CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2016; **28**:4429–4442
Published online 28 January 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3767

Dynamic resource demand prediction and allocation in multi-tenant service clouds‡

Manish Verma[1], G. R. Gangadharan[2,*,†], Nanjangud C. Narendra[3], Ravi Vadlamani[2], Vidyadhar Inamdar[1] Lakshmi Ramachandran[4], Rodrigo N. Calheiros[5] and Rajkumar Buyya[5]

[1]*University of Hyderabad, India*
[2]*IDRBT, Hyderabad, India*
[3]*Ericsson Research, Bangalore, India*
[4]*North Carolina state University, Raleigh, USA*
[5]*The University of Melbourne, Melbourne, Australia*

SUMMARY

Cloud computing is emerging as an increasingly popular computing paradigm, allowing dynamic scaling of resources available to users as needed. This requires a highly accurate demand prediction and resource allocation methodology that can provision resources in advance, thereby minimizing the virtual machine downtime required for resource provisioning. In this paper, we present a dynamic resource demand prediction and allocation framework in multi-tenant service clouds. The novel contribution of our proposed framework is that it classifies the service tenants as per whether their resource requirements would increase or not; based on this classification, our framework prioritizes prediction for those service tenants in which resource demand would increase, thereby minimizing the time needed for prediction. Furthermore, our approach adds the service tenants to matched virtual machines and allocates the virtual machines to physical host machines using a best-fit heuristic approach. Performance results demonstrate how our best-fit heuristic approach could efficiently allocate virtual machines to hosts so that the hosts are utilized to their fullest capacity.

Received 9 February 2015; Revised 6 June 2015; Accepted 13 December 2015

KEY WORDS: service tenants; prediction; time series; dynamic resource allocation; VM placement

1. INTRODUCTION

In multi-tenant cloud-based systems, multiple requests are served concurrently by one or more instances of a hosted application based on a shared hardware and software infrastructure [1]. Each tenant can interact with the service application as if it was the user, and the tenant cannot access or view the data of another tenant [2]. This is typically implemented through virtual machines (VMs) running on physical servers, which help allocate the resources to service tenants. Static resource allocation limit consolidation and increase the cost of running the services; hence, dynamic resource allocation methods are preferred. However, the benefits of dynamic resource allocation are offset by the time taken to shut down and restart VMs for resource allocation/re-allocation. Hence minimizing this downtime becomes crucial in dynamic resource allocation [3, 4]. This, in turn, introduces the need for accurate resource demand prediction.

In order to predict the future demand accurately, historical data of past demand profiles/patterns of service tenants needs to be leveraged. This introduces the challenge of periodically updating the

*Correspondence to: G. R. Gangadharan, IDRBT, Hyderabad, India.
†E-mail: geeyaar@gmail.com/grgangadharan@idrbt.ac.in
‡This paper is a revised and extended version of our earlier conference paper [29].

prediction framework as the demand profiles/patterns of service tenants keep changing. Apart from creating and provisioning functionalities required by a service tenant, a service provider must be able to add service tenants to a specified VM and to allocate a VM to the host efficiently, i.e. ensuring that the capacity of the host is utilized effectively and with minimum wastage. The amount of resources that a service tenant needs to carry out its operations varies depending on factors such as type of operations performed, time of day during which the operation is performed, and load that the service tenant is experiencing at a certain point in time. Thus, the resource requirements of a service tenant are likely to change. This is an important consideration while handling resource allocation for a single service tenant as well as across multiple service tenants.

Our proposed dynamic resource prediction and allocation methodology uses a tenant provider component [3] to monitor the resource requirements of service tenants during the execution of their instances. Our methodology follows a two-step process, the first of which involves deciding if resources could be allocated to a service tenant and the second involves making the allocation. The main contributions of our work are:

- A novel approach to predict the future resource demand extracting high level characteristics from service tenants' historical data and classifying service tenants depending on the criteria of increase or decrease of resource demand (thereby reducing execution time for prediction); and
- A dynamic resource allocation approach adding the service tenants to matched VMs and allocating the VM into physical host machine using the best-fit decreasing heuristic approach [26].

This paper is a revised and extended version of our earlier conference paper [29]. We have enhanced our previous work by integrating the prediction concepts [29] with a novel dynamic resource allocation methodology. The proposed prediction and resource allocation framework is updated and refined (see Section 4). Experimental evaluation on prediction is performed on a large-scale cloud system with 10 000 service tenants (see Section 5). A novel dynamic resource allocation approach using best-fit decreasing heuristic algorithm and performance evaluation via simulation is proposed in detail in Section 6.

The rest of the paper is organized as follows. Section 2 compares our paper against related work. Section 3 discusses service tenant-related characteristics that influence prediction and resource allocation. Section 4 describes our proposed prediction and dynamic resource allocation framework. Section 5 presents prediction experimental evaluation using various machine learning techniques. Section 6 presents our proposed dynamic resource allocation methodology and performance evaluation, followed by concluding remarks in Section 7.

2. RELATED WORK

Demand prediction has always been important for planning and operational decisions. Jiang et al. [5] propose a VM provisioning system that uses aggregated time series considering VM type and request time. In our proposed framework, we apply time series for the prediction of VM demands considering several service tenant-related characteristics including resource utilization, functional priority, functional size, complexity index, activity period, and state information. Gong et al. [6] present an approach for online prediction of dynamic application resource requirements using lightweight signal processing and statistical learning algorithms. In contrast, our system makes predictions for both immediate and long-term demand. Besides, Gong et al. [6] have not considered response time characteristics.

Reig and Guitard [7] combine statistical and machine learning techniques to predict the immediate and long-term resource demand of web applications. Islam et al. [8] develop a set of resource provisioning strategies using neural network and linear regression techniques to satisfy resource demand. Prevost et al. [9] apply neural network and autoregressive linear prediction algorithms to forecast loads in Cloud data center applications. Zhang et al. [10] present a resource management method that predicts the changing workload of VMs using exponential smoothing techniques. In contrast to these previous works, our proposed framework predicts the service tenant requirement based on short-term and long-term service tenant demand behavior, using time series and polynomial regression.

DOI: 10.1002/cpe

Dynamic resource allocation in cloud computing is a challenging problem [11–15]. Warneke and Kao [11] described Nephele, a data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for task scheduling and execution. Yazir et al. [12] present an approach for dynamic autonomous resource management using distributed multiple criteria decision analysis with the PROMETHEE method. Chang et al. [13] presented an approximation algorithm with bounded ratio for optimal resource allocation in cloud environment. Inomata et al. [14] proposed a dynamic resource allocation method based on the load of VMs on IaaS that enables users to dynamically add and/or delete one or more instances on the basis of the load and the conditions specified by the user. Calcavecchia et al. [15] present a practical model of VM placement under a stream of requests and describe the past demand behavior of a VM to a candidate target host using backward speculative placement technique. In contrast to the above works, our proposed resource allocation framework implements dynamic resource allocation based on activity period and checks the conditions for existing running service tenants using service tenant characteristics and continuous resource utilization factors. In our proposal, we apply prediction results in resource allocation for provisioning of resource requirements of service tenants and for preparing the correct type of VMs in advance.

Nejad et al. [16] formulate the VM provisioning and allocation problem in clouds as an integer programming problem and design truthful greedy mechanisms to solve it. Zhu and Agrawal [17] describe a multi-input-multi-output feedback control based dynamic resource provisioning algorithm that adopts reinforcement learning to adjust adaptive parameters to guarantee the optimal application benefit within the time constraint. In contrast to these works, in our proposed dynamic resource allocation approach, we apply the best-fit decreasing heuristic algorithm. A best-fit decreasing heuristic algorithm sorts the items in decreasing order and places the next item into that bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, it starts a new bin. Thus, significant improvements are possible in solving bin packing problems. A best-fit decreasing heuristics algorithm uses no more 11/9 OPT+1 bins (where OPT is the number of bins given by the optimal solution) [26], thereby reducing the running time of our proposed resource allocation approach.

Espadaz et al. [18] describe formal measurements for under and over provisioning of virtual resources in Cloud infrastructure, specially for SaaS platform deployment and propose a resource allocation model to deploy SaaS applications over Cloud computing platforms by taking into account their multi-tenancy, thus creating a cost-effective scalable environment. Zhuang et al. [19] propose a model for simulating the process of tenants seeking better performing instances in the cloud. Xiao et al. [20] describe a method of enhancing the resource utilization of a server by minimizing the unevenness in the utilization of a server under multidimensional resource constraints. Comparing to these approaches, our proposed framework dynamically allocates resources based on service tenant characteristics and historical and future utilization behavior in a multi-tenant service Cloud environment.

3. MODELING SERVICE TENANT-RELATED COMPONENTS AND CHARACTERISTICS

We define a (service) tenant as a Software as a Service (SaaS) application instantiated to fulfill a client request. An important requirement for a SaaS application is the support of multiple tenants [3], which enables multiple services to share access to common resources. Tenant instances may not be required to be operational at all times of the day by their clients. Hence, depending on when a tenant is operational and when it is not, the state of a tenant changes and the resource requirements of a tenant are different when it is in different states.

3.1. Tenant requirements and provider models and tenant state information

The tenant requirements model (TRM) and tenant provider model (TPM) represent the behavior and constraints of tenants and providers. These models maintain state information (*created*, *active*, and *paused*) that plays a significant role in the resource allocation process. A TRM constitutes behavior,

DOI: 10.1002/cpe

constraints, and state information of a tenant and represents a tenant's functional requirements, containing inputs and outputs. A TPM contains tenant-active constraints (specifying the periods during which a service tenant is active) and state information for all existing tenants hosted by the provider.

Tenant state information is maintained in order to determine if a tenant was active at a certain point in time. A tenant could be in any one of the *created*, *active*, or *paused* state at any point. When a tenant is created after the dynamic provisioning phase is complete, it is said to be in the *created* state. During this state the tenant seeks resource allocation, so that it can transition into the *active* state, provided it was supposed to be active at that point. If a tenant was not required to be active during a certain time period, it is placed in the *paused* state and its resources de-allocated. It is evident that the transition between states takes place under strict conditions. For example, there might arise issues associated with the availability of resources or even lack of knowledge of a tenant's resource requirements. Hence, in order to guide the process of resource allocation, we identify the characteristics that are likely to influence allocation.

3.2. Service tenant-related characteristics

Let us consider that there are n service tenants and each service tenant contains any number of functions f. A service tenant ti is represented as $ti \Rightarrow fj$ (*set of input*, *set of output*) where $i = 1,..,n$ and $j = 1,..,k$. The characteristics of service tenants that are likely to influence the process of prediction and resource allocation are:

Functional Size (FS): Functional size, specified by the tenant service provider, represents the number of functionalities that a service tenant contains. Generally, a service tenant having more functions will require more resources.

Functional Priority (FP): Functional priority represents the degree of significance given to a service tenant over other tenants. This priority is specified by the client, which could range between 1 and 10 (where 1 indicates the lowest priority level).

S*tate Information (SI):* State information represents the state of a service tenant that can assume one of the following: {*created*, *active*, *paused*}.

Activity Period (AP): Activity period indicates the time period in which a service tenant is active and is defined either in terms of a specific period (range) of a day or number of hours.

Initial Resource Requirement (IR): The number of resources that are required by a service tenant at the start of its function is specified by initial resource requirement. In our approach, CPU resources are calculated in terms of the number of cores used, and memory is calculated in terms of the number of gigabytes used.

Resource Utilization (RU): Resource utilization represents the past history of resources utilized by a service tenant that is already in active state. It is specified in terms of percentage of utilization of CPU and other resources.

4. PREDICTION AND RESOURCE ALLOCATION FRAMEWORK

Based on the characteristics of service tenants, the resource requirement $R(t_i)$ of a tenant is represented as a function of the following characteristics:

$$R(ti) = \{FS(ti), FP(ti), SI(ti), AP(ti), IR(ti), RU(ti)\}.$$

Given the past behavior of resource requirements of tenants, we develop a classification model to classify the service tenants in order to understand whether their resource requirements are increasing or decreasing. We integrate the classification model with our resource allocation approach to

DOI: 10.1002/cpe

identify if resources can be allocated or de-allocated to a service tenant. Our work focuses on allocation of resources after the service tenant has been composed (before being provisioned) and during its execution.

Figure 1 presents our proposed prediction and resource allocation framework which comprises the following components:

(1) *Data Pre-processor*: Data pre-processor processes the historical data by filtering out unnecessary information from raw data and removes inconsistencies.
(2) *Trainer*, *Tester*, *and Validator*: This component trains and tests the pre-processed data of service tenant characteristics. Several classification techniques [21–23] are applied to determine the optimal classification technique. We follow a binary classification problem that determines whether in future the resource demand for a specific service tenant will increase or not.
(3) *Demand Predictive Classification Model*: The demand predictive classification model classifies the service tenants based on whether resource requirement would increase or not, by applying the optimal classification algorithm. This model is periodically updated as the service tenants dynamically change their resource requirements. Based on the classification result, prediction techniques determine (short term or long term) the future resource requirements of those service tenants for which resource requirement demand is expected to increase.
(4) *Short-term Forecasting Model*: Short-term forecasting model predicts the resource demand in those service tenants where the future resource requirement demand is expected to increase in the near future (e.g. in the next hour). We apply exponential moving average and trend seasonality model (time-series techniques) for short-term prediction [24].
(5) *Long-term Forecasting Model*: Long-term forecasting model predicts the resource demand for those service tenants in which the future resource demand is expected to increase, say for next week or for next month, based on weekly or monthly resource requirements. We apply

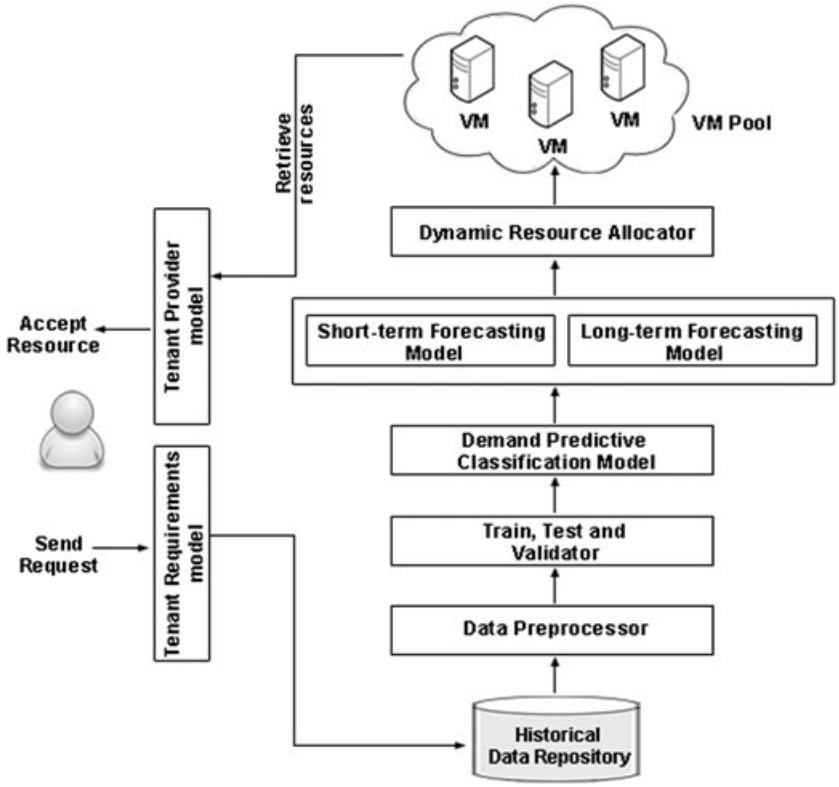

Figure 1. Prediction and resource allocation framework.

DOI: 10.1002/cpe

polynomial regression, auto-regressive with external input (ARX), and auto-regressive moving average with external input (ARMAX) for long-term predictions [25].

(6) *Dynamic Resource Allocator*: Dynamic Resource Allocator allocates CPU and other resources from our proposed dynamic resource allocation methodology by applying the best-fit heuristic algorithm.

The prediction framework predicts the future resources demand based on past utilization, and provisions resources in advance to minimize the down-time for running service tenants. Thus, prediction is useful in resource allocation for provisioning of resource requirements of service tenants and for preparing the correct type of VMs in advance. Based on prediction results of newly composed and already running service tenants, the dynamic resource allocator allocates the CPU and other resources from the VM pool using the tenant provider model.

5. EXPERIMENTAL EVALUATION OF RESOURCE PREDICTION

For our experiments, we set up a Cloud architecture running banking services. We conducted experiments on a set of 10 000 service tenants from the banking domain. Our experimental data set contains tenant information including functional size, functional priority, duration, state, CPU, memory, and utilization. Ten-fold cross validation [21, 22] is used to select the best parameter combination in each service tenant.

The values of accuracy and sensitivity obtained by different classification techniques are presented in Table I. Accuracy measures the actual percentage of test records (number of service tenants for which resource demand would increase) that are correctly classified. Sensitivity measures the actual percentage of positive class records (number of service tenants for which resource demand would increase) that are correctly classified. A total of 8000 service tenants are applied for training and 2000 service tenants are applied for testing purpose using 10-fold cross validation technique. We observe that reduced error pruning tree (REPTree) [30] produces better result with 97.05% accuracy and 95.9% sensitivity compared to other techniques. Our experiment result indicates that prediction of future resource requirement is needed only for service tenants in which the resource demand will increase, thereby reducing the computational time and cost for prediction.

Based on Table I, we selected REPTree as an optimal classification technique. Further, to check the difference in statistical significance of REPTree with other classification techniques, we applied *t*-test [21, 22]. The *t*-test results of LR and MLP techniques are lower than 2.83 compared to REPTree. The *t*-test result indicates that we could also use other classification techniques which have no significant statistical difference with REPTree. We analyzed the REPTree structures and observed that the following tenant-related characteristics played a vital role in building the predictive framework: resource utilization, functional priority, functional size, complexity index, activity period, and state information.

We applied time-series techniques for predicting the short-term future resource requirements. Figures 2 and 3 illustrate the pattern of the actual CPU utilization and predicted resource demand using trend seasonality model and exponential moving average respectively. We compared the mean square error (in term of CPU cores) of short-term prediction techniques in order to check the accuracy of prediction techniques, as illustrated in Table II. Based on our observations, trend seasonality model generates more accurate prediction compared to exponential moving average technique.

Table I. Classification Results.

Classification techniques	Accuracy	Sensitivity
Logistic regression (LR) [31]	95.8%	94.5%
Multilayer perceptron (MLP) [32]	93.3%	90.6%
Support vector machine (SVM) [33]	88.6%	83.4%
Reduced error pruning Tree (REPTree) [30]	97.05%	95.9%

DOI: 10.1002/cpe

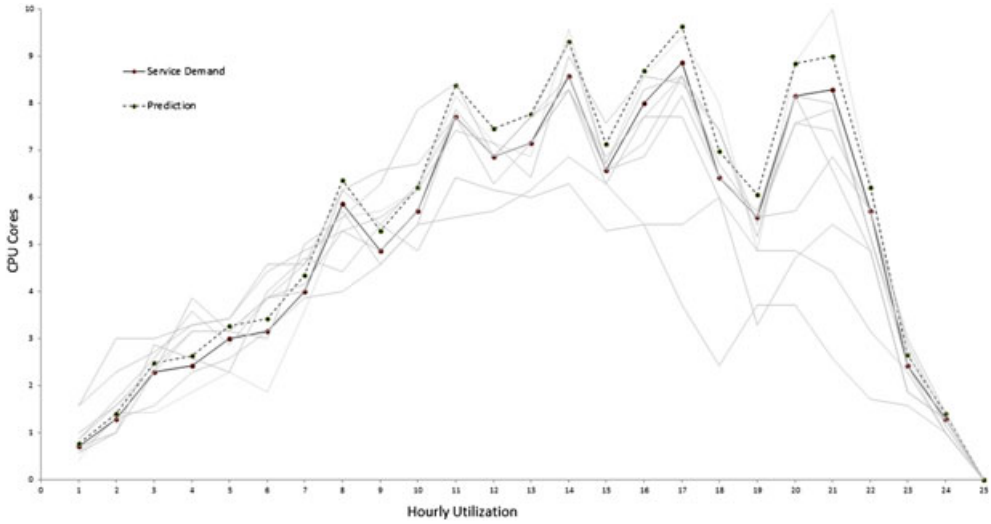

Figure 2. Trend seasonality model prediction.

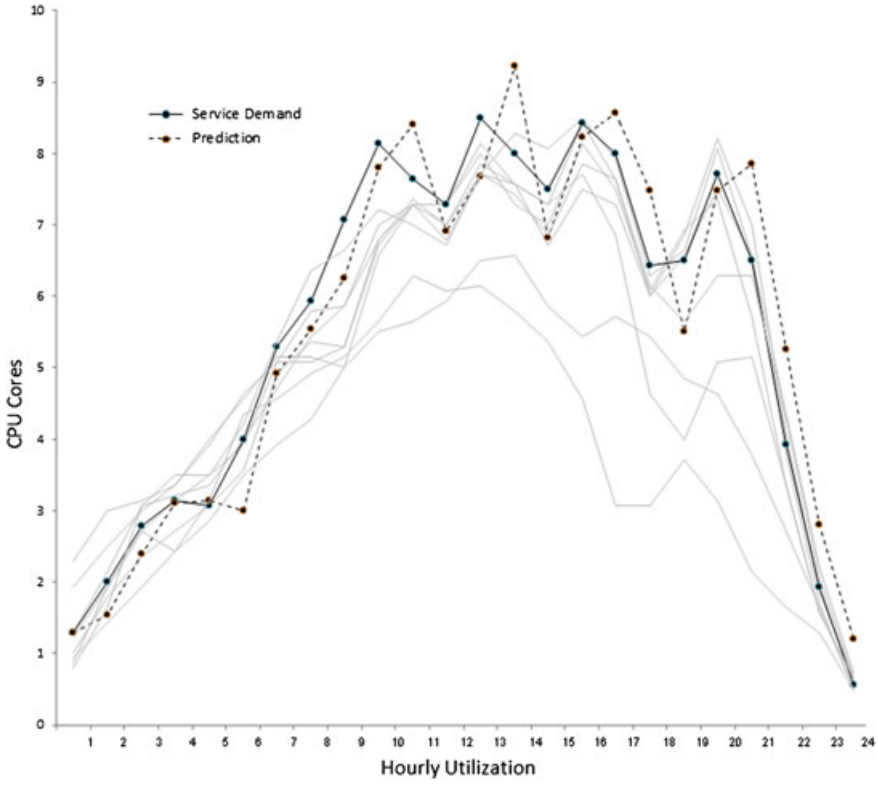

Figure 3. Exponential moving average prediction.

Table II. Short-term prediction results.

Time series technique	Mean squared error
Trend seasonality model	0.425
Exponential moving average	0.470

DOI: 10.1002/cpe

For long-term prediction, we analyzed the past 1-week hourly utilization of CPU workloads of service tenants and extracted the demand pattern. Figure 4 presents the long-term prediction of resource demand of service tenants using polynomial regression.

In Figure 4, prediction error pattern is shown as the difference between average demand pattern and polynomial regression prediction. We minimize the prediction error by keeping the optimal polynomial degree factor (in the range between 1 and 10) that reduces actual and anticipated demand gap. Figure 5 shows the long-term prediction using ARX/ARMAX.

The results of long-term prediction methods together with mean squared error are presented in Table III. ARX(10,3,1) polynomial coefficient order gives the more precise result and less mean squared error compared to other techniques. Long-term predictor gives more accurate result compared to short-term predictor as the long-term predictor learns more from historical data. However, it is not recommended to use the long-term predictor to predict the short-term future resource requirement as the long-term predictor requires a learning phase.

6. DYNAMIC RESOURCE ALLOCATION METHODOLOGY

Our dynamic resource allocation methodology integrates the prediction approach with resource allocation. Initially, resources are allocated using our dynamic resource allocation methodology (see Section 6.4). Our prediction approach predicts the future resource requirements (as explained in Section 4). Then, our dynamic resource allocation uses best-fit decreasing heuristic algorithm which assigns VMs to the smallest residual host if it fits. A new host is created if newly provisioned VMs cannot fit in any initialized host.

6.1. Defining the resource allocation problem

Let T be a set of service tenants that have been created and provisioned by a service provider and R be the amount of resources available to the provider. Given a set of service tenants with different resource requirements and a set of resources of varying capacities, let us assign resources to service tenants in such a way that the number of service tenants whose resource requirements are satisfied is maximized and the amount of resource waste is minimized. Let C_t denote the set of characteristics that influence resource allocation (see Section 2) for each service tenant t. A service tenant is allocated to resources only if all of its characteristics are satisfied. Hence, at any time instant n, the function (with the objective to maximize) indicating the number of service tenants allocated to resources is represented as follows:

$$f(n) = \sum\nolimits_{t \in T} \begin{cases} 1, & \textit{if } C_t \textit{ is satisfied} \\ 0, & \textit{if } C_t \textit{ is not satisfied} \end{cases}.$$

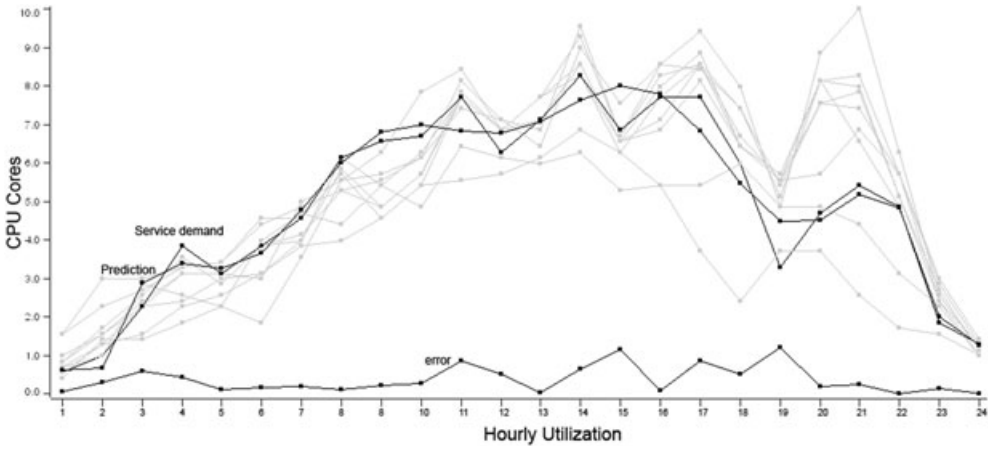

Figure 4. Polynomial regression prediction.

DOI: 10.1002/cpe

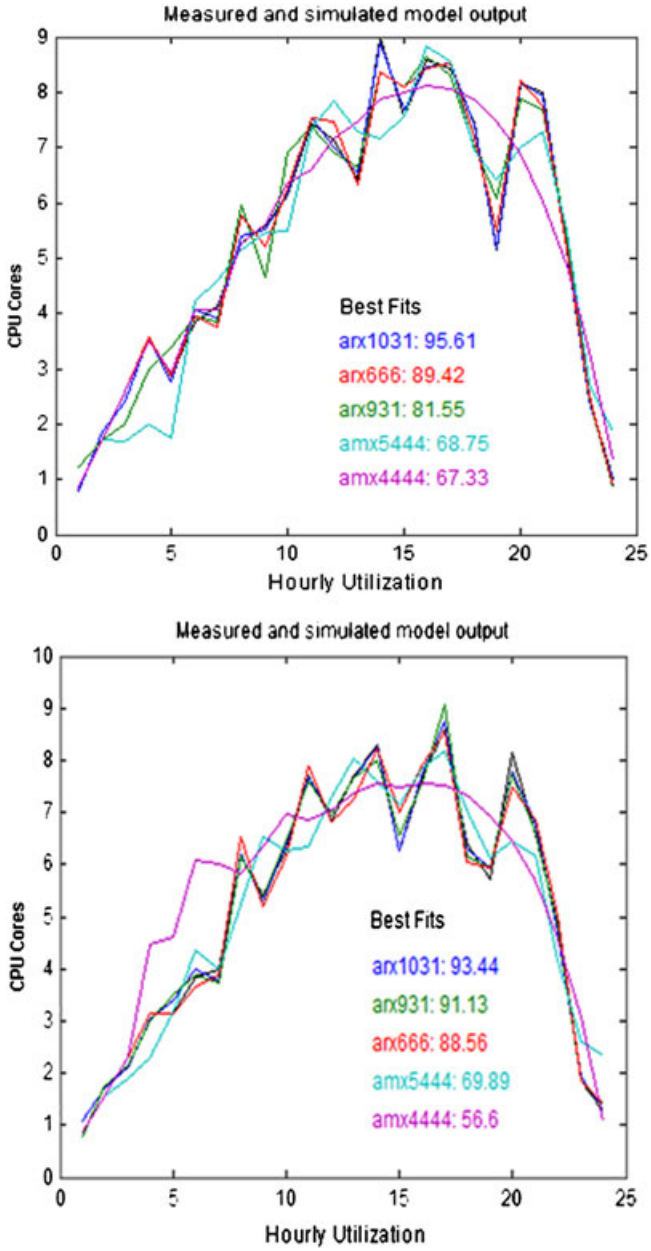

Figure 5. ARX and ARMAX prediction.

Table III. Long-term prediction results.

Technique	Best fit	Mean squared error
ARX 10 3 1	93.09	0.067
ARX 9 3 1	84.75	0.185
ARX 6 6 6	81.6	0.233
ARMX 5 4 4 4	70.8	0.458
ARMX 4 4 4 4	67.04	0.516
Polynomial regression	—	0.393

Let $r(t)$ denote the resource requirement of a service tenant t, which is modeled by the characteristics in Ct, $\forall$ t$\in$T. Thus, the amount of resources wasted during an allocation process can be modeled as follows:

DOI: 10.1002/cpe

$$w(n) = \sum_{t \in T} \begin{cases} 0, & \text{if } a_t = r(t) \\ a(t) - r(t), & \text{if } a_t > r(t) \end{cases}.$$

Here, $a(t)$ refers to the amount of resources allocated to a service tenant $t \in T$. The objective is to minimize $w(n)$.

6.2. Deciding whether resources can be allocated

Figure 6 represents a decision tree for every service tenant that represents the set of characteristics that need to be solved for allocating resources. Here, state refers to the current state of the service tenant. The time when it becomes active refers to the time at which the presently paused service tenant would become active. The time when it becomes active, i.e. current time > threshold, states that if the service tenant becomes active after the threshold hours, then resources of that service tenant could be freed. The decision tree is built by using service tenant characteristics such as state information, activity periods, and initial resource requirement of a service tenant. Depending on the values of the service tenant characteristics at any point, one of the following decisions could be made, namely *free resources*, *hold resources*, *allocate resources*, and *do not allocate resources*. A threshold value for activity period is set by the service provider.

6.3. Checking conditions for service tenant resource allocation

The following are the conditions that are required to be checked before and during resource allocation (see Figure 7).

- Identify the state of a service tenant whether resources could be allocated or de-allocated.
- If an existing tenant has not specified a state, then, based on the past active state information, predict at which time the service tenant will become active.
- Identify whether there is any change in the state of service tenant (active or paused). If the state of service tenant is active, then identify whether the resource requirements are specified or not. If requirements are not specified, then check for the resource requirement based on past utilization and waits until resource becomes available.
- If the requirements of tenants are specified, then determine the amount of resource available in repository. If enough resources are available, allocate resources and set the tenant's state as active. If resources are not available, then wait until resources become available.
- If the service tenant becomes active with in a stipulated time, then resources are kept as allocated (not freed). As explained in the previous subsection, the decision tree decides whether a resource can be allocated based on the state information. Otherwise, the state information is saved and the resources of the service tenant are freed.

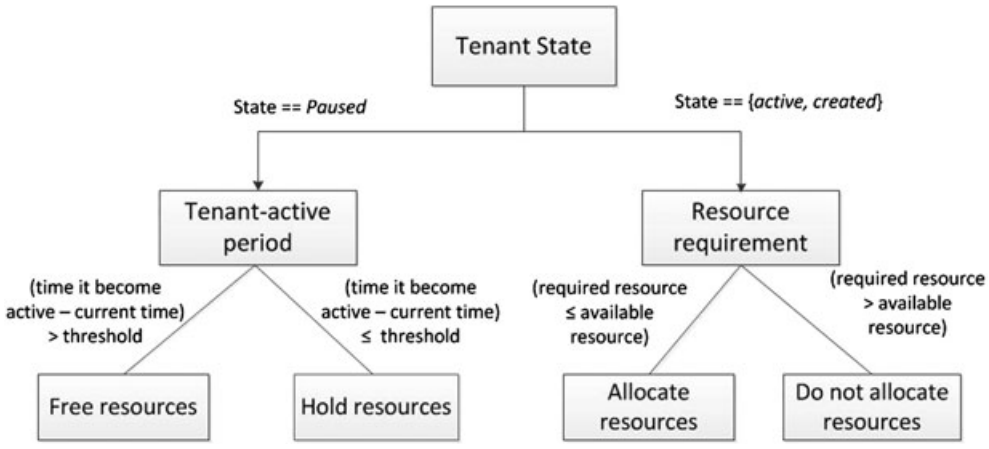

Figure 6. Decision tree for a single tenant.

DOI: 10.1002/cpe

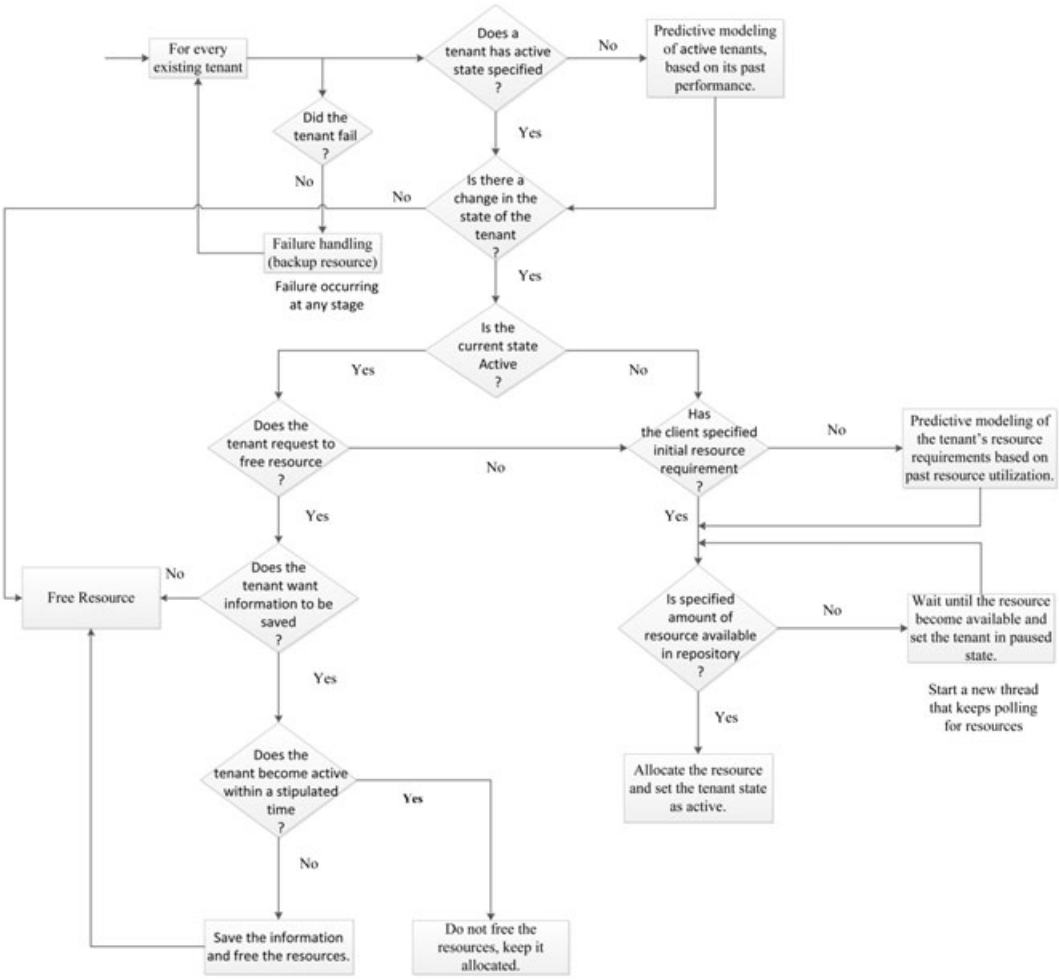

Figure 7. Service tenant resource allocation flow-chart for checking condition.

- During de-allocation or freeing the resource, conditions such as desire to save data for future references and the time before which the service tenant will become active are considered.
- Identify whether an existing running service tenant failed or not. If an existing running service tenant failed, then resources are backed up using a failure handling process. Our framework keeps records of service tenant information in a repository and the same resources are provided to the failed service.

6.4. Resource allocation process

Service tenants having different resource requirements must be added to a finite number of VMs in such a way that it minimizes the number of VMs used. We consider a set of service tenants (T1,T2,....,T10) along with their initial resource requirements and demonstrate how the resource requirements are matched. Service tenants are matched with the VM configuration table and categorized into 'small', 'medium', and 'large' VMs according to their resource requirements. The matched service tenants are added to active VMs based on the current VM utilization and maximum user handling capacity. The unmatched service tenants are provisioned to newly configured VMs.

Algorithm 1 gives a formal description of VM placement on the physical host with minimum remaining resources capacity. Service tenants are added to the VMs that are already running based on the matching configuration. We sort all the VMs in decreasing order based on CPU requirement.

DOI: 10.1002/cpe

Each time VMs are placed into physical host machine using best-fit decreasing heuristic algorithm [26]. The best-fit decreasing heuristic algorithm assigns VMs to the smallest residual host if it fits. A new host is created if newly provisioned VMs cannot fit in any initialized host. Because all the VMs and physical hosts are examined for each placement step, the algorithm has a running time of $O(nm)$, where n is the number of VMs and m is the number of hosts.

Algorithm 1 VM Allocation on Physical host: *Best-fit Decreasing Heuristic Algorithm*

```
1: procedure VM Allocation (VMj, Hosti) // number of VMs and number of physical hosts
2:       Add service tenants to VM based on matching configuration
3:       Sort VMs in decreasing order based on CPU requirements
4:       for all VM j to n do
5:             for all Host i to m do
6:                   if VMj fits in Hosti then
7:                         Calculate the remaining CPU capacity of Host after VM has been added
8:                   else No processing element left in Hosts or failed to be allocated
9:                    end if
10:            end for
11:            Start a new Host in the datacenter, and allocate remaining VM into a new Host
12:      end for
13: end procedure
```

6.5. Evaluation using CloudSim

CloudSim [27] is a generalized, and extensible simulation toolkit that enables seamless modeling, simulation, and experimentation of Cloud computing infrastructures and application services. It supports modeling and simulation of large-scale Cloud computing data centers and supports for user-defined policies for allocation of hosts to VMs and policies for allocation of host resources to

Table IV. Comparison of VM allocation strategies using CloudSim.

Round-robin allocation					
	Hosts	CPU	Allocated VMs	Remaining CPU	Failed VM allocations
Datacenter 1	Host2	8	VM4	2	VM7
Datacenter 1	Host1	8	VM2, VM6	1	VM7
Datacenter 1	Host0	4	VM0	2	VM6
Datacenter 2	Host2	8	VM5	1	VM7
Datacenter 2	Host1	8	VM3	3	VM7
Datacenter 2	Host0	4	VM1	3	VM7
Shortest job first allocation					
	Hosts	CPU	Allocated VMs	Remaining CPU	Failed VM allocations
Datacenter 1	Host2	8	VM2	5	VM4, VM5, VM7
Datacenter 1	Host1	8	VM0,VM3,VM6	1	VM7,VM4, VM5
Datacenter 1	Host0	4	VM1	3	VM6,VM5, VM4, VM7
Datacenter 2	Host2	8	VM7	0	VM4
Datacenter 2	Host1	8	VM5	1	VM4
Datacenter 2	Host0	4	—	4	VM4
Best-fit heuristic decreasing algorithm					
	Hosts	CPU	Allocated VMs	Remaining CPU	Failed VM allocations
Datacenter 1	Host2	8	VM7	0	VM4,VM3
Datacenter 1	Host1	8	VM5, VM1	0	VM2
Datacenter 1	Host0	4	VM6	0	VM0
Datacenter 2	Host2	8	VM4, VM0	0	—
Datacenter 2	Host1	8	VM3, VM2	0	—
Datacenter 2	Host0	4	—	—	—

DOI: 10.1002/cpe

VMs. We used CloudSim to model a Cloud environment and evaluate our algorithm. In CloudSim, we used a configuration with the following characteristics: two data centers, and three hosts per data center. For each data center, we allocated four cores, eight cores, and eight cores respectively for each host. We assumed eight VMs to be allocated, with the following requirements in number of cores: 2,1,3,5,6,7,4, and 8. We simulated each core in each VM with 1000 million instruction per second (MIPS) and kept each VM's requirement as 1000 MIPS so that the cores are not shared by different VMs during execution. We evaluated our proposed best-fit decreasing heuristic algorithm against different VM allocation strategies [28] and the results are depicted in Table IV.

In round-robin allocation, the allocation of VM7 failed in both data centers and the hosts in both the data centers are not utilized efficiently. In shortest job first allocation, the allocation of VM4 failed in both the data centers and the hosts in both the data centers are not utilized efficiently. In our proposed best-fit decreasing heuristic algorithm, at the time of allocation, a host is selected in such a way that it is the best fit for the VM and there is no wasting of resources in the host. Based on Table IV, all the VMs are accommodated in five hosts keeping one host still available. Furthermore, the hosts are utilized efficiently without wasting any resources.

7. CONCLUDING REMARKS

We presented a novel approach for resource demand prediction and dynamic resource allocation in multi-tenant service Cloud environments. To the best of our knowledge, our prediction approach is the first one that classifies service tenants in which resource requirement would increase or not, and predicts the resource demand for only those service tenants in which resource demand would increase, thereby minimizing the time needed for prediction. Our dynamic resource allocation approach adds the service tenants to matched VMs and allocates the VMs into physical host machines using the best-fit heuristic approach. Experiments demonstrated how our best-fit heuristic approach can efficiently allocate VMs to hosts so that the hosts are utilized to their fullest capacity. Future work will involve testing and evaluating our approach on larger data sets in various domains. Also, as an extension of the ideas presented in this paper, we will also investigate integrated dynamic resource allocation for service compositions that represent services working together as part of a business process execution.

REFERENCES

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4):50–58.
2. Mietzner R, Leymann F, Papazoglou MP. Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns. *In*: *Proceeding of the 3rd International Conference on Internet and Web Applications and Services*, 2008.
3. Ramachandran L, Narendra NC, Ponnalagu K. Dynamic provisioning in multi-tenant service Clouds. In *Service Oriented Computing and Applications*. Springer-Verlag: Germany, 2012.
4. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *FGCS* 2012; **28**(5):755–768.
5. Jiang Y, Perng CS, Li T, Chang R. ASAP: a self-adaptive prediction system for instant cloud resource demand provisioning. *In*: *Proceeding of the 11th International Conference on Data Mining* (*ICDM*), pp. 1104–1109, 2011.
6. Gong Z, Gu X, Wilkes J. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. *In*: *Proceeding of the 6th International Conference on Network and Service Management*, 2010.
7. Reig G, Guitart J. On the anticipation of resource demands to fulfill the QoS of SaaS web applications. *In*: *Proceeding of the International Conference on Grid Computing*, pp.147–154, 2012.
8. Islam S, Keung J, Lee K, Liu A. Empirical prediction models for adaptive resource provisioning in the Cloud. *FGCS* Elsevier, 2012; **28**(1):155–162.
9. Prevost JJ, Nagothu K, Kelley B, Jamshidi M. Prediction of cloud datacenter net works loads using stochastic and neural models. *In*: *Proceeding of the 6th International Conference on System of Systems Engineering* (*SoSE*), pp.276–281, 2011.
10. Zhang Z, Xiao L, Li Y, Ruan L. AVM-based resource management method using statistics. *In*: *Proceeding of the International Conference on Parallel and Distributed Systems*, 2012.
11. Warneke D, Kao O. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**(6):985–997.

DOI: 10.1002/cpe

12. Yazir YO, Matthews C, Farahbod R, Neville S, Guitouni A, Ganti S, Coady Y. Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. *In*: *Proceeding of IEEE 3rd International Conference on Cloud Computing*, pp. 91–98, 2010.
13. Chang N, Ren J, Viswanathan R. Optimal resource allocation in clouds. *In*: *Proceeding of the 3rd IEEE International Conference on Cloud Computing*, pp. 418–425, 2010.
14. Inomata A, Morikawa T, Ikebe M, Rahman M. Proposal and evaluation of dynamic resource allocation method based on the load of VMs on IaaS. *In*: *Proceeding of 4th IFIP International Conference on New Technologies*, *Mobility and Security*, 2011.
15. Calcavecchia NM, Biran O, Hadad E, Moatti Y. VM placement strategies for cloud scenario. *In*: *Proc o f5th International Conference on Cloud Computing*, pp.852–859, 2012.
16. Nejad MM, Mashayekhy L, Grosu D. A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. *In*: *Proceeding of the IEEE 6th International Conference on Cloud Computing*. pp. 188–195. 2013.
17. Zhu Q, Agrawal G. Resource provisioning with budget constraints for adaptive applications in cloud environments. *IEEE Transactions on Services Computing* 2012; **5**(4):497–511.
18. Espadas J, Molina A, Jimnez G, Molina M, Ramrez R, Concha D. A tenant-based resource allocation model for scaling Software-as-a-Service applications over Cloud computing infrastructures. *Future Generation Computer Systems* 2013; **29**(1):273–286.
19. Zhuang H, Liu X, Ou Z, Aberer K. Impact of instance seeking strategies on resource allocation in cloud data centers. *In*: *Proc. of the 6th IEEE International Conference on Cloud Computing*. pp. 27–34.2013.
20. Xiao Z, Song W, Chen Q. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems* 2013; **24**(6):1107–1117.
21. Witten IH, Frank E, Hall MA. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann series: San Francisco, CA, 2011.
22. Han J, Kamber M, Pei J. *Data Mining: Concepts and Techniques* (3rd edn). Morgan Kaufmann series: Wyman street, Waltham, MA, USA, 2011.
23. Specht DF. Probabilistic neural networks. *Neural Networks* 1990; **3**:109–118.
24. Shmueli G. *Practical Time Series Forecasting: A Hands on Guide* (2nd edn). Create Space Independent Publishing Platform, Galit Shmueli & Statistics.com LLC, 2011.
25. Pintelon R, Schoukens J. *System Identification: A Frequency Domain Approach* (2nd edn). Wiley & Sons: Hoboken, New Jersey, 2012.
26. Yue M. A simple proof of the inequality FFD (L) $<$11/9 OPT (L) + 1, for all for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica* 1991; **7**(4):321–331.
27. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a tool kit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice & Experience* 2011; **41**(1):23–50.
28. Sotomayor B, Montero RS, Llorente IM, Foster I. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing* 2009; **13**(5):14–22.
29. Verma M, Gangadharan GR, Vadlamani R, Narendra NC. Resource demand prediction in multi-tenant service clouds. *In*: *Proc. of the IEEE International Conference on Cloud Computing for Emerging Markets* (*CCEM*), 2013.
30. Elomaa T, Kaariainen M. An analysis of reduced error pruning. *Journal of Artificial Intelligence Research* 2001; **15**:163–187.
31. Hosmer DW, Lemeshow S. *Applied Logistic Regressions* (2nd edn). Wiley: Canada, 2004.
32. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by error propagation. In *Book: Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. **1**. MIT Press: Cambridge, MA, USA, 1986; 318–362.
33. Cortes C, Vapnik V. Support vector networks. *Machine Learning* 1995; **20**(3):273–297.

DOI: 10.1002/cpe