Achieving *Fair* Load Balancing by Invoking a Learning Automata-based Two Time Scale Separation Paradigm

Anis Yazidi, *Senior Member, IEEE,* Ismail Hassan, Hugo L. Hammer, B. John Oommen, *Life Fellow, IEEE*

***Abstract*—In this paper[1], we consider the problem of Load Balancing (LB), but unlike the approaches that have been proposed earlier, we attempt to resolve the problem in a *fair* manner (or rather, it would probably be more appropriate to describe it as an ϵ-fair manner, because although the LB can, probably, never be totally fair, we achieve this by being "as close to fair as possible"). The solution we propose invokes a novel stochastic Learning Automata (LA) scheme, so as to attain a distribution of the load to a number of nodes, where the performance level at the different nodes is approximately equal, and where each user experiences approximately the same Quality of the Service (QoS) irrespective of which node he is connected to. Since the load is dynamically varying, static resource allocation schemes are doomed to underperform. This is further relevant in cloud environments, where we need dynamic approaches because the available resources are unpredictable (or rather, uncertain) by virtue of the shared nature of the resource pool. Further, we prove here that there is a coupling involving the LA's probabilities and the dynamics of the rewards themselves, which renders the Environments to be non-stationary. This leads to the emergence of the so-called property of "stochastic diminishing rewards". Our newly-proposed novel LA algorithm ϵ-optimally solves the problem, and this is done by resorting to a two-time scale based stochastic learning paradigm. As far as we know, the results presented here are of a pioneering sort, and we are unaware of any comparable results.**

***Index Terms*—Continuous Learning Automata, Resource Allocation, Fair Load Balancing.**

I. Introduction

In this paper, we consider the problem of Load Balancing (LB), which is extremely pertinent in today's highly-connected world. To put the problem in the right perspective, we observe that, unarguably, computers and information technology, have experienced enormous growth and development over the past three decades. This unalterable trend has profoundly affected societies worldwide, in every sense of the word. Products and services that were traditionally delivered through other means are, currently, online services.

Unlike the scenario a few decades ago, where one "connected" directly to an institution's machine, most of these services are now being executed on the internet. Since more than 50 billion devices will be connected to the Internet by 2020 [28], one understands that the traditional model of having in-house computers and resources is not going to be a sustainable and viable option. Rather, to cope with the sheer increase in the number of users and devices interacting with the machines, and the respective services delivered online, government and business institutions are reducing their investments in on-premise IT infrastructure. Indeed, to mitigate the super-exponential increases in the corresponding communication and computational costs, they are moving to, and increasing their spending on, cloud-based services [33]. In this context, we mention that the National Institute of Standards and Technology (NIST) defines "cloud computing" as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [19]. Clearly, one has to now consider how all these services can be distributed over the cloud of computers. This, precisely, involves the problem of LB.

LB, like many other related problems [13], many instances of LB are considered NP-Hard problems [31]. Thus, we will never be able to solve the problem so as to allocate the resources in a perfectly-balanced manner. Unlike the approaches that have been proposed in the literature [11] such as Round Robin, Weighted Round Robin, Power of two choices, Least Connection, Weighted Least Connection etc..., we attempt to resolve the problem in an "almost fair" manner, and we shall refer to such an allocation as as an ϵ-fair balance. In other words, we attempt to achieve this by being "as close to fair as possible". While one can attempt to do this intelligently using any of the available AI-based paradigms, the solution we propose invokes a novel stochastic Learning Automata (LA) scheme. Our LA-based solution distributes the load to a number of nodes, where the performance level at the different nodes is

A. Yazidi, Ismail Hassan and H. Hammer are with the Department of Computer Science, Oslo Metropolitan University, Oslo, Norway.

B. John Oommen: Life Fellow IEEE and Fellow IAPR, School of Computer Science, Carleton University, Ottawa, ON: K1S5B6, Canada; oommen@scs.carleton.ca. This author is also an Adjunct Professor at the University of Agder, in Grimstad, Norway.

[1]The work of the last author was partially supported by NSERC, the Natural Sciences and Engineering Council of Canada. We are very grateful for the feedback from the anonymous Referees of the *original* submission. Their input significantly improved the quality of this final version.

approximately equal, and where each user experiences approximately the same Quality of the Service (QoS) irrespective of the node he is connected to. Although LA has been applied classically to solve many resource allocation problems including load balancing, to the best of our knowledge the current work is distinct in two aspects. First, to the best of our knowledge, there is no theoretical analysis of any load balancing algorithm in the field of LA. Load balancing induces usually dynamicity of the environment as the LA actions will continously alter the load distribution and consequently render the environment non-stationary. Thus, such settings deviate from the classical multi-armed bandit settings where the environment is rather static and the reward distribution is not influenced by the actions of the LA. The analysis of such cases is much more involved than the stationary cases. Please note that, probably, the most notable example of a theoretical treatment is due to [31], [47] where the load balancing problem is mapped into a coordinated strategic game. Second, the success of adopting an LA for solving a rea-life problem is dependent on an appropriate choice, or more precisely, an appropriate engineering of the reward function. Most of the engineerred reward functions in the context of LA-based load balancing solutions rely on ”the response time” of a server solely. In this paper, we used a modified version of the reward that can infer fairness based on a dynamic comparison threshold.

How then should a cloud-based service model differ from a more traditional model? From the reported literature [12], we submit that a cloud-based infrastructure should make it easy for a customer to request a resource, and to have that resource provisioned and ready for use, in minutes, rather than days or weeks. The ability to scale the available resources *on demand*, with little or no downtime, is another factor which makes the cloud preferable over traditional enterprise data centers.

The cloud-based computing paradigm has transformed the IT industry profoundly, paving the way to foster new concepts such as DevOps, and microservices. However, to stay competitive and to also ensure customer satisfaction, companies offering online services aim to quickly deliver new features to their customers. Developing and deploying software as a monolithic application does not fully take advantage of the benefits of a "cloud computing" paradigm, and many companies are considering migrating towards microservices [7] and a Cloud-Native Application approach.

While having many benefits, cloud computing still has some challenges when it comes to offering an optimized system, and a fair allocation of resources. Available cloud models do not adequately capture uncertainty, non-homogeneity, and dynamic performance changes that are inherent to non-uniform and shared infrastructures [41]. One of the viable ways to address challenges related to dynamic performance changes associated with any uncertainties in the load distribution, is to employ a LB technique.

LB is the process of distributing workloads fairly among multiple hosts. The major advantage of deploying a LB solution is to be able to handle more traffic than a single host can tackle. Another advantage of LB is that such a system offers high availability such that if one service fails, others are available to ensure that the application stays up and running.

In order to achieve an optimal distribution of workloads to any numbers of hosts, several algorithms have been developed throughout the years. LB algorithms are mainly classified as being static or dynamic.

Static LB schemes assume that the information governing the LB-oriented decisions are known in advance [32]. The LB decisions are made deterministically or probabilistically when the system starts or boots, and remain constant during runtime. Every time the system restarts, the same values gets loaded. Static LB algorithms are mostly suitable for stable environments with homogeneous systems.

The nature of a data center or of a cloud implicitly requires dealing with a mixture of stochastic processes [40]. In contrast to static algorithms, dynamic LB algorithms do not require prior knowledge or configuration of the system. To make more fair load distribution decisions, dynamic LB algorithms monitor the current runtime state of the system, and adapt to changing loads. The experiments that we report tacitly imply that the servers are not homogenous, as they need not necessarily be homogenous specially in cloud environment. Indeed, one of the reasons for this is the types of hardware used for the servers may be different as well as the unpredictability of the resources in a cloud environment. ANIS TO JOHN:I CORRECTED THE TEXT HERE THAT YOU HAD IT IN RED COLOR AND CHANGED IT TO BLUE. THE SERVERS ARE NOT HOMOGENEOUS IN OUR EXPERIMENTS BCS THEIR SERVICE RATES ARE DIFFERENT.

A. Distinctive Properties of Our Solution

Without going into any details of the arguments presented in the body of the paper, it is prudent to mention the distinctive properties of our proposed solution when it concerns the learning mechanism itself, and the associated analysis. In all brevity, they can be listed as below:

1) By virtue of the "fair balance" paradigm, the learning algorithm initiated by the LA proposed here is distinct from all the families of LA described in the LA-based load balancing literature such as [23]. This includes those from the previously-reported families of fixed structure, variable structure, discretized and estimator-based LA.
2) To achieve a fair load balance, we encounter an irony. Thus is, indeed, the fact that the more often an "action" is chosen, the likelihood of the LA choosing it even more, must subsequently decrease. In other words, the rewards that are received for any action must decrease as the action is chosen more frequently. This is contrary to what the

properties of absolute expedience and ϵ-optimality entail, especially since, in these cases, the LA aim to converge to an absorbing barrier in the probability space. To the best of our knowledge, LA which possess the phenomenon mentioned here have not been proposed in the literature.

3) Our solution is characterized by the amazing property that as any specific $p_i(t)$ increases, the corresponding reward probability of the action in question, decreases. We refer to this phenomenon as the "stochastic diminishing return property". Informally, this means that the more an action is chosen, the less it will be rewarded. This involves the two-time scale LA with barriers, as we shall explain presently, and also facilitates fair LB.
4) The analysis methods we use here are both distinct and unique. The mathematical techniques used for the various families of LA described in the literature are each distinct in their own right. The methodology for the family of Fixed Structure Stochastic Automata (FSSA) involves formulating the Markov chain for the LA, computing its equilibrium probabilities, and then computing the asymptotic action selection probabilities. The proofs of convergence for Variable Structure Stochastic Automata (VSSA) involve the theory of small-step Markov processes, distance diminishing operators, and the theory of regular functions. The proofs for discretized LA involve the asymptotic analysis of the Markov chain that represents the LA in the discretized space, whence the *total* probability of convergence to the various actions is evaluated. The proof of Estimator/Pursuit algorithms concerns two intertwined phenomena, i.e., the convergence of the reward estimates and the convergence of the action probabilities themselves. The proof methodology considered in this paper utilizes the theory of small-step Markov processes and distance diminishing operators, but unlike the existing LA, they do not converge to absorbing barriers but fixed-points in the corresponding probability vector space.
5) Historically, the metric for analyzing LA has generally been ϵ-optimality and Absolute Expedience. Indeed, the concept of the Lypanuov stability of an LA solution has been rarely used with few exceptions [9]. This is, indeed, the metric that we have invoked.

B. Contributions of the Paper

The contributions of this paper can be summarized as follows:

- We present an LA solution for ensuring fairness of load distribution in the field of LB.
- We present deep theoretical results that prove the convergence of our scheme.
- We use some of the most recent advances in the field of LA which combines the time-separation paradigm and the phenomenon of artificial barriers, introduced by Yazidi and Hammer in [51] and [52] respectively.
- We prove that the equilibrium point which the algorithm converges to is asymptotically Lyapunov stable. The concept of the Lypanuov stability of an LA solution has been rarely used, except for a very few reported results [9].
- We provide some experimental results that confirm and justify our theoretical assertions.

C. Organization of the Paper

The paper is organized as follows. The background and related work is first presented in Section II. In Section III, we give an introduction to the theory of LA which is central to this paper. Section IV includes the details of our proposed solution, where we present the scheme itself in Subsection IV-C, and report the theoretical results proving its convergence to an optimal equilibrium, in Subsection IV-D. Thereafter, in Section V, we include the results of rigorous simulations that confirm the theoretical results. Section VI concludes the paper.

II. Background and Related Work

Historically, LB and task scheduling have been two closely-related research areas that have been widely investigated. However, the literature that reports the use of stochastic LA to achieve these, has been limited.

A stochastic LA model for the decentralized control of job scheduling in distributed processing systems was presented by [22]. The algorithm proposed by these authors operates with absolutely no prior knowledge about the job, but rather adapts to the changing loads of the hosts. The aim of the proposed algorithm was to provide load balanced jobs to a number of hosts, and to improve the response time while achieving this.

To minimize the response time, a heuristic LB scheme based on the concept of a stochastic LA was implemented by the author of [15]. Depending on the status of the current load distribution, a new task would be scheduled to be executed either locally or on a remote host. The paper employed a learning scheme with a reward constant $\boldsymbol{A}$ of ***0.25*** and a penalty constant $\boldsymbol{B}$ of ***0.3***. Although many fine details were not reported in the paper, the author claimed to also have examined other influences on different numbers of automaton states, and the behavior of the scheduler under different network sizes.

The authors of [23] presented a framework based on LA that is capable of addressing some of the challenges and demands of various cloud applications. The proposed framework analysis invoked various performance metrics such as response time, parallel execution speed, and job priority. These metrics were then used to select the appropriate resources, using LA.

A Cost Aware S-model Reward Epsilon-Penalty (CA-S) method was proposed in [46]. The authors employed a LA-based solution that sought to reduce the average cost in serving web requests with replicated web servers, deployed in different geographical regions. To minimize the cost, the LA made routing decisions for each incoming request by assessing response times and energy prices at the different server locations through action selection probabilities. The paper reported very promising experimental results by using the proposed CA-S method. These results demonstrated that the total average cost of serving web responses could be reduced up to 33% when compared to the minimum cost flow dynamic server selection algorithm, and up to 49.2% compared to the traditional Round-Robin method.

The problem of optimal priority assignment among two streams of jobs with unknown characteristics, each with a random service time and a random arrival time, was addressed in the doctoral thesis of Meybodi [20]. A threshold was computed which was the average service time taken over both streams, and the response time of a served dispatched request from the chosen stream by the LA was compared to that threshold for the inferred LA response. This idea of using response time-based threshold in a queuing system as a mechanism for inferring the response of LA to constitute the rewards/penalties appeared also in [3]. Similarly, in our current work we resort to a dynamic threshold computed using a type of moving average, in contrast to a stationary type of estimator found in the seminal work of Meybodi [20] and Meybodi and Lakshmivarhan [21]. Apart from the queuing system model, our paper is different from the work in [20]. Indeed, the LA proposed by Meybodi was absorbing because the optimal solution was to be exclusively chosen from one of the priority streams. Furthermore, the dynamics of the reward probabilities addressed here are much more complicated in our problem setting. An alternative solution to the priority assignment problem based on FSSA was proposed by Srikantakumar [14]. The problem was revisited recently using the theory of Petri Nets and LA in [43], [44]. But as our model and solution are distinct, in the interest of brevity, we shall not expand on these papers any further. Finally, we emphasize that although LA has been applied in few instances in the litterature for solving load balancing problems, we are not aware of any theoretical treatments of the problems. The theoretical analysis is for this type of LA applications is intrinsically hard due the dynamic nature of the environment as the reward dynamics are coupled with the action changes. LA algorithms in this direction are rather presented as heuristics with no theoretical guarantees. The only may be attempt to cast an LA-based load balancing algorithm into a theoretical framework was done in a series of work by the same research group [31], [47] where the load balancing problem is mapped into a coordinated game.

III. Stochastic Learning Automata

We shall now proceed to present a brief overview of LA [1], which is the toolkit we will use to solve the problem.

In psychology, learning is characterized as the act of modifying one's behavior as a result of acquiring knowledge from past experience. In the field of automata theory, an automaton can be described as a self-operating machine or control mechanism consisting of a set of states, a set of outputs or actions, an input, a function that maps the current state and input to the next state, and a function that maps a current state (and input) to the current output.

The term Learning Automata (LA) was first presented in the survey paper by Narendra and Thathachar (cited in [1]). LA are well suited for systems with noisy and incomplete information about the Environment in which they function [1], [16], [26], [27], [29], [34]. The Environment is generally stochastic and the LA lacks prior knowledge as to which action is the optimal one. Stochastic LA, which are probabilistic finite state machines, attempt to solve this problem by choosing an initial action randomly, and then updating the action probabilities based on the response received. The action chosen is dependent on the action probability distribution vector, which, in turn, is updated based on the reward/penalty input that the LA receives from the random Environment. This process is repeated until the optimal action is, hopefully, achieved.

The research on LA is comprehensive and over the past decades, several classes have been proposed. LA are mainly categorized as being Fixed Structure Stochastic Automata (FSSA) or Variable Structure Stochastic Automata (VSSA). In FSSA, the mapping between transition and output functions is time invariant. Initial research into LA was mainly focused on FSSA. Tsetlin, Krylov, and Krinsky [42] demonstrated several models of this class of automata. Gradually, research into LA advanced towards VSSA. LA schemes in this category possess transition and output functions which evolve as the learning process proceeds [30]. In VSSA, the state transitions or the action probabilities are updated at every time step. This class of automata was introduced by Varshavskii and Vorontsova in the early 1960's [45].

LA can further be classified as either ergodic or endowed with absorbing barriers based on their Markovian properties. In an ergodic LA system, the final steady state is independent of the initial state. As opposed to this, for LA with absorbing barriers, the steady state depends on the initial state and once the LA has converged, it will be locked into a so-called absorbing barrier. Furthermore, while ergodic VSSA are suitable for non-stationary environments, absorbing barrier VSSA are preferred in stationary environments. As opposed to these, a unique property of the work in [53] is that the action with the highest probability may not be the same one being chosen most frequently.

Stochastic LA had been utilized in many applications over the years. Recent applications of LA include resource usage prediction algorithm for cloud computing environment [35], channel selection in cognitive radio/dynamic spectrum access for WiMAX networks [24], distributed network formation [6], solutions to the single elevator problem [10], efficient decision making mechanism for stochastic nonlinear resource allocation [52], dynamic cost-aware routing of web requests [46], learning periodic spatio-temporal patterns [50], content placement in cooperative caching [49], resource selection in computational grids [8], determining proper subset size in high-dimensional spaces [38] and image segmentation [37], to mention a few.

IV. LA Load Balancing Model

In this section we present our LA model for LB, as well as the theoretical proofs for the solution's convergence.

A. Model

We consider a scenario where we have a set of r servers. Each server is modelled as an $M/M/1$ queue, which means that arrivals are modelled by a Poisson process with some intensity λ_i, and the job service times have an exponential distribution with a service rate μ_i.

In our model, we assume that an LA is responsible for dispatching the request. The LA sends the request to server i with probability $p_i(t)$. We will later define the update equations for the LA. However, for the sake of simplicity, we shall give first the overall idea for the different updates involved here at the two time scale, and subsequently, in the next section, we shall delve into the LA's detailed update equations.

By virtue of the $M/M/1$ queue, the mean-response time at server i is:

$$MRT_i(t) = \frac{1}{\mu_i - \lambda_i(t)}, \tag{1}$$

where $\lambda_i(t)$ is the average arrival rate at server i. If the $\{p_i\}$ are constant or vary slowly over time, then $\lambda_i(t)$ can be approximated using $\lambda_i(t) = p_i(t)\lambda$, which is a consequence of the $M/M/1$ queue model [17].

Let $s_i(t)$ be the instantaneous response time of server i at time t. In order to estimate the average response time of each server (i.e., $\hat{s}_i$), we merely use the exponential moving average approach with learning parameter α. The parameter α is the learning parameter of the scheme, and is similar to the parameter used in any learning algorithm. It is a hyper-parameter determined by a "rule of thumb" or trial-and-error for the particular setting. A larger value of α implies a larger step away from the current value, and vice versa. This, in turn, illustrates the speed-accuracy dilemma of the estimate.

Let $\hat{s}_i(t)$ be the estimate of the average response time of server i.

Once the action i is polled, i.e., the request is dispatched to server i, the estimate $\hat{s}_i(t+1)$ of the average is immediately updated using an adaptive estimator, namely, the exponential moving average given by:

$$\hat{s}_i(t+1) = \hat{s}_i(t) + \alpha(s_i(t) - \hat{s}_i(t)). \tag{2}$$

The average response time for the other severs (actions) are left unchanged. In other words:

$$\hat{s}_j(t+1) = \hat{s}_j(t) \text{ for } j \neq i, j \in [1, n]. \tag{3}$$

We now consider how the corresponding rewards and penalties are constructed. If action i is chosen, the reward or penalty is constructed as below using some type of dynamic threshold:

- Reward if $\hat{s}_i \leq \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k$
- Penalty if $\hat{s}_i > \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k$.

With these definitions as a back-drop, we are able to formally present the steps of our algorithm.

B. Initialization Criteria

Without any knowledge of $\hat{s}_i$, which is estimated using an exponential moving average as per Eq. (2), we have opted to initialize $\hat{s}_i$ to a random low value of response time close to zero. As in any exponential moving average scheme, the value what we use for this initialization is not critical. In our experiments, we assigned this value as $\hat{s}_i(0)$ for all the r servers. This is in line with the spirit of what is done in LA, where the initialization is achieved by values that are equal. In fact, without the accurate knowledge of initial LA's action probability, the initial probability for each action i is usually set to ($p_i(0) = \frac{1}{r}$). In our case, we have also verified experimentally that the initial value of $\hat{s}_i$ does not have any effect on the *long term convergence* behavior of the scheme which is an observation consistent with the behavior of the exponential moving average schemes, which are ergodic by nature. However, in real life settings, the experimenter might assign an initial value of $\hat{s}_i$ that is more informed based on an *a priori* knowledge of server i. In this case, one might also alter the initial LA probabilities so as to move away from the uniform distribution, i.e., $p_i(0) = \frac{1}{r}$.

C. Details of our Solution: Two-time Scale LA with Barriers

The first step in our solution process is to see how we can transform the Markov process given by the probability space from being absorbing, into being ergodic. The reader who is aware of the field of Markov Chains will immediately recognize that this is, actually, the converse of what the literature [5], [39][2] reports when an ergodic chain in rendered artificially absorbing, as in the families of Artificially Absorbing Discretized LA such as the ADL_{RP} and ADL_{IP} [30]. Rather than use the actual limits of the probability space to be zero and

[2]The projection method is a classical method in constrained optimization [5] that ensures that the solution is mapped back in the feasible search space whenever it falls outside it. The relative reward LA devised [39] adopts artificial barriers for more than 2 actions using the projection method.

unity, we work with the constraint that no probability value can take on a value below a pre-specified lower threshold of p_{min}, or a value above a pre-specified upper threshold of p_{max} [51]. The action-choosing probability values, which traditionally move proportionally towards zero and unity for the L_{RI} scheme, for example, are now made to move towards the respective values of p_{min} and p_{max} respectively. Interestingly enough, this minor modification renders the scheme to be ergodic, making the analysis also to be correspondingly distinct from that of the L_{RI} and similar schemes.

To achieve this, we enforce a minimal value p_{min}, where $0 < p_{min} < 1$ for each selection probability x_i, where $1 \leq i \leq r$ and r is the number of actions. As a result, the maximum value any selection probability p_i, where $1 \leq i \leq r$, can achieve is $p_{max} = 1 - (r-1)p_{min}$. This happens when the other $r-1$ actions take their minimum value p_{min}, while the action with the highest probability takes the value p_{max}. Consequently, p_i, for $1 \leq i \leq r$, will take values in the interval $[p_{min}, p_{max}]$.

To proceed with the formulation, let $\alpha(t)$ be the index of the chosen action at time instant t. Then, the value of $p_i(t)$ is updated as per the following simple rule (the rules for other values of $p_j(t), j \neq i$, are analogous):

$$\begin{aligned} p_i(t+1) \quad & \leftarrow p_i(t) + \theta(p_{max} - p_i(t)) \\ & \text{when } \alpha(t) = i \text{ and } v_i = 1 \\ p_i(t+1) \quad & \leftarrow p_i(t) + \theta(p_{min} - p_i(t)) \\ & \text{when } \alpha(t) = j, j \neq i \text{ and } v_i = 1, \end{aligned}$$

where θ is a user-defined parameter $0 < \theta < 1$, typically close to zero. Further, v_i is a reward function indicator defined by:

- $v_i = 1$, reward, if the instantaneous response of the chosen server is under the running moving average of the mean response time $s_i \leq \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k$
- $v_i = 0$, penalty, if the instantaneous response of the chosen server exceeds the running moving average of the mean response time, $\hat{s}_i > \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k$.

In our algorithm, we avoid using a classical projection method to map the solution to our feasible space, meaning all components of the probability vector are within the interval $[p_{min}, p_{max}]$. Projection methods were used in the field of LA for enforcing artificial barriers. A prominent example is due to Simha and Kurose [39] who tackle a number actions $r > 2$, which is a more challenging scenario than the 2 actions scenario. However, our approach does not involve projection methods as the update equations will always ensure that the probabilities will be in our feasible space. Furthermore, in contrast to projection methods, our LA update methodology *naturally* ensures that the probability vector will always sum to 1 in a manner than can be seen as a generalization of the Linear Reward Inaction LA. The classical Linear Reward Inaction LA can be seen as an instance of our algorithm with $p_{max} = 1$.

Let the average of all the instantaneous response times of all the nodes at time t be given by $\hat{s}(t)$ defined by:

$$\hat{s}(t) = \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k(t). \tag{4}$$

We also introduce the following notation:

$$D_i(t) = Prob(s_i(t) \leq \hat{s}(t)), \tag{5}$$

where $\hat{s}(t)$ is given by Eq.(4).

A consequence of these assignments is a scheme formalized by the pseudo-code given in Algorithm 1. The algorithm proceeds as follows in a loop. Each time a request is received, the LA probability vector is used to choose a server by polling an action, which corresponds here to a server among the r severs. The server choice corresponds to step 1 in the pseudo-code given in Algorithm 1. Once the server is chosen, the instantaneous response time of the chosen server for that requested is observed. Then, in step 2, based on this observation we update the average response time of the chosen server of the pseudo-code using exponential moving average. The estimates for the other ”unchosen” servers will be kept unchanged. In step 3, the chosen action receives a reward or penalty by comparing the estimated response time of the chosen server to a dynamic threshold: the mean of the individual average response times of the r servers. In step 4, we operate with the same rules of the classical Linear Reward-Inaction LA but with the exception of accommodating artificial barriers. If the chosen action resulted in a reward, its probability is increased, while the probabilities the rest of the $r-1$ actions are decreased. However, if the chosen action results into a penalty, the probability vector is kept unchanged as per the Linear Reward-Inaction LA philosophy.

With these definitions in place, we are in a position to analyze the scheme and give the theoretical results. This is done in the next section. We show that as $p_i(t)$ increases this quantity, $Prob(s_i(t) > \hat{s}(t))$, decreases. This is an extremely interesting observation because the latter quantity is, quite simply, the reward probability when choosing action i. This is referred to as the “stochastic diminishing return” property, which, informally, means that the more an action is chosen, the less its reward will be. Thereafter, we will prove the scheme’s convergence.

Algorithm 1 The Two-Time Scale based Learning Automata Solution

```
Loop
  1. Poll an action at time instant $t$ according to the probability vector $[p_1, p_2, \dots, p_r]$.
  2. Updating the response time estimates.
    • Update the response time of the chosen action:
        $\hat{s}_i(t+1) = \hat{s}_i(t) + \alpha(s_i(t) - \hat{s}_i(t))$
    • The response estimates for the other actions are kept unchanged, and so,
        $\hat{s}_j(t+1) = \hat{s}_j(t)$ for $j \neq i, j \in [1, r]$
  3. Environment response: Reward/Penalty.
    • $v_i = 1$ (Reward) if $\hat{s}_i \leq \frac{1}{r}\sum_{k=1}^{r} \hat{s}_k$;
    • Otherwise, $v_i = 0$ (Penalty).
  4. Let $\alpha(t)$ be the index of the chosen action at time instant $t$. The value of $p_i(t)$ is updated as per the following simple rule below, (where the update rules for other values of $p_j(t), j \neq i$, are similar.):
        $p_i(t+1) \leftarrow p_i(t) + \theta(p_{max} - p_i(t))$
            when $\alpha(t) = i$ and $v_i = 1$
        $p_i(t+1) \leftarrow p_i(t) + \theta(p_{min} - p_i(t))$
            when $\alpha(t) = j, j \neq i$ and $v_i = 1$.
```

D. Theoretical analysis

In this section we shall investigate and analyze the asymptotic behavior of our LA based two-time scale separation solution with artificial barriers. We shall analyze our scheme in terms of both its convergence and stability.

Theorem 1. *For a sufficiently small α and for $\theta << \alpha$, $\hat{s}_i(t)$ can be approximated by $MRT_i(p_i(t)) = \frac{1}{\mu_i - \lambda_i p_i(t)}$ for all $1 \leq i \leq r$.*

Proof. We will prove that for $1 \leq i \leq r$, $\hat{s}_i(t)$ converges to $\bar{s}_i(p_i(t))$ where $\bar{s}_i$ denotes the MRT_i.

The proof is based on the theory of stochastic approximation [2]. Since θ is much smaller that α, the p_i's evolve at a slower time scale compared to the $\hat{s}_i$'s, which, in turn, guarantees the two-time scale separation. Using the notation that $\alpha(t) = i$ means that action i is chosen at time t, we can write:

$$\hat{s}_i(t+M) = \hat{s}_i(t) + \alpha \sum_{k=0}^{M-1} I_{\{\alpha(t+k+1)=i\}}(s_i(t+k) - \hat{s}_i(t+k))$$

As per the theory of small step processes, we can assume that whenever α is small enough, the vector $[\hat{s}_1(t), \hat{s}_2(t), ..., \hat{s}_r(t)]$ remains almost unchanged in the discrete interval $\{t, t+1, \dots, t+M\}$. Thus, we can write the following approximate equations for $1 \leq i \leq r$:

$$\hat{s}_i(t+M) \approx \hat{s}_i(t) + M\alpha(R_i(t, M) - Q_i(t, M)\hat{s}_i(t)) \quad (6)$$

For $i \in [1, r]$, when the values of the estimates $\{\hat{s}_1(.), \hat{s}_2(.), \dots, \hat{s}_r(.)\}$ are respectively considered fixed at $\{\hat{s}_1(t), \hat{s}_2(t), \dots, \hat{s}_r(t)\}$, and M is large, we now approximate the quantities:

$$R_i(t, M) = \frac{\sum_{k=0}^{M-1} I_{\{\alpha(t+k+1)=i\}} s_i(t+k)}{M},$$

as well as

$$Q_i(t, M) = \frac{\sum_{k=0}^{M-1} I_{\{\alpha(t+k+1)=i\}}}{M}.$$

The probability vector $p_1(.), p_2(.), \dots, p_r(.)$, too, can be regarded to be essentially constant in the interval $\{t, t+1, \dots, t+M\}$, because we have affirmed that p_i evolves at slower time scale compared to $\hat{s}_i$. Note that the fact that θ is much smaller than α permits the separation in this time scale.

Now, assuming that M is large enough such that the law of large numbers is in effect, the average

$$Q_i(t, M) = \frac{\sum_{k=0}^{M-1} I_{\{\alpha(t+k+1)=i\}}}{M},$$

which is the fraction of time the action i was chosen in the interval $[t, t+M]$, converges to $p_i(t)$.

By reckoning the actions' probabilities to be fixed, the response time processes $s_i(.)$, can converge to a stationary distribution, with the mean being denoted by $\bar{s}_i(p_i(t))$.

Further, the quantities:

$$R_i(t, M) = \frac{\sum_{k=0}^{M-1} I_{\{\alpha(t+k+1)=i\}} s_i(t+k)}{M}$$

can be approximated by $p_i(t)\bar{s}_i(p_i(t))$.

Employing the approximations as described above, we notice from Eq. (6), that the evolution of the vector $[\hat{s}_1(.), \hat{s}_2(.), \dots, \hat{s}_r(.)]$ reduces to the following ODE system when α is sufficiently small:

$$\frac{\hat{s}_i(t)}{dt} = p_i(t).(\bar{s}_i(p_i(t)) - \hat{s}_i(t)). \quad (7)$$

One observes that Eq. (7), reduces to having the running response time estimates, given by $[\hat{s}_1(.), \hat{s}_2(.), \dots, \hat{s}_r(.)]$, converging to a steady state vector $[\bar{s}_1(p_1(t)), \bar{s}_2(p_2(t)), \bar{s}_r(p_r(t))]$, whenever α tends to 0.

We now invoke the properties of the $M/M/1$ queue model, aluded to above. As per the properties of the $M/M/1$ queue model, we know that:

$$\bar{s}_i(p_i(t)) = MRT_i(p_i(t)) = \frac{1}{\mu_i - \lambda_i p_i(t)}. \quad (8)$$

This, indeed, concludes the proof. □

In the next theorem, we shall prove the diminishing property of our designed feedback mechanism. In fact, our reward is defined by the fact that the instantaneous response time observed when we choose a server is smaller than $\hat{s}(t)$, which is the arithmetic mean of $\hat{s}_i(t)$ for $1 \leq i \leq n$, where $\hat{s}_i(t)$ is the running estimate (i.e., the exponential moving average) of the response time at server i. Using the notation of Eq. (5), we will show that the reward probability decreases as we increase p_i.

Theorem 2. *$D_i(t)$ is monotonically strictly decreasing as a function of p_i.*

Proof. We consider the reward probability $D_i(t) = Prob(s_i(t) \leq \hat{s}(t))$ where $\hat{s}(t)$ is given by Eq.(4).

As a consequence of the previous result from Theorem 1, if the $\hat{s}_i$'s evolve at a slower time scale than the p_i's, we can approximate $\hat{s}(t)$ by the sum of the mean response times of each server, i.e., sum of $MRT_i(t)$, $1 \leq i \leq r$. In other words $\hat{s}(t) \approx \frac{\sum_{k=i}^{r} \bar{s}_i(p_i(t))}{r} = \frac{\sum_{k=i}^{r} MRT_i(p_i(t))}{r}$.

The probability that the response time of server i, $s_i(t)$ exceeds $\hat{s}(t)$ is [36]:

$$D_i(t) = Prob(s_i(t) \leq \hat{s}(t)) = 1 - exp(-\hat{s}(t)(\mu_i - \lambda_i(t))). \tag{9}$$

We need to show that as $p_i(t)$ increases, this quantity decreases. To achieve this, consider $\frac{dD_i(t)}{dp_i}$ given by: $\frac{dD_i(t)}{dp_i} = \frac{\delta D_i(t)}{\delta p_i} + \sum_{\substack{j=1\\j\neq i}}^{r} \frac{\delta D_i(t)}{\delta p_j}\frac{\delta p_j}{\delta p_j}$.

In order to apply the chain rule for the derivation, we resort to a subtle mathematical trick similar to the one used in [18], [48]. We define arbitrary constants $b_j \geq 0$ for $j \neq i$, whence, following a derivation similar to the one in [18], [48], we have:

$$p_1 = b_1 p_i, p_2 = b_2 p_i, \ldots p_r = b_r p_i, \text{ with } b_j \geq 0 \text{ for } j \neq i.$$

Consequently,

$$\begin{aligned} p_1 &= \frac{b_1(1-p_i)}{\sum_m b_m} \\ \ldots &= \ldots \\ p_j &= \frac{b_j(1-p_i)}{\sum_m b_m} \\ \ldots &= \ldots \\ p_r &= \frac{b_r(1-p_i)}{\sum_m b_m} \end{aligned} \tag{10}$$

Now, since $\sum_m p_m = 1$, we can obtain: $\frac{dp_j}{dp_i} = \frac{-b_j}{\sum_{m\neq j} b_m} < 0$ for all $j \neq i$.

Considering the expression for $D_i(t)$ we see that:

$$\begin{aligned} D_i(t) &= 1 - exp(-\hat{s}(t)(\mu_i - \lambda_i(t)) \\ &= 1 - exp(-\frac{\mu_i - \lambda_i(t)}{r}\sum_k \frac{1}{\mu_k - \lambda_k(t)}) \\ &= 1 - exp(-\frac{1}{r} - \sum_{\substack{k=1\\k\neq i}}^{r} \frac{1}{r(\mu_k - \lambda p_k(t))}). \end{aligned}$$

This expression is independent of p_i, which implies that $\frac{\delta D_i(t)}{\delta p_i} = 0$. Consequently, $\frac{\delta D_i(t)}{\delta p_i}$ reduces to $\frac{dD_i(t)}{dp_i} = \sum_{\substack{j=1\\j\neq i}}^{r} \frac{\delta D_i}{\delta p_j}\frac{\delta p_j}{\delta p_i}$.

Algebraic simplification leads to:

$$\frac{\delta D_i}{\delta p_j} = \frac{\lambda}{r(\mu_i - \lambda p_i(t))^2} exp(-\hat{s}(t)(\mu_i - \lambda_i(t))).$$

Furthermore, since $\frac{dp_j}{dp_i} < 0$, $\frac{\delta D_i(t)}{\delta p_i} = \sum_{\substack{j=1\\j\neq i}}^{r} \frac{\delta D_i}{\delta p_j}\frac{dp_j}{dp_i} < 0$, since all the terms in the above sum are strictly negative.

Hence the theorem! □

Theorem 3. *For a sufficiently small p_{min} approaching 0, the system of update equations characterizing the LA has a unique fixed point equilibrium.*

Proof.

$$\begin{aligned} E[p_i(t+1) - p_i(t)|p(t)] &= p_i D_i(p_i)[\theta(1-p_i)] \\ &+ \sum_{\substack{j=1\\j\neq i}}^{r} p_j D_j(p_j).[\theta(p_{min} - p_i)]. \end{aligned}$$

Then:

$$\begin{aligned} &E[p_i(t+1) - p_i(t)|p(t) = p] = \\ &\quad p_i D_i(p_i).[\theta(1 - p_{max} + 1 - p_i)] \\ &\quad + \sum_{\substack{j=1\\j\neq i}}^{r} p_j D_j(p_j)[\theta(p_{min} - p_i)] \\ &= p_i D_i(p_i).[\theta(1 - p_{max} + \sum_{\substack{j=1\\j\neq i}}^{r} p_j)] \\ &\quad + \sum_{\substack{j=1\\j\neq i}}^{r} p_j D_j(p_j)[\theta(p_{min} - p_i)]. \end{aligned} \tag{11}$$

(12)

By taking into account the fact that $1 - p_{max} = (r-1)p_{min}$, Eq. (12) can be simplified (after some algebraic manipulations) and written as:

$$\begin{aligned} &E[p_i(t+1) - p_i(t)|p(t) = p] = \\ &\quad \theta \sum_{\substack{j=1\\j\neq i}}^{r} p_i p_j (D_i(p_i) - D_j(p_j)) \\ &\quad + \theta p_{min}(\sum_{\substack{j=1\\j\neq i}}^{r} p_j D_j(p_j)) \\ &\quad - \theta(r-1)p_{min} p_i D_i(p_i) \\ &= \theta \sum_{\substack{j=1\\j\neq i}}^{r} p_i p_j (D_i(p_i) - D_j(p_j)) \\ &\quad + \theta p_{min} \sum_{\substack{j=1\\j\neq i}}^{r} (p_j D_j(p_j) - p_i D_i(p_i)) \\ &\approx \theta w_i(p), \end{aligned}$$

where $w_i(p)$ is defined by $w_i(p) = \sum_{\substack{j=1\\j\neq i}}^{r} p_i p_j (D_i(p_i) - D_j(p_j))$.

For small values of p_{min}, i.e., as $p_{min} \to 0$, we can approximate $E[p_i(t+1) - p_i(t)|p(t) = p]$ by:

$$E[p_i(t+1) - p_i(t)|p(t) = p] = \theta w_i(p). \tag{13}$$

We can thus write:

$$\frac{dp_i(t+1)}{dt} \quad = \quad \theta w_i(p). \tag{14}$$

Using the above result, we shall now proceed with the details of the proof.

a) Existence and Uniqueness:: We will show that $w(p) = (w_1(p), w_2(p), ..., w_r(p))$ has a unique zero in the neighborhood of $p^* = (p_1^*, ..., p_r^*)$, which means that we have a fixed point.

The above assertions imply the system of r equalities:

$$\begin{cases} \sum_{\substack{j=1 \\ j\neq 1}}^{r} p_1 p_j (D_1(p_1) - D_j(p_j)) = 0 \\ \sum_{\substack{j=1 \\ j\neq 2}}^{r} p_2 p_j (D_2(p_2) - D_j(p_j)) = 0 \\ \vdots \\ \sum_{\substack{j=1 \\ j\neq r}}^{r} p_n p_j (D_r(p_r) - D_j(p_j)) = 0. \end{cases}$$

$$\Leftrightarrow \begin{cases} p_1 \sum_{\substack{j=1 \\ j\neq 2}}^{r} p_j (D_1(p_1) - D_j(p_j)) = 0 \\ p_2 \sum_{\substack{j=1 \\ j\neq 2}}^{r} p_j (D_2(p_2) - D_j(p_j)) = 0 \\ \vdots \\ p_n \sum_{\substack{j=1 \\ j\neq r}}^{r} p_j (D_r(p_r) - D_j(p_j)) = 0. \end{cases}$$

The reader should observe that a crucial concept in our approach is that we are using the barrier p_{min}, which ensures that $p_1 \neq 0$, $p_2 \neq 0 ... p_r \neq 0$. We can thus confidently divide the first equation by p_1, the second equation by p_2 and so on, yielding:

$$\Leftrightarrow \begin{cases} \sum_{\substack{j=1 \\ j\neq 1}}^{r} p_j (D_1(p_1) - D_j(p_j)) = 0 \\ \vdots \\ \sum_{\substack{j=1 \\ j\neq 2}}^{r} p_j (D_2(p_2) - D_j(p_j)) = 0 \\ \vdots \\ \sum_{\substack{j=1 \\ j\neq r}}^{r} p_j (D_r(p_r) - D_j(p_j)) = 0. \end{cases}$$

After invoking some algebraic manipulations, we obtain that:

$$\Leftrightarrow \begin{cases} D_1(p_1) = \sum_{j=1}^{r} p_j D_j(p_j) \\ \vdots \\ D_2(p_2) = \sum_{j=1}^{r} p_j D_j(p_j) \\ \vdots \\ D_r(p_r) = \sum_{j=1}^{r} p_j D_j(p_j), \end{cases}$$

which guarantees that $D_1(p_1) = D_2(p_2) = \ldots = D_r(p_r)$.

Now we will show that the solution is unique.

b) Uniqueness:: The uniqueness of p^* is proven by contradiction. Suppose there exists $q^* = (q_1^*, q_2^*, ..., q_n^*)$ that is a zero of $w(q)$ such that $q^* \neq p^*$.

Without loss of generality since p^* and q^* are two probability vectors such that $p^* \neq q^*$, we can confidently affirm that they have at least two components i and j such that $p_i^* > q_i^*$ and $p_j^* < q_j^*$. Observe that the result is general, and that it applies for any two distinct probability vectors. Intuitively, this means that if we increase any one component of a probability vector, we should decrease another component so as to ensure that the sum of the components is unity.

Suppose now that $p_i^* > q_i^*$. Then, by invoking the monotonicity of the function $D_i(.)$, we obtain that $D_i(p_i^*) < D_i(q_i^*)$. On the other hand, the condition $p_j^* < q_j^*$ implies that $D_j(p_j^*) > D_j(q_j^*)$, where this is obtained by virtue of the monotonicity of $D_j(.)$. But since p^* and q^* are equilibrium points, we know that $D_i(p_i^*) = D_j(p_j^*)$ and that $D_i(q_i^*) = D_j(q_j^*)$. This forces a contradiction since it is impossible to simultaneously maintain that : $D_i(p_i^*) < D_i(q_i^*)$ which is equivalent to $D_j(p_j^*) < D_j(q_j^*)$ and $D_j(p_j^*) > D_j(q_j^*)$.

Therefore p^* is unique.

□

Theorem 4. *The equilibrium point to which the algorithm converges, is asymptotically Lyapunov stable.*

Proof. Consider the following Lyapunov function:

$$V(p(t)) \quad = \quad \sum_{k=i}^{r} \int_0^{p_t} D_k(z) dz.$$

Consider now its derivative:

$$\frac{dV(p(t))}{dt} = \sum_{i=1}^{r} \frac{dV(p(t))}{dp_i} \frac{dp_i}{dt}. \tag{15}$$

It is easy to note that by virtue of the integral derivation, $\frac{dV(p(t))}{dp_i} = D_i(t)$. Furthermore, according to Eq. (14), $\frac{dp_i}{dt} = \theta w_i(p)$.

Thus

$$\frac{dV(p(t))}{dt} = \theta \sum_{k=1}^{r} D_k(t) w_k(p), \tag{16}$$

where $w_i(p)$ is defined by $w_i(p) = \sum_{\substack{j=1 \\ j\neq i}}^{r} p_i p_j (D_i(p_i) - D_j(p_j))$. Therefore,

$$\begin{aligned} \frac{dV(p(t))}{dt} &= \theta \sum_{i=1}^{r} D_i \sum_{j=1}^{r} p_i p_j (D_i - D_j) \\ &= \theta \sum_{i=1}^{r} \sum_{j=1}^{r} p_i p_j (D_i^2 - D_i D_j) \\ &= -\frac{\theta}{2} \sum_{i=1}^{r} \sum_{j=1}^{r} p_i p_j (D_i - D_j)^2 \end{aligned}$$

Therefore $\frac{dV(p(t))}{dt} \leq 0$.

Observe though that the Lyapunov function must be zero at its equilibrium point, and thus: $\frac{dV(p(t))}{dt} = 0$. This, in turn, means that for every i, j, we have $p_i^* p_j^* (D_i^* - D_j^*)^2 = 0$. However, since $p_i^* > p_{min}$ and $p_j^* > p_{min}$, the equality $D_i^*(p_i^*) - D_j^*(p_j^*) = 0$ must necessarily be true, and consequently for all i, j

$$D_i^*(p_i^*) = D_j^*(p_j^*) = 0.$$

The result follows, since by Lyapunov theorem, we have shown that p^* is an asymptotically Lyapunov stable equilibrium point of the scheme. $\square$

E. Summary outlining of the theoretical results

In the first theorem, Theorem 1, we show that by imposing a two-time scale separation where we slowly update the LA probabilities, while we update the response times in faster scale we are able to approximate the estimated response times at each server. For any probability vector that is slowly varying, the response time estimates converge to a steady state depending on the probability vector given by Eq. (8). Once we have characterized the response times, we can analyze the behavior of the reward probabilities of our LA. This is treated in Theorem 2 where we show an intuitive property which states that the reward probability of an action is monotonically strictly decreasing as a function of its respective action probability. The next theorem: Theorem 3 characterizes the fixed point of the LA update equations. The artificial barriers as well as the monoticity of the reward functions yield a unique fixed point. Interestingly, the fixed point achieves fairness as the reward probabilities of the action are "equalized", thus, the LA will be indifferent between the choices of the servers at this point. The last theorem, Theorem 4 shows the algorithm converges to is asymptotically Lyapunov stable by defining an appropriate Lyapunov function.

V. Experimental Verification

In this section, we will briefly confirm that the theoretical results that were derived in the previous section, hold true. To achieve this, we conducted two types of experiments. The intent of the first set of experiments was to prove the claims for a small-scale system, namely, for one with only three servers. The second, and more extensive testing, involved a larger pool of servers, i.e., 15. Clearly, such a setting is well in-line with real-life load balancing problems. Furthermore, we tested both of the scenarios on two types of environments, namely, static and dynamic. For the dynamic case, we report experiments where we either changed the serving rates of the services or the arrival rate of the requests.

A. Experiments with three servers

1) Static environment with 3 servers: In this first set of experiments, we simulated three servers characterized by the parameters $\mu_1 = 50$, $\mu_2 = 33.33$ and $\mu_3 = 25$ respectively. We assumed that the total arrival rate was $\lambda_1 + \lambda_2 + \lambda_3 = 50$. Further, we used $p_{min} = 0.01$. For the time scale separation, we used two values $\theta = 0.001$ for updating the LA action probabilities, and $\alpha = 0.01$ for the estimation of the response times. The intention of our experiments was to observe the action probabilities and the corresponding response times, when the protocol was tested for 9,000 iterations, for an ensemble of 1000 experiments.

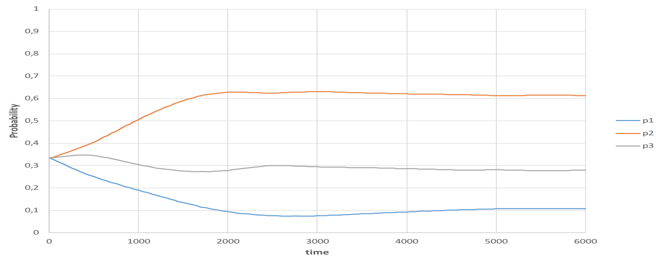

Fig. 1. Evolution of the action selection probabilities in a static environment.

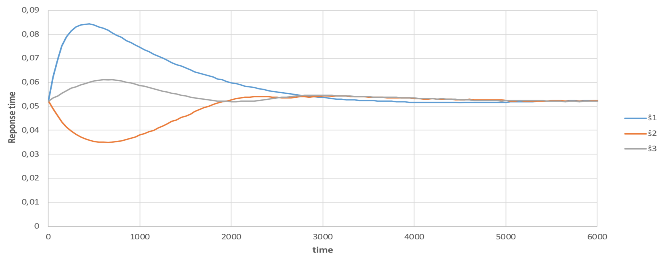

Fig. 2. Evolution of the response times of the different servers in a static environment.

The evolution of the probability and response time are plotted in Figure 1 and Figure 2 respectively. Interestingly, from Figure 2, we observed that the response time was equalized on the different servers. The results are quite amazing because even though the corresponding action probabilities converged to different values, the composite effect of the convergence was to make the overall response times to be almost equal.

2) Dynamic serving rates with 3 servers: To investigate the performance of the scheme for time-varying systems, we also ran another experiment where we dynamically shuffled the serving rates of the three servers every 5,000 iterations. In this case, the experiments were run for 15,000 iterations and the number of experiments (over which the ensemble average was obtained) was 1000. We again observed that the system stabilized after some time and that it was again capable of equalizing the response times. Figure 3 and Figure 4 respectively depict the evolution of the action probabilities of each server, as well as the evolution of the estimated response times. Clearly, the results demonstrated that even though the corresponding action probabilities converged to completely different values, the overall effect of the scheme's convergence was to make the overall response times to be almost equal.

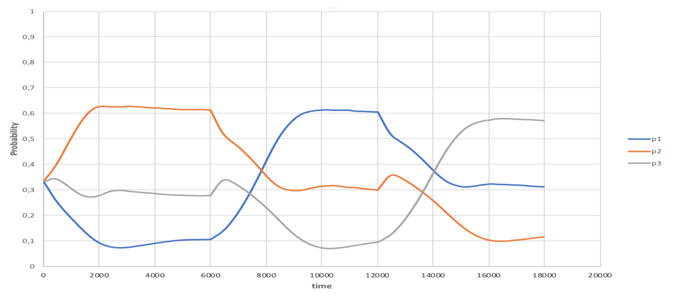

Fig. 3. Evolution of the action selection probabilities in an environment with dynamic serving rates.

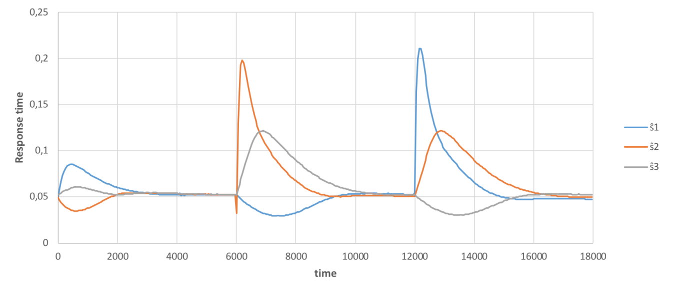

Fig. 4. Evolution of the response times of the different servers in an environment with dynamic serving rates.

The power of the scheme to balance the loads in an ϵ-optimal manner is obvious!

3) Dynamic arrival rate with 3 servers: In real-life scenarios, it is more common that the arrival rate of the traffic changes over time, while the serving rate is usually stable over time. This is because the latter is an intrinsic characteristic of the server, and does not, usually, change due to extrinsic factors related to the environment[3]. In our simulations, we adjusted the arrival rate every 6,000 iterations. We started with a total arrival rate of 50, and after the first switch this number was increased to 60. It was then lowered to 54 after the second switch. Figure 5 and Figure 6 report respectively the evolution of the probabilities and the evolution of the response time estimates for this dynamic environment characterized by varying arrival rates. One should observe that our scheme is able to equalize the response times of the servers after around 3,000 iterations.

[3]The serving rate of a server might degrade slightly over time due to hardware issues. It is also possible to upgrade the servers for improved performance. However, such changes are beyond the scope of this work and are still rare within the life-time of a server.

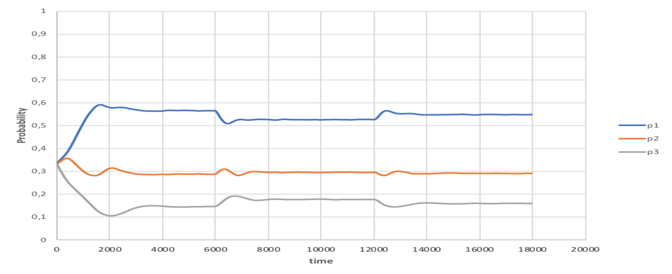

Fig. 5. Evolution of the action selection probabilities in a environment with varying arrival rates and with 3 servers.

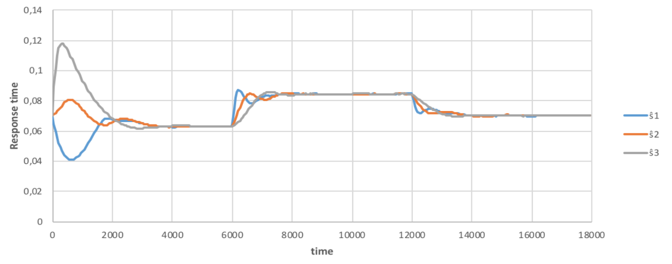

Fig. 6. Evolution of the response times of the different servers in a environment with varying arrival rate with 3 servers.

B. Larger Scale Experiments

1) Static environment with 15 servers: In the second set of experiments, we increased the number of servers to 15 as per the following parameters: 5 servers with $\mu = 50$, 5 servers with $\mu = 60$ and 5 servers with $\mu = 80$. We assumed that the total arrival rate was $\lambda = 350$. Figure 7 and Figure 8 illustrate respectively the evolution of the probabilities and the evolution of the response time estimates for a static environment. When it comes to the parameters of the algorithm, we use the same tuning parameters as in the previous experiment involving 3 servers, i.e., $\theta = 0.001$ and $\alpha = 0.01$. Interestingly, even though the environment was intrinsically more challenging than the case of having three servers, we observed that our scheme was able to stabilize the response times of the 15 servers after around 5,000 iterations.

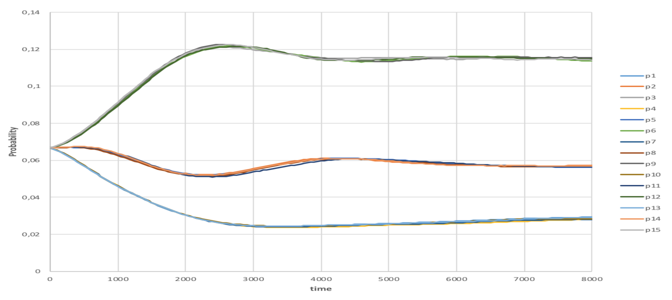

Fig. 7. Evolution of the action selection probabilities in a static environment with 15 servers.

2) Dynamic serving rates with 15 servers: In order to test the adaptivity of our scheme in a large scale settings, we executed a "switch" in the environment by

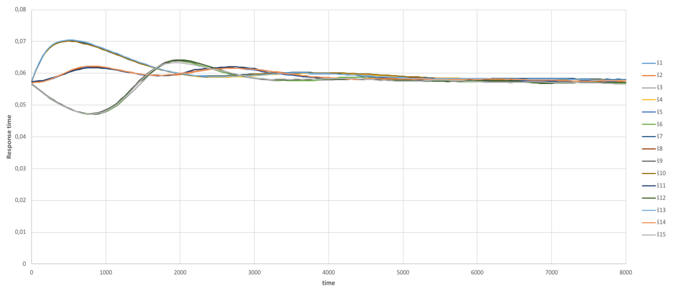

Fig. 8. Evolution of the response times of the different servers in a static environment with 15 servers.

modifying the serving rates every 30,000 iterations. The switch was a right-circular shift of a single position of the serving rate vector. For example, before the first switch, the serving rate vector of the 15 servers was (μ_1 =50, μ_2 =60, μ_3 =80, μ_4 =50,μ_5 =60, μ_6 =80, . . ., $\mu_1 3$ =50,μ_{14} =60,μ_{15} =80) respectively, and after the first switch, the serving rates became (60, 80,50,. . ., 60, 80, 50). Figure 9 and Figure 10 respectively depict the evolution of the action probabilities of each server, as well as the evolution of the estimated response times.

However, with such a large number of servers, it is clear that we could run into stability issues of the queues whenever the arrival rate at a given server became higher than its serving rate as a consequence of the shift in the serving rates. Formally, this instability can be seen to be a consequence of violating the condition $\mu_i - \lambda_i > 0$ where $\lambda_i = \lambda p_i$. For instance, this happens after the first switch, as we can easily observe. Consider, in this case, the third server. Before the switch, p_3 stabilized to 0.152 while the processing rate was as high as 80. Abruptly, however, after the switch in the serving rates, the same server, i.e., 3, obtained a new processing rate 50 which was much lower than before while p_3 was 0.152. This, clearly, led to a queue instability, since in this case $\mu_3 - \lambda_3 = 50 - 350 \times 0.152 = -3.2 < 0$, because the server was receiving more traffic than it could serve, which it clearly, could not handle.

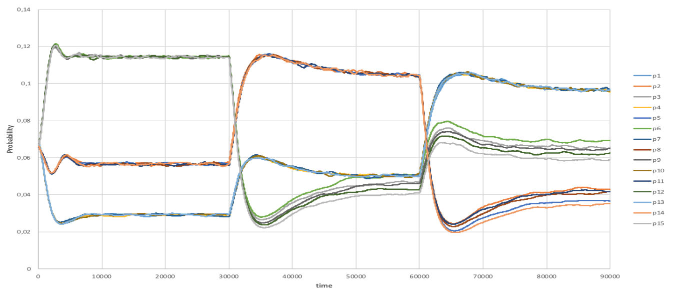

Fig. 9. Evolution of the action selection probabilities in a dynamic environment with dynamic serving rates and with 15 servers.

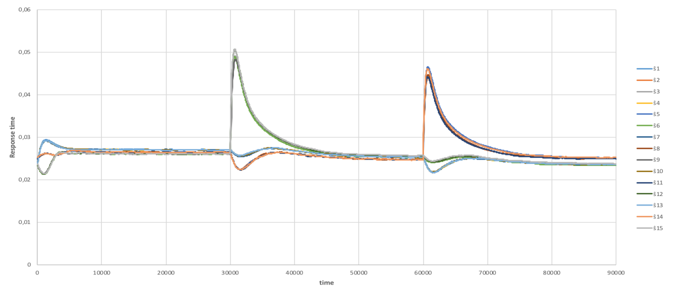

Fig. 10. Evolution of the response times of the different servers in a dynamic varying environment with dynamic serving rates and with 15 servers.

3) Dynamic arrival rate with 15 servers: In order to test the adaptivity of our scheme when facing changes in the arrival rate, we executed an environment switch every 30,000 iterations. We started with an arrival rate of 350, and after the first switch we dropped it to 280, and then we invoked a further drop to 252.

Figure 11 and Figure 12 respectively depict the evolution of the action probabilities of each server, as well as the evolution of the estimated response times.

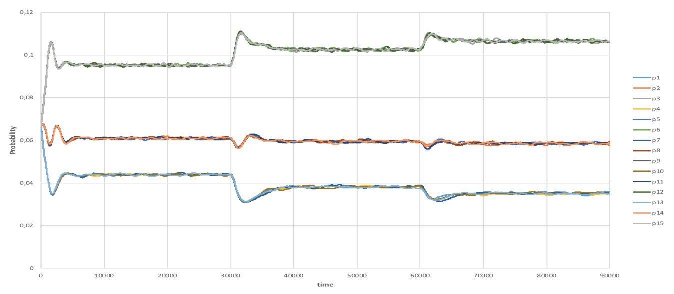

Fig. 11. Evolution of the action selection probabilities in a dynamic environment with varying arrival rate and with 15 servers.

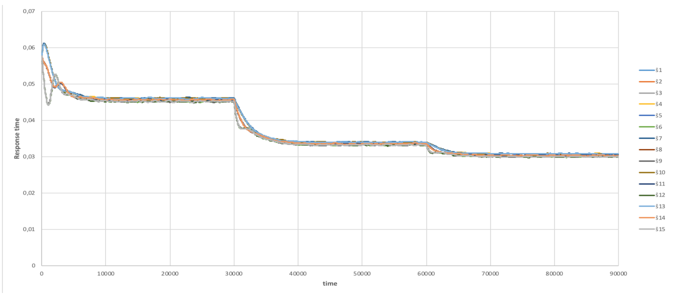

Fig. 12. Evolution of the response times of the different servers in a dynamic environment with varying arrival rate and with 15 servers.

C. *Comparison results in terms of fairness*

In this paper, we claimed that our algorithm is fair in the sense that the response times from the different servers are equalized, and thus a client will experience in average the same QoS measured in terms of average response time independently of the chosen server. It is not out place to compare the presented algorithm to other load balancing algorithms in the literature in terms of fairness. To this end, we resort to 3 commonly deployed load balancing algorithms: Round Robin (RR), Weighted Round Robin (WRR), Power of 2 choices (Po2) algorithm [25]. When it to comes to WRR, each server's weight is proportional to the service rate of the server. The fairness will be measured utilizing the Jain's fairness index [4], using the formula:

$$JFI = \frac{(\sum_{i=1}^{r} \hat{s}_i(t))^2}{r \sum_{i=1}^{r} \hat{s}_i(t)^2} \tag{17}$$

If the estimated response times of the different servers are equalized, the JFI will be equal to unity, its max value. The JFI by definition ranges between zero and one. In Figure 13, we report the ensemble average over 1000 experiments of the fairness index (JFI) for our LA algorithm against the aforementioned comparison algorithm for the case of 3 servers. The environment is static and the settings of the environment are the same settings as in Section V-A1. From Figure 13 we see that our LA algorithm achieves the highest JFI followed by the WRR. Please note that the WRR operates with extra knowledge than the LA algorithm as it assumes full knowledge of the servers serving rates to define its weights. Thus, we state that the LA algorithm is a superior solution in the sense that its achieves almost optimal JFI value, around unity, with little information, meaning no knowledge of the serving rates of the servers. Similarly, we conduct an experiment with 15 servers using the same settings as in Section V-B1. Figure 14 illustrates the behavior of our LA algorithms versus the state-of-the-art comparison algorithms. We observe similar remarks to those of the case of 3 servers reported in Figure 13. In fact, the LA algorithms is the most superior algorithm in terms of JFI followed by WRR. However, the Po2 achieves the lowest performance which was not the case when we used 3 servers (namely Figure 13). The main reason is that, as the number of servers increases, the Po2 will by definition select among two random servers among 15 servers, which gives it a limited view of the environment composed of a high number of servers, in this case 15, and consequently leads to poor performance.

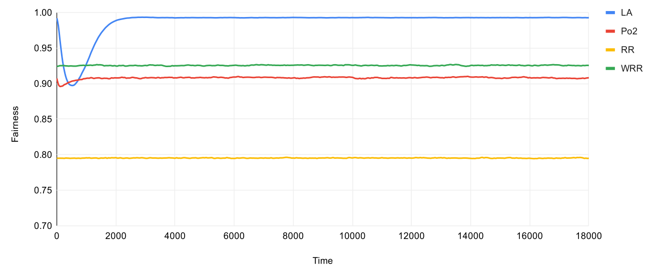

Fig. 13. Fairness comparison of the different load balancing algorithms in a static environment with varying arrival rate and with 3 servers.

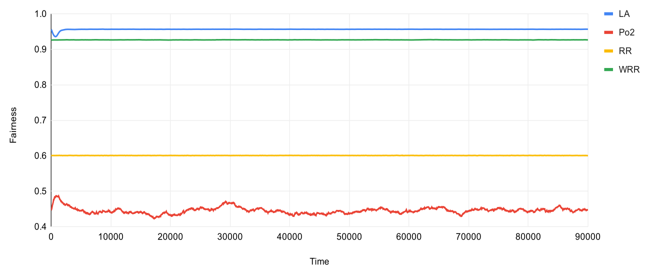

Fig. 14. Fairness comparison of the different load balancing algorithms in a static environment with varying arrival rate and with 15 servers.

VI. Conclusion

With the proliferation of network-based services, and the increasing popularity of the cloud, approaches that allow fair Load Balancing (LB) are becoming more important than before. Cloud computing is characterized by the volatility of resources, and the variability that makes static LB approaches inefficient. In this paper, we presented a dynamic LB approach that aspires to achieve "almost optimal" fairness between different servers in terms of a QoS-based metric. We have used the theory of Learning Automata (LA) to deal with the problem, and have designed a sophisticated LA which combined the time-separation paradigm and the concept of "artificial" ergodic (i.e., non-absorbing) barriers, which was recently introduced by Yazidi and Hammer in [51] and [52] respectively. In contrast to classical LA, the Environment considered was modeled to be non-stationary, and the reward probabilities were shown to be characterized by a law of diminishing returns.

As a future research endeavor, we intend to implement our solution in a real-life cloud setting, and to test its efficiency and fairness when compared to other classical approaches.

References

[1] M. Agache and B. J. Oommen. Generalized pursuit learning schemes: New families of continuous and discretized learning automata. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):738–749, December 2002.
[2] Albert Benveniste, Pierre Priouret, and Michel Métivier. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.
[3] Edward A Billard. Stabilizing distributed queuing systems using feedback based on diversity. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(2):251–256, 1997.
[4] Per Nikolaj D Bukh and Raj Jain. The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling, 1992.
[5] Paul H Calamai and Jorge J Moré. Projected gradient methods for linearly constrained problems. *Mathematical programming*, 39(1):93–116, 1987.
[6] G. C. Chasparis and J. S. Shamma. Network formation: Neighborhood structures, establishment costs, and distributed learning. *IEEE Transactions on Cybernetics*, 43(6):1950–1962, 2013.
[7] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*, pages 195–216. Springer, 2017.

[8] Alireza Enami, Javad Akbari Torkestani, and Abbas Karimi. Resource selection in computational grids based on learning automata. *Expert Systems with Applications*, 125:369–377, 2019.

[9] Mina Fahimi and Abdorasoul Ghasemi. A distributed learning automata scheme for spectrum management in self-organized cognitive radio network. *IEEE Transactions on Mobile Computing*, 16(6):1490–1501, 2016.

[10] O Ghaleb and B John Oommen. Learning automata-based solutions to the single elevator problem. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 439–450. Springer, 2019.

[11] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 88:50–71, 2017.

[12] Robert L Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[13] Jon Kleinberg and Eva Tardos. *Algorithm Design: Pearson New International Edition*. Pearson Higher Ed, 2013.

[14] PR Srikanta Kumar. A simple learning scheme for priority assignment at a single-server queue. *IEEE transactions on systems, man, and cybernetics*, 16(5):751–754, 1986.

[15] Thomas Kunz. The influence of different workload descriptions on a heuristic load balancing scheme. *IEEE transactions on software engineering*, 17(7):725–730, 1991.

[16] S. Lakshmivarahan. *Learning Algorithms Theory and Applications*. Springer-Verlag, 1981.

[17] John DC Little and Stephen C Graves. Little's law. In *Building intuition*, pages 81–100. Springer, 2008.

[18] L Mason. An optimal learning algorithm for s-model environments. *IEEE Transactions on Automatic Control*, 18(5):493–496, 1973.

[19] Peter Mell and Tim Grance. The nist definition of cloud computing, 2018. https://csrc.nist.gov/publications/detail/sp/800-145/final, Last seen 16/09/2019.

[20] Mohammad Reza Meybodi. *Learning Automata and Its Application to Priority Assignment in a Queueing System with Unknown Characteristics*. PhD thesis, The University of Oklahoma, 1983.

[21] MR Meybodi and S Lakshmivarhan. A learning approach to priority assignment in a two class m/m/1 queuing system with unknown parameters. In *Proceedings of Third Yale Workshop on Applications of Adaptive System Theory, Yale University*, pages 106–109, 1983.

[22] Ravi Mirchandaney and John A Stankovic. Using stochastic learning automata for job scheduling in distributed processing systems. *Journal of Parallel and Distributed Computing*, 3(4):527–552, 1986.

[23] S. Misra, P.V. Krishna, K. Kalaiselvan, V. Saritha, and M.S. Obaidat. Learning automata-based qos framework for cloud iaas. *IEEE Transactions on Network and Service Management*, 11(1):15–24, March 2014.

[24] Sudip Misra, Shankha Subhra Chatterjee, and Mohsen Guizani. Stochastic learning automata-based channel selection in cognitive radio/dynamic spectrum access for wimax networks. *International Journal of Communication Systems*, 28(5):801–817, 2015.

[25] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[26] K. Najim and A. S. Poznyak. *Learning Automata: Theory and Applications*. Pergamon Press, Oxford, 1994.

[27] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, New Jersey, 1989.

[28] Amy Nordrum. Popular internet of things forecast of 50 billion devices by 2020 is outdated, 2018. https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated, Last seen 16/09/2019.

[29] M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis. Learning automata: Theory, paradigms, and applications. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(6):706–709, December 2002.

[30] B. J. Oommen. Absorbing and ergodic discretized two action learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, 16:282–293, March/April 1986.

[31] Johan Parent, Katja Verbeeck, Jan Lemeire, Ann Nowe, Kris Steenhaut, and Erik Dirkx. Adaptive load balancing of parallel applications with multi-agent reinforcement learning on heterogeneous systems. *Scientific Programming*, 12(2):71–79, 2004.

[32] Deepak Kumar Patel, Devashree Tripathy, and Chita Ranjan Tripathy. Survey of load balancing techniques for grid. *Journal of Network and Computer Applications*, 65:103–119, 2016.

[33] Christy Pettey. Cloud shift impacts all it markets, 2018. https://www.gartner.com/smarterwithgartner/cloud-shift-impacts-all-it-markets/, Last seen 16/09/2019.

[34] A. S. Poznyak and K. Najim. *Learning Automata and Stochastic Optimization*. Springer-Verlag, Berlin, 1997.

[35] Ali Asghar Rahmanian, Mostafa Ghobaei-Arani, and Sajjad Tofighy. A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. *Future Generation Computer Systems*, 79:54–71, 2018.

[36] Heejun Roh, Cheoulhoon Jung, Wonjun Lee, and Ding-Zhu Du. Resource pricing game in geo-distributed clouds. In *2013 Proceedings IEEE INFOCOM*, pages 1519–1527. IEEE, 2013.

[37] Qian Sang, Zongli Lin, and Scott T Acton. Learning automata for image segmentation. *Pattern Recognition Letters*, 74:46–52, 2016.

[38] Seyyed Hossein Seyyedi and Behrouz Minaei-Bidgoli. Using learning automata to determine proper subset size in high-dimensional spaces. *Journal of Experimental & Theoretical Artificial Intelligence*, 29(2):415–432, 2017.

[39] Rahul Simha and James F Kurose. Relative reward strength algorithms for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(2):388–398, 1989.

[40] Alexandru-Adrian Tantar, Anh Quan Nguyen, Pascal Bouvry, Bernabé Dorronsoro, and El-Ghazali Talbi. Computational intelligence for cloud management current trends and opportunities. In *2013 IEEE Congress on Evolutionary Computation*, pages 1286–1293. IEEE, 2013.

[41] Andrei Tchernykh, Uwe Schwiegelsohn, Vassil Alexandrov, and El-ghazali Talbi. Towards understanding uncertainty in cloud computing resource provisioning. *Procedia Computer Science*, 51:1772–1781, 2015.

[42] M. L. Tsetlin. *Automaton Theory and the Modeling of Biological Systems*. Academic Press, New York, 1973.

[43] S Mehdi Vahidipour and Mehdi Esnaashari. Priority assignment in queuing systems with unknown characteristics using learning automata and adaptive stochastic petri nets. *Journal of computational science*, 24:343–357, 2018.

[44] S Mehdi Vahidipour, Mohammad Reza Meybodi, and Mehdi Esnaashari. Learning automata-based adaptive petri net and its application to priority assignment in queuing systems with unknown parameters. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(10):1373–1384, 2015.

[45] V. I. Varshavskii and I. P. Vorontsova. On the behavior of stochastic automata with a variable structure. *Automation and Remote Control*, 24:327–333, 1963.

[46] Gandhimathi Velusamy and Ricardo Lent. Dynamic cost-aware routing of web requests. *Future Internet*, 10(7):57, 2018.

[47] Katja Verbeeck, Ann Nowé, and Karl Tuyls. Coordinated exploration in multi-agent reinforcement learning: an application to load-balancing. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1105–1106, 2005.

[48] R Wheeler and K Narendra. Decentralized learning in finite markov chains. *IEEE Transactions on Automatic Control*, 31(6):519–526, 1986.

[49] Zhong Yang, Yuanwei Liu, Yue Chen, and Lei Jiao. Learning automata based q-learning for content placement in cooperative caching. *arXiv preprint arXiv:1903.06235*, 2019.

[50] A. Yazidi, O.-C. Granmo, and B. J. Oommen. Learning-automaton-based online discovery and tracking of spatiotemporal event patterns. *IEEE Transactions on Cybernetics*, 43(3):1118–1130, 2013.

[51] Anis Yazidi and Hugo L Hammer. Solving stochastic nonlinear resource allocation problems using continuous learning automata. *Applied Intelligence*, 48(11):4392–4411, 2018.

[52] Anis Yazidi, Hugo L Hammer, and Tore M Jonassen. Two-time scale learning automata: an efficient decision making mechanism for stochastic nonlinear resource allocation. *Applied Intelligence*, pages 1–14, 2019.

[53] Junqi Zhang, Cheng Wang, Di Zang, and Mengchu Zhou. Incorporation of optimal computing budget allocation for ordinal optimization into learning automata. *IEEE Transactions on Automation Science and Engineering*, 13(2):1008–1017, 2015.