ARTICLE IN PRESS

Information Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Applying Reinforcement Learning towards automating energy efficient virtual machine consolidation in cloud data centers

Rachael Shaw [*], Enda Howley, Enda Barrett

School of Computer Science, National University of Ireland, Galway, Ireland

A R T I C L E I N F O

Article history:
Received 9 June 2020
Received in revised form 17 November 2020
Accepted 13 January 2021
Available online xxxx
Recommended by Gottfried Vossen

Keywords:
Energy efficiency
Virtual machine consolidation
Reinforcement learning
Artificial intelligence

A B S T R A C T

Energy awareness presents an immense challenge for cloud computing infrastructure and the development of next generation data centers. Virtual Machine (VM) consolidation is one technique that can be harnessed to reduce energy related costs and environmental sustainability issues of data centers. In recent times intelligent learning approaches have proven to be effective for managing resources in cloud data centers. In this paper we explore the application of Reinforcement Learning (RL) algorithms for the VM consolidation problem demonstrating their capacity to optimize the distribution of virtual machines across the data center for improved resource management. Determining efficient policies in dynamic environments can be a difficult task, however, the proposed RL approach learns optimal behavior in the absence of complete knowledge due to its innate ability to reason under uncertainty. Using real workload data we provide a comparative analysis of popular RL algorithms including SARSA and Q-learning. Our empirical results demonstrate how our approach improves energy efficiency by 25% while also reducing service violations by 63% over the popular Power-Aware heuristic algorithm.

1. Introduction

Cloud computing services offered by companies such as Amazon and Google deliver on-demand virtualized resources which can be accessed over the internet and charged on a pay as-you-go basis. The ability to scale up or down computing resources in response to current demand coupled with the expanding web of smart devices and the eclectic range of cloud services has resulted in the exponential growth in demand for data centers causing them to become one of the fastest growing users of electricity [1]. As the global climate crisis escalates and the substantial energy related costs and associated carbon emissions of data centers continue to linger, cloud providers have come under mounting pressure to make a concerted effort towards the continued improvements in data center efficiency. Over the last number of years energy related costs and environmental sustainability issues have had a discernible impact on the direction of research initiatives due to the sheer enormity of energy consumption rates from existing data centers. In particular, research has shown that in 2015, Google data centers consumed the same amount of energy as the entire city of San Francisco [2]. Studies have reported that this sector alone currently utilizes approximately 7% of global electricity, this figure is predicted to rise up to 13% by the year 2030 [3]. This sector is also responsible for an estimated 2% of global CO_2 emissions, a figure which is on par with that produced by the aviation industry [3]. Other studies have also confirmed that the number of mobile internet users is expected to increase from 3.6 billion in 2018 to 5 billion by 2025, while the amount of Internet of Things (IoT) connections is projected to triple to over 25 billion by 2025 [4]. These trends are the main contributing factor behind the exponential growth in demand for data centers and cloud services at large. Over the last number of years improvements have been made in the design and operation of data centers to stabilize energy consumption. However, given the capricious nature of the tech environment coupled with the substantial growth in the use of cloud based services it is anticipated that future demand could exceed efficiency improvements [5]. This motivates greater efforts into the continued development of novel energy efficient approaches in data center operations.

However, despite the benefits gained from the advancements in virtualization technologies one of the major inefficiencies in data center deployments is caused by poorly managed and idle resources [6]. Research has shown that even hosts that operate at low utilization such as 10% have been proven to use up to 50% of their maximum power resulting in significant draws on energy consumption [7]. To add, given the continued growth of cloud services, resource management has become an even more complex and challenging issue in cloud data centers due to their stochastic nature, consisting of vast amounts of resources and wide ranging and diverse application workloads with dynamic resource requirements [8]. VM consolidation is one mechanism that can

* Corresponding author.
E-mail address: r.shaw4@nuigalway.ie (R. Shaw).

https://doi.org/10.1016/j.is.2021.101722
0306-4379/© 2021 Elsevier Ltd. All rights reserved.

be employed to improve resource management in the cloud. This approach enables better utilization of available resources through the consolidation of multiple workloads on the same physical machine. Each VM executes in apparent isolation running its own Operating System (OS) and applications [9]. In addition, the use of virtualization technology enables live migration of VMs between different hosts in the data center [10]. The importance and relevance of resource optimization in cloud environments has led to the proposal of many consolidation mechanisms which consider different factors from performance and resource availability, interference reduction, security issues, communication costs, while also proving to be an effective method for reducing energy consumption in data center operations [11–14]. Although the resource management mechanisms of public clouds are not made known to the public due to business regulations, research has shown that one of the major limitations of open-source cloud infrastructure management platforms such as OpenStack and Eucalyptus is the lack of energy-aware consolidation mechanisms [15]. VM consolidation greatly improves resource management and data center efficiency by packing a larger number of VMs onto a reduced number of servers using live migration in an effort to optimize resource usage while also satisfying user specified Service Level Agreements (SLA) [16]. Consolidating a larger number of VM instances on an already loaded host can however, cause a surge in energy consumption while also potentially causing the host to become overloaded incurring service violations and further requiring VMs to be migrated to additional hosts in the datacenter [17]. Conversely, placing a VM on an underutilized host promotes the continuation of poor resource utilization. As a result, striking a balance between competing objectives such as energy efficiency and the delivery of service guarantees is essential to achieving high performance while reducing overall energy consumption.

To date there has been an extensive amount of research devoted to classical heuristic based algorithms for the VM consolidation problem which have proven to be popular and widely used approaches due to their simplicity and also their ability to deliver relatively good results. Heuristic algorithms are designed to provide a good solution to the problem in a relatively short time frame. However, one significant limitation of many heuristic based VM consolidation algorithms is that they are relatively static and as a result they are limited in their effectiveness to reduce energy consumption in more dynamic and volatile cloud environments. In the cloud resource demands are dynamic and stochastic by nature. Application workloads invariable fluctuate overtime and often exhibit varying demand patterns. As a result these types of approaches can quickly result in redundant consolidation decisions having a negative impact on energy and performance. Intelligent automation is becoming a more essential component in order to develop self-sufficient resource management algorithms in the cloud which have the capacity to learn optimal decisions and adapt their behavior even in the absence of complete knowledge due to their ability to reason under uncertainty. Reinforcement Learning (RL) [18] is a machine learning technique which enables an intelligent agent to learn an optimal policy by selecting the most lucrative action out of all possible actions without prior knowledge of the environment. Other popular statistical and ML techniques have also been recently explored in the literature in a bid to improve resource management in the cloud, these include neural networks [19], SVM [20] and ARIMA modeling [21]. Furthermore, more specialized neural networks such as a Recurrent Neural Network (RNN) have also been proposed in the literature to estimate future resource demands. In particular, Duggan et al. [22] implemented a multiple time step look ahead algorithm using an RNN to predict both CPU and bandwidth resources to improve live migration in cloud data centers, they compared their results against both traditional one step ahead nonlinear and linear forecasting algorithms. Hicham et al. [23] demonstrated the benefits of applying a ANN learning algorithm to improve CPU scheduling in the cloud. In their study they compared several types of ANNs which have been successfully applied across multiple fields, including a multilayer perceptron and RNN. Janardhanan et al. [24] explored the impact of employing a Long Short Term Memory (LSTM) RNN and compared its performance to an ARIMA model for CPU load prediction of machines in a data center. For more information on RNN models refer to the work of Khalil et al. [25]. However, although these models have shown to be effective in their application, one limitation of these proposed approaches is that they are trained in an off-line manner, as a result in order to incorporate and take advantage of newly available data these models must be retrained. In contrast, an RL approach has the capacity to effectively operate in an online manner even when a complete environmental model is unavailable. Through repeated interactions with the environment an RL agent has the capacity to dynamically learn a good policy and generate improved decision making to operate effectively in its environment. When new information becomes available it is incorporated into the action value function estimates of the RL model. From this point of view, given the capricious and inherently dynamic nature of the cloud environment an RL approach offers a more flexible, robust and practical solution whose experiential learning style is more suited to learning in cloud environments.

In this work we present a self-optimizing RL based VM consolidation algorithm to optimize the allocation of VM instances to achieve greater energy gains while also improving the delivery of service guarantees in the data center. The RL agent continues to learn an optimal resource allocation policy depending on the current state of the system through repeated interactions with the environment. In the Machine Learning (ML) literature, many different learning techniques have been developed and proposed over the years. However, there is no global learning algorithm that works best for all types of problem domains and therefore a particular algorithm must be selected based on its suitability to the problem [26,27]. In this work, we apply two popular RL techniques namely Q-learning and SARSA and we compare their performance for the VM consolidation problem using a selection of cloud performance metrics. One of the fundamental challenges faced by model-free learning agents is the tradeoff between exploration and exploitation in the agents quest to learn a good policy, in this work we apply two of the most widely implemented mechanisms to manage this trade off and show and they perform with each of the RL learning techniques. Lastly, in this work we also explore a reward shaping technique known as Potential Based Reward Shaping (PBRS). One of the more profound limitations of standard RL algorithms is the slow rate at which they converge to an optimal policy [28]. PBRS allows expert advice to be included in the learning model to assist the agent to learn more rapidly and as a result encouraging more optimal decision making in the earlier stages of learning. Our results show the RL agents ability to learn and adapt in a highly volatile cloud environment while also delivering an intelligent energy performance tradeoff capability resulting in improved energy efficiency and performance.

The contributions of this paper are the following:

- Using an intelligent RL learning agent we present an autonomous VM consolidation model capable of optimizing the distribution of VMs across the data center in order to achieve greater energy efficiency while also delivering improved performance.

- We provide a comparative study to evaluate the performance of two state-of-the-art RL learning algorithms along with alternative exploration mechanisms and we demonstrate the application of these approaches in managing the energy performance tradeoff to resolve the VM consolidation problem.
- In contrast to existing approaches we also explore the application of PBRS which is an advanced RL technique that incorporates domain knowledge into the reward structure allowing for improved guidance during the learning process.
- Unlike heuristic based algorithms this approach is more suitable for decision making in highly dynamic environments while also considering allocation decisions that could possibly suffer from delayed consequences. Using real workload traces we evaluate our solution against a widely known energy aware consolidation heuristic which has shown to deliver good results [16]. Overall we demonstrate the benefits of incorporating intelligent and adaptive RL modeling and show the ability of our RL based consolidation algorithm to achieve improved energy efficiency and performance.

The remainder of this paper is structured as follows: Section 2 discusses related work and background material. Section 3 formulates the research problem and the proposed solution. Section 4 presents our experimental setup and performance metrics. Section 5 evaluates the performance of popular RL learning algorithms and techniques and selects an appropriate learning agent to incorporate into our consolidation approach. Lastly, Section 6 concludes the paper.

2. Related research & background

This section discusses related research while also providing the necessary background material for the classification techniques that have been compared in this work.

2.1. Resource management in the cloud

Over the last number of years VM consolidation has gained widespread attention in the literature. Heuristic based resource management algorithms are the most commonly used methodologies to optimize resource usage in the cloud due to their simplicity and their ability to deliver good results. Lee et al. [6] proposed two task consolidation heuristics which seek to maximize resource utilization while considering active and idle energy consumption in consolidation decisions, their results showed promising energy savings. Verma et al. [29] presented pMapper, a power and migration cost aware placement controller which aims to dynamically minimize energy consumption and migration costs while maintaining service guarantees. Cardosa et al. [30] explored the impact of the min, max and shares features of virtualisation technologies to improve energy and resource usage. The authors used these features to inform their consolidation approach while also providing a mechanism for managing the power-performance tradeoff. One of the most important well known contributions in the field of VM consolidation is accredited to the work of Beloglazov et al. as outlined in two of their more highly cited papers [31,32]. They proposed and explored various heuristics for the three stages of VM resource optimization consisting of Host overutilization/underutilization detection, VM selection and VM consolidation. In particular, they proposed the Lr-MMT algorithm which manages host utilization detection and VM selection. In addition, they proposed the Power Aware consolidation algorithm. This heuristic considers the heterogeneity of cloud resources by selecting the most energy efficient hosts first in order to reallocate VMs in the data center. Their experimental results concluded that the composition of Lr-MMT in conjunction with Power-Aware significantly outperformed all other resource management approaches resulting in a profound reduction in energy consumption and SLA violations.

Alternatively, research initiatives continue to explore the application of ML methodologies in order to evolve a new generation of autonomic resource management strategies. Farahnakian et al. [10] introduced a VM consolidation approach which employs a K-nearest neighbors forecasting methodology to improve energy and SLA. Nguyen et al. [33] employed Multiple Linear Regression (MLR) to predict over and under-loaded server capacity and demonstrated how it can be integrated with existing consolidation solutions to improve energy consumption. Shaw et al. [34] proposed a predictive anti-correlated VM consolidation algorithm which also considers optimal data transfer times. These works focus on improving resource usage and energy efficiency in the data center. Slama et al. [35] proposed an interference aware consolidation algorithm using Fuzzy Formal Concepts Analysis to minimize interference effects of co-located VMs while also reducing communication costs. In addition, a number of research studies have demonstrated the effectiveness and benefits of employing RL approaches to manage resources in the cloud. Duggan et al. [36] introduced an RL network-aware live migration strategy, their model enables an agent to learn the optimal times to schedule VM migrations to improve the usage of limited network resources. Tesauro et al. [37] presented an RL approach to discover optimal control policies for managing Central Processing Unit (CPU) power consumption and performance in application servers. Their proposed method consists of an RL based power manager which optimizes the power performance tradeoff over discrete workloads. Barrett et al. [38] introduced an agent based learning architecture for scheduling workflow applications, in particular they employed an RL agent to optimally guide workflow scheduling according to the state of the environment. Rao et al. [39] introduced VCONF an RL based VM auto configuration agent to dynamically re-configure VM resource allocations in order to respond effectively to variations in application demands. VCONF operates in the control domain of virtualized software, it leverages model based RL techniques to speed up the rate of convergence in non-deterministic environments. Das et al. [40] introduced a multiagent based approach to manage the power performance tradeoff by specifically focusing on powering down underutilized hosts. Dutreil et al. [41] also showed how RL methodologies are a promising approach for achieving autonomic resource allocation in the cloud. They proposed an RL controller for dynamically allocating and deallocating resources to applications in response to workload variations. Using convergence speed up techniques, appropriate Q function initializations and model change detection mechanisms they were able to expedite the learning process. In the literature, the most similar work to ours is found in the work of Farahnakian et al. [42]. Their work also focused on the consolidation problem overall, however, they proposed an RL agent using a Q-learning algorithm specifically to determine the power mode of a host to reduce energy consumption in the data center. The key differences offered in this work is that our work implements an RL agent to determine how to consolidate VMs to reduce energy consumption and improve performance guarantees. Furthermore, we also explore the application of prominent RL learning algorithms (Q-learning & SARSA) while also exploring mechanisms including PBRS and demonstrate their impact on the consolidation problem. In the literature many approaches employ a single RL learning algorithm such as SARSA or Q-learning [36,39,41,42]. Unlike in our previous study where we initially proposed an RL approach with PBRS [43], we extend this work by providing a comparative study, as mentioned above, we compare popular RL learning algorithms

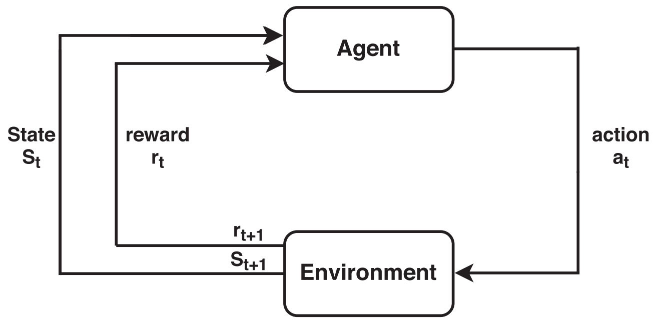

Fig. 1. The agent environment interaction model [44].

and also various exploration mechanisms which is necessary in ensuring the best possible performance from the proposed RL agent. Furthermore, this work focuses on not only demonstrating the associated benefits of our RL agent according to the selected performance metrics but also exposing the mechanics behind the proposed RL agent. In particular, arising from these findings an important and fundamental dimension of this research is to expose why an RL approach proves to be a more efficient and robust solution.

2.2. Reinforcement learning

Reinforcement Learning enables an agent to learn optimal behavior in a highly stochastic, non deterministic environment with no prior knowledge of its surrounding environment. An agent interacts with its environment in order to gain knowledge about how to optimize its behavior and discover the most optimal policy to satisfy its design objectives, this is often a cyclical learning process of state–action–reward interactions as illustrated in Fig. 1. The agent must discover which actions yield the greatest reward by means of trial and error. Notably the selected action effects both the immediate reward and also successive environmental states and therefore all subsequent rewards, in the literature this is often referred to as a delayed reward [44].

All RL control problems can be intuitively modeled as Markov Decision Process (MDP), this is considered to be the standard approach to defining learning in sequential decision making problems faced with adverse uncertainty. The MDP framework allows agents to learn an optimal policy incrementally by means of simulated trials [45]. All MDPs are governed by a Markov property. This property states that only the current state of the environment is required in the prediction of all future states [44]. As outlined by Barrett et al. [46] the MDP formalism comprises of states, actions, transition probabilities and rewards respectively (S, A, p, q) where:

- S, denotes a set of possible states;
- A, represents a set of actions;
- $p(s_{t+1}|s_t, a_t)$ represents a probability distribution governing state transitions;
- $q(s_{t+1}|s_t, a_t)$ denotes the probability distribution governing the rewards received $R(s_t, a_t)$;

The learning process as illustrated in Fig. 1 is often segregated into discrete time steps t. At the end of each time step t the learning agent occupies state $s_t \in S$. The agent selects a possible action $a_t \in A(s_t)$ where $A(s_t)$ refers to the set of possible actions available in the current state s_t. The execution of the selected action induces an environmental state transition s_{t+1} and results in the allocation of a reward $R(s_t, a_t)$ to the agent. The state transition probability $p(s_{t+1}|s_t, a_t)$ calculates the probability of a transition to s_{t+1} given that the agent occupies state s_t and selects action a_t. $q(s_{t+1}|s_t, a_t)$ denotes the expected reward received by the agent after transitioning from state s_t to state s_{t+1} by executing action $A(s_t)$ [46].

2.2.1. Temporal difference learning

RL as outlined above is primarily concerned with converging to an optimum solution. Temporal Difference (TD) methodologies as outlined by Sutton and Barto [44] are one such set of approaches which can be harnessed in order to discover an optimal policy. They implement prediction based learning by incrementally updating current estimates based on the outcome of previous estimates, a process often referred to as bootstrapping [46]. The application of TD procedures in real world prediction problems has proven to be a more effective solution in comparison to traditional approaches resulting in more accurate predictions and faster convergence rates [44]. There are several TD learning strategies, two of which are the most widely implemented when faced with an unknown model of the environment are Q-learning and SARSA.

Q-learning is one of the most well understood and widely used off-policy TD learning algorithm. Q-values denote the expected reward for each state–action pair often stored in a matrix which is a representation of the agents knowledge to date [47]. At each discrete time step t an agent updates its current Q-value estimate based on the outcome of selecting action a_t while in state s_t where the update rule is defined as [44]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big] \tag{1}$$

The discount factor γ determines the degree to which an agent favors long term rewards over short term gains. The rate of γ typically ranges between 0 and 1. A rate set closer to 1 results in an agent that is more forward looking and strives to maximize future rewards while a rate closer to 0 results in an agent that assigns a greater weight to short term rewards often referred to as a myopic agent. The learning rate α also ranges between 0 and 1. It determines the rate at which Q-values are updated. A learning rate closer to 0 results in less updates to the Q-value estimates and therefore a decline in the agents learning, while a rate of 1 induces more learning. One of the differentiating features of off-policy algorithms such as Q-learning is that in the event of an agent selecting a suboptimal action the agent still attempts to pursue the most optimum policy possible [45]. This is achieved by updating Q-values based on adding the immediate reward to the discounted estimation of the most rewarding action the agent could take in the resulting state as exemplified in the following update rule $r_{t+1} + \gamma \max_a Q(s_{t+1})$. The Q-learning algorithm is outlined below, the initialization of the Q-values denoted $Q(s, a)$ can be generated in a random manner where the value of $Q(s, a)$ is set between 0 and 1 for all s, a as indicated by Suttin and Barto [44].

Algorithm 1: *Q-Learning*

```
Initialize $Q(s, a)$ arbitrarily
Repeat(for each episode):
    Initialize $s$
    Repeat(for each step of episode):
        Choose $a$ from $s$ using policy derived from $Q$($e.g., \epsilon - greedy$)
        Take action $a$, observe $r$, $s_{t+1}$
        $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s, a)\big]$
        $s \leftarrow s_{t+1}$
    until $s$ is terminal
```

SARSA was originally introduced by Rummery and Niranjan in 1994 [48]. In their work the authors argued that the use of

$\gamma \max_a Q(s_{t+1}, a)$ as observed in Q-Learning was not a reliable estimation of the expected future return for any given state s_t. Q-learning assumes that the most optimal action yielding the greatest reward will be selected in the subsequent state and therefore updates Q-values based on this assumption. As a result the authors argument was warranted on the merits of the probability of inaccurate Q-values in the early stages of the learning process while in the latter stages the occurrence of an overestimated maximum value. SARSA is an acronym for state, action, reward, state, action. Its name is derived from a quintuple of events that must occur in order to transition from one state to the next and update the current Q-value approximations. The update rule as outlined below [44]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big] \quad (2)$$

In contrast to Q-learning the SARSA learning algorithm is considered to be an on-policy algorithm which updates action value estimates strictly based on the experience gained as a result of selecting an action according to some policy. In this regard, an agent implementing a SARSA learning algorithm in some way recognizes that a suboptimal action may be invoked and therefore this approach is often likely to produce more accurate estimates. As a result on-policy algorithms are more favorable in highly volatile environments, they tend to avoid parts of the state–action space where exploration poses greater danger [44]. The SARSA algorithm is presented below, the initialization of the Q-values denoted $Q(s, a)$ can be generated in a random manner where the value of $Q(s, a)$ is set between 0 and 1 for all s, a as indicted by Sutton and Barto [44].

Algorithm 2: *SARSA*

```
Initialize $Q(s, a)$ arbitrarily
Repeat(for each episode):
    Initialize $s$
    Choose $a$ from $s$ using policy derived from $Q(e.g., \epsilon - greedy)$
    Repeat(for each step of episode):
        Take action $a$, observe $r$, $s_{t+1}$
        Choose $a_{t+1}$ from $s_{t+1}$ using policy derived from $Q(e.g., \epsilon - greedy)$
        $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s, a)\big]$
        $s \leftarrow s_{t+1}$     $a \leftarrow a_{t+1}$
    until $s$ is terminal
```

2.2.2. *Action selection strategies*

One of the fundamental challenges faced by model-free learning agents is the tradeoff between exploration and exploitation [44]. The inherently complex and uncertain nature of RL based control problems require the agent to explore all states infinitely often in order to discover a more optimal policy, this is particularly vital in highly dynamic environments in order to maintain an up to date policy [45,49]. However, this is often conducive to the selection of suboptimal actions resulting in a local optimum [50]. Therefore, in order to perform well the agent should exploit the knowledge it has already gained. Balancing such a tradeoff is critical as it impacts greatly on the speed of converging to an optimal policy and also the overall quality of the learned policy. Two of the most widely implemented methodologies to manage this trade off are ϵ-greedy and softmax.

ϵ-greedy exploration mechanism has gained widespread recognition across the research community due to its capacity to achieve near optimal performance with the use of a single parameter when applied to a variety of problem domains [51]. This probability parameter is known as epsilon and it controls the rate of exploration in the state space. At each time step t the probability metric is compared against a random value and if a correlation exists between both values a random action is selected by the agent. One of the major drawbacks of ϵ-greedy is that in order to explore it chooses equally among all actions and therefore, it is equally probable to choose the worst appearing action as opposed to the next best action. In order to overcome this problem the Softmax mechanism assigns action probabilities according to the expected utility, thus ensuring higher rewarding actions are more likely to be explored.

2.2.3. *Reward shaping*

One of the more profound limitations of conventional RL algorithms is the slow rate at which the agent converges to an optimal policy [52]. In order to expedite the learning process the research community have developed more advanced techniques which incorporate domain knowledge in the form or an additional reward into the reward structure so that learning agents are guided faster to more optimal solutions [53]. PBRS is an extension of the standard reward shaping approach which proves to be a more reliable and effective solution in reducing the time required by the agent to learn [54]. PBRS associates a potential to each state, thus the formulation of PBRS is the difference between both potentials [54] which can be formally expressed as:

$$F(s, a, s_{t+1}) = \gamma \Phi(s_{t+1}) - \Phi(s) \quad (3)$$

where Φ is the potential function which maps states to associated potentials and γ is defined as the same discount factor applied in the update rule.

3. Dynamic RL VM consolidation model

In order to reach new frontiers in energy efficient cloud infrastructure we propose an RL Consolidation Agent known as Advanced Reinforcement Learning Consolidation Agent (ARLCA) which is capable of driving both efficiency and improved delivery of service guarantees by dynamically adjusting its behavior in response to changes in workload variability. More specifically, ARLCA is presented with a list of VMs that require allocation to suitable hosts in the datacenter. Through repeated interactions with the environment ARLCA discovers the optimal balance in the dispersal of VMs across the data center so as to prevent hosts becoming overloaded too quickly but also ensuring that resources are operating efficiently.

This work presents a more effective VM consolidation strategy which leverages a state-of-the-art RL approach to foster a more intelligent energy and performance aware consolidation approach. We model the problem using a simulated data center consisting of heterogeneous resources, including m hosts and n VMs, where $PM = \{pm_1, pm_2, \ldots, pm_m\}$ and $VM = \{vm_1, vm_2, \ldots, vm_n\}$. The proposed system model is depicted in Fig. 2. In the proposed system VMs in the data center are initially allocated onto a host based on the requested resources which dynamically change over time. To reduce energy consumption and improve the delivery of SLA it is critical to optimize the distribution of VMs across the data center using live migration and reallocate VMs to other hosts in the data center. In this work we consider VM resource optimization to be a three-stage process consisting of host overload detection, VM selection and VM consolidation. In this work, first the system continuously monitors the resource usage of each host in the data center to detect hosts that are likely to become overloaded, next the system selects VMs to be migrated and finally these VMs get consolidated onto a more suitable host that will drive energy efficiency while being cognizant of performance. In our system model the global resource manager is responsible for managing resource allocation and live migration. It consists of three key components: the performance monitor, the VM selection algorithm and the proposed ARLCA consolidation algorithm which provides decision support for VM consolidation. The performance monitor continuously monitors

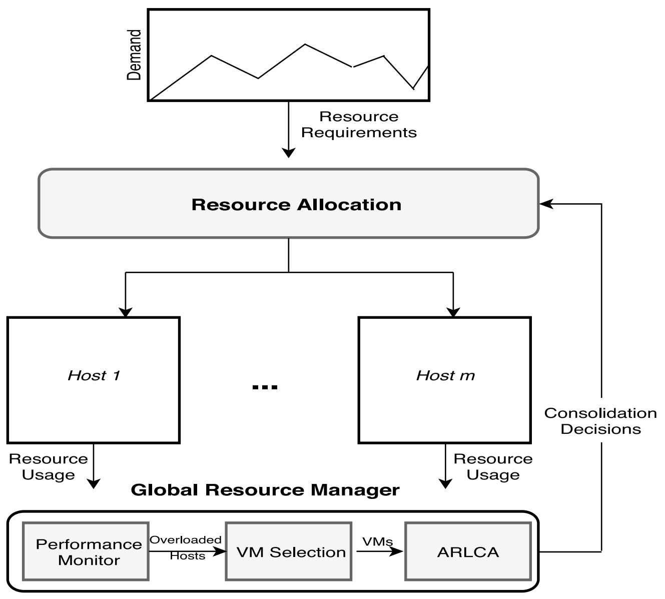

Fig. 2. System model.

resource usage on each host and stores this as historical resource usage data. Next the popular Local Regression algorithm [31] is used to infer the probability of a host becoming loaded, next the Minimum Migration Time algorithm also proposed by Beloglazov et al. [31] is used to selects which VMs to migrate. ARLCA is then used to generate consolidation decisions for the data center. Through repeated interactions with the environment ARLCA discovers an improved balance in the dispersal of VMs across the data center so as to prevent hosts becoming overloaded too quickly but also ensuring that resources are operating efficiently.

3.1. ARLCA learning models

The application of RL in more complex problem domains requires careful definition of the state–action space. RL learning algorithms such as Q-learning and SARSA can suffer from exponentially large state–action spaces often referred to as the curse of dimensionality [55]. This problem can often have a severally negative impact on the overall performance of the agent while also limiting its ability to operative effectively. The implication of such a problem within a cloud infrastructure would impact both performance and energy consumption in the data center as the agent continues an exhaustive search of the state–action space. This would result in slower decision making and prolonging the total amount of time resources are being poorly utilized. It is therefore imperative to the design of the proposed RL energy model to define a more cohesive and sustainable state–action space.

The state space S represents the global state of the environment. We design a novel state–action space by defining all possible states and actions as a percentage ranging from 0%–100%. We use increments of 1% which provided the best performance according to an experimental parameter sweep. The state space S is defined below in Eq. (4). It can be represented as the sum of the number of active hosts a_i in the environment as a percentage of the total number of hosts M.

$$S = 100x\frac{\sum_{i=1}^{n} a_i}{M} \tag{4}$$

where a_i is an active host in the environment stored in a list that must be processed and M is the total amount of hosts.

An action A is a combined variable composed of the utilization rate of a possible host coupled with the size of the VM to be placed. This calculation is defined in Eq. (5), where the host utilization rate U is calculated as the sum of the total requested resources for each VM residing on the host as a percentage of the hosts capacity C. Additionally, VM utilization vmu is computed as the VMs requested resources rr returned as a percentage of the total host capacity C. This determines the size (CPU resource requirements) of the VM to be placed. Research has shown that CPU is one of the most dominant factors in energy consumption and performance reliability [1,16]. As a result, we characterize the resource capacities of each host and resource usage by VMs by the CPU parameter. Defining the state–action space as percentages from 0%–100% significantly reduces the size of the state–action space thus preventing the agent from engaging in an exhaustive search. This type of approach allows for the deployment of a more agile and efficient agent capable of pursuing its design objectives.

$$A = \left[U = 100x\frac{\sum_{i=1}^{n} trr_i}{C} + vmu = 100x\frac{rr}{C} \right] \tag{5}$$

where trr_i is the total requested resources of VM i allocated to host m.

Rewards are determined based on the performance of each host in terms of the delivery of SLA to the allocated VMs and also the energy consumption rate. Similar to the reward function defined by Farahnakian et al. [42]. We compute the SLA violation in a given time step t as the difference between the requested and actual resources for all VMs a denoted below in Eq. (6).

$$SLA_t = \sum_{i=1}^{n} (trr_i - ar_i) \tag{6}$$

where trr_i is the total requested resources of VM i, ar_i is the actual resources allocated and n is the number of VMs. We calculate the magnitude of the score after the action has been executed by the agent according to the SLA at time step t divided by the previous SLA.

The energy score is calculated by dividing the power consumption at the current time step t by the power consumption at the previous time step as denoted below in Eq. (7).

$$P_t = \sum_{i=1}^{M} \frac{power_t}{power_{t-1}} \tag{7}$$

where m is the total number of hosts, $power_t$ is the current power consumption and $power_{t-1}$ is the previous rate.

3.2. Q-learning implementation

The Q-learning based RL learning algorithm for VM consolidation is outlined below. Initially, the global state of the environment is calculated according to Eq. (5) and a list of all possible hosts in the data center is created excluding the host the VM was removed from and also hosts which lack available resources to run the VM. A VM is selected from the migration list (generated by the VM selection algorithm above in Fig. 2) and the host utilization rate for each host on the host list is calculated according to Eq. (5) and returned as a percent ranging between 0%–100%. Next the size of the VM denoted (*VMSize*) is calculated and a list of possible actions is generated using the combined action variable as denoted in Eq. (6). The agent selects a host based on the action selection strategy being followed e.g ϵ-greedy or softmax denoted below in Algorithm 3 as π. Once the host is selected, the VM gets allocated and the global state is recalculated. A reward is also received by the agent calculated according to Eqs. (7) and (8). Next the Q-value update rule is calculated and the result is stored in the Q-value matrix. Lastly, the global state is updated for the next iteration. This algorithm continues until all VMs in the migration list have been reallocated on to other hosts in the environment.

Algorithm 3: *Q-learning VM Consolidation Algorithm*

```
calculate globalState
foreach VM ← VMMigrationList do
    foreach host ← hostList do
        calculate hostUtilisationRate
    end
    calculate VMSize
    foreach host do
        possibleActions ← VMSize + hostUtilisationRate
    end
    select host from possibleActions using $\pi$
    allocate VM
    observe $globalState + 1$, reward
    calculate $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_a Q(s_{t+1}, a) - Q(s,a)\big]$
    update QValueMatrix
    globalState ← $globalState + 1$
end
```

3.3. SARSA implementation

The customized SARSA version of the learning algorithm is presented below in Algorithm 4. The fundamental difference between both algorithms is that SARSA requires that a quintuple of events consisting of state, action, reward, state, action must occur in order to transition from one state to the next and subsequently update the Q-value estimates. SARSA updates Q-value estimates based on the action that will be implemented in the subsequent state denoted in the update rule as $Q(s_{t+1}, a_{t+1})$. As seen in the Q-learning approach, the global state of the environment is calculated according to Eq. (5) and a list of all possible hosts in the data center is created excluding the host the VM was removed from and also hosts which lack available resources to run the VM. The host utilization rate for each host on the host list is calculated according to Eq. (5). A VM is then selected from the migration list and the size of the VM is calculated and a list of possible actions is generated using the combined action variable as denoted in Eq. (6). The agent selects a host based on the action selection strategy being followed e.g ϵ-greedy or softmax denoted below in Algorithm 4 as π. Once the host is selected, that specific VM gets allocated and the global state is recalculated denoted *globalState*+ 1. A reward is also received by the agent calculated according to Eqs. (7) and (8). At this point in the algorithm the first three of five events consisting of state, action, reward have executed. Next, the host utilization rate for each host gets calculated again and we select another VM from the migration list and calculate its size denoted *nextVMSize*. We then generate the range of actions available as above and select an action according to π. The Q-value SARSA update rule is calculated and the result is stored in the Q-value matrix. Lastly, the global state is updated for the next iteration and the selected host become the action in the next iteration.

Algorithm 4: *SARSA VM Consolidation Algorithm*

```
calculate globalState
foreach host ← hostList do
    calculate hostUtilisationRate
end
VM ← VMMigrationList
calculate VMSize
foreach hostUtilisationRate do
    possibleActions ← VMSize + hostUtilisation
end
select host from possibleActions using $\pi$
foreach VM ← migrationList do
    allocate VM
    observe $globalState + 1$, reward
    foreach host ← hostList do
        calculate hostUtilisationRate
    end
    calculate nextVMSize
    foreach hostUtilisationRate do
        possibleActions ← nextVMSize + hostUtilisation
    end
    select host from possibleActions using $\pi$
    calculate $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s,a)\big]$
    update QValueMatrix
    globalState ← $globalState + 1$
    action ← host
end
```

4. Experimental setup

To evaluate the efficiency of the proposed energy-aware learning agent we use the CloudSim toolkit [31] which is a widely used simulation framework for conducting cloud computing research. In CloudSim we model the problem using a data center consisting of 800 physical machines. In particular, the simulator consists of two types of physical hosts modeled as HP ProLiant ML110G4(Intel Xeon 3075, 2 cores 1860 MHz, 4 GB) and HP ProLiant ML110G5(Intel Xeon 3075, 2 cores 2660 MHz, 4 GB). Furthermore, we consider four types of VMs consisting of High-CPU Medium Instances (2500 MIPS, 0.85 GB), Extra Large Instances (2000 MIPS, 3.75 GB), Small Instances (1000 MIPS, 1.7 GB) and also Micro Instances (500 MIPS, 613 MB). To validate our approach the CPU workload used in this study was generated based on a large real-world data set provided by the CoMon project, a monitoring infrastructure for PlanetLab [31]. The traces provide the CPU demand collected from more than a thousand VMs running on servers situated in more than 500 locations globally. The CPU usage is measured in 5 min intervals resulting in 288 intervals in a given trace and measurements are provided for 10 days generating traces from over 10,000 VMs in total. We evaluate and compare the performance of the learning agent using three sets of experiments. Firstly we compare the performance of the learning agent using Q-learning and SARSA with different action selection strategies. Next we analyze the performance of the agent when PBRS is applied. Lastly, we compare the best performing RL algorithm from the previous experimental analysis to the popular PowerAware VM consolidation algorithm proposed by Beloglazov et al. in one of their most highly cited works [16]. Although there exists numerous approaches for the VM consolidation problem the most well-known and commonly used benchmark is the PowerAware approach when energy efficiency is one of the core objectives and therefore this benchmark is an important measure in this work. The PowerAware algorithm considers the heterogeneity of cloud resources and efficiently allocates VM instances to hosts by selecting the most energy efficient hosts first. To produce an accurate and reliable experiment we evaluate the algorithms over a time period of 30 days. In the research field many researchers adopt the PlanetLab data to evaluate and measure the added benefits of their proposed approaches. The PlanetLab data provides a 10 day workload which is considered a standard benchmark dataset and time horizon employed by many in the field to evaluate their proposed resource management algorithms [10,21,31,33]. In this work, we generate a larger workload based on the PlanetLab data which equates to a 30 day stochastic workload to provide a more robust experimental evaluation. In

Table 1
Power consumption of hosts at various load levels in Watts.

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

doing so we seek to provide a more thorough measurement of the adaptability of our learning agent in a cloud environment over an extended time period of 30 days. Furthermore, we argue that an extended 30 day workload provides us with enough oversight to gain insights into the expected performance of the proposed RL agent in a dynamic cloud environment.

4.1. Performance metrics

The main goal of the proposed RL based VM consolidation algorithm is to generate a consolidation approach that reduces energy consumption while also improving the ability of service providers to deliver service guarantees. As a result the key cloud performance metrics used to evaluate the effectiveness of the proposed algorithm are:

- **Energy Consumption**: In this research the energy consumption metric is computed by CloudSim where energy is defined as the total energy consumed in the data center by computational resources as a result of processing application workloads. In a data center power consumption incurred by the servers is mostly determined by resource usage such as CPU, memory and disk storage. However, CPU consumes the most energy [16]. As a result, in this work power consumption of a host is represented by the CPU utilization on the host and reported as energy consumption over time (kWh). To provide an accurate and realistic evaluation we use real data on power consumption provided by the results of the SPECpower benchmark implemented in CloudSim[1] Table 1 illustrates the power consumption rates of the selected servers under different load levels.
- **Service Level Agreement Violations**: The ability of cloud providers to deliver service guarantees is critical to their service offerings. A metric proposed in [31] can be used to calculate the SLAV incurred by VMs in the data center, this metric is also implemented by CloudSim exactly as it is defined in [31]. In particular, service violations are measured as the performance degradation experienced by each VM due to both hosts becoming overloaded and also the number of VM migrations incurred by the cloud system. Specifically, the metric is calculated using both Service Level Agreement Violation Time per Active Host (SLATAH) and Performance Degradation Due to Migrations (PDM). SLATAH is the percentage of time during which each active host in the environment has experienced 100% utilization of its resources and is denoted below in Eq. (8).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}}\ . \tag{8}$$

where N is defined as the total number of hosts, T_{si} is the total time host i has experienced 100% utilization of its resources and T_{ai} is defined as the total time that host i is actively serving VMs.

Live migration enables VMs to be transferred between physical hosts in the data center to improve resource management. However, the migration process also has a negative impact on the performance of applications executing on migrating VMs in the data center. PDM calculates the overall decrease in performance experienced by migrating VMs. The PDM equation is presented below in Eq. (9).

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}}\ . \tag{9}$$

where M is defined as the number of VMs, C_{dj} is the estimated performance degradation for VM_j, in our experiments C_{dj} is set as 10% of the CPU utilization for all migrations of VM_j, while C_{rj} is the total requested CPU by VM_j over its lifespan.
The combined metric used to measure the overall service violations in the data center is shown below in Eq. (10).

$$SLAV = SLATAH \times PDM \tag{10}$$

- **Number of Migrations**: In our evaluation we also compare the total number of VM migrations incurred by each algorithm. Live migration can cause interference between source and destination host resulting in additional SLAV. Therefore the number of migrations should also be reduced.
- **ESV**: Energy Service Level Agreement Violations (ESV) is a combined metric which recognizes the multi-objective nature of energy aware resource allocation. The goal of this approach is to reduce energy consumption while also delivering the required level of service, therefore, the ESV metric allows for a more fine-grained measurement of how well this tradeoff is being managed in the data center. The lower the result of the ESV metric means that the approach has an improved capacity to reduce the energy rate while also making less SLAV. The ESV calculation is shown below.

$$ESV = Energy \times SLAV \tag{11}$$

5. Results & discussion

In this section we evaluate the performance of the proposed RL approach over a 30 day workload using the above cloud performance metrics. We firstly evaluate the performance of an RL agent following Q-learning and SARSA learning methodologies coupled with either an ϵ-greedy or softmax action selection strategy. Next we evaluate the application of PBRS into the best performing algorithm from the first set of experiments. Lastly, we compare the proposed RL learner against the benchmark consolidation algorithm using the performance metrics outlined above.

5.1. Comparison of RL methodologies

The goal of the following chapter is to conduct a series of experiments in order to compare the performance of both algorithms utilizing two different action selection strategies, namely ϵ-greedy and softmax. This analysis is necessary in determining the most robust combination of learning mechanisms which will ensure the best possible performance from the proposed RL learning agent. This set of experiments compares the following combination of learning and action selection policies:

1. Q-learning and ϵ-greedy.
2. Q-learning and softmax.
3. SARSA and ϵ-greedy.
4. SARSA and softmax.

[1] The SPECpower benchmark. http://www.spec.org/power_ssj2008/.

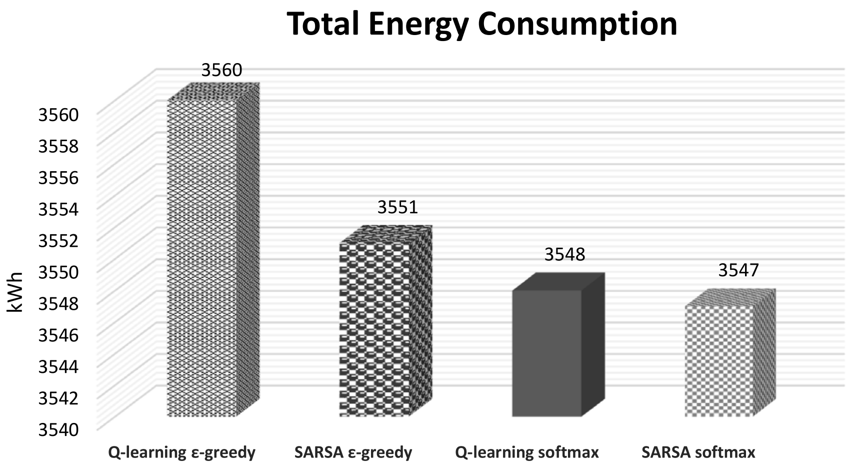

Fig. 3. Total energy consumption over 30 day workload.

In order to conduct a thorough examination of each policy there are two broad divisions under which the performance of each policy must be analyzed. These categories include the overall robustness and adaptability of our agent when faced with a more dynamic workload and also the rate at which the agent converges to an optimum solution over a single iterative workload. In light of this, each combination of the above policies will be run over both a 30 day stochastic workload and also a single workload over 100 iterations. The 30 day workload has been created using the PlanetLab workload traces. It has been implemented in order to examine the performance of our agent under a more volatile workload which the agent has not been continuously learning over. As a result this will expose the capability of our agent in a more realistic cloud scenario. Additionally, the performance of each combination of policies must be consistently measured in exchange for reliable and accurate findings. Therefore, the performance metrics outlined below will be used to evaluate each algorithmic approach. Fig. 3 illustrates the total energy output for each combination of policy over a rigorous 30 day stochastic workload. One of the observations to be noted is that both variations in the Q-learning algorithm incurred an increase in energy consumption compared to SARSA relative to the action selection policy being implemented. Furthermore, both Q-learning softmax and SARSA softmax consumed the least amount of energy. SARSA softmax prevailed with a total energy consumption of 3547 kWh with an average of 118.23 kWh. The performance difference of each approach in relation to energy is minimal given the difference between both SARSA and Q-learning is between 0.2% and 0.02%. In the early stages of learning the Q-learning agent due to the nature of its policy attempts to update Q-values based on the best action that might be selected in the subsequent state. However this off-policy approach to learning can result in the agent taking high risks generating allocation decisions that cause higher energy increase at the beginning of learning. Conversely, SARSA is more conservative avoiding high risk and in an online environment is also important as we care about the rewards gained while learning.

Adhering to Service level agreements is essential in the delivery of cloud based services. Service level agreements govern the availability, reliability and overall performance of the service. Failure to deliver the pre agreed QoS can have a detrimental impact on both profitability and the ability of the service provider to gain a competitive advantage. In this respect, it is imperative in this analysis to consider the number of SLAV incurred by each variation of policy. Fig. 4 depicts the total amount of SLAV over the 30 day workload. As shown, similar to the results above Q-learning ϵ-greedy was subjected to the greatest number of violations while SARSA softmax once again generated a slightly better result.

While migrating VM instances across discrete physical hosts in the data center provides a plausible mechanism for performance optimization and also plays a pivotal role in the reduction of energy and power consumption. The process of spawning a VM image and instantiating it on a new host can have an impact on the delivery of service guarantees. Therefore, it is vital to implement a migration strategy which strikes a balance between both energy and performance and thus preventing excessive migrations. Fig. 5 depicts the number of migrations incurred by each approach. Both SARSA ϵ-greedy and Q-learning ϵ-greedy resulted in the greatest number of migrations at 466,424 and 465,282 respectively. Once again Q-learning softmax and SARSA softmax produced the least amount of migrations improving the performance of the agent. In terms of the SARSA learning algorithm, through the implementation of the softmax action selection the agent generated a reduction of approximately 10,128 (2%) in the number of migrations. Similar results were achieved when using Q-learning softmax, reducing migrations by 7957 (2%) over its ϵ-greedy alternative. However, again we note that the difference between both SARSA and Q-learning for this metric is between 0.2% and 0.7%.

One of the challenging problems in achieving improved energy conservation in data center deployments is that there can also be an inverse relationship between energy and the level of service provided. A reduction in energy consumption can often cause a surge in the number of SLAVs. This is often caused by an aggressive consolidation approach. One of the key objectives of the proposed RL model is to strike a balance between both variables in order to manage this relationship and drive a more globalized optimization strategy. The ESV metric plays a pivotal role in addressing this energy performance tradeoff, as previously mentioned ESV combines both energy and SLAV in order to provide a more fine-grained measure of this impact. In a broad sense, this will allow for an unambiguous representation of the data center performance. Fig. 6 present the performance of the data center using each variation of policy under the ESV metric. Once again as shown Q-learning ϵ-greedy performed least best while SARSA softmax again performed marginally better than Q-learning softmax.

Overall, SARSA softmax performed slightly better across all performance metrics. In most instances, it showed a small improvement of less that 1% over the other approaches relative to the action selection strategy applied. Although it can be said that both SARSA and Q-learning resulted in similar performance. However, it remains important in identifying the most promising

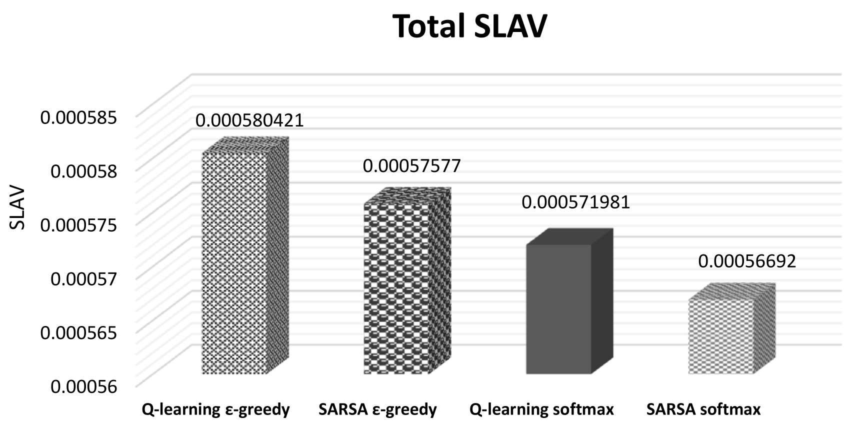

Fig. 4. Total SLAV over 30 day workload.

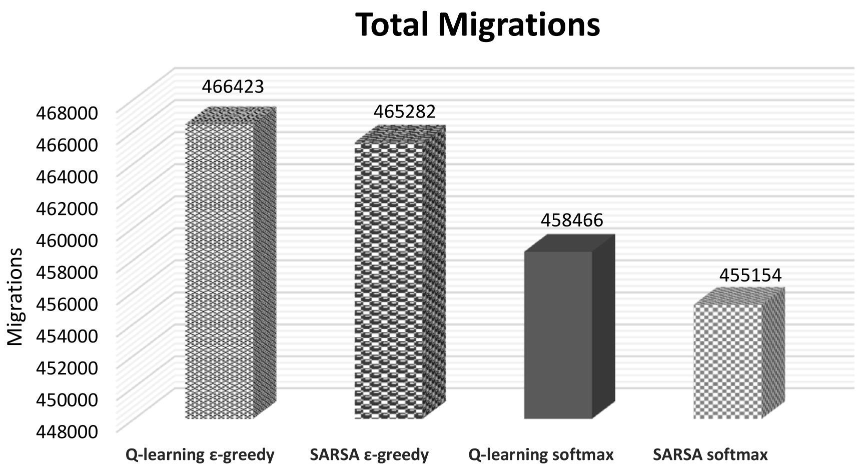

Fig. 5. Total migrations over 30 day workload.

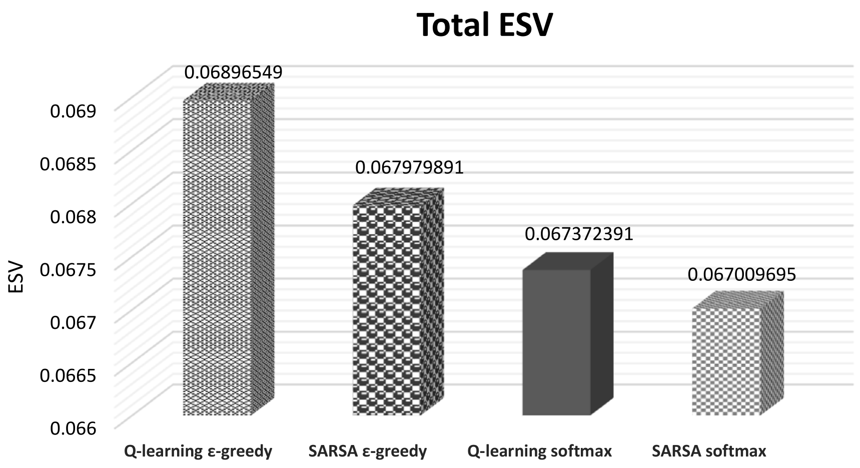

Fig. 6. ESV over 30 day workload.

policy for further analysis. The Q-learning algorithm is an off-policy algorithm which assumes that the action yielding the greatest reward will be selected in the subsequent state, As shown in the work of [44] in a simplistic Gridworld environment, this approach overtime can generate a more optimal policy over SARSA, however, the agent can also receive less lucrative rewards by taking more risk and has shown that its online performance is worse than SARSA. This is because it over estimates the Q-values during the learning process. In this work our reward structure is based on both energy and SLAVs which have an impact on other metrics such as migrations and ESV. As a result, over a realistic the 30 day stochastic workload it makes consolidation decisions which result in overall a small increase in total energy consumption and SLAV. Similiar results were also reported in the work of Arabnejad et al. [26]. They also applied SARSA and Q-learning for auto-scaling cloud resources. Their results showed that both approaches can perform effectively to improve QoS and reduce cost, however the main difference that emerged is that when using a real life workload with a bursty and unpredictable pattern their SARSA approach behaved slightly better in terms of the

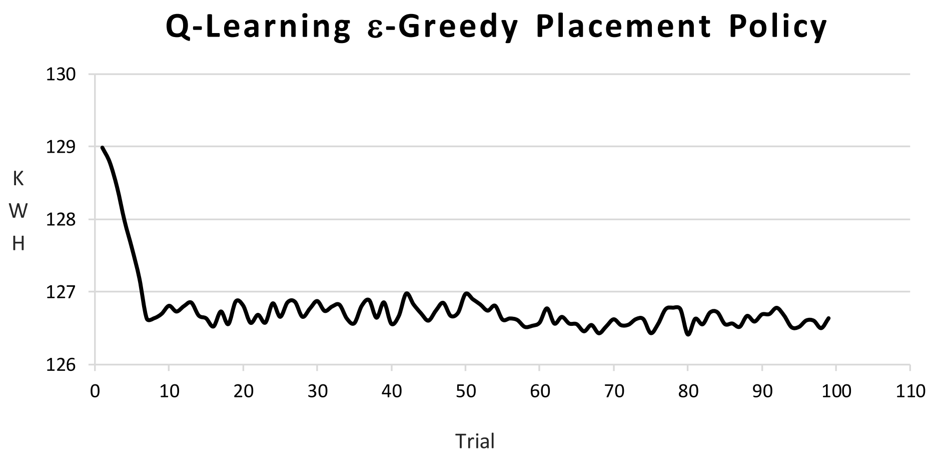

Fig. 7. Energy consumption over 100 iterations.

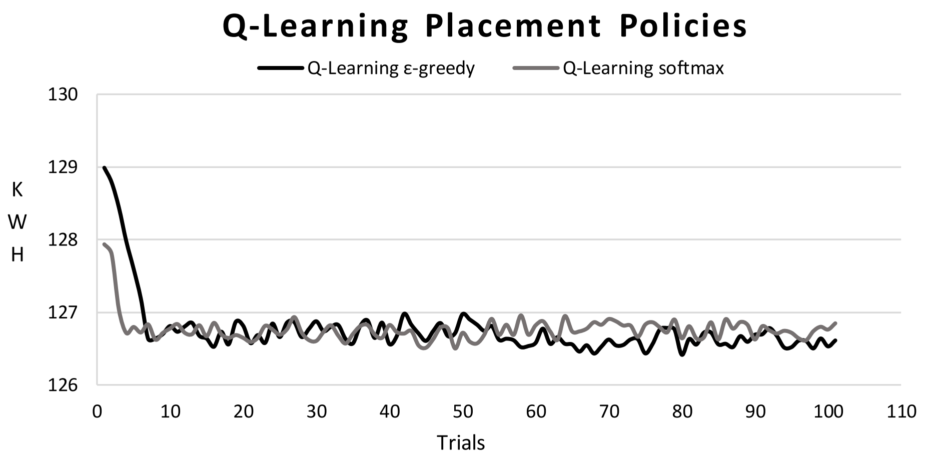

Fig. 8. Energy consumption Q-Learning with ϵ-greedy and softmax.

measured response time. In relation to the action selection policy being pursued the softmax action selection strategy performed better than ϵ-greedy in a more stochastic environment. This is largely due to the weighted approach used in the softmax policy, this prevents an agent from choosing equally among all actions which is a fundamental behavioral characteristic of ϵ-greedy. By implementing a softmax action selection strategy when entering a phase of exploration it is more probable that the agent will select a host to allocate a VM which will result in a more lucrative reward. From these results the learning policy that marginally outperforms all other variations under each metric over the 30 day stochastic workload is SARSA softmax which will be utilized in further experiments.

We further investigate the convergence rate of the RL approaches considered in this work. In order to demonstrate the application of RL in provisioning and managing cloud resources Fig. 7 portrays the energy consumption of the data center over 100 learning trials utilizing Q-Learning combined with an ϵ-greedy action selection policy. The 126 kWh energy range has been deemed the optimum energy rate across all four combinations of policies and will be used to measure the rate at which each policy converges. As illustrated, in the early stages of the learning process energy consumption was measured close to 129 kWh. Between trials 0–7 the line shows a downward trend as the energy rate proceeded to decrease demonstrating the agents ability to learn in more dynamic environments. In trial No.7 the energy consumption stabilized to a certain extent at 126.63 kWh indicating that the agent converged to an optimal policy. However, it also began to fluctuate marginally due to the set rate of ϵ which caused the agent to select a random action periodically. Overall the energy rate remained within the 126 kWh energy span.

The Q-Learning algorithm was compared utilizing both ϵ-greedy and softmax action selection strategies as depicted in Fig. 8. Q-Learning ϵ-greedy recorded close to 129 kWh of energy in the early stages of learning while Q-Learning softmax resulted in sub 128 kWh. It can be observed that Q-Learning softmax appears to have converged at a faster rate on trial No.4 in comparison to Q-Learning ϵ-greedy which converged on trial No.7. Additionally, post the 60th iteration mark Q-Learning softmax generated a slightly higher rate of energy averaging at 126.77 kWh in comparison to Q-Learning ϵ-greedy which averaged at 126.60 kWh.

Fig. 9 illustrates a comparison of the SARSA learning algorithm coupled with ϵ-greedy and softmax action selection strategies. As displayed, SARSA ϵ-greedy commenced learning at an energy rate of 128.56 kWh. On iteration No.6 it achieved sub 127 kWh and converged to an optimum policy. In contrast, SARSA softmax initially recorded energy consumption of 127.33 kWh and converged much quicker on iteration No.2 than SARSA ϵ-greedy. Both Q-Learning and SARSA algorithms paired with a softmax selection strategy resulted in a faster rate of convergence. Below Fig. 10 compares both policies relative to the rate at which they converge.

As previously highlighted, Q-Learning softmax recorded an initial energy consumption rate of 127.93 kWh prior to converging

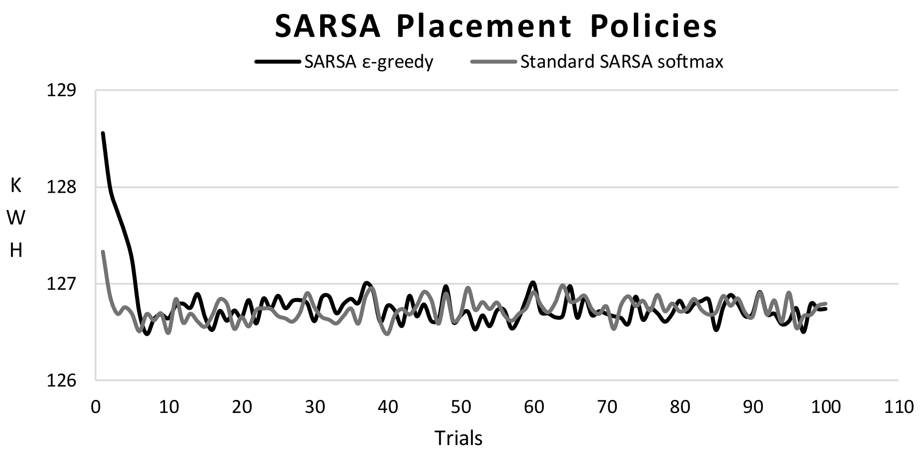

Fig. 9. Energy consumption SARSA with ϵ-greedy and softmax.

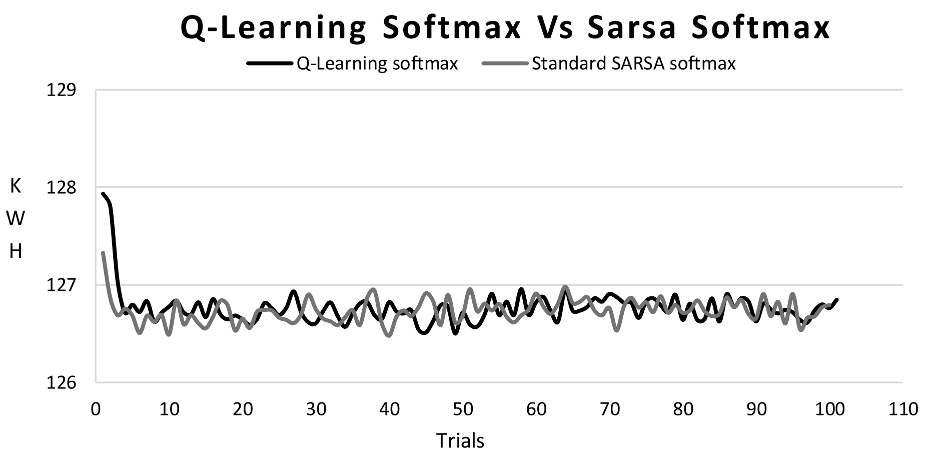

Fig. 10. Energy consumption Q-Learning softmax and Sarsa softmax.

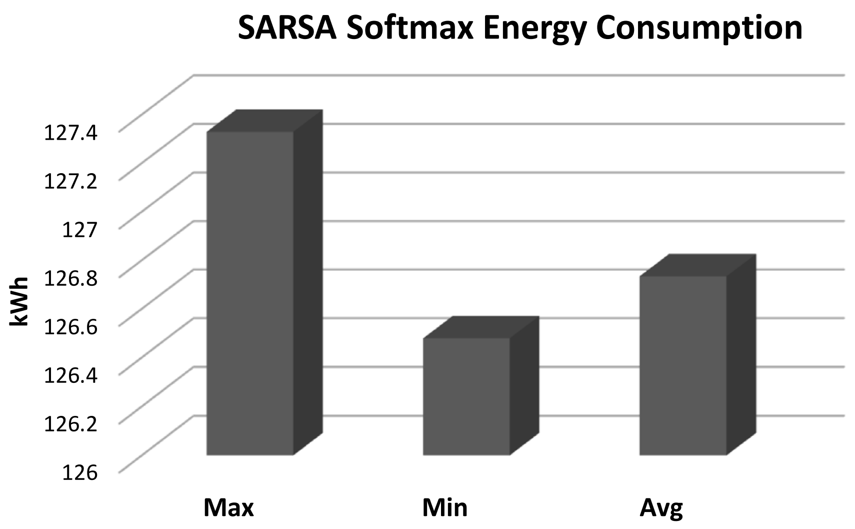

Fig. 11. Maximum, minimum and average Energy consumption over 100 iterations SARSA softmax.

on trial No.4 as indicated in Fig. 10. SARSA softmax produced a slightly lower energy rate of 127.33 kWh in the early stages of the learning process before experiencing a descent which caused it to converge on trial No.2. In terms of average energy consumption rates SARSA softmax experienced a minimal reduction in energy with a rate of 126.73 kWh and a standard deviation of 0.126115. Q-learning softmax averaged at 126.77 kWh with a higher deviation of 0.188252. By calculating the standard deviation of both policies we can infer that Q-Learning softmax resulted in a greater dispersion and variation in energy while SARSA softmax produced a more refined and overall reduced rate of energy.

In addition to the convergence of the algorithms we also present the maximum, minimum and average results of both

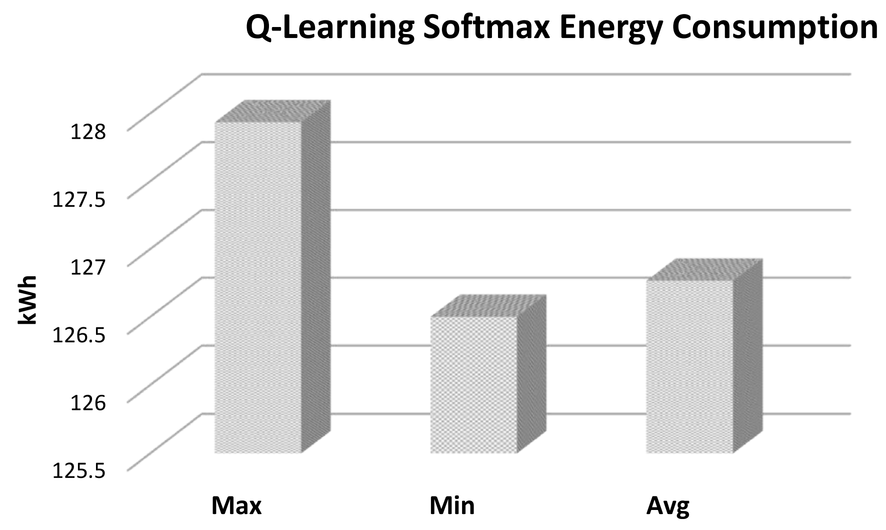

Fig. 12. Maximum, minimum and average Energy consumption over 100 iterations Q-Learning softmax.

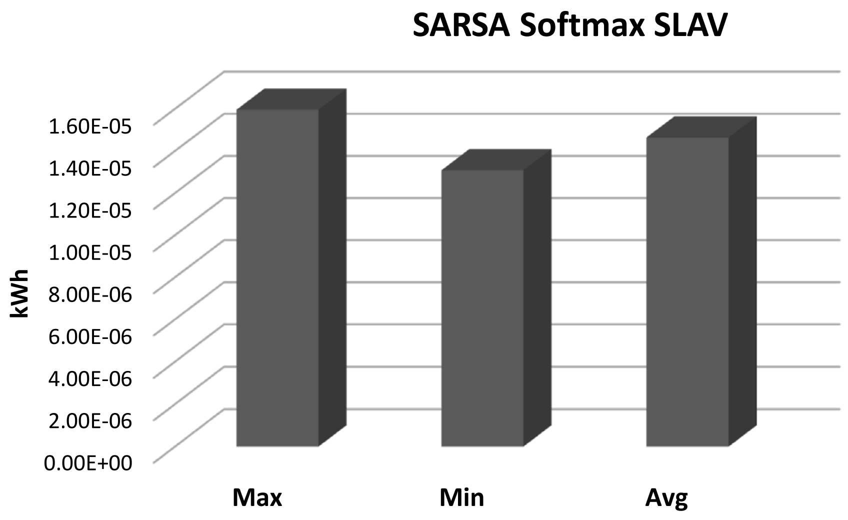

Fig. 13. Maximum, minimum and average SLAV over 100 iterations SARSA softmax.

the Q-Learning and SARSA softmax variations which in previous results generated improved performance. To supplement the simulations of the RL algorithms we demonstrate their performance over 100 iterations in terms of their ability to reduce energy and deliver improved performance in the data center which are the two key performance metrics considered in our simulations. Figs. 11 and 12 present the min, max and avg energy consumption for both the Q-Learning and SARSA RL algorithms. As previously stated both algorithms generate similar performance with only a slight difference evident between both. The Q-Learning softmax approach resulted in an average energy consumption of 126.77 kWh, a minimum of 126.50 kWh and a maximum rate of 127.93. The SARSA softmax variation generated an average energy rate of 126.73 kWh, a minimum rate of 126.48 kWh and a maximum energy rating of 127.33 kWh.

The minimum, maximum and average performance in terms of the number of service violations experienced in the data center is also presented in Figs. 13 and 14 for both the SARSA and Q-Learning softmax variation. Once again a similar result is achieved. Over 100 iterations SARSA softmax generated a slight decrease in the average number of service violations in comparison to the Q-Learning variation and this finding holds true when considering both the max and min number of service violations.

5.2. Performance evaluation of PBRS

The following experimental evaluation seeks to reinforce and provide a more robust algorithm by exploring the effects of introducing PBRS. SARSA softmax has been evaluated as the slightly more promising learning algorithm in the previous experiment. As a result, this experiment will measure the performance of PBRS Sarsa softmax against the standard SARSA softmax approach. As outlined earlier, PBRS requires the mapping of states to potentials in order to offer the agent an additional reward which conveys the designers range of preferred states [54], this potential function is defined as:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s) \tag{12}$$

The incorporation of PBRS requires a modification to the underlying MDP reward function. For the purposes of this work we modify the SARSA update rule as outlined below.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + F(s, s') + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big] \tag{13}$$

In order to implement PBRS the calculation used to compute the potentials and to ultimately shape the reward must be defined

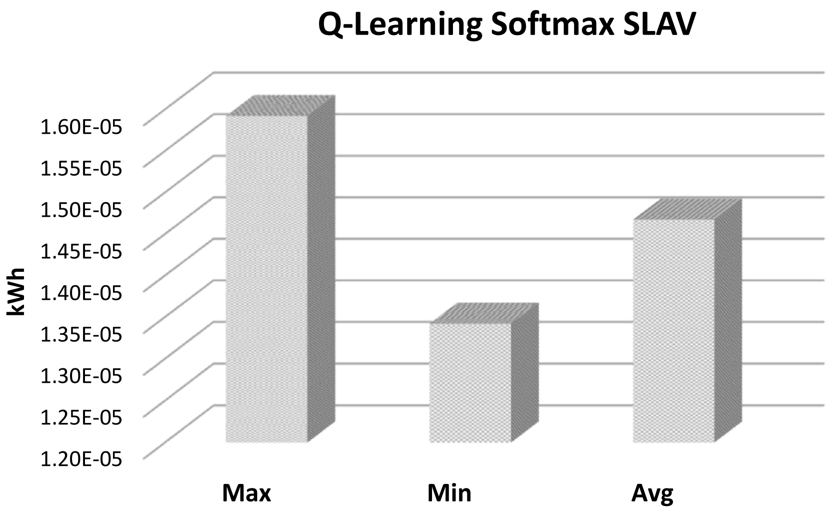

Fig. 14. Maximum, minimum and average SLAV over 100 iterations Q-learning softmax.

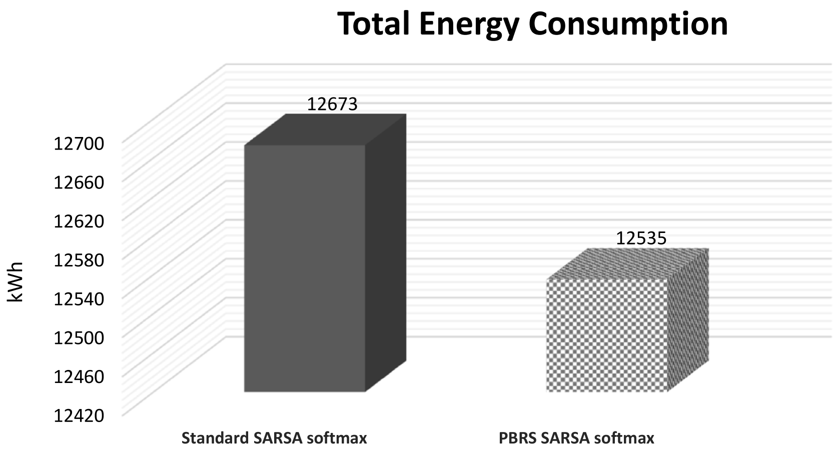

Fig. 15. Total energy consumption over 100 iterations for PBRS SARSA and standard SARSA.

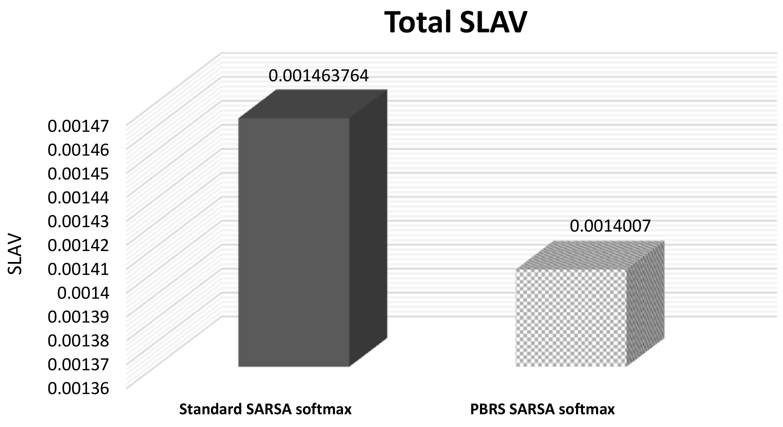

Fig. 16. PBRS SARSA vs. standard SARSA total SLAV.

relative to the environment in which the agent is being deployed. In light of this, the values in the Q-value matrix from the previous results chapter were leveraged in order to help shape the reward by providing a greater perspective on the most rewarding VM allocation strategy which resulted in host utilization states between 40%–70%. This allows for communicating the most optimal strategy to the agent in the early stages of learning in order to expedite the learning process. The algorithm will be run over a single iterative workload over 100 iterations.

Fig. 15 illustrates the total amount of energy consumed over 100 iterative trials. The standard approach incurred a total of 12,673 kWh of energy while the incorporation of PBRS resulted in 12,535 kWh, a 138 kWh (1.09%) reduction in energy over a single iterative workload. PBRS better guides the agent during learning and as a result speeding up the learning process by encouraging

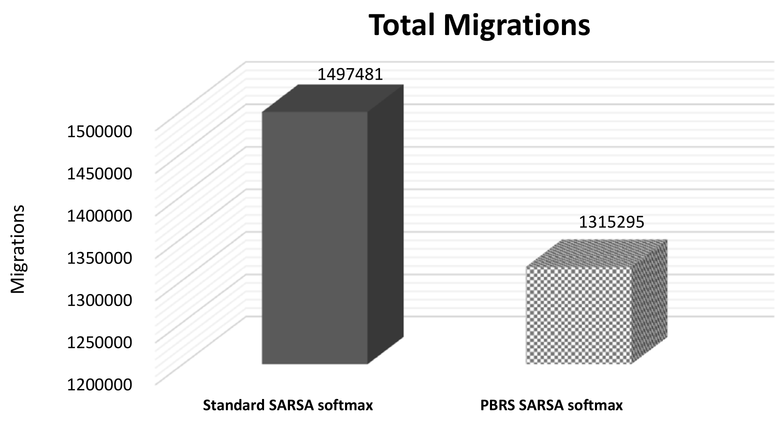

Fig. 17. PBRS SARSA vs. standard SARSA total migrations over 100 iterations.

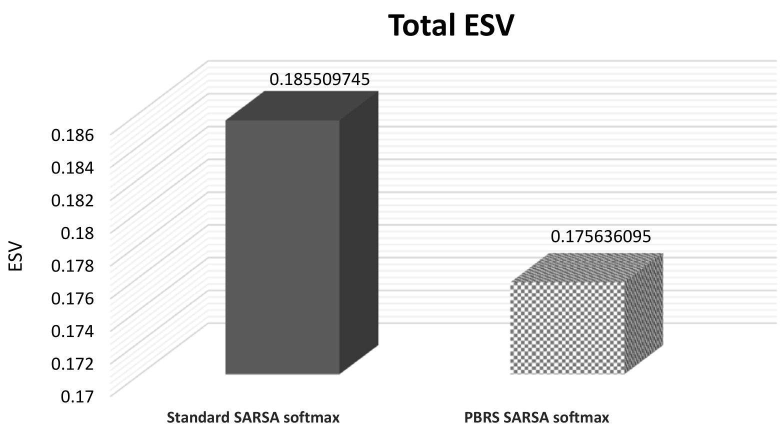

Fig. 18. PBRS SARSA vs. standard SARSA ESV.

the agent to be select more lucrative actions early on. The ability to learn faster allows the PBRS SARSA Softmax agent to make better consolidation decisions in the earlier stages of learning and results in a small improvement in energy overall.

The application of PBRS has also had a positive impact on the delivery of service agreements as depicted below in Fig. 16. As illustrated, the standard approach incurred the greatest amount of service violations while PBRS SARSA resulted in a 4.3% reduction in SLAV which helps in the delivery of a more robust and reliable cloud service better capable of meeting the needs of its users.

The effects of introducing PBRS have also had a positive impact on the number of migrations. As shown in Fig. 17 the standard SARSA algorithm resulted in an average of close to 14,975 migrations. Conversely, the PBRS SARSA algorithm migrated on average 13,153 VMs which resulted in an average reduction of up to 1,822 migrations per trial and a total reduction of 182,186 migrations (12.2%) over 100 iterations.

As illustrated below in Fig. 18 the PBRS SARSA algorithm outperformed the standard Sarsa version. It resulted in an overall 5.3% decrease in the ESV rating. Once more, this demonstrates the improved efficiency achievable by levering this technique in conjunction with the SARSA algorithm. The reduction on the ESV metric suggests that by harnessing PBRS it allows for an improvement in the management of the energy-performance tradeoff associated with energy aware resource allocation in cloud infrastructures. This is achieved by further guiding the agent early in the learning process.

5.3. Performance evaluation against benchmark

Next we evaluate the application of the proposed PBRS SARSA softmax algorithm known as ARLCA against the PowerAware consolidation algorithm which is often used as a benchmark when energy efficiency is the core objective [16]. The objective of ARLCA is to strategically allocate VMs on to a reduced number of hosts while idle servers can be effectively powered down. This in effect should reduce idle power costs and overall promote improved energy conservation in data center deployments while also improving the number of SLAVs. Both algorithms are evaluated over the 30 day workload in order to accurately analyze the performance and measure the robustness of both approaches under realistic workload variability.

Fig. 19 illustrates the behavior of both algorithms in relation to daily energy consumption over the 30 day stochastic workload. As evident in Fig. 19, there is variation in the energy consumption over the 30 day workload. In this work a stochastic workload is implemented based on the widely employed PlanetLab workloads which provide us with 10 days of CPU resource usage. To measure the robustness of the proposed learning algorithm we used these traces to generate a randomized 30 day workload. In doing so, we generate workloads which have added complexity and are overall more volatile and as a result, more accurately represent the dynamic nature of the cloud environment. The demand for CPU resources is driven by the workload data which dynamically changes overtime in the environment. In this research, we use the CloudSim simulator where energy is defined as the total energy consumed in the data center by computational resources as a result of processing application workloads represented by the CPU utilization on the host and reported as energy consumption over time. By implementing a more volatile workload it evaluates the capacity of our learning agent to manage a highly dynamic cloud environment. The implementation of ARLCA resulted in reduced energy consumption overall by a total of 25.35% (1,191

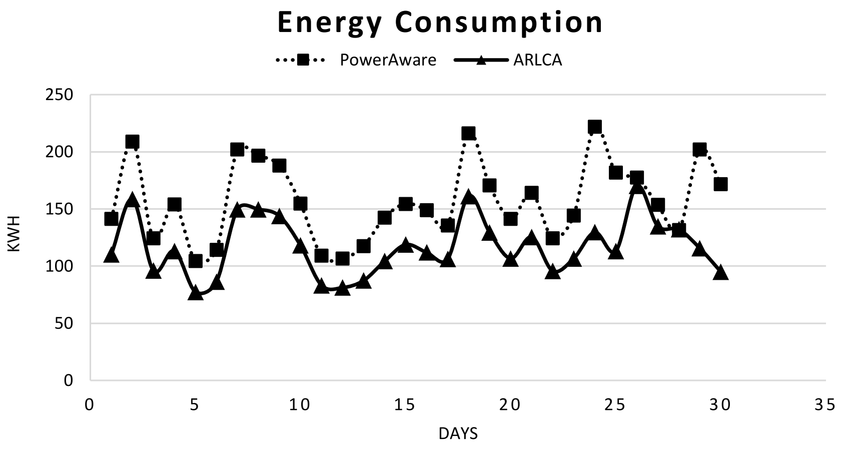

Fig. 19. Energy consumption PowerAware vs. ARLCA over 30 day workload.

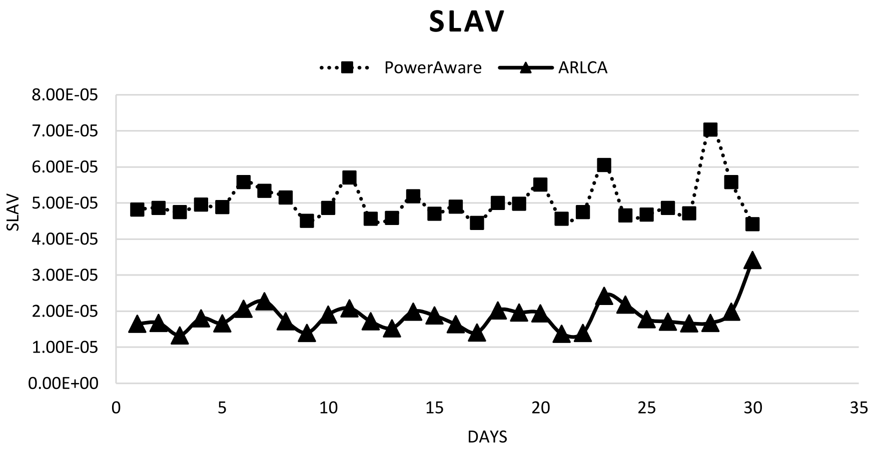

Fig. 20. SLAV PowerAware vs. ARLCA over 30 day workload.

kWh). Furthermore, our results indicated that on day 28 ARLCA failed to outperform the PowerAware algorithm resulting in a slight increase in energy of 0.5%.

An important dimension in achieving greater energy efficiency through resource optimization is managing the tradeoff between energy and performance which are inextricably linked. Notably, an interesting observation in Fig. 19 was that on day 28 the PowerAware algorithm generated a slight improvement in energy consumption of a mere 0.5%. However, Fig. 20 confirms that this marginal improvement was achieved at the cost of increased SLAV as indicated by the spike generated in the number of violations by the PowerAware algorithm on day 28. This suggests that PowerAware consolidated VMs more aggressively in an attempt to successfully reduce energy but failed to efficiently manage the energy performance tradeoff resulting in a surge in the number of service violations. In contrast, ARLCA strikes a more precise balance with such a tradeoff by generating a relatively similar energy rating of just .5% in the difference. However, more profoundly the agent also achieves a significant 76.2% decrease in the number of SLA violations on day 28 alone. Furthermore, the implementation of ARLCA resulted in an overall 63% decrease in the mean number of SLAV per day.

Displayed in Fig. 21 is the number of migrations incurred by both algorithms over the 30 day stochastic workload. As illustrated, the application of ARLCA also had a positive impact on the number of migrations. The RL agent reduced migrations by a substantial 49.17% in total (394,587 migrations). In addition it reduced the mean number of migrations per day by 13,153.

The results for the ESV metric are illustrated in Fig. 22. These results support the findings of both energy and SLAV shown above. The RL agent achieved a vast reduction of 72% in the total ESV rating again demonstrating its ability to effectively mange the tradeoff between energy and performance for a more robust and reliable solution to the consolidation problem.

Above we showed how the deployment of a learning agent to allocate VMs in a data center outperforms the prevalent PowerAware consolidation algorithm. However, arising from these findings an important and fundamental dimension of this research is to expose why the proposed RL learner proves to be a more efficient and robust solution. We provide further insights by analyzing each approach according to the number of active hosts, server shutdowns and also server utilization rates.

In order to gain a greater understanding and insight into the behavior of both approaches we firstly compare the number of active servers in the environment. As illustrated below in Fig. 23, the PowerAware algorithm generated a substantial increase in the number of required hosts to manage each workload over the 30 days. More specifically, PowerAware resulted in an overall total of 1290 hosts in comparison to the RL approach which required 887 an overall reduction of 31%. These considerable reductions attainable through the deployment ARLCA are further displayed as the agent achieves as much as a 50% decrease in the number of hosts on days 11 and 21, this difference resulted in the requirement of 32 additional hosts for PowerAware on day 21 alone. Although PowerAware seeks to drive efficiency it falls short as it allocates VMs onto hosts which cause the least increase in energy. However, in doing so this strategy exposes a major

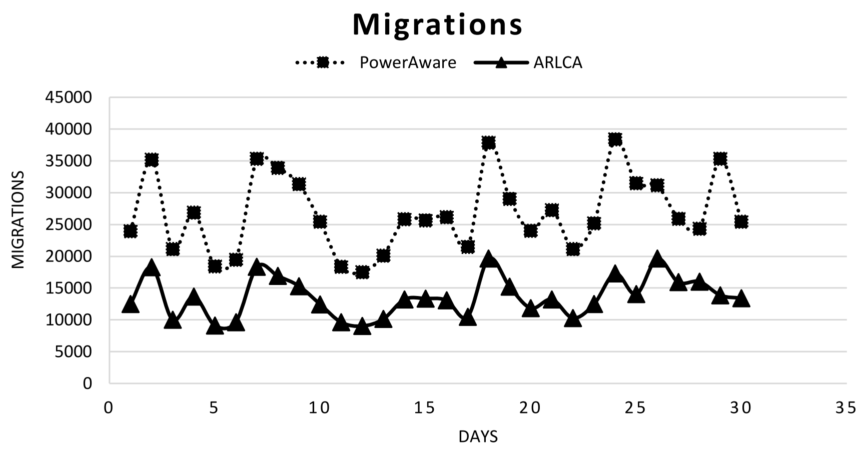

Fig. 21. Migrations PowerAware vs. ARLCA over 30 day workload.

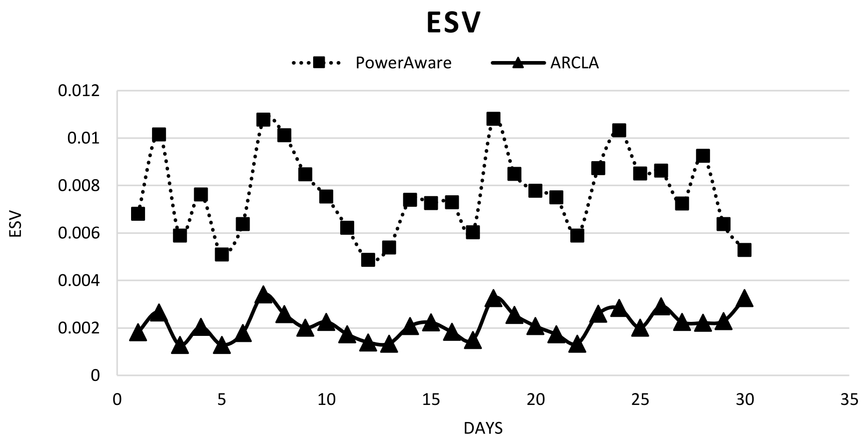

Fig. 22. ESV PowerAware vs. ARLCA over 30 day workload.

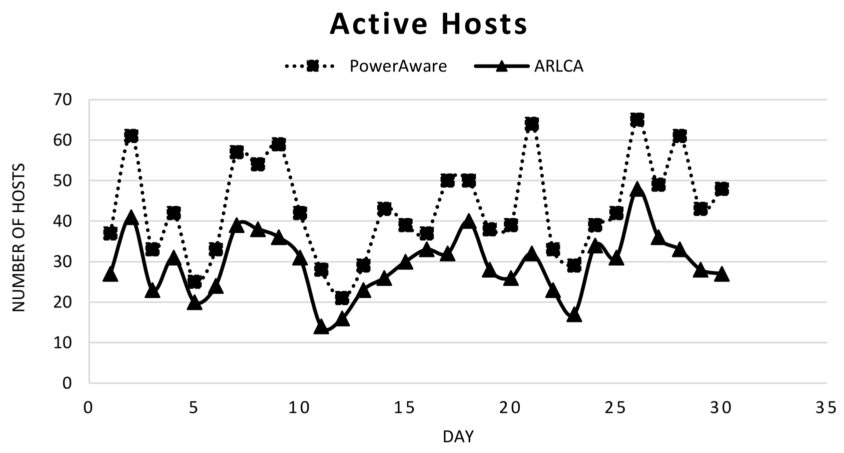

Fig. 23. Number of active hosts PowerAware vs. ARLCA over 30 day workload.

limitation pertaining to the number of active hosts booted up in the environment and results in the dispersal of VMs across an increased number of hosts causing a surge in energy. These results indicate the improved efficiency achievable through the deployment of an intelligent agent capable of consolidating VMs on to a reduced number of hosts in order to drive energy conservation and achieve overall improved use of the data center capacity.

To further expose the variations evident in both approaches we analyze the number of host shutdowns which occurred over the 30 day workload. Depending on how much of the available resources on each host remain idle the server is shutdown and all VMs operating on that host will be migrated off in an attempt to drive efficiency. As shown below in Fig. 24, PowerAware resulted in the greatest amount of shutdowns with a total of 144,660 while the ARLCA incurred 34,911 shutdowns. In addition, the average number of shutdowns for the PowerAware algorithm was 4822 in comparison to ARLCA which achieved a reduced average of 1163 shutdowns per day. Furthermore, as evident in our results we see far greater variation and volatility in the number of

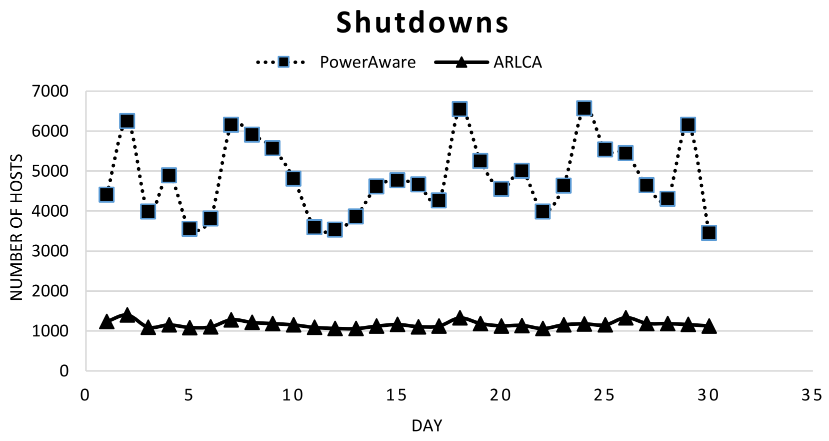

Fig. 24. Host shutdowns PowerAware vs. ARLCA over 30 day workload.

shutdowns generated by the PowerAware algorithm while the proposed ARLCA algorithm produces a more consistent result once again alluding to the added benefits gained through the deployments of the proposed approach. The objective of ARLCA is to strategically allocate VMs on to a reduced number of hosts while idle servers can be effectively powered down. Due to the innate learning capabilities of our RL agent it has the capacity to adapt and make more intelligent consolidation decisions under uncertainty early on in the simulation, as a result striking a better balance between both energy and performance. The agent generates a more optimized allocation of VMs across the data center such that hosts are not becoming overloaded too quickly but also the hosts in the environment are less likely to become underutilized which would result in the host being shutdown and the VMs reallocated elsewhere in the data center. Conversely, as shown in Fig. 23, the PowerAware algorithm distributes the workload across a greater number of hosts in the datacenter. As a result, the host's resource usage is not optimized and often results in the hosts entering an underutilized state and thus generating an increase in the number of hosts shutdowns in the data center. These results overall indicate that the PowerAware algorithm deploys a greater number of hosts which are operating at largely underutilized levels. This in turn causes hosts to be powered off and VMs to be migrated to new hosts in an effort to reduce idle resources.

Having highlighted the overall variations evident in both approaches in terms of the number of active hosts and shutdowns we now focus on portraying the impact of both algorithms on the internal management of server resources. The results in the previous sections demonstrated that ARLCA managed application workloads on a reduced number of hosts which has a significant impact on the number of shutdowns, energy consumption, migrations and SLAVs as shown in the initial results. In order to further highlight the superior efficiency of ARLCA we compare both approaches with regards to host utilization levels for a single day. For the purposes of this experiment day 21 was selected for analysis as it showed the greatest contrast in the number of active hosts being managed by both algorithms. Fig. 25 portrays the resource usage of active hosts at the end of the workload simulation on day 21. This provides a snapshot of the internal management of host resource usage. As illustrated, the PowerAware algorithm resulted in a total of 64 hosts in operation at the end of the workload with an average resource usage of a mere 40%. As illustrated, for 33 of these hosts their resource usage levels fell between the 0%–40% utilization range. More profoundly, of the 33 hosts which operated at this capacity 32 of them were found to be running at an extremely suboptimal 0%–10% utilization rate. The remaining hosts operated largely between the 62%–82% and 83%+ categories with 18 in total operating in the latter usage range. Conversely, ARLCA resulted in the operation of a significant 50% less hosts with an average host utilization of 74.8%. A total of 18 hosts were found to be operating between 62%–82% utilization of which 72% of those hosts were running at 70%–79% capacity. Furthermore, only 1 host fell under the 0%–40% capacity and more accurately it was found to operate at 28% which is significantly higher than the 32 hosts which operated at 0%–10% as evident in the PowerAware consolidation algorithm. The PowerAware consolidation algorithm deploys an increased number of hosts which are operating at largely underutilized levels in order to manage the workload. This in turn causes hosts to be powered off and VMs migrated to new hosts in an effort to reduce idle resources. However as previously mentioned, the PowerAware algorithm once again falls short as it allocates VMs onto hosts resulting in the least increase in power. Often this logic results in the placement of VMs onto hosts that are severely underutilized but with no guarantee that such a strategy will drastically improve resource utilization. As a result it produces cohorts of servers that remain severely underutilized generating a surge in the number of server shutdowns and subsequently the number of migrations as shown above in Figs. 21 and 24. Furthermore, as illustrated in Fig. 24, there are stark variations and overall fluctuations evident in the number of shutdowns incurred over the 30 day cycle as PowerAware endeavors to drive efficiency. In contrast, the ARLCA learning algorithm produces more consistency in the number of shutdowns, improves energy efficiency, reduce migrations and generates more optimal usage of the available resources. This is achieved as the intelligent capabilities underlying ARLCA proves to be more robust in a dynamic cloud environment. The learning agent strikes an improved balance between energy and performance for a more optimum resource management strategy. These results overall demonstrate that ARLCA has the capacity to optimize the distribution of VMs across a reduced number of hosts resulting in a larger portion of host utilization concentrated between the 62%–82% utilization.

Overall, as clearly evident in the PowerAware consolidation algorithm, workloads distributed over an increased number of hosts leads to underutilized resources and as a result drive up idle power costs, host shutdowns and migrations. More significantly, this strategy results in reduced efficiency and also an increase in the number of service violations as workloads are invariably migrated and reallocated in an attempt to gain the efficiency lost through a less plausible consolidation strategy. The key points arising out of these results are that through the deployment of a more dynamic and flexible approach which is better suited to the volatility evident in a cloud environment we achieve a more sustainable energy efficient resource management model. Above

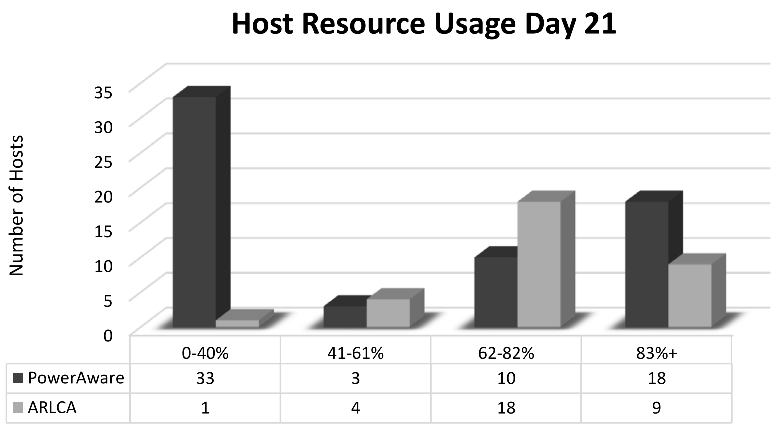

Fig. 25. Active hosts resource utilization PowerAware vs. ARLCA.

Table 2
Summary of the average performance.

Performance gains over 30 day workload (Average)		
Metric	PABFD	ARLCA
Energy consumption	156.7698735	121.6661199
SLAV	9.770940091	9.75834798
Migrations	26,750	15,769
ESV	0.007605409	0.002308703
Active hosts	43	30

all the results demonstrate the immense potential of ARLCA, in particular they display the ability of the RL agent to strike a balance between energy and performance which is a crucial dimension in the design of any given energy efficient resource management model. In Table 2 we provide a conclusive summary of the average performance of the approaches considered in our evaluation. In particular, we provide a summary of the average performance generated by each algorithm using the selected performance metrics including energy consumption, SLAV, the number of migrations, ESV and also the average number of active hosts in the environment.

We argue that the cost associated with implementing an RL driven VM consolidation approach such as ARLCA is less costly in comparison to some of the other types of approaches proposed in the literature. In more recent years there has been a significant surge in the number of researchers exploring the applications of Machine Learning and predictive based methodologies to improve resource management mechanisms in the cloud. Studies which focus on delivering consolidation decisions based on predictive approaches such as time series ARIMA modeling [21,24] can be a more costly approach overall. This is primarily due to the nature of these algorithms where they are applied to resource usage data in an offline manner. From this perspective, overtime as the cloud continues to grow these approaches would require modifications to refit the predictive models based on more recent data. Conversely, the major advantage of the proposed RL approach lies in its ability to reason under uncertainty in an online manner. However, one of the limitations or associated costs of an RL based consolidation approach is that the initial decisions taken by the agent can result poor outcomes. To optimize resource management the agent requires interactions with the environment in order to learn a good policy. In the context of the cloud this can result in the agent making suboptimal consolidation decisions in early stages of learning. However, taking this cost into consideration as demonstrated in our results the agent overall generates effective energy gains and drives performance throughout the data center.

The focus of this work is primarily devoted to the exploration of RL in the development of a novel consolidation algorithm to improve the energy performance tradeoff. However, another important and interesting research direction is identifying the optimal point to both start and end the VM consolidation process which is currently beyond the scope of this research but warrants discussion. The entire VM consolidation process typically consists of several different tasks including host overload detection, VM selection and VM consolidation. The vast majority of VM consolidation approaches often apply static resource utilization thresholds to trigger the VM consolidation process [31]. When resource usage goes beyond the threshold value it causes VMs to be migrated to various hosts in the data center in an attempt to restore resource usage on the original host to what is deemed to be an acceptable level. The consolidation process usually ends when enough VMs have been migrated from the host and resource usage becomes stable again. However, these types of approaches pose several limitations. In particular, they are largely reactive and as a result they do not work well given the inherently dynamic nature of cloud environments and can often result in service violations.

To foster a more proactive consolidation process it is beneficial to identify an optimal point at which to start and end VM consolidation. Using predictive modeling is becoming a necessary component across all aspects of VM consolidation including host overload detection [10]. By employing predictive solutions for host overload detection the optimal point at which to start consolidation can be better identified. If algorithms can predict server loads the process of migrating VMs and consolidating them on other hosts across the data center can occur prior to a spike occurring in the resource usage on the host. The predictions generated could also be potentially used to identify the most optimal point to end VM consolidation. By predicting resource usage on the host over future time steps it would be possible to identify the point at which usage is predicted to stabilize and VM consolidation can end. This research is largely one of the core points of focus within the area of host overload detection. In this work, given the primary focus is devoted to the consolidation problem we employ the popular LR-MMT algorithm [31] to initiate the consolidation process. LR-MMT can be used to infer the probability of a host becoming loaded and to select which VMs to migrate. The LR-MMT algorithm continues to migrate VMs from the host as long as the host usage is predicted to be overloaded based on the latest usage values. This algorithm is widely used to manage host overload detection and VM selection when the consolidation problem is the core objective of the proposed work.

6. Conclusion

Energy related costs and environmental sustainability of data centers have become challenging issues for cloud computing practitioners who seek new innovative approaches to significantly reduce the colossal energy costs of data center operations. Through the innovative application of more sophisticated and advanced RL methodologies adopted from the field of Machine Learning we propose ARLCA, an intelligent solution to optimize the distribution of VMs across the data center. ARLCA demonstrates the potential of more advanced intelligent solutions capable of reaching new frontiers in data center energy efficiency while also achieving significant improvements in the quality of the service provided. Unlike existing approaches in the literature, we provide a comparative analysis of popular RL learning algorithms, we compare alternative exploration mechanisms while also harnessing PBRS. We demonstrate the application of these approaches in managing the energy performance tradeoff to resolve the VM consolidation problem. Furthermore, we compare our approach to the well known PowerAware consolidation algorithm which is used as a benchmark. Our results showed the ability of ARLCA to reduce energy consumption by 25% while also reducing the number of SLAV by a significant 63% in comparison to the popular PowerAware consolidation approach. We also providing further insights into the behavior of the RL agent and how it outperforms the PowerAware algorithm. This solution could potentially be integrated into the VM consolidation strategy of industrial data centers or open-source cloud infrastructure management platforms in a bid to reach new frontiers in data center energy efficiency while also improving the delivery of service guarantees for multi-tenant cloud environments. Lastly, this research also has wider implications as it stands to provide a more sustainable green cloud infrastructure in support of global environmental sustainability.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is funded by the Irish Research Council through the Government of Ireland Postgraduate Scholarship Scheme grant number GOIPG/2016/631.

References

[1] Thiago Lara Vasques, Pedro Moura, Aníbal de Almeida, A review on energy efficiency and demand response with focus on small and medium data centers, Energy Effic. (2019) 1–30.
[2] M.A. Khan, A.P. Paplinski, A.M. Khan, M. Murshed, R. Buyya, Exploiting user provided information in dynamic consolidation of virtual machines to minimize energy consumption of cloud data centers, in: Fog and Mobile Edge Computing (FMEC), 2018 Third International Conference on, IEEE, 2018, pp. 105–114.
[3] Maria Avgerinou, Paolo Bertoldi, Luca Castellazzi, Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency, Energies 10 (10) (2017) 1470.
[4] J. Stryjak, M. Sivakumaran, The mobile economy 2019, GSMA Intell. (2019).
[5] Anders S.G. Andrae, Tomas Edler, On global electricity usage of communication technology: trends to 2030, Challenges 6 (1) (2015) 117–157.
[6] Y.C. Lee, A.Y. Zomaya, Energy efficient utilisation of resources in cloud computing systems, J. Supercomput. 60 (2012) 268–280.
[7] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, Feng Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, 2008, pp. 337–350.
[8] Ali Asghar Rahmanian, Mostafa Ghobaei-Arani, Sajjad Tofighy, A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment, Future Gener. Comput. Syst. 79 (2018) 55–71.
[9] R. Shaw, E. Howley, E. Barrett, An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers, in: Proc. 12th International Conference for Internet Technology and Secured Transactions, IEEE, 2017, pp. 61–66.
[10] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N.T. Hieu, H. Tenhunen, Energy-aware vm consolidation in cloud data centers using utilisation prediction model, IEEE Trans. Cloud Comput. (2016).
[11] L.C. Jersak, T. Ferreto, Performance-aware server consolidation with adjustable interference levels, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 420–425.
[12] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Shiraz, Abdullah Yousafzai, Feng Xia, A survey on virtual machine migration and server consolidation frameworks for cloud data centers, J. Netw. Comput. Appl. 52 (2015) 11–25.
[13] Shreenath Acharya, Demian Antony D'Mello, A taxonomy of Live Virtual Machine (VM) Migration mechanisms in cloud computing environment, in: 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), IEEE, 2013, pp. 809–815.
[14] Manoel C. Silva Filho, Claudio C. Monteiro, Pedro R.M. Inácio, Mário M. Freire, Approaches for optimizing virtual machine placement and migration in cloud environments: A survey, J. Parallel Distrib. Comput. 111 (2018) 220–250.
[15] Zaigham Mahmood, Cloud Computing: Challenges, Limitations and R & D Solutions, Springer, 2014.
[16] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (2012) 755–768.
[17] Anton Beloglazov, Rajkumar Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, Concurr. Comput.: Pract. Exper. 24 (13) (2012) 1397–1420.
[18] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, 1998.
[19] Tao Chen, Yaoming Zhu, Xiaofeng Gao, Linghe Kong, Guihai Chen, Yongjian Wang, Improving resource utilization via virtual machine placement in data center networks, Mob. Netw. Appl. 23 (2) (2018) 227–238.
[20] Rachael Shaw, Enda Howley, Enda Barrett, An energy efficient and interference aware virtual machine consolidation algorithm using workload classification, in: International Conference on Service-Oriented Computing, 2019, pp. 251–266.
[21] Xiong Fu, Chen Zhou, Predicted affinity based virtual machine placement in cloud computing environments, IEEE Trans. Cloud Comput. (2017).
[22] M. Duggan, R. Shaw, J. Duggan, E. Howley, E. Barrett, A multitime steps ahead prediction approach for scheduling live migration in cloud data centers, Softw. - Pract. Exp. 49 (4) (2019) 617–639.
[23] G.T. Hicham, E. Chaker, E. Lotfi, Comparative study of neural networks algorithms for cloud computing CPU scheduling, Int. J. Electr. Comput. Eng. (IJECE) 7 (1) (2017) 3570–3577.
[24] D. Janardhanan, E. Barrett, CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models, in: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017, pp. 55–60.
[25] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, Economic LSTM approach for recurrent neural networks, IEEE Trans. Circuits Syst. II: Express Briefs 66 (11) (2019) 1885–1889, http://dx.doi.org/10.1109/TCSII.2019.2924663.
[26] Hamid Arabnejad, Claus Pahl, Pooyan Jamshidi, Giovani Estrada, A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling, in: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, 2017, pp. 64–73.
[27] P. Saripalli, G.V.R. Kiran, R.R. Shankar, H. Narware, N. Bindal, Load prediction and hot spot detection models for autonomic cloud computing, in: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, IEEE, 2011, pp. 397–402.
[28] Marek Grześ, Daniel Kudenko, Multigrid reinforcement learning with reward shaping, in: International Conference on Artificial Neural Networks, 2008, pp. 357–366.
[29] A. Verma, P. Ahuja, A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, in: Proc. the 9th ACM/IFIP/USENIX International Conference on Middleware, 2008, pp. 243–264.
[30] M. Cardosa, M.R. Korupolu, A. Singh, Shares and utilities based power consolidation in virtualized host environments, in: Proc. the 11th IFIP/IEEE International Conference on Symposium on Integrated Network Management, IEEE Press, 2008, pp. 327–334.
[31] A. Beloglazov, R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, Concurr. Comput.: Pract. Exper. 24 (2012) 1397–1420.

[32] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.

[33] T.H. Nguyen, M. Di Francesco, A. Yla-Jaaski, Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers, IEEE Trans. Serv. Comput. (2017).

[34] R. Shaw, E. Howley, E. Barrett, An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions, Simul. Model. Pract. Theory (2019) http://dx.doi.org/10.1016/j.simpat.2018.09.019.

[35] W.B. Slama, Z. Brahmi, Interference-aware virtual machine placement in cloud computing system approach based on fuzzy formal concepts analysis, in: 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, 2018, pp. 48–53.

[36] Martin Duggan, Jim Duggan, Enda Howley, Enda Barrett, A network aware approach for the scheduling of virtual machine migration during peak loads, Cluster Comput. 20 (3) (2017) 2083–2094.

[37] Gerald Tesauro, Rajarshi Das, Hoi Chan, Jeffrey Kephart, David Levine, Freeman Rawson, Charles Lefurgy, Managing power consumption and performance of computing systems using reinforcement learning, in: Advances in Neural Information Processing Systems, 2008, pp. 1497–1504.

[38] E. Barrett, E. Howley, J. Duggan, A learning architecture for scheduling workflow applications in the cloud, in: Web Services (ECOWS), 2011 Ninth IEEE European Conference on, IEEE, 2011, pp. 83–90.

[39] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, George Yin, VCONF: a reinforcement learning approach to virtual machines auto-configuration, in: Proceedings of the 6th International Conference on Autonomic Computing, 2009, pp. 137–146.

[40] Rajarshi Das, Jeffrey O. Kephart, Charles Lefurgy, Gerald Tesauro, David W. Levine, Hoi Chan, VCONF: Autonomic multi-agent management of power and performance in data centers, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, 2008, pp. 107–114.

[41] Xavier Dutreilh, Sergey Kirgizov, Olga Melekhova, Jacques Malenfant, Nicolas Rivierre, Isis Truck, Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow, in: ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems, 2011, pp. 67–74.

[42] Fahimeh Farahnakian, Pasi Liljeberg, Juha Plosila, Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning, in: 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, IEEE, 2014, pp. 500–507.

[43] Rachael Shaw, Enda Howley, Enda Barrett, An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers, in: 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017, pp. 61–66.

[44] Richard S. Sutton, Andrew G. Barto, et al., Introduction to Reinforcement Learning, MIT press, Cambridge, 1998.

[45] Marco Wiering, Martijn Van Otterlo, Reinforcement learning, in: Adaptation, Learning, and Optimization, Vol. 12, 2012, p. 3.

[46] Enda Barrett, Enda Howley, Jim Duggan, Applying reinforcement learning towards automating resource allocation and application scalability in the cloud, Concurr. Comput.: Pract. Exper. 25 (12) (2013) 1656–1674.

[47] Patrick Mannion, Jim Duggan, Enda Howley, Parallel learning using heterogeneous agents, in: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015), 2015.

[48] Gavin A. Rummery, Mahesan Niranjan, Online Q-Learning using Connectionist Systems, University of Cambridge, Department of Engineering, 1994.

[49] Patrick Mannion, Jim Duggan, Enda Howley, Learning traffic signal control with advice, in: Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015), 2015.

[50] Patrick Mannion, Jim Duggan, Enda Howley, An experimental review of reinforcement learning algorithms for adaptive traffic signal control, in: Autonomic Road Transport Support Systems, 2015.

[51] Michel Tokic, Adaptive ε-greedy exploration in reinforcement learning based on value differences, in: KI 2010: Advances in Artificial Intelligence, 2010, pp. 203–210.

[52] Marek Grześ, Daniel Kudenko, Multigrid reinforcement learning with reward shaping, in: Artificial Neural Networks-ICANN, 2008, pp. 357–366.

[53] Marek Grzes, Reward shaping in episodic reinforcement learning, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, ACM, 2017, pp. 565–573.

[54] Andrew Y. Ng, Daishi Harada, Stuart Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: ICML, 1999, pp. 278–287.

[55] Richard Bellman, Dynamic Programming, Princeton University Press, 1957.