1-1-2019

An Optimal Strategy for Resource Utilization in Cloud Data Centers

Wenlong Ni
Jiangxi Normal University

Yuhong Zhang
Texas Southern University

Wei W. Li
Texas Southern University

Received October 2, 2019, accepted October 22, 2019, date of publication October 30, 2019, date of current version November 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2950435

An Optimal Strategy for Resource Utilization in Cloud Data Centers

WENLONG NI[1], **(Member, IEEE), YUHONG ZHANG**[2], **AND WEI W. LI**[3], **(Senior Member, IEEE)**

[1]School of Computer Information Engineering, Jiangxi Normal University, Nanchang 330022, China
[2]Department of Engineering, Texas Southern University, Houston, TX 77004, USA
[3]Department of Computer Science, Texas Southern University, Houston, TX 77004, USA

Corresponding author: Wenlong Ni (wni@jxnu.edu.cn)

This work was supported in part by the U.S. National Science Foundation under Grant HRD-1137732, Grant CNS-1827940, and Grant HRD-1915995.

ABSTRACT Cloud computing has emerged in recent years as one of the most interesting developments in technology. With the gaining popularity of cloud-based solutions, more and more applications are migrating into the Cloud and thus have highly demanding critical requirements for networking resources. Virtual technology associated with a Data Center consists of a set of servers, storage and network devices, power systems, cooling systems, etc., and makes it possible for the resource management of physical machines to be more finely tuned and thus support multiple virtual machines well. The growing challenge, however, is how to efficiently provision these resources to meet the requirements of the different qualities of service levels. This paper offers and investigates a general situation wherein a datacenter can determine the cost of using resources and a Cloud service user can decide whether it will pay the price for the resource or not for an incoming task. By establishing a Continuous-Time Markov Decision Process model for both an average reward model and a discounted expected reward model, the optimal policy of each model for admitting tasks can be verified to be a State-related control limit (threshold) policy, respectively. Further, a detailed statement and verification of the upper boundaries for such an optimal policy, and a comprehensive set of experiments on the various cases to validate this proposed solution are provided. Particularly, the machine learning method is implemented to obtain the optimal threshold values by using a feed-forward neural network model. Several numerical examples are also provided on how to derive optimal threshold values. The results offered in this paper can be easily utilized to help datacenter operate in an economically optimal way when providing different needed application services to Cloud service users.

INDEX TERMS Cloud computing, continuous-time Markov decision process, datacenter, optimal control policy, resource allocation, virtual machine.

I. INTRODUCTION

Cloud computing [1]–[6] has emerged in recent years as one of the most interesting developments in technology. Given the gaining popularity of cloud-based solutions, more and more applications are now migrating into the Cloud. An increasing number of these applications have highly demanding critical requirements for networking different resources. Indeed, Cloud computing has emerged as a new computing paradigm that enables ubiquitous, convenient, on-demand network access to a large amount of remote, distributed, and shared computing resources. As of today, a simple web search request may touch 1000+ servers, while a large computing request can involve thousands of machines. In Cloud computing, different kinds of computing resources are provided to users as services and these users have access to computing resources (e.g., networks, servers, storage, applications, and services) from anywhere in the world based on their needs where they are. Cloud computing has been known as software as a service, infrastructure as a service, and platform as a service. Cloud computing requires the underlying network infrastructure to be fast, carry large amounts of traffic, and be scalable.

Many existing cloud service platforms, such as Amazon EC2 [7], the Google App Engine [8] and Microsoft Azure [9], have proven their success and been opened to the public as a

The associate editor coordinating the review of this manuscript and approving it for publication was Nan Wu.

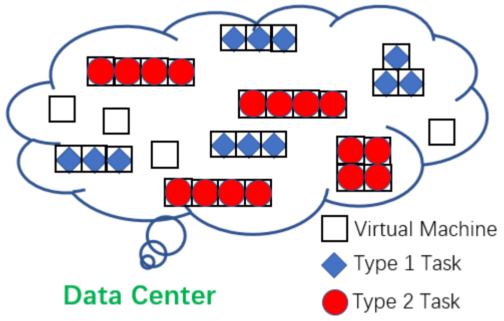


FIGURE 1. **Datacenters with virtual machines.**

pay-as-you-go service. More and more enterprises and organization build their own Cloud computing infrastructures or resort to using a hybrid cloud. Efficient resource management not only enhances the quality of the service, but also reduces the consumption of key resources. The growing challenge is how to efficiently provision the resources to meet the requirements of quality of service (QoS). Resource provision is the most fundamental issue of Cloud computing service deployment.

Virtual technology makes resource management of physical machines (PMs) possible, and so they can be more fine-grained by supporting multiple Virtual Machine (VMs) [10]. Each VM can be equipped with different physical resources, e.g., CPU, memory, bandwidth, and disk storage. Thus, the overall resource can be multiplexed to improve the resource utilization. VMs can also be migrated if necessary. The virtualized server cluster thus constitutes the datacenter (DC) [11]–[13] in different topology structures. This paper proposes a cloud service model that follows the cloud service framework of [14], where a virtual machine (VM) is the minimal portion of the cloud resource that can be allocated to a cloud service. When user sends a service request to the cloud system, one or multiple VMs are dedicatedly assigned to this task. Furthermore, Liang *et al.* [15] propose a SMDP [16] service decision making system for interdomain service transfer to balance the computation loads among multiple cloud domains when the holding cost of multiple VMs is a linear function of the number of VMs, and the Wang *et al.* [11] and Barroso *et al.* [17] investigate the similar problems when the relationship of power cost and energy efficiency over the number of VMs is nonlinear function. Our research here targets on the problems when the relationship of power cost and energy efficiency is a function of the number of VMs. Figure 1 shows an example of a datacenter with many Virtual Machines serving two types of Cloud service tasks. Depending on the task requirement, each type of task may involve different number of VMs. Here type-1 task takes 3 VMs and type-2 task needs 4 VMs for their services.

DC is a set of servers, storage and network devices, power systems, cooling systems, etc. A DC can work alone, which is suited for small scale applications. A large number of distributed DCs in different geo-locations [18]–[20] can work collaboratively as a whole entity for large-scale service applications, such as Online Businesses, Smart Grid, and scientific computation. These DCs are connected using high-speed Internet or dedicated high-bandwidth communication links. Service can thus be provided by DCs that are closer to users. This networking scheme is more scalable and also more eligible for successful larger scale applications.

Some studies [21] indicates that the VM demands for certain resources are highly bursty traffic, and thus can be modelled as stochastic processes. In other words, the real demands of these stochastic resources fluctuate, so it is difficult to obtain an accurate fixed-value measure. One such example is network bandwidth. The bandwidth demands of VMs in datacenters will be determined by the classes of the tasks. This paper proposes a novel resource allocation schemes in DCs for a CSP to maximize its expected average reward or total discounted expected reward for serving Cloud computing tasks from any initial state. The problem under investigation in our research is for a general situation wherein a datacenter can determine the cost of using resources and a Cloud service user can decide whether it will pay the price for the resource or not for an incoming task. We verified in this paper that our proposed CTMDP method and model has successfully reached our goals in finding the optimal policy. Assuming that the arrival processes of tasks is a Poisson process [22], their departure process follows negative exponential distribution, and different application tasks require different amounts of resources. The major contributions of this paper are listed below:

1) A Continuous Time Markov Decision Process (CTMDP) model for the DCs is established, respectively, to gain the optimal policy of a DC on when to admit or reject a task in order to achieve the maximum average reward and the total discounted expected reward for any initial state. As far as we know, this is the first time that the DCs as described in this paper is modelled as a CTMDP model with several major theoretic results obtained.
2) A detailed statement and verification of the optimal control limit policy, as obtained in [23]–[26] for other different models, for both the average model and the discounted expected reward model is provided for multimedia tasks requiring different amount of resources. In [26], only the optimal objective function with discounted model is studied. The major additional contribution of the current research over the previous work in [26]: (1) the consideration of average model including the objective development, optimization analysis and corresponding optimal threshold policy, (2) upper bound analysis of the optimal threshold for both discount model and average model; and (3) and machine learning methods to derive the optimal thresholds, threshold to value method to verify the theoretical result, etc. for both discount model and average model.
3) A detailed statement and verification of a novel result in the upper bound of the optimal policy, and a

comprehensive set of experiments on various cases to validate our proposed solution is justified both mathematically and numerically.

4) CTMDP is in general a powerful tool in dealing with stochastic dynamic problems. However, based on the complexity of the problems, it is always a challenge to suitably define the model and the related parameters to be able to theoretically resolve the concerned problems successfully. Even though in practice this method still takes too much time through iteration methods to reach the obtained optimal solution, which is particularly not feasible in a constantly changing online environment. Thus machine learning methods are proposed to derive the optimal policy based on limited available data. Our numerical results, as explained in the tables and diagrams, are consistent with our theoretical results.

The remainder of this paper is organized as follows. Section II discusses the modelling, Section III describes the structure of the optimal policy, a control limit policy for both the average model and the discounted expected reward model, and includes the verification process for both models. Section IV is devoted to a simulation using the machine learning method. Section V offers a numerical analysis with various tables and diagrams that validate the theoretical results. Finally, Section VI offers concluding remarks.

II. MODEL FORMULATION

In this section we build the models on the datacenters. The first part introduces the system model, and the second part defines the CTMDP models based on the assumptions for the system model.

A. THE SYSTEM MODEL

There are basically two parties in the Cloud computing paradigm, the cloud service providers (CSPs) and the clients, who act their own roles by providing and using the computing resources. While enjoying the convenient on-demand access to computing resources or services, the clients need to pay for these accesses. CSPs can make a profit by providing services and charging the clients for these services. Clients can avoid the costs associated with 'inhouse' provisioning of computing resources and also have access to a larger pool of computing resources than they could possibly own by themselves. There are many different aspects, however, to consider to evaluate the Cloud computing service quality, and these can be optimized using various optimization methods [7], [11], [27]–[32].

The datacenter offers various features to help organize the Cloud computing and includes the following advantages:

1) A datacenter permits the connection of thousands of datacenter servers in an efficient way, so the Cloud computing can expand its service easily.
2) The datacenter delivers traffic reliability and efficiency to massive machine-to-machine communications wherein the activities from Cloud computing will emerge as the workloads then distributed on the datacenter servers.
3) The datacenter supports various virtualization techniques that help the DC to create a Virtual Machine (VM), virtual network, and virtual function.

Note that in general, there are two types of applications in the datacenter: (i) service applications and (ii) batch applications [17]. Service applications tend to generate many requests with low processing needs whereas batch applications tend to small number of requests with large processing needs. Unlike the batch applications that are throughput-sensitive, service applications are typically response time-sensitive. This paper considers a datacenter with both service applications (type-1) and batch applications (type-2), each of which will require resources in the datacenter for service. The other basic assumptions for the DC are as follows:

1) The resources for a Cloud computing task can be defined as a specific number of VMs designed to a certain type of task. The total number of VMs defines the capacity of resources from the DCs. There are a total number of C VMs in the system.
2) There are two types of $Tasks$ (T_1 and T_2) in the system, and each needs a number of b_1, b_2 VMs for service. The arriving time for tasks T_1 and T_2 are Poisson processes with rates λ_1 and λ_2, respectively. The task processing time for the tasks follows the negative exponential distributions with rates μ_1 and μ_2, respectively.
3) When a task comes to the system and there are enough free VMs, the CSP will decide whether to admit/reject the task based on the current state of the system. However, if the system is full when a task is coming, which means there is no free VM, the task will leave the system.
4) Serving a type-1 (type-2) task would contribute $R_1(R_2)$ units of reward to the CSP. However, for each task, the CSP needs to pay a price at rate $f(b_1, n_1, b_2, n_2)$ to manage the VMs (resources) in the DCs when there are already n_1 type-1 tasks and n_2 type-2 tasks in service. Here we consider b_i, $i = 1, 2$ as constants.

B. THE CTMDP MODEL

First, let us introduce some basic concepts in general CTMDP models. Each model has a state space, Action space, Transition Probabilities between states, reward functions and decision epochs. Also in the CTMDP models, a decision rule prescribes a procedure for action selection in each state at a specified decision epoch. Decision rules range in general from deterministic Markovian to randomized History Dependent, depending on how the rules incorporate past information and how they select the actions. Deterministic Markovian decision rules specify the action choice when the system occupies a state s at decision epoch t. A policy π specifies the decision rule to be used at every decision epoch. It gives the decision maker a prescription for action selection for any possible future system state or history.

TABLE 1. List of important notations.

C	Maximal number of VMs in the datacenter
$T_i, i = 1, 2$	type-i task
$b_i, i = 1, 2$	Number of VMs needed for type-i task
$n_i, i = 1, 2$	Number of type-i tasks
$\lambda_i, i = 1, 2$	Arrival rate of type-i task
$\mu_i, i = 1, 2$	Service rate of type-i task
$R_i, i = 1, 2$	Reward of type-i task
$A_i, i = 1, 2$	Arrival event of type-i task
$D_i, i = 1, 2$	Departure event of type-i task
α	Continuous-time discount factor

1) OBJECTIVE FUNCTIONS

In this paper, denote by s_t for the state at time t, a_t for the action to take at state s_t, and $r(s_t, a_t)$ for the reward obtained when action a_t is selected at state s_t, we have two objectives as follows:

1) The first objective is to find a policy π_g that can bring the maximum average expected reward $g^\pi(s)$ for every initial state s among all the possible polices.

$$g^\pi(s) = \lim_{N\to\infty} \frac{1}{N} E_s^\pi \left\{ \sum_{t=0}^{N} r(s_t, a_t) \right\}. \tag{1}$$

2) For each policy π, let $v_\alpha^\pi(s)$ denote the total expected infinite-horizon discounted reward with α as the discount factor, given that the process occupies state s at the first decision epoch. In this paper, our second objective is to find an optimal policy π_α that can bring the maximum total expected discounted reward $v_\alpha^\pi(s)$ for every initial state s.

$$v_\alpha^\pi(s) = E_s^\pi \left\{ \int_0^\infty e^{-\alpha t} r(s_t, a_t) dt \right\}. \tag{2}$$

It is known that if decisions are made frequently, so that the discount rate is very close to 1, or when performance criterion cannot easily be described in economic terms, the decision maker (CSP) may prefer to compare policies on the basis of their average expected reward instead of their total discounted reward. In this paper, both the average optimal policy and discounted optimal policy are studied so that the CSP can choose any optimal policy if needed. It is worth to point out, from reference [33], that the average reward is the average expected reward of current epoch while the discounted reward is for long-term accumulated reward. The decision variables are the actions on either entering the system or leaving from the system at each time epoch in this stochastic dynamic programming problem.

2) MODEL CONSTRUCTION

We now introduce the CTMDP models as follows:

1) Let state space be $S = \{s : s = (n_1, n_2)\}$, where integers n_1 and n_2 satisfy $b_1 n_1 + b_2 n_2 \le C$; and event space be $E = \{D_1, D_2, A_1, A_2\}$, where D_1 and D_2 means a T_1 and T_2 departure from the system after service, while A_1 means an arrival of a T_1 task, A_2 is an arrival of T_2 task. Since the states migration not only depends on the number of tasks in the system but also depends on the happening departure and arrival events, we will need to define a new state space as $\hat{S} = S \times E$. By doing so a state could be written as $\hat{s} = \langle s, e\rangle = \langle (n_1, n_2), e\rangle$, where n_1 and n_2 are the numbers for T_1 and T_2 tasks, e stands for the event which will probably happen on state (n_1, n_2), $e \in \{D_1, D_2, A_1, A_2\}$. Please be noticed that the specification of the event in this paper is one of major technical differences from that in paper [25].

2) In states $\langle (n_1, n_2), D_1\rangle$ and $\langle (n_1, n_2), D_2\rangle$, if denote by a_C as the action to continue and by noting that $b_1 n_1 + b_2 n_2 \le C$ is a precondition in the state space, the action space is then given by

$$\begin{aligned} A_{\langle (n_1,n_2),D_1\rangle} &= \{a_C\}, n_1 > 0, n_2 \ge 0; \\ A_{\langle (n_1,n_2),D_2\rangle} &= \{a_C\}, n_1 \ge 0, n_2 > 0. \end{aligned}$$

Similarly, in states $\langle (n_1, n_2), A_1\rangle$ and $\langle (n_1, n_2), A_2\rangle$, if denote by a_R as the action to reject the request and a_A as the action to admit, the action space will be

$$\begin{aligned} A_{\langle (n_1,n_2),A_1\rangle} &= \{a_R, a_A\}, n_1 \ge 0, n_2 \ge 0; \\ A_{\langle (n_1,n_2),A_2\rangle} &= \{a_R, a_A\}, n_1 \ge 0, n_2 \ge 0. \end{aligned}$$

3) The decision epochs are those time points when a call arriving or leaving the system. Based on our assumption, it is not too hard to know that the distribution of time between two epochs is

$$F(t|\hat{s}, a) = 1 - e^{-\beta(\hat{s},a)t}, t \ge 0,$$

where for each state $\hat{s} = \langle ((n_1, n_2)), b\rangle$ and action a, $\beta_0(s) = \lambda_1 + \lambda_2 + n_1\mu_1 + n_2\mu_2$, since a departure event only happens when there is a task in the system, the $\beta(\hat{s}, a)$ will be represented as

$$\begin{cases} \beta_0(s) - \mu_1, & b = D_1, a = a_C, n_1 > 0, \\ \beta_0(s) - \mu_2, & b = D_2, a = a_C, n_2 > 0, \\ \beta_0(s) + \mu_1, & b = A_1, a = a_A, n_1 \ge 0, n_2 \ge 0, \\ & b_1 n_1 + b_2 n_2 \le C - b_1, \\ \beta_0(s) + \mu_2, & b = A_2, a = a_A, n_1 \ge 0, n_2 \ge 0, \\ & b_1 n_1 + b_2 n_2 \le C - b_2, \\ \beta_0(s), & b = \{A_1, A_2\}, a = a_R, n_1 \ge 0, n_2 \ge 0. \end{cases}$$

4) Let $q(j|\hat{s}, a)$ denote the probability that the system occupies state j in the next epoch, if at the current epoch the system is at state $\hat{s}$ and the decision maker takes action $a \in A_{\hat{s}}$. For the cases of departure events, e.g. for a departure event of D_1 under the condition of $(n_1 > 0, n_2 \ge 0)$, $(\hat{s}, a) = (\langle (n_1, n_2), D_1\rangle, a_C)$, if denote by $s_n = (n_1 - 1, n_2)$, then we will have $q(j|\hat{s}, a)$

as

$$\begin{cases} \lambda_1/\beta_0(s_n), & j=\langle(n_1-1,n_2),A_1\rangle, \\ \lambda_2/\beta_0(s_n), & j=\langle(n_1-1,n_2),A_2\rangle, \\ (n_1-1)\mu_1/\beta_0(s_n), & j=\langle(n_1-1,n_2),D_1\rangle, \\ n_2\mu_2/\beta_0(s_n), & j=\langle(n_1-1,n_2),D_2\rangle. \end{cases}$$

Similar equations can be derived for cases like $(\hat{s},a) = (\langle(n_1,n_2),D_2\rangle,a_C)$. For the cases of arrival events, $(\hat{s},a) = (\langle(n_1,n_2),A_1\rangle,a_A)$, $(\hat{s},a) = (\langle(n_1,n_2),A_2\rangle,a_A)$, since admitting an incoming call migrates the system state immediately (adding one user or not), we will get $q(j|\hat{s},a)$ as

$$\begin{cases} q(j|\langle(n_1+2,n_2),D_1\rangle,a_C), & b=A_1,a=a_A, \\ q(j|\langle(n_1+1,n_2),D_1\rangle,a_C), & b=A_1,a=a_R, \\ q(j|\langle(n_1,n_2+2),D_2\rangle,a_C), & b=A_2,a=a_A, \\ q(j|\langle(n_1,n_2+1),D_2\rangle,a_C), & b=A_2,a=a_R. \end{cases}$$

5) Because the system state does not change between decision epochs, from our assumptions, the expected average reward between epochs satisfies

$$\begin{aligned} r(\hat{s},a) &= k(\hat{s},a)+c(\hat{s},a)E^a_{\hat{s}}\{\tau_1\} \\ &= k(\hat{s},a)+\frac{c(\hat{s},a)}{\beta(\hat{s},a)}. \end{aligned}$$

Also from *Chp 11.5.2* [33] and our assumptions, the expected discounted reward between epochs satisfies

$$\begin{aligned} r(\hat{s},a) &= k(\hat{s},a)+c(\hat{s},a)E^a_{\hat{s}}\left\{\int_0^{\tau_1} e^{-\alpha t}dt\right\} \\ &= k(\hat{s},a)+c(\hat{s},a)E^a_{\hat{s}}\left\{[1-e^{-\alpha\tau_1}]/\alpha\right\} \\ &= k(\hat{s},a)+\frac{c(\hat{s},a)}{\alpha+\beta(\hat{s},a)}, \end{aligned}$$

where

$$k(\hat{s},a)=\begin{cases} 0, & b=\{D_1,D_2\},a=a_C, \\ 0, & b=\{A_1,A_2\},a=a_R, \\ R_1, & b=A_1,a=a_A, \\ R_2, & b=A_2,a=a_A. \end{cases}$$

Here, since we will get $R_1(R_2)$ unites of reward after the service of a $T_1(T_2)$ task, we can treat this as that we get the reward at the time of accepting the task, thus making the problem to be an admission control problem. Also, we have the cost function $c(\hat{s},a)$ as

$$\begin{cases} -f(b_1,n_1-1,b_2,n_2), & b=D_1,a=a_C,n_1>0, \\ -f(b_1,n_1,b_2,n_2-1), & b=D_2,a=a_C,n_2>0, \\ -f(b_1,n_1+1,b_2,n_2), & b=A_1,a=a_A, \\ & b_1n_1+b_2n_2\le C-b_1, \\ -f(b_1,n_1,b_2,n_2+1), & b=A_2,a=a_A, \\ & b_1n_1+b_2n_2\le C-b_2, \\ -f(b_1,n_1,b_2,n_2), & b=\{A_1,A_2\},a=a_R. \end{cases}$$

In the next section we will prove that there exists a state-related threshold for accepting the tasks if the cost function has some special properties.

III. OPTIMAL POLICY

A policy is stationary if, for each decision epoch t, decision rule at t epoch $d_t = d$ is the same, which can be denoted by d^∞. In our CTMDP model, an average optimal policy π_g means that it is with optimal average expected reward $g^\pi(\hat{s})$ for every initial state $\hat{s}$. Similarly, an discounted optimal policy π_α means that it can bring the maximum total expected discounted reward $v^\pi_\alpha(\hat{s})$ for every initial state $\hat{s}$.

In our CTMDP model, since both the state space $\hat{S}$ and the action space A are finite, the reward functions $r(\hat{s},a)$ for both average optimal policy and discounted optimal plicy are also finite, then from *Theorem 8.4.5* and *Theorem 11.3.2* of [33], both the average optimal policy and discounted optimal policy are stationary deterministic policy d^∞, so our problem can be reduced to find the deterministic decision rules d of the optimal policies.

A. THRESHOLD POLICY

For our admission problem, a policy is called a control limit (threshold) policy for a given number of **Tasks** n_1 and n_2 in the system, say for T_1 task, is there existing a constant or threshold $D(n_2) \ge 0$ such that the system will accept the **arriving** T_1 whenever the number of T_1 currently in the system is less than $D(n_2)$, that means the decision rule for T_1 is:

$$d(n_1,n_2)=\begin{cases} Admit, & n_1\le D(n_2),n_2\ge 0, \\ Reject, & n_1>D(n_2),n_2\ge 0. \end{cases} \tag{3}$$

Similar definitions can be found with T_2 tasks. It is observed that a control limit policy is a stationary deterministic policy.

B. RATE UNIFORMIZATION

Based on the assumptions, our process fits the condition of *Assumption 11.5.1* of [33], which is $[1-q(\hat{s}|\hat{s},a)]\beta(\hat{s},a)\le c, \forall\hat{s}\in\hat{S},a\in A$, here c is a constant. So, we can define a uniformization of our process with components denoted by $\sim$. For each deterministic decision rule d, let $q_d(j|\hat{s}) = q(j|\hat{s},d(\hat{s}))$, $r_d(\hat{s}) = r(\hat{s},d(\hat{s}))$ and $\beta_d(\hat{s}) = \beta(\hat{s},d(\hat{s}))$, $c = \lambda_1+\lambda_2+C*\max(\mu_1,\mu_2)$, from *Chp 11.5.2* [33], we have

$$\tilde{q}(j|\hat{s},a)=\begin{cases} 1-\dfrac{[1-q(\hat{s}|\hat{s},a)]\beta(\hat{s},a)}{c}, & j=s, \\ \dfrac{q(j|\hat{s},a)\beta(\hat{s},a)}{c}, & j\ne\hat{s}. \end{cases}$$

From *Theorem 8.4.5* of [33], with a stationary deterministic policy d^∞, the expected average reward (gain) per time unit is a constant function that is $g(s) = g$, where g is a scalar. Also, from *Chp 11.5.3* [33], the equation can be written in component notation as

$$h^{d^\infty}(\hat{s}) = r_d(\hat{s}) - \frac{g}{\beta(s,a)} + \sum_{j\in\hat{S}} q_d(j|\hat{s})h^{d^\infty}(j). \tag{4}$$

Here $h(s)$ is the bias function for each state s, which is interpreted as if starting from state s, the expected total difference between the reward and the average reward (stationary reward).

Furthermore, for the reward functions of average policies, we have $\tilde{r}(\hat{s},a) \equiv r(\hat{s},a)\frac{\beta(\hat{s},a)}{c}$.

Then for each d^∞ policy and $\hat{s} \in \hat{S}$, we have

$$\tilde{h}^{d^\infty}(\hat{s}) = \tilde{r}_d(\hat{s}) - \tilde{g} + \sum_{j\in\hat{S}} q_d(j|\hat{s})\tilde{h}^{d^\infty}(j). \tag{5}$$

Since g denotes the optimal average reward per time unit and the expected time between transitions is $\frac{1}{c}$, then $\tilde{g} = g/c$ can be interpreted as the optimal average reward per transition. Also, we have

$$\tilde{h}^{d^\infty}(\hat{s}) = h^{d^\infty}(\hat{s}). \tag{6}$$

From equation (4) and (5) , the optimality equation of $h(\hat{s})$ would have the form of

$$h(\hat{s}) = \max_{a\in A_{\hat{s}}}\left\{\tilde{r}(\hat{s},a) - \tilde{g} + \sum_{j\in\hat{S}}\tilde{q}(j|\hat{s},a)h(j)\right\}. \tag{7}$$

Similarly for discounted polices, from *Chp 11.5.2* [33], we have

$$v_\alpha^{d^\infty}(\hat{s}) = r_d(\hat{s}) + \frac{\beta_d(\hat{s})}{\alpha+\beta_d(\hat{s})}\sum_{j\in\hat{S}} q_d(j|\hat{s})v_\alpha^{d^\infty}(j). \tag{8}$$

For the reward functions, we have $\tilde{r}(\hat{s},a) \equiv r(\hat{s},a)\frac{\alpha+\beta(\hat{s},a)}{\alpha+c}$. From *Proposition 11.5.1* [33], for each d^∞ policy and $\hat{s}\in\hat{S}$, we have

$$\tilde{v}_\alpha^{d^\infty}(\hat{s}) = v_\alpha^{d^\infty}(\hat{s}). \tag{9}$$

From equation (8), the optimal equation of $v(\hat{s})$ for maximum $v_\alpha^\pi(\hat{s})$ would have the form of

$$v(\hat{s}) = \max_{a\in A_{\hat{s}}}\left\{\tilde{r}(\hat{s},a) + \lambda\sum_{j\in\hat{S}}\tilde{q}(j|\hat{s},a)v(j)\right\}, \tag{10}$$

where $\lambda \equiv \frac{c}{c+\alpha}$.

C. AVERAGE MODEL

From equation (7) we get

$$\begin{aligned}
&h(\langle(n_1+1,n_2),D_1\rangle)\\
&\quad= \frac{1}{c}[-f(b_1,n_1,b_2,n_2) - g\\
&\qquad+\lambda_1\, h(\langle(n_1,n_2),A_1\rangle) + \lambda_2\, h(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1\, h(\langle(n_1,n_2),D_1\rangle) + n_2\mu_2\, h(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(s))h(\langle(n_1+1,n_2),D_1\rangle)].
\end{aligned}$$

This means that if we define a new cost function $f_h(b_1,n_1,b_2,n_2) = f(b_1,n_1,b_2,n_2) + g$, we have

$$\begin{aligned}
&h(\langle(n_1+1,n_2),D_1\rangle)\\
&\quad= \frac{1}{\beta_0(n_1,n_2)}[-f_h(b_1,n_1,b_2,n_2)\\
&\qquad+\lambda_1 h(\langle(n_1,n_2),A_1\rangle) + \lambda_2 h(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 h(\langle(n_1,n_2),D_1\rangle)\\
&\qquad+n_2\mu_2 h(\langle(n_1,n_2),D_2\rangle)].
\end{aligned} \tag{11}$$

Similarly, it is easily found that

$$h(\langle(n_1+1,n_2),D_1\rangle) = h(\langle(n_1,n_2+1),D_2\rangle),$$

which shows the equality between different departure events, similar results can also be seen among arrival events or even between departure and arrival events. This leads us to define a new function $B(s)$, $s=(n_1,n_2)$, $n_1\geq 0$, $n_2\geq 0$ which is

$$B(s) = h(\langle(n_1+1,n_2),D_1\rangle) = h(\langle(n_1,n_2+1),D_2\rangle).$$

It is noticed that $X(n_1,n_2)$, or sometime using $X(s)$ only for a short expression, is only related to the state, but not with the happening event, which can greatly simplify the proof process.

Similar as above results for a departure event, we can consider an arrive event and will get the following results:

$$\begin{aligned}
&h(\langle(n_1,n_2),A_1\rangle,a_A)\\
&\quad= R_1\frac{\beta_0(n_1+1,n_2)}{c} + \frac{1}{c}\Big[-f_h(b_1,n_1+1,b_2,n_2)\\
&\qquad+\lambda_1 h(\langle(n_1+1,n_2),A_1\rangle) + \lambda_2 h(\langle(n_1+1,n_2),A_2\rangle)\\
&\qquad+(n_1+1)\mu_1 h(\langle(n_1+1,n_2),D_1\rangle)\\
&\qquad+n_2\mu_2 h(\langle(n_1+1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1+1,n_2))h(\langle(n_1,n_2),A_1\rangle)\Big],
\end{aligned} \tag{12}$$

and

$$\begin{aligned}
&h(\langle(n_1,n_2),A_1\rangle,a_R)\\
&\quad= \frac{1}{c}\Big[-f_h(b_1,n_1,b_2,n_2) + \lambda_1 h(\langle(n_1,n_2),A_1\rangle)\\
&\qquad+\lambda_2 h(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 h(\langle(n_1,n_2),D_1\rangle) + n_2\mu_2 h(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2))h(\langle(n_1,n_2),A_1\rangle)\Big].
\end{aligned} \tag{13}$$

Similarly, we will have

$$\begin{aligned}
&h(\langle(n_1,n_2),A_2\rangle,a_A)\\
&\quad= R_2\frac{\beta_0(n_1,n_2+1)}{c} + \frac{1}{c}\Big[-f_h(b_1,n_1,b_2,n_2+1)\\
&\qquad+\lambda_1 h(\langle(n_1,n_2+1),A_2\rangle) + \lambda_2 h(\langle(n_1,n_2+1),A_2\rangle)\\
&\qquad+n_1\mu_1 h(\langle(n_1,n_2+1),D_1\rangle)\\
&\qquad+(n_2+1)\mu_2 h(\langle(n_1,n_2+1),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2+1))h(\langle(n_1,n_2),A_2\rangle)\Big],
\end{aligned} \tag{14}$$

and

$$\begin{aligned}
&h(\langle(n_1,n_2),A_2\rangle,a_R)\\
&\quad= \frac{1}{c}\Big[-f_h(b_1,n_1,b_2,n_2) + \lambda_1 h(\langle(n_1,n_2),A_1\rangle)\\
&\qquad+\lambda_2 h(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 h(\langle(n_1,n_2),D_1\rangle) + n_2\mu_2 h(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2))h(\langle(n_1,n_2),A_2\rangle)\Big].
\end{aligned} \tag{15}$$

Since admitting a call migrates the system state immediately, we get

$$h(\langle(n_1,n_2),A_1\rangle,a_A) \geq R_1 + B((n_1+1,n_2)),$$
$$h(\langle(n_1,n_2),A_1\rangle,a_R) \geq B((n_1,n_2)),$$
$$h(\langle(n_1,n_2),A_2\rangle,a_A) \geq R_2 + B((n_1,n_2+1)),$$
$$h(\langle(n_1,n_2),A_2\rangle,a_R) \geq B((n_1,n_2)).$$

And furthermore, the above inequalities will be the equality when the corresponding action a_A (whenever an A_1 arrives), a_R (whenever an A_1 arrives), a_A (whenever an A_2 arrives) and a_R (whenever an A_2 arrives), is the best action, respectively. This also includes the situation when $b_1n_1 + b_2n_2 = C$, the action a_R is the the best action for any arrival of A_1 and A_2.

From these analysis, it is not too hard to verify that

$$\begin{aligned} &h(\langle(n_1,n_2),A_1\rangle) \\ &\quad= \max\Big[B((n_1,n_2)), R_1 + B((n_1+1,n_2))\Big], \end{aligned} \tag{16}$$
$$\begin{aligned} &h(\langle(n_1,n_2),A_2\rangle) \\ &\quad= \max\Big[B((n_1,n_2)), R_2 + B((n_1,n_2+1))\Big]. \end{aligned} \tag{17}$$

Before proceeding to our major theory of optimal strategy, we will need to introduce the following two lemmas first.

Lemma 1: If $h(\langle(n_1,n_2),A_1\rangle)$, $h(\langle(n_1,n_2),A_2\rangle)$ and $f_h(b_1,n_1,b_2,n_2)$ are all concave functions for n_1 and n_2 respectively, then $B((n_1,n_2))$ is also concave function for n_1 and n_2 respectively.

Proof: By using equation (11) and the notation of $B((n_1,n_2))$, we will have

$$\begin{aligned} &B((n_1,n_2)) \\ &\quad= \frac{1}{\beta_0(n_1,n_2)}[-f_h(b_1,n_1,b_2,n_2) + \lambda_1 h(\langle(n_1,n_2),A_1\rangle) \\ &\qquad+\lambda_2 h(\langle(n_1,n_2),A_2\rangle) \\ &\qquad+n_1\mu_1 B((n_1-1,n_2)) + n_2\mu_2 B((n_1,n_2-1))]. \end{aligned} \tag{18}$$

For any function $g(n_1,n_2)$ $(n_1 \geq 0, n_2 \geq 0)$, if denote by

$$\Delta_{n_1} g(n_1,n_2) = g(n_1+1,n_2) - g(n_1,n_2),$$
$$\Delta_{n_2} g(n_1,n_2) = g(n_1,n_2+1) - g(n_1,n_2),$$

we will have the results by a mathematical implementation on above equation (18) as follows:

$$\begin{aligned} &\beta_0(n_1+1,n_2)\big[B(n_1+1,n_2) - B(n_1,n_2)\big] \\ &\quad= f_h(b_1,n_1,b_2,n_2) - f_h(b_1,n_1+1,b_2,n_2) \\ &\qquad+\lambda_1\Big[h(\langle(n_1+1,n_2),A_1\rangle) - h(\langle(n_1,n_2),A_1\rangle)\Big] \\ &\qquad+\lambda_2\Big[h(\langle(n_1+1,n_2),A_2\rangle) - h(\langle(n_1,n_2),A_2\rangle)\Big] \\ &\qquad+n_1\mu_1\Big[B(n_1,n_2) - B(n_1-1,n_2)\Big] \\ &\qquad+n_2\mu_2\Big[B(n_1+1,n_2-1) - B(n_1,n_2-1)\Big], \end{aligned} \tag{19}$$

and

$$\begin{aligned} &\beta_0(n_1+2,n_2)\big[\Delta B_{n_1}(n_1+1,n_2) - \Delta B_{n_1}(n_1,n_2))\big] \\ &\quad= \Delta_{n_1} f_h(b_1,n_1,b_2,n_2) - \Delta_{n_1} f_h(b_1,n_1+1,b_2,n_2) \\ &\qquad+\lambda_1\big[\Delta_{n_1} h(\langle(n_1+1,n_2),A_1\rangle) - \Delta_{n_1} h(\langle(n_1,n_2),A_1\rangle)\big] \\ &\qquad+\lambda_2\big[\Delta_{n_1} h(\langle(n_1+1,n_2),A_2\rangle) - \Delta_{n_1} h(\langle(n_1,n_2),A_2\rangle)\big] \\ &\qquad+n_1\mu_1\big[\Delta_{n_1} B(n_1,n_2) - \Delta_{n_1} B(n_1-1,n_2)\big] \\ &\qquad+n_2\mu_2\big[\Delta_{n_1} B(n_1,n_2-1) - \Delta_{n_1} B(n_1-1,n_2-1)\big]. \end{aligned} \tag{20}$$

From these equation, by using a combined mathematical induction method and iteration method, we can verify that the left side of equation (19) and (20) are all non-positive, that means for any n_1 and n_2,

$$B(n_1+1,n_2) - B(n_1,n_2) \leq 0,$$

and

$$\Delta B_{n_1}(n_1+1,n_2) - \Delta B_{n_1}(n_1,n_2)) \leq 0.$$

Thus $B(n_1,n_2)$ is a non-increasing function on n_1 from above first inequality and is a concave function on n_1 from above second inequality. Similarly, $B(n_1,n_2)$ is also a concave and non-increasing function on n_2.

Another important result to be used in our major theory of both average and discounted optimal strategies is given as follow:

Lemma 2 [25]: If an integer function $p(i)$ $(i \geq 0)$ is concave, then for any constant $R \geq 0$, the function

$$q(i) \equiv \max\{p(i), R + p(i+1)\},$$

is also concave on $i \geq 0$.

For the cost function $f(b_1,n_1,b_2,n_2)$ $(n_1 \geq 0, n_2 \geq 0)$, if denote by

$$\begin{aligned} \Delta_{n_1} f(b_1,n_1,b_2,n_2) &= f(b_1,n_1+1,b_2,n_2) \\ &\quad -f(b_1,n_1,b_2,n_2), \\ \Delta_{n_2} f(b_1,n_1,b_2,n_2) &= f(b_1,n_1,b_2,n_2+1) \\ &\quad -f(b_1,n_1,b_2,n_2), \end{aligned}$$

then from the equations (27) and (28), we can obtain the following theorem.

Theorem 1: If $f(b_1,n_1,b_2,n_2)$ is convex and increasing for nonnegative integers n_1 and n_2, respectively, and

$$\Delta_{n_1} f(b_1,n_1,b_2,n_2) \geq 0, \quad \textit{and } \Delta_{n_2} f(b_1,n_1,b_2,n_2) \geq 0,$$

the optimal policy for the average model is a control limit policy.

Proof: It is straightforward to note that

$$\Delta_{n_1} f_h(b_1,n_1,b_2,n_2) = \Delta_{n_1} f(b_1,n_1,b_2,n_2),$$
$$\Delta_{n_2} f_h(b_1,n_1,b_2,n_2) = \Delta_{n_2} f(b_1,n_1,b_2,n_2).$$

Under the given condition, we use Value Iteration Method to show that for all states $B(n_1,n_2)$ is concave and nonincreasing for nonnegative integers n_1 and n_2, respectively.

1) Set $B^0(n_1,n_2) = 0$, and substitute this into equation (21), we will have
$$B^1(n_1,n_2) = -\frac{f_h(b_1,n_1,b_2,n_2)}{c}, \quad n_1 \geq 0, n_2 \geq 0.$$
By concavity and monotony of $f_h(b_1,n_1,b_2,n_2)$, $B^1(n_1,n_2)$ is therefore concave nonincreasing for n_1 and n_2, respectively.
2) By using equation (16) and (17) as well as the result in Lemma 2, we know that $h^1(\langle n_1,n_2,A_1\rangle)$ and $h^1(\langle n_1,n_2,A_2\rangle)$ are all concave and non-increasing functions.
3) Set n=n+1, by noting the result in Lemma 1, and by using equation (16) and (17) as well as the result in Lemma 2 again, we can finally verify $B^{n+1}(n_1,n_2)$ is also concave nonincreasing for n_1 and n_2, respectively
4) As the iteration continues, with n goes to ∞, $B(n_1,n_2)$ is always concave nonincreasing for n_1 and n_2, respectively.

Finally, by noting the *Theorem 8.4.4* and *Theorem 8.4.5* of [33] that the optimality equation has the unique solution, we know the value iteration $B^n(n_1,n_2)$ will uniquely converges to a concave and nonincreasing. Furthermore, by using equation (16) and (17), as well as the concavity property of $h(\langle(n_1,n_2),A_i\rangle)$, it is straight forward to know that the optimal policy must be a control limit policy as stated in the Theory.

D. DISCOUNTED MODEL

From equation (10) we get
$$\begin{aligned}
&v(\langle(n_1+1,n_2),D_1\rangle)\\
&\quad= \frac{1}{\alpha+c}[-f(b_1,n_1,b_2,n_2)\\
&\qquad+\lambda_1 v(\langle(n_1,n_2),A_1\rangle)+\lambda_2 v(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 v(\langle(n_1,n_2),D_1\rangle)+n_2\mu_2 v(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(s))v(\langle(n_1+1,n_2),D_1\rangle)]. \qquad (21)
\end{aligned}$$

This means that
$$\begin{aligned}
&v(\langle(n_1+1,n_2),D_1\rangle)\\
&\quad= \frac{1}{\alpha+\beta_0(n_1,n_2)}[-f(b_1,n_1,b_2,n_2)\\
&\qquad+\lambda_1 v(\langle(n_1,n_2),A_1\rangle)+\lambda_2 v(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 v(\langle(n_1,n_2),D_1\rangle)\\
&\qquad+n_2\mu_2 v(\langle(n_1,n_2),D_2\rangle)]. \qquad (22)
\end{aligned}$$

Similarly, it is easily found that
$$v(\langle(n_1+1,n_2),D_1\rangle) = v(\langle(n_1,n_2+1),D_2\rangle),$$
which shows the equality between different departure events. This leads us to define a new function $X(s), s = (n_1,n_2), n_1 \geq 0, n_2 \geq 0$ which is
$$X(n_1,n_2) = v(\langle(n_1+1,n_2),D_1\rangle) = v(\langle(n_1,n_2+1),D_2\rangle).$$
It is noticed that $X(n_1,n_2)$, or sometime using $X(s)$ only for a short expression, is only related to the state, but not with the happening event, which can greatly simplify the proof process.

Similar as above results for a departure event, we can consider an arrive event and will get the following results:
$$\begin{aligned}
&v(\langle(n_1,n_2),A_1\rangle,a_A)\\
&\quad= R_1\frac{\alpha+\beta_0(n_1+1,n_2)}{\alpha+c}\\
&\qquad+\frac{1}{\alpha+c}\Big[-f(b_1,n_1+1,b_2,n_2)\\
&\qquad+\lambda_1 v(\langle(n_1+1,n_2),A_1\rangle)+\lambda_2 v(\langle(n_1+1,n_2),A_2\rangle)\\
&\qquad+(n_1+1)\mu_1 v(\langle(n_1+1,n_2),D_1\rangle)\\
&\qquad+n_2\mu_2 v(\langle(n_1+1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1+1,n_2))v(\langle(n_1,n_2),A_1\rangle)\Big], \qquad (23)
\end{aligned}$$
and
$$\begin{aligned}
&v(\langle(n_1,n_2),A_1\rangle,a_R)\\
&\quad= \frac{1}{\alpha+c}\Big[-f(b_1,n_1,b_2,n_2)+\lambda_1 v(\langle(n_1,n_2),A_1\rangle)\\
&\qquad+\lambda_2 v(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 v(\langle(n_1,n_2),D_1\rangle)+n_2\mu_2 v(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2))v(\langle(n_1,n_2),A_1\rangle)\Big]. \qquad (24)
\end{aligned}$$
Similarly, we will have
$$\begin{aligned}
&v(\langle(n_1,n_2),A_2\rangle,a_A)\\
&\quad= R_2\frac{\alpha+\beta_0(n_1,n_2+1)}{\alpha+c}\\
&\qquad+\frac{1}{\alpha+c}\Big[-f(b_1,n_1,b_2,n_2+1)\\
&\qquad+\lambda_1 v(\langle(n_1,n_2+1),A_2\rangle)+\lambda_2 v(\langle(n_1,n_2+1),A_2\rangle)\\
&\qquad+n_1\mu_1 v(\langle(n_1,n_2+1),D_1\rangle)\\
&\qquad+(n_2+1)\mu_2 v(\langle(n_1,n_2+1),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2+1))v(\langle(n_1,n_2),A_2\rangle)\Big], \qquad (25)
\end{aligned}$$
and
$$\begin{aligned}
&v(\langle(n_1,n_2),A_2\rangle,a_R)\\
&\quad= \frac{1}{\alpha+c}\Big[-f(b_1,n_1,b_2,n_2)+\lambda_1 v(\langle(n_1,n_2),A_1\rangle)\\
&\qquad+\lambda_2 v(\langle(n_1,n_2),A_2\rangle)\\
&\qquad+n_1\mu_1 v(\langle(n_1,n_2),D_1\rangle)+n_2\mu_2 v(\langle(n_1,n_2),D_2\rangle)\\
&\qquad+(c-\beta_0(n_1,n_2))v(\langle(n_1,n_2),A_2\rangle)\Big]. \qquad (26)
\end{aligned}$$

From above equations, we can easily get
$$\begin{aligned}
&v(\langle(n_1,n_2),A_1\rangle,a_A) \geq R_1+X((n_1+1,n_2)),\\
&v(\langle(n_1,n_2),A_1\rangle,a_R) \geq X((n_1,n_2)),\\
&v(\langle(n_1,n_2),A_2\rangle,a_A) \geq R_2+X((n_1,n_2+1)),\\
&v(\langle(n_1,n_2),A_2\rangle,a_R) \geq X((n_1,n_2)).
\end{aligned}$$

In fact, the above inequalities will be the equalities when the corresponding action a_A (whenever an A_1 arrives), a_R (whenever an A_1 arrives), a_A (whenever an A_2 arrives) and a_R (whenever an A_2 arrives), is the best action, respectively.

This also includes the situation when $n_1+n_2=C$, the action a_R is the the best action for any arrival of A_1 and A_2.

From these analysis, it is not too hard to verify that

$$\begin{aligned} &v(\langle(n_1,n_2),A_1\rangle)\\ &\quad=\max\Big[X((n_1,n_2)),R_1+X((n_1+1,n_2))\Big], \qquad (27)\\ &v(\langle(n_1,n_2),A_2\rangle)\\ &\quad=\max\Big[X((n_1,n_2)),R_2+X((n_1,n_2+1))\Big]. \qquad (28)\end{aligned}$$

Before proceeding to our major theory of optimal strategy, we will need to introduce the following two lemmas first.

Lemma 3: If $v(\langle(n_1,n_2),A_1\rangle)$, $v(\langle(n_1,n_2),A_2\rangle)$ and $f(b_1,n_1,b_2,n_2)$ are all concave functions for n_1 and n_2 respectively, then $X((n_1,n_2))$ is also concave function for n_1 and n_2 respectively.

Proof: By using equation (22) and the notation of $X((n_1,n_2))$, we will have

$$\begin{aligned} &X((n_1,n_2))\\ &\quad=\frac{1}{\alpha+\beta_0(n_1,n_2)}[-f(b_1,n_1,b_2,n_2)\\ &\qquad+\lambda_1 v(\langle(n_1,n_2),A_1\rangle)\\ &\qquad+\lambda_2 v(\langle(n_1,n_2),A_2\rangle)+n_1\mu_1X((n_1-1,n_2))\\ &\qquad+n_2\mu_2X((n_1,n_2-1))]. \qquad (29)\end{aligned}$$

we will obtain the following results by a mathematical implementation on above equation (29):

$$\begin{aligned} &\big(\alpha+\beta_0(n_1+1,n_2)\big)\big[X(n_1+1,n_2)-X(n_1,n_2)\big]\\ &\quad=f(b_1,n_1,b_2,n_2)-f(b_1,n_1+1,b_2,n_2)\\ &\qquad+\lambda_1\Big[v(\langle(n_1+1,n_2),A_1\rangle)-v(\langle(n_1,n_2),A_1\rangle)\Big]\\ &\qquad+\lambda_2\Big[v(\langle(n_1+1,n_2),A_2\rangle)-v(\langle(n_1,n_2),A_2\rangle)\Big]\\ &\qquad+n_1\mu_1\Big[X(n_1,n_2)-X(n_1-1,n_2)\Big]\\ &\qquad+n_2\mu_2\Big[X(n_1+1,n_2-1)-X(n_1,n_2-1)\Big], \qquad (30)\end{aligned}$$

and

$$\begin{aligned} &(\alpha+\beta_0(n_1+2,n_2))\big[\Delta_{n_1}X(n_1+1,n_2)-\Delta_{n_1}X(n_1,n_2))\big]\\ &\quad=\Delta_{n_1}f(n_1,n_2)-\Delta_{n_1}f(n_1+1,n_2)\\ &\qquad+\lambda_1\big[\Delta_{n_1}v(\langle(n_1+1,n_2),A_1\rangle)-\Delta_{n_1}v(\langle(n_1,n_2),A_1\rangle)\big]\\ &\qquad+\lambda_2\big[\Delta_{n_1}v(\langle(n_1+1,n_2),A_2\rangle)-\Delta_{n_1}v(\langle(n_1,n_2),A_2\rangle)\big]\\ &\qquad+n_1\mu_1\big[\Delta_{n_1}X(n_1,n_2)-\Delta_{n_1}X(n_1-1,n_2)\big]\\ &\qquad+n_2\mu_2\big[\Delta_{n_1}X(n_1,n_2-1)-\Delta_{n_1}X(n_1-1,n_2-1)\big]. \qquad (31)\end{aligned}$$

From these equation, by using a combined mathematical induction method and iteration method, we can verify that the left side of equation (30) and (31) are all non-positive, that means for any n_1 and n_2,

$$X(n_1+1,n_2)-X(n_1,n_2)\le 0,$$

and

$$\Delta X_{n_1}(n_1+1,n_2)-\Delta X_{n_1}(n_1,n_2))\le 0.$$

Thus $X(n_1,n_2)$ is a non-increasing function on n_1 from above first inequality and is a concave function on n_1 from above second inequality. Similarly, $X(n_1,n_2)$ is also a concave and non-increasing function on n_2.

Another important result to be used in our major theory of optimal strategy is given as follow:

Based on above Lemma and by a similar method as in the proof of Theorem 1, we can prove the following major result:

Theorem 2: If $f(b_1,n_1,b_2,n_2)$ is convex and increasing for nonnegative integers n_1 and n_2, respectively, then the optimal policy for the discounted model is also a control limit policy, i.e., for any state (n_1,n_2) in which $n_1+n_2\le C$, there must exist two integers, say N_1 and N_2, such that decision

$$a_{\langle(n_1,n_2),A_1\rangle}=\begin{cases}a_A, & \text{if } n_1\le N_1;\\ a_R, & \text{if } n_1>N_1.\end{cases} \qquad (32)$$

and

$$a_{\langle(n_1,n_2),A_2\rangle}=\begin{cases}a_A, & \text{if } n_2\le N_2;\\ a_R, & \text{if } n_2>N_2.\end{cases} \qquad (33)$$

E. UPPER BOUND ANALYSIS

From the above analysis we have known that both the average optimal policy and discounted optimal policy is a control limit policy if the cost function $f(b_1,n_1,b_2,n_2)$ is convex and increasing for nonnegative integers n_1 and n_2, respectively. How to determine the corresponding threshold value of the control limit policy is a challenging issue. However, under some conditions, we can identify the upper bound easily as follows:

Theorem 3: In the average model, for any non-negative integer n_2, if there exists an integer n such that $b_1n+b_2n_2\le C$ and

$$\frac{\Delta_{n_1}f_h(b_1,n,b_2,n_2)}{\beta_0(n+1,n_2)}>R_1, \qquad (34)$$

then an upper bound of the threshold value when any task-1 arrives at the system is determined by

$$N_1^*=\min\left\{n:\frac{\Delta_{n_1}f_h(b_1,n,b_2,n_2)}{\beta_0(n+1,n_2)}>R_1\right\},$$

that means the optimal threshold value $N_1\le N_1^*$. Similarly, for any non-negative integer n_1, if there exists an integer n such that $b_1n_1+b_2n\le C$ and

$$\frac{\Delta_{n_2}f_h(b_1,n_1,b_2,n)}{\beta_0(n_1,n+1)}>R_2, \qquad (35)$$

then an upper bound of the threshold value when any task-2 arrives at the system is determined by

$$N_2^*=\min\left\{n:\frac{\Delta_{n_2}f_h(b_1,n_1,b_2,n)}{\beta_0(n_1,n+1)}>R_2\right\},$$

that means the optimal threshold value $N_2\le N_2^*$.

Proof: By noticing the non-increasing property of the concave functions $h(\langle n_1, n_2, A_1\rangle)$, $h(\langle n_1, n_2, A_2\rangle)$ and $B(n_1, n_2)$, we will have $h(\langle n_1 + 1, n_2, A_1\rangle) < h(\langle n_1, n_2, A_1\rangle)$, $h(\langle n_1 + 1, n_2, A_2\rangle) < h(\langle n_1, n_2, A_2\rangle)$, $B(n_1, n_2) < B(n_1 - 1, n_2)$ and $B(n_1 + 1, n_2 - 1) < B(n_1, n_2 - 1)$. If furthermore,

$$\frac{1}{\beta_0(n_1+1, n_2)}\Delta_{n_1} f_h(b_1, n_1, b_2, n_2) > R_1, \qquad (36)$$

holds from the given condition, from equation (19), we know

$$R_1 + B(\langle n_1 + 1, n_2\rangle) < B(\langle n_1, n_2\rangle).$$

According to equation (16), the above inequality means that the system will take a reject action for incoming A_1 arrivals. Thus, the n in the condition is an upper bound and therefore the first result is verified. The result for the upper bound of task-2 can be derived similarly.

Similarly, we can obtain the upper result for the discounted model and only list the result as follows:

Theorem 4: In the discounted model, for any non-negative integer n_2, if there exists an integer n such that $b_1 n + b_2 n_2 \le C$ and

$$\frac{\Delta_{n_1} f(b_1, n, b_2, n_2)}{\alpha + \beta_0(n+1, n_2)} > R_1, \qquad (37)$$

then an upper bound of the threshold value when any task-1 arrives at the system is determined by

$$N_1^* = \min\left\{n : \frac{\Delta_{n_1} f(b_1, n, b_2, n_2)}{\alpha + \beta_0(n+1, n_2)} > R_1\right\},$$

that means the optimal threshold value $N_1 \le N_1^*$. Similarly, for any non-negative integer n_1, if there exists an integer n such that $b_1 n_1 + b_2 n \le C$ and

$$\frac{\Delta_{n_2} f(b_1, n_1, b_2, n)}{\alpha + \beta_0(n_1, n+1)} > R_2, \qquad (38)$$

then an upper bound of the threshold value when any task-2 arrives at the system is determined by

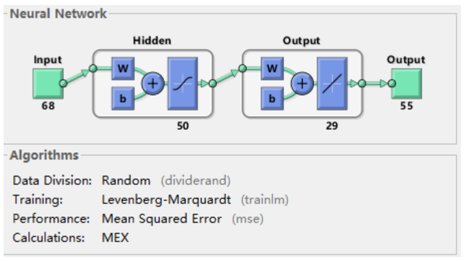
$$N_2^* = \min\left\{n : \frac{\Delta_{n_2} f(b_1, n_1, b_2, n)}{\alpha + \beta_0(n_1, n+1)} > R_2\right\},$$

that means the optimal threshold value $N_2 \le N_2^*$.

IV. SIMULATION WITH MACHINE LEARNING

We have theoretically verified that the optimal policy to maximizing the expected discounted reward in equation (2) is a control limit policy or a threshold policy for accepting task-1 and task-2 arrivals. However, it is always a challenging problem in exactly finding this threshold value or optimal objective value for a given problem, especially in a continuously changing environment, such as the changes of arrival rates, service rates, rewards in the real world. While we have demonstrated in above subsection on how to derive the thresholds by using value iteration method, the calculation through this way is always a time-consuming issue.

In this subsection, we propose the machine learning method and then demonstrate it to obtain or estimate the threshold value and the optimal objective value by using a



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

FIGURE 2. Neural network model created with Matlab.

TABLE 2. Parameter value setting.

	λ	μ	b
T_1	4	10	4
T_2	2	4	10

TABLE 3. Training dataset for neural network model.

R_1	0.2, 0.4, 0.6, 0.8, 1
R_2	0.2, 0.4, 0.6, 0.8, 1
a	0.1, 0.5, 1, 2

feed-forward neural network model. A feed-forward neural network is an artificial neural network where connections between the units do not form a cycle. Figure 2 shows the structure of the neural network model.

Our CTMDP model consists of several parameters like arrival rates, departure rates, rewards and cost function, etc. Here for simulation and numerical analysis purpose we set some parameters as those listed in Table 2. It can be seen from the Table 2 that the loads for T_1 and T_2 tasks are $\rho_1 = \frac{\lambda_1}{\mu_1} = 0.4$ and $\rho_2 = \frac{\lambda_2}{\mu_2} = 0.5$, which means the system are medium loaded, and the rewards for T_2 is a little higher than the T_1 task. The total capacity of the system is set to be 40.

Based on these development, we can build the training data sets in Table 3, here reward R_1 and R_2 is chosen from 0.2, 0.4, 0.6, 0.8, 1, and the cost function $f(b_1, n_1, b_2, n_2)$ is with the format as

$$f(b_1, n_1, b_2, n_2) = a(n_1 + n_2) + 0.001b_1 + 0.002b_2$$

Here a is some constants ≥ 0, the terms $0.001b_1 + 0.002b_2$ stands for setup cost of binding and releasing b_1 and b_2 VMs. In Table 3 a is chosen from 0.1, 0.5, 1, 2, while keeping all the other parameters unchanged.

The combination of these parameters will give us $5 * 5 * 4 = 100$ training data inputs totally. More specifically, as shown in Figure 2, this is a two-layer neural network with 50 hidden neurons. The inputs are the set of parameters in our CTMDP model, such as reward, cost function, arrival rates, departure rates, etc which makes the total input parameters be 68 (including all the values in the cost function); output are the $B(n_1, n_2)$ values for each state, in this case there are $11 * 5 = 55$ outputs, but since $b_1 n_1 + b_2 n_2 \le C$, the effective outputs is only 29, which is reflected in Figure 2.

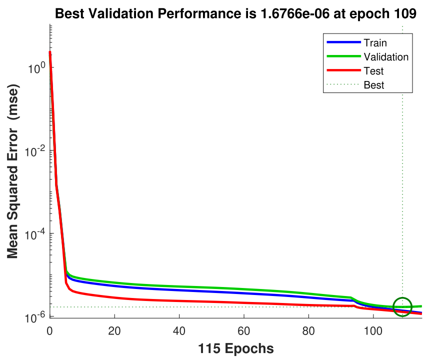


FIGURE 3. Neural network model performance plot.

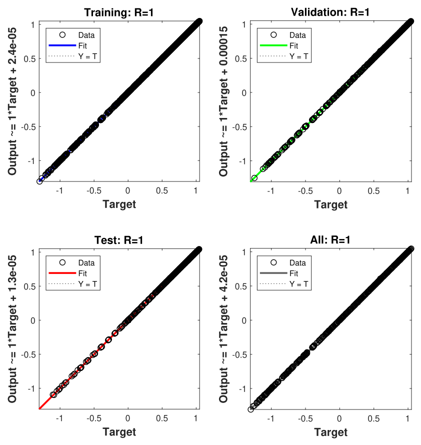


FIGURE 4. Neural network model regression plot.

After the model is trained, we can test the model with some other parameter settings which can be different from those in the initial training dataset. As seen from the performance and regression plot from the Neural Network model, it is observed that the machine learning model can do a good estimation of the $B(n_1, n_2)$ values.

First we test the parameter set with $R_1 = 0.2$, $R_2 = 0.3$ and cost function $a = 1$, which is not in the training dataset since $R_2 = 0.3$. It is seen that the values generated from Neural Network model (shown in Table 4) quite fits the values in Table 5, with their differences shown in Table 6.

Next we test the parameter set with $R_1 = 0.3$, $R_2 = 0.3$ and $a = 1$, while keeping all the other parameters unchanged. As seen from Table 7 and 8, the values from Neural Network model also fits with the real values from value iteration method. Their differences are listed in Table 9.

V. NUMERICAL ANALYSIS

In the last section, we have observed that we can get a good estimation of $B(n_1, n_2)$ values by using machine learning with Neural Network method for average optimal policies. Here we will do the numerical analysis with the same parameter setting as those in the last section, both with average models and discounted models.

TABLE 4. $B(n_1, n_2)$ values from neural network model at $R_1 = 0.2$ and $R_2 = 0.3$.

	0		$n_2 \rightarrow$		4
0	1.0050	0.7552	0.5030	0.2493	-0.0362
	0.9044	0.6548	0.4031	0.1418	0
	0.8049	0.5544	0.3024	0.0284	0
	0.7048	0.4527	0.1968	0	0
	0.6053	0.3533	0.0961	0	0
$n_1 \downarrow$	0.5051	0.2538	-0.0127	0	0
	0.4037	0.1502	0	0	0
	0.3033	0.0425	0	0	0
	0.2027	0	0	0	0
	0.1006	0	0	0	0
10	-0.0057	0	0	0	0

TABLE 5. $B(n_1, n_2)$ values with $R_1 = 0.2$ and $R_2 = 0.3$.

	0		$n_2 \rightarrow$		4
0	1.0041	0.7535	0.5018	0.2455	-0.0357
	0.9041	0.6534	0.4013	0.1419	0
	0.8041	0.5533	0.3001	0.0294	0
	0.7040	0.4531	0.1979	0	0
	0.6040	0.3528	0.0960	0	0
$n_1 \downarrow$	0.5039	0.2523	-0.0117	0	0
	0.4038	0.1504	0	0	0
	0.3035	0.0438	0	0	0
	0.2022	0	0	0	0
	0.1009	0	0	0	0
10	-0.0041	0	0	0	0

TABLE 6. Differences from Table 4 and Table 5.

	0		$n_2 \rightarrow$		4
0	0.0009	0.0017	0.0012	0.0038	-0.0005
	0.0003	0.0014	0.0018	-0.0002	0
	0.0009	0.0011	0.0023	-0.0010	0
	0.0008	-0.0004	-0.0011	0	0
	0.0013	0.0005	0.0001	0	0
$n_1 \downarrow$	0.0011	0.0015	-0.0011	0	0
	-0.0002	-0.0002	0	0	0
	-0.0002	-0.0014	0	0	0
	0.0005	0	0	0	0
	-0.0003	0	0	0	0
10	-0.0016	0	0	0	0

A. MODEL PERFORMANCE

First, we choose $a = 1$, the cost function becomes

$$f_1(b_1, n_1, b_2, n_2) = (n_1 + n_2) + 0.001b_1 + 0.002b_2.$$

With $R_1 = 0.2$ and $R_2 = 0.3$, the values of bias function $B(n_1, n_2)$ are listed at Table 5, "0" values means there are no such state in the system. It is seen that $B(n_1, n_2)$ values are concave decreasing on both n_1 and n_2 directions.

TABLE 7. $B(n_1, n_2)$ values generated from neural network model when $R_1 = 0.3$ and $R_2 = 0.3$.

	0		$n_2 \rightarrow$		4
0	1.0083	0.7585	0.5056	0.2487	-0.0615
	0.9079	0.6578	0.4055	0.1406	0
	0.8083	0.5577	0.3040	0.0160	0
	0.7084	0.4561	0.1991	0	0
	0.6084	0.3563	0.0976	0	0
$n_1 \downarrow$	0.5085	0.2562	-0.0179	0	0
	0.4074	0.1526	0	0	0
	0.3067	0.0397	0	0	0
	0.2058	0	0	0	0
	0.1036	0	0	0	0
10	-0.0065	0	0	0	0

TABLE 8. $B(n_1, n_2)$ values with $R_1 = 0.3$ and $R_2 = 0.3$.

	0		$n_2 \rightarrow$		4
0	1.0075	0.7566	0.5042	0.2446	-0.0615
	0.9075	0.6565	0.4035	0.1406	0
	0.8075	0.5563	0.3016	0.0167	0
	0.7075	0.4561	0.1997	0	0
	0.6074	0.3558	0.0975	0	0
$n_1 \downarrow$	0.5073	0.2550	-0.0170	0	0
	0.4072	0.1529	0	0	0
	0.3067	0.0409	0	0	0
	0.2054	0	0	0	0
	0.1039	0	0	0	0
10	-0.0051	0	0	0	0

TABLE 9. Differences from Table 7 and 8.

	0		$n_2 \rightarrow$		4
0	0.0008	0.0018	0.0015	0.0040	-0.0000
	0.0004	0.0012	0.0021	-0.0000	0
	0.0008	0.0013	0.0024	-0.0006	0
	0.0010	0.0000	-0.0006	0	0
	0.0010	0.0005	0.0001	0	0
$n_1 \downarrow$	0.0012	0.0012	-0.0009	0	0
	0.0002	-0.0003	0	0	0
	-0.0000	-0.0012	0	0	0
	0.0004	0	0	0	0
	-0.0004	0	0	0	0
10	-0.0014	0	0	0	0

As seen from Table 10 and 11, "1" means the system will accept the arrival, "0" means the system will reject the arrival and "-1" means there is no such state in the sytem. From the table we see the system will accept any T_1 and T_2 arrivals until it reaches the capacity limit. The average reward is $g = 0.4747$.

Next, we decrease R_1 to 0.1, and keep everything else unchanged,

TABLE 10. Actions for T_1 arrivals for cost function f_1.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	1	-1
	1	1	1	0	-1
	1	1	1	-1	-1
	1	1	1	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	1	-1	-1	-1
	1	0	-1	-1	-1
	1	-1	-1	-1	-1
	1	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 11. Actions for T_2 arrivals for cost function f_1.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 12. $B(n_1, n_2)$ values with $R_1 = 0.1$ and $R_2 = 0.3$.

	0		$n_2 \rightarrow$		4
0	1.0007	0.7506	0.5003	0.2493	-0.0069
	0.9006	0.6505	0.3999	0.1451	0
	0.8006	0.5505	0.2996	0.0437	0
	0.7006	0.4503	0.1972	0	0
	0.6006	0.3501	0.0956	0	0
$n_1 \downarrow$	0.5006	0.25	-0.0055	0	0
	0.4005	0.1484	0	0	0
	0.3004	0.0472	0	0	0
	0.1992	0	0	0	0
	0.0981	0	0	0	0
10	-0.0029	0	0	0	0

It is observed from Table 13 and 14, due to the drop in R_1, the model will not accept any T_1 arrivals to the system, meanwhile the actions for T_2 arrival is not affected, the model will accept the T_2 arrivals until the capacity limit. Also, the average reward drops to $g = 0.0758$.

Next, set $a = 0.1$ and keep everything else unchanged, the cost functions becomes

$$f_2(b_1, n_1, b_2, n_2) = 0.1n_1 + 0.1n_2 + 0.001b_1 + 0.002b_2$$

TABLE 13. Actions for T_1 arrivals for $R_1 = 0.1$.

	0		$n_2 \rightarrow$		4
0	0	0	0	0	0
	0	0	0	0	-1
	0	0	0	0	-1
	0	0	0	-1	-1
	0	0	0	-1	-1
$n_1 \downarrow$	0	0	0	-1	-1
	0	0	-1	-1	-1
	0	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 14. Actions for T_2 arrivals for $R_1 = 0.1$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 15. $B(n_1, n_2)$ values with $R_1 = 0.1$ and $a = 0.1$.

	0		$n_2 \rightarrow$		4
0	1.0076	0.9808	0.9511	0.9102	0.8285
	0.9975	0.9705	0.9387	0.8797	0
	0.9875	0.96	0.9253	0.8529	0
	0.9773	0.949	0.9019	0	0
	0.9672	0.9378	0.883	0	0
$n_1 \downarrow$	0.957	0.9261	0.8612	0	0
	0.9465	0.9071	0	0	0
	0.9354	0.8856	0	0	0
	0.9183	0	0	0	0
	0.9018	0	0	0	0
10	0.8828	0	0	0	0

TABLE 16. Actions for T_1 arrivals for $R_1 = 0.1$ and $a = 0.1$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	1	-1
	1	1	1	0	-1
	1	1	1	-1	-1
	1	1	1	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	1	-1	-1	-1
	1	0	-1	-1	-1
	1	-1	-1	-1	-1
	1	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 17. Actions for T_2 arrivals for $R_1 = 0.1$ and $a = 0.1$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 18. $B(n_1, n_2)$ values with $R_1 = 0.2$ and $a = 0.1$.

	0		$n_2 \rightarrow$		4
0	1.011	0.9839	0.9533	0.9082	0.8016
	1.001	0.9736	0.9408	0.8779	0
	0.9909	0.963	0.9267	0.8398	0
	0.9808	0.952	0.9035	0	0
	0.9706	0.9408	0.8843	0	0
$n_1 \downarrow$	0.9604	0.9288	0.8559	0	0
	0.9499	0.9096	0	0	0
	0.9386	0.8827	0	0	0
	0.9215	0	0	0	0
	0.9049	0	0	0	0
10	0.8818	0	0	0	0

It is observed from Table 16 and 17, with a much smaller cost function $a = 0.1$, the model again will accept any T_1 or T_2 arrivals to the system until the capacity limit. Also, the average reward comes to $g = 0.8802$.

Next, we increase R_1 back to 0.2, the values for the bias function are listed in Table 18. Since $R_1 = 0.2$ is larger, the actions for T_1 and T_2 arrivals would be the same as the previous case with $R_1 = 0.1$ which is that it will receive any arrivals up to the capacity limit. And the average reward increases to $g = 1.2815$.

We have done multiple cases with average model, next we do some tests with discounted model. With the default parameter settings, set discount factor be $\alpha = 0.1$,

From Table 19 it is observed that $X(n_1, n_2)$ values are concave decreasing on both n_1 and n_2 directions. From Table 20 and 21 the model would receive any arrivals up to the capacity limit.

TABLE 19. $X(n_1, n_2)$ values with $\alpha = 0.1$.

	0		$n_2 \rightarrow$		4
0	4.9085	4.664	4.4184	4.168	3.8922
	4.8095	4.5649	4.3188	4.0649	0
	4.7105	4.4658	4.2186	3.9532	0
	4.6114	4.3665	4.1171	0	0
	4.5124	4.2673	4.016	0	0
$n_1 \downarrow$	4.4133	4.1677	3.9091	0	0
	4.3142	4.0666	0	0	0
	4.2148	3.9608	0	0	0
	4.1144	0	0	0	0
	4.0139	0	0	0	0
10	3.9098	0	0	0	0

TABLE 20. Actions for T_1 arrivals for $\alpha = 0.1$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	1	-1
	1	1	1	0	-1
	1	1	1	-1	-1
	1	1	1	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	1	-1	-1	-1
	1	0	-1	-1	-1
	1	-1	-1	-1	-1
	1	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 21. Actions for T_2 arrivals for $\alpha = 0.1$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 22. $X(n_1, n_2)$ values with $\alpha = 0.9$.

	0		$n_2 \rightarrow$		4
0	0.6662	0.4615	0.2556	0.044	-0.1965
	0.5744	0.3696	0.163	-0.0547	0
	0.4827	0.2777	0.0696	-0.1606	0
	0.3909	0.1856	-0.0265	0	0
	0.2991	0.0935	-0.1214	0	0
$n_1 \downarrow$	0.2073	0.001	-0.2222	0	0
	0.1153	-0.0941	0	0	0
	0.0231	-0.1938	0	0	0
	-0.0711	0	0	0	0
	-0.1652	0	0	0	0
10	-0.263	0	0	0	0

TABLE 23. Actions for T_1 arrivals for $\alpha = 0.9$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	1	-1
	1	1	1	0	-1
	1	1	1	-1	-1
	1	1	1	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	1	-1	-1	-1
	1	0	-1	-1	-1
	1	-1	-1	-1	-1
	1	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 24. Actions for T_2 arrivals for $\alpha = 0.9$.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

Next we change the discount factor to 0.9, the $X(n_1, n_2)$ values with $\alpha = 0.9$ are listed in Table 22. It is observed with a larger discount factor, the $X(n_1, n_2)$ values drop a lot. But from Table 23 and 24 the actions for T_1 and T_2 are the same as the discount factor $\alpha = 0.1$.

B. THRESHOLD TO VALUES

From equation (1), the average optimal policy would get the maximum expected average reward if starting from an initial state. Therefore, by using the obtained thresholds in a policy, we can also calculate the corresponding expected average reward using equation (1) as shown in the following Table 25 and 26. With the default parameter settings and cost function f_2, we know the average reward is $g = 1.2815$. Using rate uniformization technique, the expected time between two epoch is $\frac{1}{c}$ and the expected average reward between two epochs is $\frac{1.2815}{c}$. The calculation runs about 10000 epochs, which means we can collect expected average reward 10000 times.

TABLE 25. Threshold to values at $\alpha = 0.1$ and cost function f_2.

	0		$n_2 \rightarrow$		4
0	110.4938	110.4676	110.4377	110.3936	110.2879
	110.4846	110.4581	110.4261	110.3641	0
	110.4754	110.4484	110.4129	110.3268	0
	110.4661	110.4382	110.3906	0	0
	110.4568	110.4278	110.3723	0	0
$n_1 \downarrow$	110.4475	110.4167	110.3447	0	0
	110.4378	110.3983	0	0	0
	110.4274	110.3723	0	0	0
	110.4112	0	0	0	0
	110.3954	0	0	0	0
10	110.3732	0	0	0	0

TABLE 26. $X(n_1, n_2)$ values with $\alpha = 0.1$ and cost function f_2.

	0		$n_2 \rightarrow$		4
0	100.0195	99.9932	99.9633	99.9191	99.8133
	100.0103	99.9836	99.9516	99.8895	0
	100.0009	99.9739	99.9383	99.8522	0
	99.9916	99.9636	99.9159	0	0
	99.9822	99.9532	99.8975	0	0
$n_1 \downarrow$	99.9728	99.9419	99.8698	0	0
	99.9631	99.9235	0	0	0
	99.9526	99.8974	0	0	0
	99.9362	0	0	0	0
	99.9204	0	0	0	0
10	99.8981	0	0	0	0

First set $c = 116$, we have $\tilde{g} = \frac{g}{c} = 0.011$, the reward collected if starting from each state after 10000 epochs is shown in Table 25, which is around 110.

Next set $c = 128.15$, this time the expected average reward between two epochs is $\tilde{g} = \frac{g}{c} = 0.01$, the reward collected if starting from each state after 10000 epochs is shown in Table 26, which is around 100.

Keeping the parameter settings unchanged, for discounted models, from equation (2), the total discounted expected reward is the expected discounted summation of reward if starting from an initial state. Therefore, by using the obtained thresholds in a policy, we can also calculate the corresponding expected discounted reward using equation (2) as shown in the following Table 27. The calculation runs about 10000 epochs, the expected time between two epoch is $\frac{1}{c}$ after rate uniformization technique, but the change in values can be ignored after running 5000 epochs.

By comparing the Table 27 and 28, we know the difference for two different calculations is very small, thus either method can equally be utilized whenever necessary.

C. UPPER BOUND ANALYSIS

From **Theorem 3** and **Theorem 4**, the upper bound for the average model and discounted model is largely related to $\beta_0(n_1 + 1, n_2)$ and $\Delta_{n_1} f(b_1, n_1, b_2, n_2)$. It can be observed that n_1 can goes up to $\lfloor 40/4 \rfloor = 10$, and n_2 is with limit of $\lfloor 40/10 \rfloor = 4$. Then we have the values of $\beta_0(n_1, n_2)$ listed in Table 28.

TABLE 27. Threshold to values at $\alpha = 0.1$ and cost function f_2.

	0		$n_2 \rightarrow$		4
0	12.8306	12.8046	12.7752	12.7316	12.6266
	12.8211	12.7948	12.7633	12.7018	0
	12.8116	12.7849	12.7498	12.6643	0
	12.802	12.7744	12.7272	0	0
	12.7925	12.7638	12.7087	0	0
$n_1 \downarrow$	12.7828	12.7524	12.6808	0	0
	12.7729	12.7338	0	0	0
	12.7622	12.7075	0	0	0
	12.7457	0	0	0	0
	12.7296	0	0	0	0
10	12.7072	0	0	0	0

TABLE 28. $X(n_1, n_2)$ values with $\alpha = 0.1$ and cost function f_2.

	0		$n_2 \rightarrow$		4
0	12.833	12.8066	12.7767	12.7326	12.6272
	12.823	12.7963	12.7643	12.7023	0
	12.813	12.7859	12.7503	12.6644	0
	12.803	12.7749	12.7273	0	0
	12.793	12.7639	12.7082	0	0
$n_1 \downarrow$	12.7829	12.7519	12.6799	0	0
	12.7724	12.7328	0	0	0
	12.7613	12.7061	0	0	0
	12.7443	0	0	0	0
	12.7277	0	0	0	0
10	12.7048	0	0	0	0

TABLE 29. $\beta_0(n_1, n_2)$ values for each state.

	0		$n_2 \rightarrow$		4
0	6	10	14	18	22
	16	20	24	28	0
	26	30	34	38	0
	36	40	44	0	0
	46	50	54	0	0
$n_1 \downarrow$	56	60	64	0	0
	66	70	0	0	0
	76	80	0	0	0
	86	0	0	0	0
	96	0	0	0	0
10	106	0	0	0	0

Set the cost function be $f_3(b_1, n_1, b_2, n_2) = 3.3n_1 + n_2 + 0.001b_1 + 0.002b_2$, then we have $\Delta_{n_1} f_3(b_1, n_1, b_2, n_2) = 3.3$, and since $R_1 = 0.2$, $\beta_0(1, 0) = 16$, there is

$$\frac{1}{\beta_0(1,0)} \Delta_{n_1} f_3(b_1, n_1, b_2, n_2) = \frac{3.3}{16} > R_1 = 0.2,$$

According to **Theorem 3**, the system will not accept any T_1 arrivals from state $(0, 0)$, the $X(n_1, n_2)$ values and state actions for T_1 arrivals is listed in Table 25, 26 and 27. From these tables it is observed that although the system will not accept any T_1 arrivals, it would accept T_2 arrivals up to the capacity limit. Also, since no T_1 arrival is accepted, the average reward is again $g = 0.0758$.

TABLE 30. $B(n_1, n_2)$ values with cost function f_3.

	0		$n_2 \rightarrow$		4
0	1.0007	0.7506	0.5003	0.2493	-0.0069
	0.6706	0.4205	0.1699	-0.0849	0
	0.3406	0.0905	-0.1604	-0.4163	0
	0.0106	-0.2397	-0.4928	0	0
	-0.3194	-0.5699	-0.8244	0	0
$n_1 \downarrow$	-0.6494	-0.9	-1.1555	0	0
	-0.9795	-1.2316	0	0	0
	-1.3096	-1.5628	0	0	0
	-1.6408	0	0	0	0
	-1.9719	0	0	0	0
10	-2.3029	0	0	0	0

TABLE 31. Actions for T_1 arrivals with cost function f_3.

	0		$n_2 \rightarrow$		4
0	0	0	0	0	0
	0	0	0	0	-1
	0	0	0	0	-1
	0	0	0	-1	-1
	0	0	0	-1	-1
$n_1 \downarrow$	0	0	0	-1	-1
	0	0	-1	-1	-1
	0	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 32. Actions for T_2 arrivals with cost function f_3.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 33. $X(n_1, n_2)$ values with cost function f_3.

	0		$n_2 \rightarrow$		4
0	0.8801	0.6362	0.392	0.147	-0.1039
	0.5534	0.3094	0.0648	-0.1845	0
	0.2267	-0.0174	-0.2622	-0.5128	0
	-0.1001	-0.3443	-0.5918	0	0
	-0.4268	-0.6713	-0.9203	0	0
$n_1 \downarrow$	-0.7536	-0.9982	-1.2482	0	0
	-1.0804	-1.3266	0	0	0
	-1.4072	-1.6548	0	0	0
	-1.7354	0	0	0	0
	-2.0633	0	0	0	0
10	-2.3912	0	0	0	0

TABLE 34. Actions for T_1 arrivals with cost function f_3.

	0		$n_2 \rightarrow$		4
0	0	0	0	0	0
	0	0	0	0	-1
	0	0	0	0	-1
	0	0	0	-1	-1
	0	0	0	-1	-1
$n_1 \downarrow$	0	0	0	-1	-1
	0	0	-1	-1	-1
	0	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

TABLE 35. Actions for T_2 arrivals with cost function f_3.

	0		$n_2 \rightarrow$		4
0	1	1	1	1	0
	1	1	1	0	-1
	1	1	1	0	-1
	1	1	0	-1	-1
	1	1	0	-1	-1
$n_1 \downarrow$	1	1	0	-1	-1
	1	0	-1	-1	-1
	1	0	-1	-1	-1
	0	-1	-1	-1	-1
	0	-1	-1	-1	-1
10	0	-1	-1	-1	-1

Next, we do a similar test with discounted model, keeping all the parameters in the average model, and set the discount factor be $\alpha = 0.1$, we have

$$\frac{1}{\alpha + \beta_0(1,0)} \Delta_{n_1} f_3(b_1, n_1, b_2, n_2) = \frac{3.3}{16.1} > R_1 = 0.2,$$

According to **Theorem 4**, the system will not accept any T_1 arrivals from state $(0, 0)$, the $X(n_1, n_2)$ values and state actions for T_1 arrivals is listed in Table 25, 26 and 27. From these tables it is observed that although the system will not accept any T_1 arrivals, it would accept T_2 arrivals up to the capacity limit.

VI. CONCLUSION AND DISCUSSION

In this paper, a general situation is investigated for when datacenter can determine the cost of using resources (number of virtual machines) and a Cloud service user can decide whether it will pay the price for the resource or not when there is an incoming application task. By establishing a Continuous-Time Markov Decision Process (CTMDP) model for both the average optimal model and a discounted optimal model, respectively, the optimal policy for each model for admitting tasks is verified as a state-related control limit policy. Further, a detailed statement and verification of the upper bound for the optimal policy, and a comprehensive set of experiments on various cases to validate our proposed solution are provided.

Particularly, the machine learning method is incorporated to show that the complex relations between parameters of the model and the optimal decision-making policies can be learned through a feed-forward neural network, which is a quite simple tool from machine learning's view. But it is not our the main concern to compare different machine learning methods in this paper since the simple ANN method has already given us a very good result. We plan to study more machine learning methods in the future work.

In our future work, we will analyze the optimal system resources toward the maximal system rewards if only with partial system information, comparison between different machine learning methods, etc.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments and suggestions, which helped a lot in improving the quality of this paper.

REFERENCES

[1] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-based resource allocation in vehicular cloud computing systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7920–7928, Dec. 2015.

[2] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. Internet Meas. Conf.*, Nov. 2009, pp. 202–208. [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-nature-of-data-center-traffic-measurements-and-analysis/

[3] B. Zhou, J. Wu, L. Wang, F. Zhang, and Z. Liu, "Joint optimization of server and network resource utilization in cloud data centers," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[4] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: Challenges and solutions," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 98–105, Mar. 2015.

[5] P. Khanna and S. Jain, "Distributed cloud federation brokerage: A live analysis," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 738–743.

[6] F. Zhang, Y. Liu, W. Liu, H. Liang, Y. Zang, and M. Fu, "Stochastic game between cloud broker and cloudlet for mobile cloud computing," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2017, pp. 1–5.

[7] P. Prukkantragorn and K. Tientanopajai, "Price efficiency in high performance computing on Amazon elastic compute cloud provider in compute optimize packages," in *Proc. Int. Comput. Sci. Eng. Conf. (ICSEC)*, Dec. 2016, pp. 1–6.

[8] M. Malawski, M. Kuźniar, P. Wójcik, and M. Bubak, "How to use Google app engine for free computing," *IEEE Internet Comput.*, vol. 17, no. 1, pp. 50–59, Jan./Feb. 2013.

[9] C. Kotas, T. Naughton, and N. Imam, "A comparison of Amazon Web services and microsoft azure cloud platforms for high performance computing," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2018, pp. 1–4.

[10] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2505–2510.

[11] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.

[12] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, and M. J. Neely, "Power cost reduction in distributed data centers: A two-time-scale approach for delay tolerant workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 200–211, Jan. 2014.

[13] Z. Zheng, M. Li, X. Xiao, and J. Wang, "Coordinated resource provisioning and maintenance scheduling in cloud data centers," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 345–349.

[14] D. Huang, X. Zhang, M. Kang, and J. Luo, "MobiCloud: Building secure cloud framework for mobile computing and communication," in *Proc. IEEE Int. Symp. Service Oriented Syst. Eng.*, Jun. 2010, pp. 27–34.

[15] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An SMDP-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Trans. Veh. Technol.*, vol. 61, no. 5, pp. 2222–2232, Jun. 2012.

[16] K. Hazeghi and M. L. Puterman, "Markov decision processes: Discrete stochastic dynamic programming," *J. Amer. Stat. Assoc.*, vol. 90, p. 392, Mar. 1995.

[17] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. San Rafael, CA, USA: Morgan & Claypool, 2013. [Online]. Available: https://ieeexplore.ieee.org/document/6812618

[18] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. M. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 689–702, Jun. 2015.

[19] A. Taha, S. Manzoor, and N. Suri, "SLA-based service selection for multi-cloud environments," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 65–72.

[20] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 854–862.

[21] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. Internet Meas. Conf.*, Nov. 2010, pp. 267–280. [Online]. Available: https://www.microsoft.com/en-us/research/publication/network-traffic-characteristics-of-data-centers-in-the-wild/

[22] S. M. Ross, *Stochastic Processes*. Hoboken, NJ, USA: Wiley, 1995.

[23] W. Li and X. Chao, "Call admission control for an adaptive heterogeneous multimedia mobile network," *IEEE Trans. Wireless Commun.*, vol. 6, no. 2, pp. 515–525, Feb. 2007.

[24] C. Xiuli, C. Hong, and L. Wei, "Optimal control for a tandem network of queues with blocking," *Acta Math. Appl. Sinica*, vol. 13, no. 4, pp. 425–437, Oct. 1997, doi: 10.1007/BF02009552.

[25] W. Ni, W. Li, and M. Alam, "Determination of optimal call admission control policy in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 1038–1044, Feb. 2009.

[26] W. Ni and W. W. Li, "Optimal resource allocation for brokers in media cloud," in *Computational Data and Social Networks*, X. Chen, A. Sen, W. W. Li, and M. T. Thai, Eds. Cham, Switzerland: Springer, 2018, pp. 103–115.

[27] P. Zhao, X. Lin, Y. Wang, S. Chen, and M. Pedram, "Hierarchical resource allocation and consolidation framework in a multi-core server cluster using a Markov decision process model," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 2, no. 3, pp. 118–126, 2017.

[28] M. Alicherry and T. V. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 647–655.

[29] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 355–359.

[30] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, "Profit maximization for cloud brokers in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 190–203, Jan. 2019.

[31] V. Di Valerio, V. Cardellini, and F. L. Presti, "Optimal pricing and service provisioning strategies in cloud systems: A stackelberg game approach," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun./Jul. 2013, pp. 115–122.

[32] J. He, D. Wu, Y. Zeng, X. Hei, and Y. Wen, "Toward optimal deployment of cloud-assisted video distribution services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1717–1728, Oct. 2013.

[33] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2005.

WENLONG NI received the B.Sc. degree from Tsinghua University, China, in 1999, the M.Sc. degree from Kobe University, Japan, in 2003, and the Ph.D. degree from The University of Toledo, USA, in 2008. Within last ten years, since receiving the Ph.D. degree, he has been a Research Engineer in several companies, including top 500 Company like Microsoft and Sinopec. He has over ten years of industrial experience with computer technology, including high performance computation, big data analysis, and processing. During that career period, he has made several recognized major contributions in complex wireless networks research through publications, such as in the IEEE Transactions on Wireless Communications, ICDCS, Globecom, ICC, and WCNC. In 2018, he returned to his hometown and started a new position as an Oversea Elite Professor with the School of Computer Information Engineering, Jiangxi Normal University. His current research interests include performance evaluation, decision analysis and the applications in cloud computing, big data analysis, recommendation systems, communications networks, machine learning, and reinforcement learning.

YUHONG ZHANG received the B.Sc. degree from Shandong University, China, in 1984, the M.Sc. degree from Manitoba University, Canada, in 2000, and the Ph.D. degree from The University of Toledo, USA, in 2008. After graduating, she then joined the Department of Engineering, Texas Southern University, as an Assistant Professor, where she is currently an Associate Professor. She has authored or coauthored for a number of peer-reviewed articles in professional journals and the proceedings of conferences. She has been actively supported through several grants as PIs from the National Science Foundation (NSF) and the Department Homeland Security (DHS) over last ten years. Her research interests include in the areas of wireless sensor networks, digital signal processing, digital image/video processing, and the application of neural networks.

WEI W. LI (M'99–SM'06) received the B.Sc. degree from Shaanxi Normal University, Xi'an, China, in 1982, the M.Sc. degree from the Hebei University of Technology, Tianjin, China, in 1987, and the Ph.D. degree from the Chinese Academy of Sciences, Beijing, China, in 1994. He was a tenured Associate Professor with the Department of Electrical Engineering and Computer Science, The University of Toledo, Toledo, OH, USA, and a tenured Track Assistant Professor with the Department of Electrical and Computer Engineering, University of Louisiana at Lafayette, Lafayette, LA, USA. He is currently a Professor with the Department of Computer Science and the Director/PI of the NSF-CREST Center for Research on Complex Networks, Texas Southern University, Houston, TX, USA. He is also the author/coauthor of six books and over 160 peer-reviewed articles in professional journals and the proceedings of international conferences. His research interests include dynamic control and optimization of energy-efficient wireless sensor networks, evaluation, complexity, power connectivity, and coverage for wireless sensor networks, adaptation, design, and implementation of dynamic models in wireless multimedia systems, theoretic foundations and advanced analysis in real-time, hybrid, and embedded systems, and performance evaluation of biomolecule networks, queuing networks, and reliability systems. He is serving or has served as a Steering Committee Member, or the General Co-Chair, or the Technical Program Committee Co-Chair, or the Track Chair, or a Technical Program Committee Member, for a number of professional conferences, including INFOCOM, ICDCS, Globecom, ICC, and WCNC. He is currently serving as an Editor for several professional journals.