

Let us assume that InP offers VMs as NE to VNOs. InP creates VMs as the NEs consisting of different amount of CPU, memory, storage, and network interface card (NIC) speed as shown in Fig. 2 (a). The VMs are described with $|F| = 11$ features, which are the amounts and types of CPU, RAM, storage, and NIC, location of VM host, power usage, and service type. InP informs the VNOs with a subset of these features F (step 0 of Fig. 1). InP assigns $|L| = 5$ labels to each VM. First, InP classifies each VM according to the following four labels: CPU, memory, storage sizes, NIC speed (shown in Fig. 2 (b)). InP then creates b (here $b = 3$) clusters (by using hierarchical [28] or k-means [29] clustering algorithm) of VMs having location and configuration (CPU and memory combination) as common features (shown in Fig. 2 (c)). These cluster information serve as the fifth label. Note that, there is no relation between the number of labels and clusters. A collection of these classification information of all VMs serves as training dataset for the MTC algorithms. MTCAS trains a suitable MTC algorithm by using the training dataset with five labels and eleven features (step 1 of Fig. 1). A VM request in specified format containing information about required amount of CPU, memory, storage, NIC, location, and usage duration arrives from VNO to InP (step 3 of Fig. 1). InP specific features such as cluster and energy are not mentioned in the request. The trained MTC algorithm predicts and matches the request to the red circled VM in cluster 3 (shown in Fig. 2 (d)). MTCAS recommends allocating this VM to the requesting VNO. In case this VM has already been allocated to previous requests, InP allocates another VM from the same cluster 3 (step 4 of Fig. 1).

Note that we only show one type of NE in this example. If the VNO request demands x NEs, the procedures of Fig. 2 must be executed simultaneously x times.

4.3 Our MVEN Algorithm

4.3.1 Description of MVEN

We follow the similar technique as the BR method to divide the multi-target problem (dataset matrix) at hand into $|L|$ multi-class problems ($|L|$ matrices). In order to solve each of these multi-class problems, contrary to the BR method, our MVEN employs an ensemble of five different algorithms and multi-voting decision rule. The MVEN algorithm is elaborated in Fig. 3, which works as follows. For each of the given label l, $l = 1, 2, \cdots, |L|$, similar to the BR method, MVEN creates a dataset matrix $\mathbf{T}^{l}_{N\times(1+|F|)}$. During training, MVEN *simultaneously* learns about each label $l \in L$ by an ensemble of five functionally different algorithms namely the Bayesian Network (BN), k-NN, Decision Table (DT), Decision Tree C4.5, and RF. Each algorithm predicts a value for the label. The final prediction of MVEN is decided by *majority voting* i.e. the label to which the majority of algorithms in the ensemble agree is the final output label of the ensemble. To keep the training time small, we do not include NN (multilayer perceptron) and SVM algorithms. However, if training time is relaxed, NN and SVM can be included in the ensemble. A short description of working principles and parameters of the seven above mentioned algorithms are found in [34] and the references therein.

The motivation of the MVEN algorithm is to increase prediction accuracy as much as possible by using less training time than the two available adaboost and bagging MTC algorithms provided by the MEKA [35] software. The reasons are twofold. First, small improvement in prediction accuracy is considered significant in many machine learning based applications [26]. Second, prediction accuracy of two ensemble MTC algorithms increases the number of adaboost and bagging, respectively, iterations performed for a given training dataset. After reaching a certain maximum accuracy, further iterations do not change these accuracy scores. Finding the iteration number resulting in the maximum accuracy requires an exhaustive search. Therefore, the cumulative training time of exhaustive search becomes an issue.

While adopting BR method logic with MVEN, we have ignored the effect losing of label correlations. In future we shall consider feature engineering and principal component analysis method [21] with MVEN to avoid ignoring this effect.

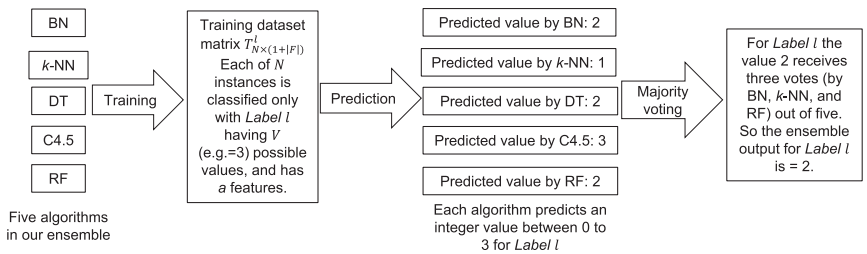


Fig. 3 An illustration of our MVEN algorithm with label l. The procedure above is *simultaneously* repeated for each label $l \in L$. The final predictions of each label $l \in L$ are combined through majority voting to make the final MTC prediction.

4.3.2 H_S, E_M, and Training Times for MVEN

Each label $l \in L$ is considered independently with MVEN. Therefore, the Eq. (2) is also suitable to compute H_S for MVEN. However, the Eq. (1) is not suitable for computing E_M for MVEN. We redefine the Eq. (1) as Eq. (3) in the following.

$$E_M \approx \min_{l\in\{1,2,\cdots,|L|\}} \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{|L|}\delta(y'_{il}, y_{il}) \tag{3}$$

where, $\frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{|L|}\delta(y'_{il}, y_{il})$ denotes the average accuracy for label l over N instances. The MVEN algorithm independently and simultaneously learns and predicts about each of the $|L|$ labels. Therefore, the label $l \in L$ that achieves the minimum average accuracy approximately determines the E_M score for MVEN. For the same reason, the total required training time equals the highest individual training time among the $l \in L$ labels.

4.4 Details of the MTCAS

The MTCAS consists of two stages: 1) Proactive categorization and training, and 2) Prediction, recommendation, and re-training. We discuss the above two stages in detail with the help of Algorithms 1 and 2. The notations used in Algorithms 1 and 2 are defined in Table 2.

The inputs to Algorithm 1 are the physical resource information ($\mathcal{P}$), the set of available MTC and our presented MVEN algorithms (M), and the set of types of NEs ($\mathcal{Z}$). For each type of NE $z \in \mathcal{Z}$, Algorithm 1 performs the categorization task (lines 2 to 6) to create the training dataset T_z and the training task (lines 7 to 10) to select the best algorithm $m_z \in M$. For each $z \in Z$ InP creates N_z instances. Each of the N_z instances is defined by f_z features according to its capabilities (line 2) and classified with $|L| - 1$ labels (Line 3). All the N_z instances are divided into b clusters (line 4). The relevant cluster number (1 to b) is added to each of the N_z instances as the $|L|^{th}$ label (line 5). The collection of all classified N_z instances with $|F|_z$ features constitute the T_z (line 6). For each algorithm $m \in M$ (line 7) we evaluate the E_M and H_S scores with T_Z and k-fold cross-validation (line 8), and select the algorithm with highest E_M and H_S as the most suitable m_z for prediction (line 10). The outputs of Algorithm 1 are T_z and m_z for each $z \in \mathcal{Z}$.

The inputs to Algorithm 2 are T_z, m_z, and the set of VNO requests Q. A VNO request $q \in Q$ demands a total of $\sum_{z\in Z} x_z$ NEs. Algorithm 2 performs three tasks for each request $q \in Q$: prediction (lines 3–5), recommendation (lines 5 to 10), and re-training (lines 11 to line 18). Algorithm 2 employs m_z and obtains predictions y'_{zr} for the x_z requested NEs of type z (lines 3 to 5), and recommends them to VNO (line 6). InP determines whether the y'_{zr}s yield a *perfect*

Table 2 Notations used in Algorithms 1 and 2

Notation	Definition
$\mathcal{P}$	Physical resource information
$\mathcal{Z}$	The set of types of NEs, e.g. $\mathcal{Z}$={VM, access point, bandwidth, gateway/router}
N_z	Number of created instances for NE type $z \in \mathcal{Z}$
p_z	Unit usage price for NE type $z \in \mathcal{Z}$
$\|F\|_z$	Associated number of features (including p_z) for NE type $z \in \mathcal{Z}$
y_{zi}	The correct classification vector for a training instance i of type z with $\|L\|$-1>1 labels and V>1 values per label
T_z	The training dataset in suitable format $\forall z \in \mathcal{Z}$
M	The set of available MTC and MVEN algorithms
m_z	The most suitable $m \in M$, used as classifier for a $z \in \mathcal{Z}$
Q	The set of VNO requests q.
x_z	Each $q \in Q$ demands for $x_z \geq 0$ NEs of type $z \in \mathcal{Z}$.
y'_{zr}	Predicated classification vector of a requested instance r of type $z \in q$, with $r \leq x_z$
c_{z1}	The number of times y'_{zi} causes over-allocation
c_{z2}	The number of times y'_{zi} causes under-allocation

Algorithm 1 Proactive categorization and training

Input: $\mathcal{P}$, M, $\mathcal{Z}$
Output: T_z, m_z $\forall z \in \mathcal{Z}$
1: **for** each $z \in \mathcal{Z}$ **do**
2: Assign f_z features to each instance $i, i = 1, 2, \cdots, N_z$.
3: Classify each instance i into y_{zi} with L-1 labels.
4: Create b clusters of all N_z instances.
5: Add cluster info as the Lth label each y_{zi}.
6: $T_z = \bigcup_{i=1}^{N_z} y_{zi} \cup a_z$
7: **for** $m \in M$ **do**
8: Train and evaluate E_M and H_S with T_z and k-fold cross-validation
9: **end for**
10: Select a m with highest E_M and H_S as m_z.
11: **end for**
12: **return** T_z, m_z

Algorithm 2 Prediction, recommendation, and re-training

Input: T_z, m_z, Q
Output: $y'_{zr} \forall q \in Q$, r=1, 2, $\cdots$, x_z, $z \in Z$, c_{z1}, c_{z2}, updated T_z, m_z

1: **for** each request $q \in Q$ **do**
2: **for** each $z \in q$ **do**
3: **for** $p = 1$ to x_z **do**
4: Employ m_z and obtain prediction y'_{zr}.
5: **end for**
6: Recommend predicted NEs y'_{zr}, $r = 1, 2, \cdots, x_z$.
7: Verify whether y'_{zr}s are equal to (*perfect*), or lower than (*under-allocation*), or higher than (*over-allocation*) the amounts requested in q.
8: **if** Predicted capabilities are either *perfect* or *over-allocation* **then**
9: VNO accepts recommended NEs
10: **end if**
11: Initialize $c_{z1} = c_{z2} = 0$
12: **if** Predicted capabilities yield *over-allocation* **then**
13: $c_{z1} = c_{z1} + 1$
14: **else if** Predicted capabilities yield *under-allocation* **then**
15: $c_{z2} = c_{z2} + 1$
16: **end if**
17: Update T_z with by adding correctly classified instances corresponding to c_{z1} and c_{z2}.
18: Re-train m_z with updated T_z for future use.
19: **end for**
20: **end for**
21: **return** y'_{xi}, $i = 1, 2, \cdots, x$, c_1, c_2, updated m_z and T_z

match, *over-allocation* or *under-allocation* (line 7). VNO accepts the recommended resources only if the predictions result in *perfect* or *over-allocation* (lines 8–10). InP also marks the predictions which result in *over-allocation* by c_{z1} and *under-allocation* by c_{z2} (lines 11–16). InP updates T_z by adding the correctly classified instances that corresponds to c_{z1} and c_{z2} (line 17). The algorithms m_z are also re-trained with the updated T_z (line 18).

5. Preparation of Training Dataset

We prepare of two training datasets for MTCAS, named D50 and D84, based on real VM allocation data referred and obtained from two sources [5] and [6], respectively. D50 and D84 consist of 50 and 84 classified VM instances, respectively. We prepare training data in "ARFF" [36], specific to WEKA [37] and MEKA [35] data mining software.

5.1 Preparing D50

For 1750 VMs, the information about provisioned and used amounts of resources (CPU, memory, storage, and network) with timestamps are available in [5], [38]. We randomly selected data for 50 VMs, and extracted the provisioned number of CPU cores, CPU speeds, allocated RAM, and network storage type (fast or slow). We added additional data to create training data in "ARFF" format with $|L| = 7$ labels (including "Label" string) and $|F| = 8$ features (including "Feature" string), as shown in Fig. 4. Note that, the three strings "relation", "attribute", and "data" are reserved for "ARFF" format.

A VM instance 2,6,SAN,1,1,2,4,4,11.43,64,F,10,1,3,0.75, as shown in Fig. 4, is interpreted as follows. The VM is classified as classes "2,6,SAN,1,1,2,4", indicating that it consists of $2^2 = 4$ CPU cores, $2^6 = 64$ GB memory (CPU core and memory requests for VM follow the power of 2 trend [5]), access to fast storage, and 10 Gbps NIC (type 1). The VM resides in a physical machine at location 1, spends high energy (class 2), belongs to optional cluster number 4. Three additional pieces of information about the VM are available from this instance. The four CPU cores have a total speed of 11.43 GHz. The VM is intended to use for scientific (type S) computation service. The last feature price-per-hour valued 0.75 indicates that a VNO should pay 0.75 currency units to use the requested resource.

```
@relation 'Virtual-Machine: -C 7'
@attribute Label1-CPU {0,1,2,3,4,5}
@attribute Label2-RAM {1,2,3,4,5,6}
@attribute Label3-STORAGE {SAN,NAS}
@attribute Label4-NIC {1,2,3}
@attribute Label5-LOCATION {1,2,3}
@attribute Label6-ENERGY {1,2,3}
@attribute Label7-Cluster {1,...,k}
@attribute Feature1-num-CPU numeric
@attribute Feature2-Speed-CPU numeric
@attribute Feature3-size-RAM numeric
@attribute Feature4-type-storage {F,S}
@attribute Feature5-num-NIC numeric
@attribute Feature6-num-LOCATION numeric
@attribute Feature7-type-service {S,M,C}
@attribute Feature8-price-per-hour numeric
@data
2,6,SAN,1,1,2,4,4,11.43,64,F,10,1,S,0.75
.........
```

Fig. 4 Training data in 'ARFF' with $|L| = 7$ labels and eight features.

A VM request must include required number of CPU cores, amount of memory, storage type, NIC bandwidth, and desired usage duration, which InP should satisfy quantitatively. The optional features, which a VNO may mention in a VM request, are CPU speed per core, desired VM location, service type, and willing to pay usage price per unit time. InP may override a location request made by VNO for the purpose of balancing resource utilization of various locations when available resources at the requested location are not sufficient to meet the demand. In this situation, InP updates the location related features in the training data and retrains the MTC algorithms so that the solution finds the best location for allocating VMs depending on the updated training data and resource availability. The first five labels are determined from the VM request. The remaining energy and cluster labels are specific to InP. InP knows the power usage of each NE, so it can allocate VMs so that total power usage of allocated resources does not exceed some threshold. Cluster feature is utilized by InP to allocate an alternative resource in case a predicted resource by classifier is unavailable.

In addition to a VM, a VNO requests for access points, gateway, and end-to-end link bandwidth to prepare a networking platform, that an ASP will use to provide intended services to the end users. Following above descriptions, we can describe access points with features like location,

capacity per user equipment (UE), total number of UE it can support, price, etc. We assume that access point ensures the requested end-to-end bandwidth up to gateway. The gateway node can be described with features like OS type, interfaces types, supported protocols type, location etc. The links between gateway node and cloud VM locations are divided into several virtual links of different capacities and categorized with features like available bandwidth, delay, physical link types (LAN or optical fiber cable), failure probability, usage price per unit bandwidth, etc.

5.2 Preparing D84

Following the same procedure described above, we prepare D84 in "ARFF" format with 16 features following the specifications mentioned in Amazon EC2 instance website [6]. A total of 84 VM instances with different configurations are specified in [6]. Among the 16 features, we select the most relevant three, seven, and twelve features and created corresponding labels $|L| = \{3, 7, 12\}$. For example, a classified vector $y_i = \{1, 1, 1\}$ ($|L| = 3$) indicates the "t2.nano" VM instance offered by Amazon Web Service. The remaining procedure for the preparation of D84 dataset is similar to that of D50 dataset.

6. Numerical Evaluations

In our evaluation, we consider that InP is a cloud provider and owns sufficient number of VMs to meet VNO requests. Therefore, a resource request response is rejected by VNO due to under-allocation, which does not meet QoS metrics. A VNO request includes the desired usage duration of requested VM. So VM migration is not considered here.

Our evaluation consists of two parts. First, we evaluate the MVEN algorithm in comparison with several available MTC algorithms in terms of E_M, H_S, and training time by varying the number of labels $|L| = \{3, 7, 12\}$ (i.e. number of QoS metrics related to requested VMs to be satisfied) with both D50 and D84 datasets, and select the most suitable MTC algorithm(s) with respect to accuracy and training time to use as classifier in our scheme. The six common QoS metrics with both datasets are number of CPU cores, RAM size, storage type, NIC speed, VM location, and the service type for which VM is used. The seventh QoS requirement with D50 dataset only is cluster number. The six additional QoS metrics considered with D84 dataset only are CPU generation, CPU capability for handling 256-bit/512-bit instruction sets and turbo boosting, need for graphical processing unit, field-programmable gate array (FPGA), and storage optimization. We denote the five different combination of dataset and number of labels used in our evaluations as D50-L3, D50-L7, D84-L3, D84-L7, and D84-L12, respectively. Second, we evaluate the prediction-based resource allocation performance of MTCAS, by using the selected classifier, with respect to over-allocation, under-allocation, and average prediction time for a request.

6.1 Selecting an MTC Algorithm as Classifier

Following the procedure of Algorithm 1, we evaluate our MVEN and nine MTC algorithms provided by the MEKA [35] software, namely Bayesian Classifier Chains (BCC), Classifier Chains (CC), Classifier Chains with probabilistic output (CCp), Class Relevance (CR), Multi-target Ensemble (ENS), Bootstrap Aggregating (BAG), Super Class Classifier (SCC), Disjoint Random k-label Pruned Sets (RAkELd), and Nearest Set Replacement (NSR). Note that BCC, CC, CCp, and CR are implemented based on classifier chain method and RAkELd, NSR, and SCC are based on label powerset method. ENS and BAG use AdaboostM1 and bagging ensemble methods, respectively, on the CC algorithm.

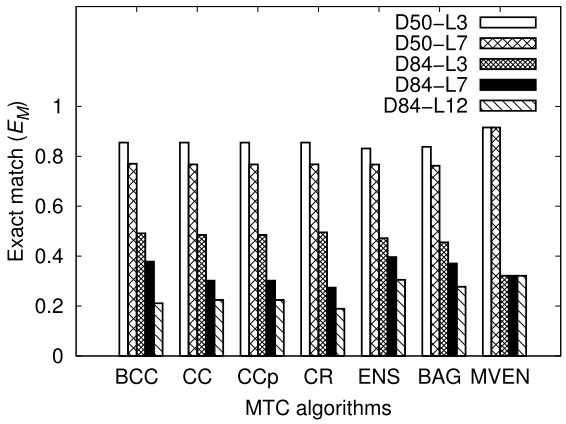
We train each of MTC algorithm with both D50 and D84 by using 10-fold cross-validation and default parameter settings in MEKA. The MVEN algorithm is implemented by using the WEKA software, where an ensemble of five algorithms are simultaneously executed with $|L|$ created datasets as explained in Sect. 4.3. The E_M and H_S scores, and training times of six MTC algorithms and MVEN, for the five dataset-$|L|$ combinations, are summarized in Figs. 5 (a), (b), and (c), respectively. We exclude the performance results of the SCC, NSR, and RAkELd algorithms in this paper, because their performance in terms of E_M and H_S are very poor. Though their training times are faster than the six other evaluated MTC algorithms, poor E_M and H_S results inhibit this advantage. The results presented in Fig. 5 are averaged over 100 evaluations by changing the random number seed between 1 to 100[†]. The decision tree C4.5 algorithm is used as the common base classifier for all MTC algorithms in our evaluation[††].

For each MTC algorithm, prediction accuracy in terms of E_M decreases (Fig. 5 (a)) and that of H_S is equal or slightly increases (Fig. 5 (b)) with the increasing number of QoS metrics (increasing value of $|L|$) to be fulfilled by InP for VNO requests. The H_S trend for our MVEN algorithm is similar to that of MTC algorithms. However, E_M scores and training times do not change for MVEN for both D50 and D84. The average training time (in ms) for each evaluated algorithm, which increase with $|L|$ for both datasets, are summarized in Fig. 5 (c). In order to evaluate the training time, the algorithms are executed in a computer equipped with Intel® Core i7-5930K CPU (3.50GHz per core), MEKA software version 1.9.1, WEKA software version 3.8.2, Windows 10 operating system, CygWin, JDK 10.0.2 64 bits.

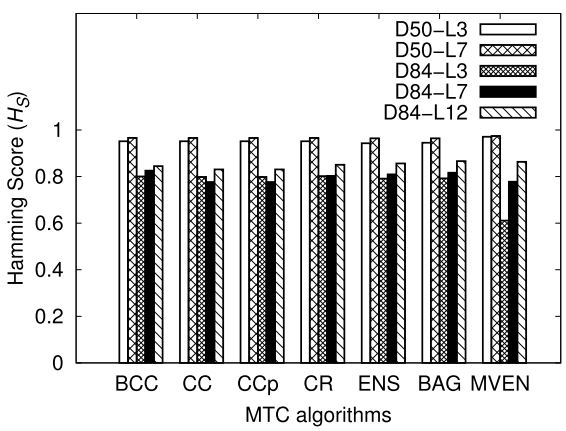
Our observations and interpretations from the results of Fig. 5 are as follows:

[†]The random number input in WEKA/MEKA enables to re-produce the same result, therefore it is suggested by WEKA/MEKA developers to average the results over several random number seed values.

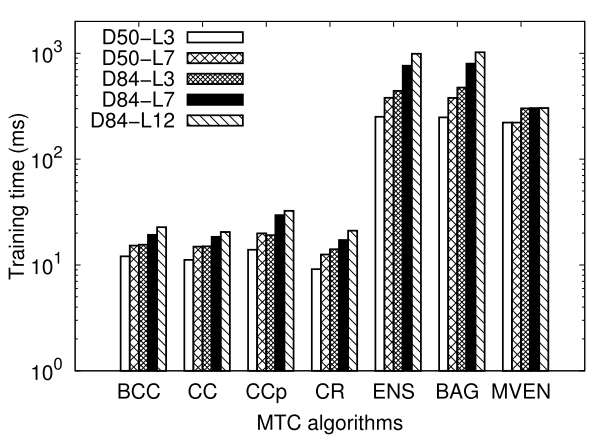
[††]In WEKA/MEKA, the Java implementation of C4.5 algorithm is denoted as J48.



(a) E_M decreases with increasing $|L|$.



(b) H_S is equal or slightly increases with increasing $|L|$.



(c) Training time increase with increasing $|L|$.

Fig. 5 E_M, H_S, and training time performances of the evaluated MTC algorithms by changing $|L|$ with D50 and D84 datasets.

- In terms of E_M (0.92) and H_S (0.97) our MVEN algorithm outperforms all MTC algorithms for the D50-L3 and D50-L7 situations, which are at least 2% and 6% higher as shown in Fig. 5 (a) and (b), respectively.
- There are more than one suitable MTC algorithm to be used as classifier in MTCAS when the offered QoS level is low ($|L| = 3, 7$). In terms of E_M and H_S, four algorithms (BCC, CC, CCp and CR) perform the same with D50-L3 (E_M=0.86 and H_S = 0.95), D50-L7 (E_M = 0.77 and H_S = 0.97), and D84-L3 (E_M = 0.49 and H_S = 0.8).
- As the offered QoS level is increased ($|L| = 7$), the BCC algorithm outperforms all other algorithms with D84-L7 (E_M = 0.38 and H_S = 0.83).
- At the highest offered QoS level ($|L| = 12$) with D84-L12 situation, MVEN achieves the highest E_M (0.32) score (Fig. 5 (a)). It achieves comparable H_S (0.86) performances when compared with the BAG and ENS (shown in Fig. 5 (b)) at the cost of 70% less training time than that of with ENS and BAG (shown in Fig. 5 (c)). With D84-L12, the BCC algorithm performs good with H_S (0.84) but poor with E_M (0.21).
- In our evaluation we observed that around 100 iterations are required for both BAG and ENS to reach a maximum E_M and H_S. So, we executed BAG and ENS with 50, 100, and 150 iterations, and choose the highest E_M and H_S scores to represent in Figs. 5 (a) and (b) respectively. Note that, the training times for ENS and BAG in Fig. 5 (c) are the sum of these three executions.
- For InPs H_S is a better choice to offer QoS guarantee to the customer VNOs. According to our evaluation, InP can offer complete QoS satisfaction for 80% to 97% of resource requests in terms of H_S. Similar prediction accuracy (85% to 90%) for virtual resource allocation was also reported in [15].
- We observe form Fig. 5 (c) that the classifier chain methods (BCC, CC, CCp, CR) are faster to train than the ensemble methods (MVEN, ENS, and BAG). The CR algorithm has the fastest training time among the six MTC algorithms.

6.2 Resource Over-Allocation and Under-Allocation

The predictive performance of MTCAS is evaluated, following the procedure of Algorithm 2, in terms of over-allocation and under-allocation. We generated 50 random VM resource requests with necessary number of CPU cores, memory size, storage type, network interface speed, desired location, and service type. We use two classifiers trained with 50% and 100% of D50 data with cross-validation. Note that the new requests do not match exactly with the classified requests in D50. The results are summarized in Table 3, where, $C_{xx\%}$ denotes a classifier trained with xx% of D50 dataset, R_r be the total amount of requested and R_p is the total amount of predicted functional resources, c_1 be the number of over-allocated and c_2 be the number of under-allocated request instances. Percentage of resource over-allocation $R_{OA}\%$ is defined by $R_{OA}\% = \frac{R_p - R_r}{R_r} \times 100$ with $R_p \geq R_r$.

Resource allocation with the classifier, trained with 50% of D50 (denoted as $C_{50\%}$ in Table 3), results in total 33.8% CPU core, 52.14% memory, and 40.6% NIC bandwidth over-allocation. Out of 50 predicted resource

Table 3 Summary of CPU, memory, and NIC resource allocation to 50 new random VM requests.

		$C_{50\%}$				$C_{100\%}$			
VM component	R_r	R_p	c_1	c_2	$R_{OA}\%$	R_p	c_1	c_2	$R_{OA}\%$
# of CPU cores	497	665	18	0	33.80%	497	0	0	0
RAM (GB)	1003	1526	33	8	52.14%	1014	11	0	1.1%
NIC (Gbps)	266	374	16	4	40.6%	374	16	4	40.6%

request classes, CPU cores, memory and NIC bandwidth are over-allocated in 18, 33, and 16 instances, and under-allocated in 0, 8, and 4 instances, respectively. In contrast, resource allocation with the classifier, trained with 100% of D50 (denoted as $C_{100\%}$ in Table 3), results in total 0% CPU core, 1.1% memory, and 40.6% NIC bandwidth over-allocations. Out of 50 predicted resource request classes, CPU cores, memory and NIC bandwidth are over-allocated in 0, 11, and 16 instances respectively, and only NIC bandwidth resource are under-allocated for 4 instances. We observe that resource over-allocation for CPU and memory are reduced by 33.8% and 51.04%, respectively, as the VM classifier with the BCC algorithm is trained with more data (100% of D50). Resource request for location and storage are correctly classified in all the requests, which are not mentioned in Table 3. However, 30 instances are not correctly classified for service types by $C_{50\%}$, which are correctly classified with $C_{100\%}$.

Note that in four prediction instances NIC bandwidth is under-allocated, resulting in four resource request rejection by VNO. So, our evaluation experiences a rejection rate of $\frac{4\times100}{50} = 8\%$.

Our scheme regularly updates the training data by adding correct classification data for the over-allocated and under-allocated resource requests history, indicated by c_1 and c_2 in Table 3. The classifiers are also regularly re-trained with the update training data. Therefore, prediction accuracy is certain to increase with time, which in turn improves multiple QoS satisfaction rate and further reduces resource over-allocation.

6.3 Balancing Resource Utilization

We consider that InP owns physical resources in three geographical locations, which are indicated by the location feature of D50 training dataset. Let us consider that InP always wants to keep 5% of computation and storage resources free for satisfying urgent requests at each location. For the next 50 requests, according to the current resource availability situations at these three locations, InP decides to allocate 15, 14, and 21 VMs from location 1, location 2, and location 3, respectively. We prepare our training data with D50 accordingly. The predictions by $C_{100\%}$ suggest InP to allocate 16, 12, and 22 VMs from location 1, location 2, and location 3, respectively. Thus, the predictions almost achieve the desired number of VM allocations, although an extra VM is allocated from both location 1 and location 3.

Table 4 Average prediction time in μsec. for one VM request.

Algorithm	D50-L3	D50-L7	D84-L3	D84-L7	D84-L12
BCC	5.8	11	7.3	11.1	24.5
MVEN	328	328	281	327.4	327.4

6.4 Prediction Times

The entire virtual network provisioning time is influenced by three issues: prediction time, provisioning time, and transfer time. Prediction time is the time required by MTCAS, on behalf of InP, to determine physical resource locations and amounts for the requested NEs. The time needed by the InP to create and activate the requested NEs in the substrate network is denoted as the provisioning time. The time required to transfer the entire VN usage right from VNO to ASP for the duration of lease is known as the transfer time. For example, if an InP is employing the OpenStack platform, the prediction and provisioning times for a requested VM are about 150 ms and 4.6 s, respectively [41]. It is also reported that when InP uses the OpenNebula (KVM based) cloud platform, the prediction and provisioning times are 200 ms and 2.5 s, respectively [41]. This implies that prediction time should be in the range of 3.26% - 8% of the provisioning time. The VN usage right transfer time from VNO to ASP can be considered equal to the network latency in 5G networks, which is about 10 ms [42]. For machine learning classifier agents, less than 10 ms of prediction time has been reported in [43]. Therefore, the prediction time in the NE allocation decision should be much less than 10 ms.

In order to show that MTCAS is suitable to handle delay sensitive requests, we evaluate average prediction time required to classify a VNO request for the BCC and our presented MVEN classifiers algorithms in MTCAS. The MVEN classifiers are executed in a aforementioned PC equipped with Intel® Core i7-5930K CPU (3.5 GHz per core), 64GB RAM, Windows 10 OS, and WEKA software (version 3.8.2). Being very small, we cannot measure the time for the BCC classifiers in the same environment as MEKA. Therefore, we execute the BCC classifier in a VM (on the same PC as MVEN) with one 3.50GHz processor, 4GB RAM, Linux OS, and MEKA software (version 1.9.1). The BCC classifiers employ the C4.5 algorithm as base classifier. We train the classifiers with 10-fold cross-validation with both D50 and D84 datasets. The results are summarized in Table 4. We obtained prediction times in the microseconds (μsec) range for one VM request, which indicates that MTCAS can be suitably adopted to the Clipper environment [26], or for applications like network slicing with

LTE [44].

7. Conclusion

We have presented MTCAS, a proactive and adaptive scheme based on MTC algorithms, for InP to allocate multiple QoS compliant set of virtual resources to VNO requests. We evaluate the MVEN and nine MTC algorithms in terms of exact match, Hamming score, and training time. We have found that MTCAS should adopt either BCC or CR algorithm for delay-sensitive requests with lower QoS metrics, ensemble algorithms (ENS and BAG) for delay-invariant requests with higher QoS metrics, and MVEN algorithms with higher QoS metrics and moderate delay-sensitivity. Prediction based resource allocation performance of MTCAS with the BCC algorithm has been evaluated with respect to new request prediction time, resource over-allocation, and request rejection. We achieved average prediction times within 5.8 to 24.5 microseconds range for the BCC and 281 to 328 microseconds for the MVEN algorithm. We obtained about 33.8% and 51.04% reductions in CPU and memory resources, respectively, for virtual machines by using classifiers trained with complete training data, and 8% request rejection by VNO in our evaluation. Our scheme regularly updates training data based on resource request history and regularly retrains classifiers with updated training data. Therefore, predictive performance is certain to improve over time.

In future, we will investigate MTCAS issues and policies regarding the determination of minimum necessary amount of training data for classifiers and simultaneous allocation of multiple NEs (e.g. VM, gateway, access points, and end-to-end bandwidth demanded in a VNO request) in order to achieve location-aware and load-balanced NE allocation, and keep energy consumption under a threshold.

References

[1] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J.J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," IEEE Commun. Mag., vol.55, no.5, pp.80–87, May 2017.

[2] K. Fujikawa, V.P. Kafle, P. Martinez-Julia, A.H.A. Muktadir, and H. Harai, "Automatic construction of name-bound virtual networks for IoT," Proc. 41st IEEE COMPSAC, pp.529–537, July 2017.

[3] J. Simão and L. Veiga, "Partial utility-driven scheduling for flexible SLA and pricing arbitration in clouds," IEEE Trans. Cloud Comput., vol.4, no.4, pp.467–480, Oct. 2016.

[4] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through Fog micro datacenter," Proc. IEEE PerCOM, pp.105–110, March 2015.

[5] S. Shen, V.V. Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," Proc. 15th IEEE/ACM CCGrid, pp.465–474, May 2015.

[6] Amazon Elastic Compute Cloud (EC2) Instance Types, https://aws.amazon.com/ec2/instance-types/.

[7] Google compute engine, https://cloud.google.com/compute/.

[8] S. Shen, K. Deng, A. Iosup, and D. Epema, Scheduling Jobs in the Cloud Using On-Demand and Reserved Instances, pp.242–254, Springer: Berlin Heidelberg, 2013.

[9] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," IEEE Trans. Netw. Service Manag., vol.10, no.4, pp.356–368, Dec. 2013.

[10] R. Yu, G. Xue, and X. Zhang, "Qos-aware and reliable traffic steering for service function chaining in mobile networks," IEEE J. Sel. Areas Commun., vol.35, no.11, pp.2522–2531, Nov. 2017.

[11] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," IEEE Commun. Lett., vol.16, no.5, pp.756–759, May 2012.

[12] T. Miyazawa and H. Harai, "Supervised learning based automatic adaptation of virtualized resource selection policy," Proc. 17th IEEE Networks, pp.170–175, Sept. 2016.

[13] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F.D. Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," Proc. IEEE NOMS, pp.1–9, May 2014.

[14] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling VNFs using machine learning to improve QoS and reduce cost," Proc. IEEE ICC, pp.1–6, May 2018.

[15] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "Topology-aware prediction of virtual network function resource requirements," IEEE Trans. Netw. Service Manag., vol.14, no.1, pp.106–120, March 2017.

[16] Y. Jiang, C.-S. Perng, T. Li, and R.N. Chang, "Cloud analytics for capacity planning and instant VM provisioning," IEEE Trans. Netw. Service Manag., vol.10, no.3, pp.312–325, Sept. 2013.

[17] G. Zhang, X. Zhu, W. Bao, H. Yan, and D. Tan, "Local storage-based consolidation with resource demand prediction and live migration in clouds," IEEE Access, vol.6, pp.26,854–26,865, April 2018.

[18] J.B. Wang, J. Wang, Y. Wu, J.Y. Wang, H. Zhu, M. Lin, and J. Wang, "A machine learning framework for resource allocation assisted by cloud computing," IEEE Netw., vol.32, no.2, pp.144–151, March 2018.

[19] L. Xu, Z. Zeng, and X. Ye, "Multi-objective optimization based virtual resource allocation strategy for cloud computing," Proc. 11th IEEE/ACIS ICIS, pp.56–61, May 2012.

[20] N. Kato, Z.M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," IEEE Wireless Commun., vol.24, no.3, pp.146–153, June 2017.

[21] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," IEEE Wireless Commun., vol.24, no.2, pp.98–105, April 2017.

[22] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When cellular networks meet artificial intelligence," IEEE Wireless Commun., vol.24, no.5, pp.175–183, Oct. 2017.

[23] J. Luo, W. Song, and L. Yin, "Reliable virtual machine placement based on multi-objective optimization with traffic-aware algorithm in industrial cloud," IEEE Access, vol.6, pp.23,043–23,052, March 2018.

[24] H.A. Alameddine, S. Sebbah, and C. Assi, "On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach," IEEE Trans. Netw. and Service Manag., vol.14, no.4, pp.860–874, Dec. 2017.

[25] Y. Xu and V.P. Kafle, "A Delay-Aware Service Function Chain Placement Scheme Based on Dynamic Programming," Proc. IEEE LANMAN, pp.110–111, June 2018.

[26] D. Crankshaw, X. Wang, G. Zhou, M.J. Franklin, J.E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," Proc. 14th USENIX NSDI, pp.613–627, April 2017.

[27] L. Chen and H. Shen, "Considering resource demand misalignments to reduce resource over-provisioning in cloud datacenters," Proc. IEEE INFOCOM, pp.1–9, May 2017.

[28] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke, "Virtual resource description and clustering for virtual network discovery," Proc. IEEE ICC Workshops, pp.1–6, June 2009.

[29] Y. Wang, Q. He, X. Zhang, D. Ye, and Y. Yang, "Efficient QoS-aware

service recommendation for multi-tenant service-based systems in cloud," IEEE Trans. Services Comput., Online, Oct. 2017.
[30] J. Read, "Scalable multi-label classification," Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, Sept. 2010.
[31] C. Bielza, G. Li, and P. Larrañaga, "Multi-dimensional classification with Bayesian networks," Int. J. Approx. Reasoning, vol.52, no.6, pp.705–727, Sept. 2011.
[32] R. Polikar, "Ensemble based systems in decision making," IEEE Circuits Syst. Mag., vol.6, no.3, pp.21–45, Sept. 2006.
[33] R. Polikar, "Ensemble learning," http://www.scholarpedia.org/article/Ensemble_learning/.
[34] Classifier of WEKA, http://weka.sourceforge.net/doc.dev/weka/classifiers/Classifier.html.
[35] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "MEKA: A multi-label/multi-target extension to WEKA," J Mach. Learning Research, vol.17, no.21, pp.1–5, 2016.
[36] ARFF Format, https://www.cs.waikato.ac.nz/ml/weka/arff.html.
[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The WEKA data mining software: an update," SIGKDD Explor. Newsl., vol.11, no.1, pp.10–18, Nov. 2009.
[38] The Grid Workloads Archive, http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.
[39] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-Provider Virtual Network Embedding With Limited Information Disclosure," IEEE Trans. Netw. and Service Manag., vol.12, no.2, pp.188–201, June 2015.
[40] J.C. Moore, H.R. Rao, and A.B. Whinston, "Multi-agent resource allocation: an incomplete information perspective," IEEE Trans. Syst., Man, Cybern., vol.24, no.8, pp.1208–1219, Aug. 1994.
[41] G. Carrozza, L. Battaglia, V. Manetti, A. Marotta, R. Canonico, and S. Avallone, "On the Evaluation of VM Provisioning Time in Cloud Platforms for Mission-Critical Infrastructures," Proc. 14th IEEE/ACM CCGrid Symp., Chicago, IL, pp.802–810, 2014.
[42] ITU-R Rec. M.2083-0, "IMT Vision Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond," Sept. 2015.
[43] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," IEEE Netw., vol.32, no.2, pp.92–99, March 2018.
[44] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5G communications: Slicing the LTE network," IEEE Commun. Mag., vol.55, no.8, pp.146–154, Aug. 2017.

Abu Hena Al Muktadir received the B.Sc. (Honors) and M.Sc. degrees from the University of Rajshahi, Bangladesh in 2004 and 2005 respectively. He was awarded Ph.D. degree on September, 2014 from The University of Electro-Communications, Tokyo, Japan. From 2007 to 2009, he served as a Lecturer at Daffodil International University, Bangladesh. His research focuses on machine learning, game theory, virtual resource management, IP networks, routing protocols, network coding, and IoT. From November, 2014 he is working as a Researcher in the National Institute of Information and Communications Technology (NICT), Japan. He is a senior member of IEEE.

Takaya Miyazawa received his Ph.D. degree in information and computer science from Keio University, Yokohama, Japan, in 2006. From 2006 to 2007, he was a visiting scholar at University of California, Davis, USA. In 2007, he joined National Institute of Information and Communications Technology (NICT), Tokyo, Japan. His research interest includes network control, network virtualization, SDN/NFV and service function chaining. He received the Hiroshi Ando Memorial Young Engineering Award in 2007 and the Funai Young Researcher Award in 2010. He is a member of IEEE.

Pedro Martinez-Julia received the Ph.D. in Computer Science and the M.S. in Advanced Information Technology and Telematics from the University of Murcia, and the B.S. degree in Computer Science from the Open University of Catalonia. He is currently a full-time researcher at the National Institute of Information and Communications Technology (NICT). His main expertise is in network architecture, control and management, with particular interest in overlay networks and distributed systems and services. He has been involved in EU-funded research projects since 2009, leading several tasks/activities and participating as researcher in others. He has presented several papers in international contributed and has several publications in international journals. He is member of ACM and IEEE.

Hiroaki Harai received M.E. and Ph.D. degrees in information and computer sciences from Osaka University, Japan in 1995 and 1998, respectively. He is currently a Director General at the National Institute of Information and Communications Technology (NICT), Tokyo, Japan, where he is leading R&D on innovation of ICT networks. He is concurrently a visiting associate professor at the Japan Advanced Institute of Science Technology, Ishikawa, Japan. His current research topic is the design and development of new generation network architecture. Dr. Harai was named an Outstanding Young Researcher at the 3rd IEEE ComSoc Asia-Pacific Young Researcher Award, 2007. He received the 2009 Young Researcher Award from the Ministry of Education, Culture, Sports, Science and Technology. He is a member of IEEE.

Ved P. Kafle is a Research Manager at the National Institute of Information and Communications Technology (NICT), and holds a concurrent visiting associate professors position at the University of Electro-Communications, Tokyo. He is also serving as a co-rapporteur of ITU-T Study Group 13s Question 15. His research interests include future networks, naming and addressing, ID/locator separation, machine-to-machine communication, Internet of Things (IoT), distributed mobility management, and privacy, security and trust in communication networks. He received the ITU Association of Japan Award in 2009 and two Best Paper Awards (second prize) at the ITU Kaleidoscope Academic Conferences in 2009 and 2014. He holds the M.S. degree from Seoul National University and the Ph.D. from the Graduate University of Advanced Studies. He is a senior member of IEEE.