Optimized Renewable Energy Use in Green Cloud Data Centers

Minxian Xu[1,2], Adel N. Toosi[2(✉)], Behrooz Bahrani[3], Reza Razzaghi[3], and Martin Singh[4]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[2] Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia
{minxian.xu,adel.n.toosi}@monash.edu
[3] Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC 3800, Australia
{behrooz.bahrani,reza.razzaghi}@monash.edu
[4] School of Earth, Atmosphere and Environment, Monash University, Clayton, VIC 3800, Australia
martin.singh@monash.edu

Abstract. The huge energy consumption of cloud data centers not only increases costs but also carbon emissions associated with such data centers. Powering data centers with renewable or green sources of energy can reduce brown energy use and consequently carbon emissions. However, powering data centers with these energy sources is challenging, as they are variable and not available at all times. In this work, we formulate the microservices management problem as finite Markov Decision Processes (MDP) to optimise renewable energy use. By dynamically switching off non-mandatory microservices and scheduling battery usage, upon the user's preference, our proposed method makes a trade-off between the workload execution and brown energy consumption. We evaluate our proposed method using traces derived from two real workloads and real-world solar data. Simulated experiments show that, compared with baseline algorithms, our proposed approach performs up to 30% more efficiently in balancing the brown energy usage and workload execution.

1 Introduction

The adoption of cloud computing has been rapid; it has been found that 70% of organizations have at least one application deployed in clouds [1]. These applications are hosted in cloud data centers which allow users to access them via the Internet. Due to the rapid growth of cloud data centers, it is anticipated that

data centers will consume significant worldwide generated electricity [1]. This huge energy consumption has an immense impact on the environment through greenhouse gas emissions. Thus, improving the energy efficiency of the cloud data center has attracted significant attention from researchers and the IT industry.

To ensure the sustainability of clouds, the carbon footprint of data centers must be reduced. Apart from reducing the total energy consumption, powering data centers with renewable (green) energy sources like wind or solar significantly reduces the carbon emissions associated with data centers. However, the limitation of renewable energy is that it is not as reliable as the grid power, and it is stochastic and intermittent in its behaviour.

Microservices are small, autonomous services that are rapidly becoming the norm for building large-scale applications in cloud data centers. With their isolation and light-weight features, microservices can be dynamically switched on/off to improve resource usage of application. This allows service providers to execute optional microservices (e.g., an analytics engine for E-commerce website) to better match the workload with the energy supply. To this end, one must develop algorithms to switch microservices on/off so that the overall executed number of microservices is maximized while minimizing the use of non-green power. In this work, we develop a method to manage resources at the microservice level to match the energy demand with the renewable energy supply of a data center.

Currently, most energy efficient scheduling algorithms for cloud data centers are heuristic-based, e.g., [2,6,7]. Heuristics are designed to provide acceptable results in a reasonable time frame. However, the entire solution space is not searched in heuristic approaches and their performance is not guaranteed. In practice, both future workloads and the availability of green energy are non-deterministic, and must be modelled probabilistically. Therefore, we consider using Markov Decision Processes (MDP) to model the stochastic nature of workloads and green energy availability in green cloud data centers. In this work, we assume that the data center is powered by an on-site renewable energy system (e.g., a photovoltaic solar power system) and the residual green energy can be stored in batteries for near future use. When battery storage is used, a challenging question to address is "when and in what capacity should batteries be discharged to maximize the overall executed number of microservices?"

We aim to find the optimised policy that maximizes the number of executed microservices while minimizing brown energy (energy produced from polluting sources) use. The policy contains actions that select how many microservices are to be switched off and whether and in what capacity the battery power is consumed in each time slot in accordance with the system administrator's preference for environmental friendliness. The key **contributions** are as follows:

- We model the green-aware microservices management problem as a *finite horizon Markov Decision Process* problem with the objective to minimize the usage of brown energy while maximizing the number of microservices deployed. In our model, we consider renewable energy (green), grid electricity (brown) and battery to power the system.

– We propose an algorithm based on MDP to dynamically switch off microservices and schedule battery usage to achieve the optimised results.
– We propose a tuning parameter which allows the system administrator to make a trade-off between the workload execution and brown energy use.
– We conduct simulation-based experiments using real data derived from workload traces and renewable energy availability. The results show that our proposed approach significantly reduces brown energy usage while deploying more microservices compared with baselines.

2   Related Work

Green cloud data centers powered by renewable energy is becoming an important topic in operating cloud data centers. To the best of our knowledge, our work is the first one to apply MDP to optimise the renewable energy use in cloud data centers. We now discuss the related work. Table 1 also shows the comparison of the related work based on key approaches, energy sources and objectives.

Markov Decision Process in Cloud Computing Environment. To model the probabilistic features of cloud computing environments and make resource management decisions, MDP has been applied in some research. Xu et al. [13] applied approximate MDP to schedule application components in cloud data centers to improve the trade-offs between the energy consumption and discount offered to users. Terefe et al. [11] adopted an MDP-based multi-site offloading algorithm for mobile cloud computing, which aims to achieve the energy-efficient objective of mobile devices. Han et al. [5] proposed a VM migration approach based on MDP to reduce data center energy consumption and resource shortage. Shen et al. [10] proposed an MDP-based approach to balance the VM loads on physical machines, which can achieve lower SLA violations and better load balancing effects than baselines.

Our work differs significantly from these MDP-based efforts in several perspectives: (1) none of them applied MDP to model the probabilistic feature of workloads and renewable energy together; (2) none of them put the efforts on optimising the use of renewable energy; and (3) none of them considered the actions on microservices and battery.

Renewable Energy Use in Data Centers. Renewable energy has been used to power data centers to reduce their carbon footprint. Zhang et al. [15] proposed a middleware system to dynamically dispatch requests to maximize the percentage of renewable energy used to power a network of distributed data centers while satisfying the desired cost/budget of the service provider. They applied a requests dispatching algorithm based on linear-fractional programming. In contrast, our approach is based MDP and considers the actions for battery to further maximize the renewable energy use.

Giori et al. [4] proposed a prototype green data center called *Parasol*, which is powered by solar panels, battery and grid power. They used linear programming to manage workloads and select sources of energy. In contrast, our approach does

Table 1. Comparison of related work

Approach	Approach			Energy sources			Objective		
	Linear programming	Prediction	MDP	Brown	Green	Battery	Energy-aware	Green-aware	QoS-aware
Xu et al. [13]			✓	✓			✓		✓
Terefe et al. [11]			✓				✓		✓
Han et al. [5]			✓	✓			✓		✓
Shen et al. [10]			✓						✓
Zhang e al. [15]	✓			✓	✓			✓	✓
Giori et al. [4]	✓	✓		✓	✓	✓		✓	✓
Liu et al. [8]		✓		✓	✓			✓	✓
Our approach			✓	✓	✓	✓		✓	✓

not need a prediction model to predict future renewable energy availability and workloads as *Parasol* does. We model workloads and renewable energy based on probabilistic model and introduce a tuning parameter (dimmer) to balance the trade-offs between the workload execution and brown energy use.

Liu et al. [8] used a holistic approach by considering renewable energy supply, electricity pricing and cooling costs to improve the sustainability of data centers. Our work differs from this one as our objective is maximising the use of renewable energy. Toosi et al. [12] proposed an approach to redirect virtual machine requests to other data centers with available renewable energy. They introduced two online deterministic algorithms for maximizing renewable energy usage. In contrast, we apply the MDP approach for a single data center and manage the actions for microservices and battery rather than virtual machines.

3   System Modeling and Problem Statement

Figure 1 shows the schematic view of the proposed system. We consider a data center that consists of multiple physical machines (servers). To power the data center, several energy sources are considered, including brown energy generated by coal-based facilities, green energy generated by solar panels and batteries that can store the surplus green energy. Applications constructed via self-contained microservices deployed on physical machines to provide services for the end users. Many applications including web applications often have microservices that fit
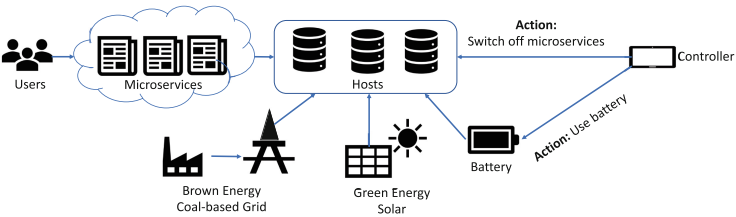

Fig. 1. Schematic view of the system.

into the **brownout** [14] feature, which can be regarded as non-mandatory components that can be dynamically activated/deactivated as need arises. For example, microservices handling the recommendation engines for a shopping website, or microservices running ad selection algorithms and optimization. Our approach only targets microservices having brownout feature and all other microservices remain untouched in the system.

3.1   States

We consider the discrete time **finite-state** MDP, and we aim to find an optimal policy (state to action mapping) to achieve best results for a single day. We discretize the time horizon into identically sized slots, i.e., each day is divided into 24 h time slots. A finer grain time slot can be used in our model. However, since weather data is often available in hourly basis[1] and service are billed per hour in well-known cloud providers such as AWS[2], here, we focus on hourly time slots throughout the day (solar cycle).

The system state $S(t)$ at time-slot t includes the status of (1) demanded microservices, (2) available renewable energy, and (3) level of battery state of charge (SoC). The state space of active microservices at time t is given as $W(t) \leq \bar{W} \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of non-negative integers and $\bar{W}$ is the maximum number of microservices the system can accommodate. The number of active microservices represents the intensity of workloads, that is, more active microservices are required when workload is high. Availability of renewable energy at time t is represented by a discrete random value $G(t) \leq \bar{G} \in \mathbb{Z}^+$. $G(t)$ represents the level of electricity generated by the renewable power system and $\bar{G}$ is the maximum level of renewable power can be generated in the system. Similarly, $B(t) \leq \bar{B} \in \mathbb{Z}^+$ is a discrete value denoting the battery level SoC, where $\bar{B}$ is the maximum charge level that battery can hold. Therefore, the state of the system at time t, $S(t)$, is denoted by:

$$S(t) \overset{\Delta}{=} [W(t), G(t), B(t)] \in \mathbb{S}, \tag{1}$$

where $\mathbb{S}$ stands for all possible states.

Figure 2a shows an example of states in our MDP-based modeling. In this example, we have four time intervals at different states. We assume one unit microservice(s) consumes one unit green energy or battery. Microservices are presumed to be grouped in a way that each group roughly consume one unit of energy. At $T0$, the demanded number of microservices is 6 units, green energy level is 3 and battery level is 2, which represents the state that green energy and battery cannot satisfy the required energy of microservices. At $T1$, the state represents that green energy is sufficient to handle the entire workload, e.g. at the noontime that solar power is adequate. In this state, no extra brown energy
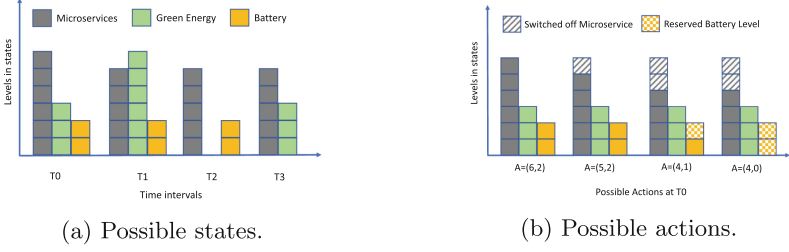

(a) Possible states.

(b) Possible actions.

Fig. 2. Example of states and actions in the system

is required. At $T2$, the state represents the situation where no green energy is available. The number of microservices is 5, green energy level is 0 and battery level is 2. At $T3$, the state represents the condition where battery is empty, so the battery level is 0.

3.2 Actions

At the beginning of each time-slot, the system determines the control action to switch off some microservices and to decide the allowed battery discharge. The next state $S(t + 1)$ depends on the current state $S(t)$ and the decision maker's action A. Our model must decide to perform the best possible action in each state. Actions are denoted as $A = (a, b)$, where a is the number of executed microservices and b denotes the maximum battery level allowed to be consumed to execute microservices. The actions change the states from one to another and achieve different rewards. The goal is to calculate the optimal policy, which is a mapping from states to actions such that the reward is maximized. The reward function will be discussed in the next subsection.

Figure 2b demonstrates a set of sample possible actions corresponding to the Fig. 2a state at $T0$. In $T0$, green energy and battery cannot fully satisfy the number of microservices. One may choose to run all 6 units of microservices using one unit of brown energy plus the entire battery charge, i.e., $A = (6, 2)$. Note that green power is always being used to the maximum a head of other sources, i.e. grid and battery. Another possible action executes only 5 out of 6 microservices and uses the entire battery storage levels so that no extra brown energy is required, i.e., $A = (5, 2)$. To reserve battery level charge for the next time slot, a possible action is to execute only 4 microservices and use one or none of the battery levels, i.e., $A = (4, 1)$ and $A = (4, 0)$.

The demanded workload and available green energy level at $S(t + 1)$ are independent of the state $S(t)$ and are determined stochastically according to the time. However, battery level at $S(t+1)$ depends on $S(t)$ and action A. If there is extra green energy in the last state, it will be used to charge battery in the next state. For $S(t) = [W(t), G(t), B(t)]$, the number of microservices that can be executed is $0 \leq a \leq W(t)$ and the battery level that can be consumed is $0 \leq b \leq min(B(t), [a - G(t)]^+)$, where $[x]^+ = max[0, x]$. Thus, the battery level

at $t + 1$ is determined according the following equation:

$$B(t+1) = min(\bar{B}, B(t) - b + [G(t) - a]^+), \qquad (2)$$

where $[G(t) - a]^+$ represents the level of not consumed green energy at t and used to charge battery.

3.3   Reward Function

At each time slot, the process is in a state $S(t)$, and we choose a possible action A. The process randomly moves to the next state $S(t+1)$ at the next time slot, and gives the corresponding reward $R(S(t), A)$. Our model intends to optimise two contradictory objectives: *minimizing brown energy consumption* and *maximizing the number of microservices executed*. Note that we target optional microservices with an interactive nature in our model that they cannot be delayed to be executed in the future. For example, current end users of a shopping website do not receive suggestions for items if the recommendation service is switched off. We introduce λ_r parameter to balance the trade-off between the number of microservices and brown energy usage. λ_r can be set by the system administrators to satisfy their cost and QoS requirements. Thus, we define the reward function $R(.)$ as:

$$R(.) = -\lambda_r \times [a - G(t) - b]^+ + (1 - \lambda_r) \times a. \qquad (3)$$

The first part $[a - G(t) - b]^+$ represents the brown energy usage and the second part, a, shows the number of executed microservices at time slot t. When $\lambda_r = 1$, the reward function only considers the brown energy usage. When $\lambda_r = 0$, the reward function is $R(.) = a$, that is, the reward function only considers the executed microservices. When λ_r is between 0 and 1, the reward function makes a balance between the number of executed microservices and brown energy consumption. The impact of λ_r will be evaluated in Sect. 5.4.

3.4   Transition Probabilities

The probability that the MDP moves into $S(t + 1)$ is influenced by the chosen action A. We assume that the decision maker has access to a long-time history of workloads demand and renewable power generation to compute probabilities. Thus, for each time slot, the probability of receiving the specific level of workload (the number of demanded microservices), $W(t)$, is known in advance and is denoted by $P(W(t))$. In Sect. 5.1, we explain how these probabilities are computed. The probability that the specific level of green power is generated at time slot t is denoted by $P(G(t))$. These values are also known to the decision maker in advance and are computed according to the history of renewable power generation. By knowing $W(t)$ and $G(t)$ and action $A = (a, b)$, we can compute the battery level SoC at $t+1$. Therefore, the transition probability from $S(t)$ to $S(t + 1)$ with a given action A is computed as:

$$P_A(S(t), S(t + 1)) = P(W(t + 1)) \times P(G(t + 1)). \qquad (4)$$

3.5   Optimal Policy

The optimal policy π^* describes the best action for each state in MDP which maximizes the expected reward in observation period, e.g. 24 h. The equation for the optimal policy is shown as follows:

$$V^{\pi^*}(S(t)) = \max_A \{R(S(t), A) + \sum_{S(t+1)} P_A(S(t), S(t+1)) \times V^{\pi^*}(S(t+1))\}, \quad (5)$$

where $V(S(t))$ is the expected reward obtained in the observation period, i.e., from the current time to the last time slot. In Eq. (5), the maximum reward that can be obtained at state $S(t)$ is computed by optimally choosing action A that maximizes the reward over all possible, next states $S(t+1)$. The above analysis converts our model to a dynamic programming problem.

4   MDP-Based Green-Aware Algorithm

Algorithm 1 shows the pseudocodes of our MDP-based green-aware algorithm.

Initializing System Information: At the beginning time interval, the algorithm uses the information to initialize system, including, the observation period, e.g. 24 h, the maximum number of levels for workloads, green energy and battery capacity (line 1).

Finding Reachable States and Possible Actions: Based on the probabilities of all states, only states with probability larger than 0 are considered as reachable. Meanwhile, based on the predefined maximum levels of workloads and green energy, possible actions can be found (lines 2–16).

Algorithm 1. MDP-based Green-aware algorithm

Input: System state, transition probabilities, observation time periods.
Output: Control actions
1: Initializing observation period T, the maximum levels of workloads $\bar{W}$ and green energy $\bar{G}$, battery capacity $\bar{B}$
2: **for** t from 0 to T **do**
3:     **for** $W(t)$ from 0 to $\bar{W}$ **do**
4:        **for** $G(t)$ from 0 to $\bar{G}$ **do**
5:            $P(S(t)) = P(W(t)) \times P(G(t))$
6:            **for** $B(t)$ from 0 to $\bar{B}$ **do**
7:               **if** $Pr[S(t)] > 0$ **then**
8:                   Adding $S(t)$ into reachable states $\mathbb{S}$
9:               **end if**
10:            **end for**
11:        **end for**
12:     **end for**
13: **end for**
14: **for** all reachable states in $\mathbb{S}$ **do**
15:     Adding action $A(a, b)$ into possible actions $A(S(t))$ for $S(t)$, $\forall\ 0 \leq a \leq W(t)$, $\forall\ 0 \leq b \leq min(B(t), [a - G]^+)$
16: **end for**
17: Updating transition probabilities $P_A(S(t), S(t+1))$ from $S(t)$ to $S(t+1)$
18: Updating the reward function $R(.)$ from $S(t)$ to $S(t+1)$
19: Calculating the optimal expected reward by algorithm 2 to find the $V^{\pi^*}(S(t))$
20: Deciding the control actions based on $V^{\pi^*}(S(t))$.
21: **return** best actions for states

Updating Transition Probabilities of States: Fetching the probabilities of different levels of workloads and availability of green energy (line 17).

Updating the Reward of States: Based on the different levels of workloads, green energy and consumed battery, the reward of each state is updated according to Eq. (3) (line 18).

Calculating the Utility Function: Using Algorithm 2, Algorithm 1 calculates the optimal expected reward value of reachable states (line 19).

Deciding the Actions: According to the optimised expected reward value that can be achieved, algorithm selects the action which maximizes the objective function for each state according to Eq. (5).

Algorithm 2 shows how to calculate the optimal expected reward by iterating over actions. The algorithm is based on value iteration to maximize the reward value, which represents the best control policy. With the inputs of reachable states and corresponding possible actions, the algorithm iterates over time periods 0 to T. The expected reward of a state is calculated in line 6. Then the algorithm iteratively updates the best reward value by going through all the reachable states and possible actions. In each iteration, the optimal expected reward value $V^{\pi*}(S(t))$ with optimal policy $\pi*$ is updated based on the expected reward in the previous state. After obtaining the optimal reward value, we can find the optimal control action.

Complexity Analysis: In our algorithm, for each state at a specific time interval, only the best action to reach the state is kept, which means the other actions are eliminated. Thus, the solutions space is $\Theta(\Lambda \times \Gamma \times \Delta)$ which is in polynomial complexity, where Λ, Γ, and Δ are the maximum level for the number of active microservices, green energy and battery, respectively.

Algorithm 2. The optimal expected reward value for all the states

---

Input: reachable states $S(t) \in \mathbb{S}$, possible actions $A(S(t))$ and estimated transition probabilities at time interval t as $P_A(S(t), S(t+1))$
Output: The optimal expected reward value
1: $t = 0$, $V(S(0)) = 0, \forall S(0) \in \mathbb{S}(0)$
2: **for** t from 0 to $T-1$ **do**
3:     $t = t + 1$
4:     $V^{\pi*}(S(t)) = -\infty$
5:     **for** $S(t) \in \mathbb{S}(t)$ **do**
6:         $V(S(t)) = \max_A \{R(S(t), A) + \sum_{S(t+1)} P_A(S(t), S(t+1)) \times V(S(t+1))\}$
7:         **if** $V(S(t)) > V^{\pi*}(S(t))$ **then**
8:             $V^{\pi*}(S(t)) = V(S(t))$
9:         **end if**
10:     **end for**
11: **end for**
12: **return** $V^{\pi*}(S(t))$ as optimal expect reward value

---

5   Performance Evaluations

In the following, we evaluate the performance of our proposed MDP-based approach. We use two workloads derived from realistic traces along with the historical solar data from the Australian Government Bureau of Meteorology[3].

5.1   Workload Traces

We use two realistic workload traces derived from Wikipedia[4] and Nectar[5] in our experiments. To convert workloads to fit into the states, we divide workloads into a set of levels with a specific range.

We use one-month Wikipedia data traces that contain 10% of all user requests issued to the Wikipedia website during this period. The total number of requests per hour ranges from 4 to 14 millions as shown in Fig. 3(a). The figure shows that the Wikipedia trace follows a typical pattern with the top and bottom number of requests during the day and night time, respectively. We divide the number of requests per hour into 10 levels, each representing a workload level in MDP. Each level is associated with a request rate range, of which the midpoint is used as the representative value for the corresponding level. For instance, the representative value for level 0 is 670 thousand and covers the range of 0 to 1340 thousand requests per hour. Figure 3(b) depicts the workload level conversion for the Wikipedia traces. It can be observed that the data still follows the pattern in Fig. 3(a), while the total workload levels are reduced to 10. Note that, in our experiment, we shifted Wikipedia data traces timing in a way that its user base would be in Australia at the same place as we consider the renewable power generation. Therefore, the peaks of workload coincides with the peaks of renewable power generation for the Wikipedia workload.

The Nectar Cloud platform provides the scalable computing infrastructure to Australian researchers and contains the traces of requests submitted for instantiating VM instances. Different from the Wikipedia trace, Nectar does not have a clear diurnal pattern and requests have start time, end time and five different VM types with the different number of vCPUs (virtual CPUs), e.g. small type has 1 vCPU and medium type has 2 vCPUs. We calculate the cumulative vCPU demand from users during 30 days as shown in Fig. 3(c), in which the demanded resource ranges from 500 to 2600 vCPUs. In Nectar traces, we cannot see an apparent pattern like that of Wikipedia. Similar to Wikipedia traces, we convert the vCPU resource consumption into 10 levels (Fig. 3(d)). For instance, level 0 represents values from 0 to 286 vCPUs.

5.2   Workload Level Probabilities

The probability $P(W(t))$ shows the likelihood that workload demand falls into a certain level in time slot t. In order to compute $P(W(t))$, we use existing

---

[3] http://www.bom.gov.au/climate/data-services/solar-information.shtml.
[4] http://www.wikibench.eu/wiki/2007-10/.
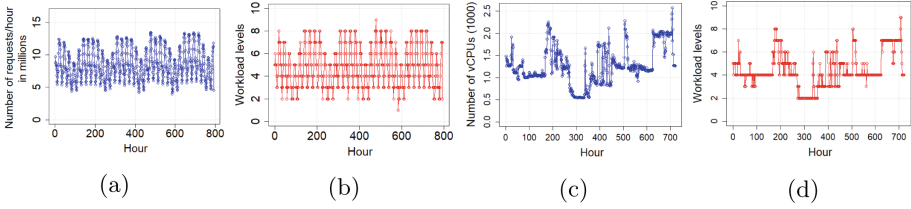[5] https://nectar.org.au/research-cloud/.

Fig. 3. (a) Original Wikipedia workloads. (b) Converted workload levels. (c) Original Nectar workloads. (d) Converted workload levels.

historical data based on a weekly cycle. Therefore, we keep different values of $P(W(t))$ for different hours and different days of a week. In order to compute $P(W(t))$ for a given time slot of a weekday (e.g., Monday, 8:00 to 9:00 am), we count the number of times that the historical workload hits a certain level (e.g., level 5) in the existing traces. Then the number is divided by the total number in all levels in time slot t to obtain the probability. This way, we create seven probability matrices (each for one day of a week) that contains 10 rows and 24 columns. Each cell shows the probability of receiving a certain level of workload at the specific time slot. The history data we used for the probability computation for Wikipedia is from September 19 to October 19, 2007 and for the Nectar is from December 1 to December 30, 2013.

5.3   Solar Power Levels

In order to make the solar data incorporated into our model, we use the hourly satellite data for the solar irradiation falling on a horizontal surface collected by the Australian Bureau of Meteorology. The trace has more than 30 years of hourly global horizontal solar irradiance (GHI) across Australia. We assume that the solar system fully converts hourly GHI value to power. Figure 4 shows the historical (original) and converted one-month solar irradiance data in Jan 2017 for the gridded data that covers Clayton campus at Monash University. In the historical data, the maximum value of GHI is $1108\,\mathrm{W/m}^2$ and the minimum value is zero. We also map GHI data into 10 levels of power, where the minimum solar level is 0 and the maximum solar level is 9.
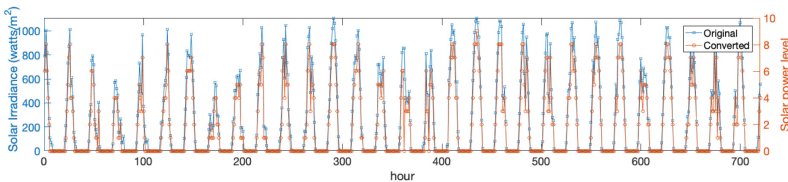

Fig. 4. Historical and converted solar irradiance.

To compute the likelihood of green power generation at a specific level $P(G(t))$, we use historical data at the same hour from the previous years. For example, if we want to calculate the probability of renewable power generation at level 2 at hour 11:00 am on the 1st of January, 2018, we look at the historical data at 11:00 am on the 1st of January from 1990 to 2017. Then, the probability that green power generation reaching level 2 is calculated based on the sample data.

Table 2. Impact of λ_r on the number of microservices and brown energy usage.

λ_r	0	0.25	0.5	0.75	1
Average number of microservices	4.26	4.26	2.47	2.47	0
Average brown energy usage	3.26	2.4	0.60	0.60	0

5.4   Evaluations with Different λ_r Values

To evaluate the impact of different λ_r values on the number of executed microservices and average energy consumption, we vary λ_r from 0 to 1 in the reward function as noted in Eq. (3) for the Wikipedia workload. The results are shown in Table 2. As we expect, the larger λ_r value, the fewer average number of microservices are executed and less brown energy is consumed. When $\lambda_r = 0$, the approach runs the maximum average number of 4.2 microservices and consume 3.3 units of brown energy. When $\lambda_r = 0.5$, the average number of microservices is reduced to 2.5 and the brown energy usage is decreased to 0.6. When $\lambda_r = 1$, no microservice is executed and no brown energy is consumed. We choose $\lambda_r = 0.5$ to balance the trade-off in the rest of experiments. In practice, the service administrators can set λ_r to fit into their preferences.

5.5   Baseline Algorithms

We use the following state-of-the-art heuristic algorithms as baselines:

DMWB (Demanded Microservices Without Battery): The algorithm executes the demanded number of microservices as it is received by the system, but does not use the battery.

SLW (Sliding Window): This sliding window algorithm [9] uses the recent historical actions to make an action. We set the sliding window size as 3 and the current action is the average number of microservices in the last 3 time intervals. SLW uses the maximum available battery capacity to power system whenever green energy is not sufficient.

BF (Best Fit): This algorithm is a representative energy management algorithm derived from [3]. It chooses the action that executes the maximum number of microservices with the least brown energy usage. The full battery capacity is consumed whenever green energy is insufficient.
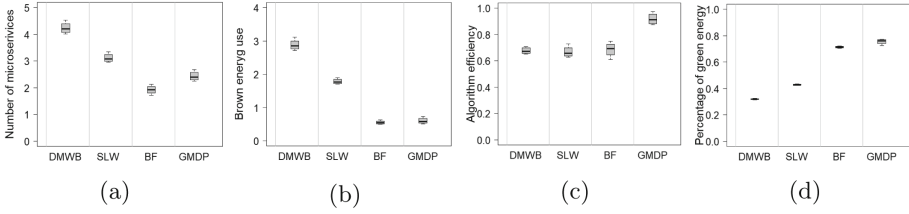
Fig. 5. Performance comparison of algorithms using Wikipedia workload.

We investigate the following metrics to evaluate system performance:

Number of Microservices: As one of the main objectives is to maximize the number of executed microservices, this metric measures the number of microservices on average over all hours.

Brown Energy Usage: Another main objective is to reduce brown energy consumption. This metric represents the average amount of brown energy usage over all hours. This metric can be read as the carbon footprint as well.

Algorithm Efficiency: The algorithm efficiency is represented as the optimization objective that considers both the brown energy usage and the number of active microservices simultaneously. The combined two objectives are derived from Eq. (3) as: $\frac{1}{T} \times \{\lambda \times \sum_{t=0}^{t=T} [a(t) - G(t) - b(t)]^+ + (1 - \lambda) \times \sum_{t=0}^{t=T} a(t)\}$, where $a(t)$ is the number of active microservices, $b(t)$ is the battery discharge at time interval t, and T is the 72-h observation period.

Percentage of Green Energy Usage: We also evaluate the percentage of green energy consumed out of the total energy usage on average over all hours.

5.6   Experimental Results

Figure 5 shows the performance comparison under the Wikipedia workload for three baseline algorithms and our proposed Green-Aware MDP-based algorithm (GMDP) over 3 days. The evaluated Wikipedia workloads start from October 20 to 22, 2007 and the evaluated Nectar workloads are from January 1 to 3, 2014. To avoid the seasonal variance of solar irradiance, we repeat our experiments with the solar data in the first three days in January, April, July and October 2017 respectively under the same Wikipedia workloads. From Fig. 5(a) and (b), DMWB represents the baseline that executes the demanded number of microservices as received, which executes 4.234 with 95% Confidence Interval (CI) (3.882, 4.586) microservices and 2.88 with 95% CI: (2.609, 3.153) brown energy usage. The SLW algorithm lowers the number of microservices and brown energy usage to 3.115 with 95% CI: (2.846, 3.384) and 1.781 with 95% CI: (1.649, 1.912) respectively. The BF algorithm supports the minimum number of microservices as 1.922 with 95% CI: (1.645, 2.199) and consumes the minimum brown energy as 0.544 with 95% CI: (0.432, 0655). Our proposed GMDP runs 27% more microservices than BF as 2.436 with 95% CI: (2.137, 2.736) and its brown energy usage is

0.60 with 95% CI: (0.449, 0.756), which is only 0.6 more than BF. Figure 5(c) and (d) depict the comparison of efficiency and percentage of green energy usage for algorithms. While GMDP reduces the number of microservices, it achieves the highest efficiency and percentage of green energy usage, which means GMDP can run the maximum number of microservices with the least brown energy usage.
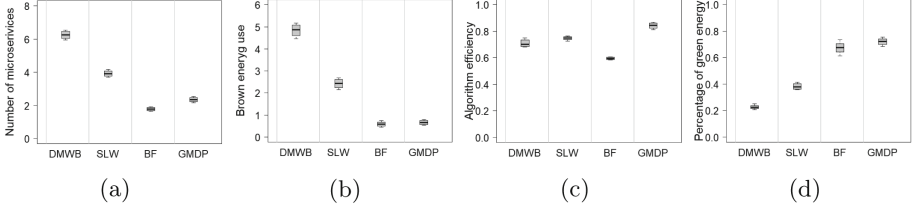

(a)   (b)   (c)   (d)

Fig. 6. Performance comparison of algorithms using Nectar workload.

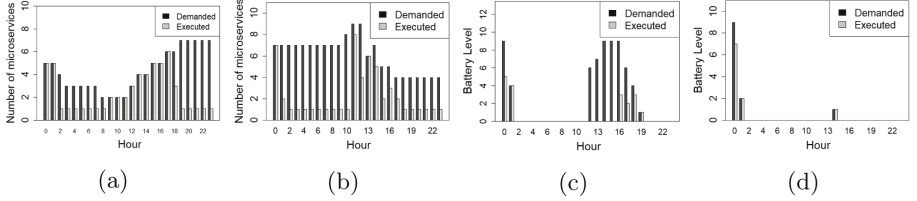(a)   (b)   (c)   (d)

Fig. 7. Actions corresponding to demanded workloads for (a) Wikipedia and (b) Nectar. Battery actions corresponding to available battery (c) Wikipedia and (d) Nectar.

Figure 6 depicts the comparison based on the Nectar workload. In Fig. 6(a) and (b), we can observe that the DMWB executes the maximum number of microservices as 6.253 with 95% CI: (5.833, 6.673) and the corresponding brown energy is 4.836 with 95% CI: (4.344, 5.329). SLW reduces the number of microservices to 3.919 with 95% CI: (3.585, 4.254) and brown energy usage to 2.425 with 95% CI: (2.064, 2.787). BF achieves the least number of 1.773 with 95% CI: (1.589, 1.958) microservices and minimum brown energy usage of 0.583 with 95% CI: (0.386, 0.782). GMDP increases microservices and brown energy usage to 2.34 with 95% CI: (2.083, 2.598) and 0.658 with 95% CI: (0.475, 0.841), respectively. Compared with BF, GMDP executes 31% more microservices with only 12% more brown energy use. GMDP has the highest efficiency and the largest percentage of green energy usage as shown in Fig. 6(c) and (d).

As a conclusion, we can say that GMDP achieves the best trade-off between the number of executed microservices and brown energy use, highest efficiency and the maximum percentage of green energy use. The DMWB algorithm executes the demanded number of microservices as received, however, its actions are

not optimised according to the availability of green energy and battery status of charge. The SLW algorithm takes the advantage of the recent actions. However, the percentage of green energy usage is not maximized. BF finds the best action at the current time period, however, the average value in the long term is not optimal. The GMDP algorithm avoids the limitations of baseline algorithms by reacting to the number of demanded microservices and searching a larger solution space. The GMDP algorithm reduces the brown energy usage to the level requested by the system administrator through lowering the number of executed microservices, and improves efficiency and percentage of green energy usage.

To demonstrate the behaviour of GMDP and the nature of selected actions, Fig. 7(a) and (b) show the executed number of microservices by GMDP versus the demanded number of microservices for the two workloads in the first day of the three-day observation period with solar irradiance in January. From Fig. 7(a), we notice that in some time periods, e.g. during time periods 0–1 and 9–17, the number of demanded microservices is the same as the executed microservices for the Wikipedia workload. This happens when the energy drawn from green sources and battery are sufficient to handle the entire workload. However, when the green energy and battery charge are not sufficient, e.g. during time periods 2–8, the executed microservices controlled by actions and are less than the demanded ones. Figure 7(b) shows similar behaviors for the Nectar workload. GMDP executes more microservices during the time period that green energy or battery charge are sufficient and efficiently reduces the executed microservices with the limitation in green energy according to the administrator preference (λ_r).

Figure 7(c) and (d) shows the sample battery status of charge and corresponding actions in the first day of our three-day observation period with solar irradiance in January for Wikipedia and Nectar workloads respectively. For the Wikipedia workload, we can notice that the initial battery is consumed in time periods 0 and 1., e.g. battery is discharged for 5 and 4 units respectively. The battery is not recharged until the time period 12, when the green energy is enough and can be charged into battery. Then the battery level is decreased to 0 during time periods 16 to 19, and there is no action for the battery in the time periods 20–23. The battery actions for Nectar workloads are much simpler compared with Wikipedia workloads. The battery is consumed in time period 0 and 1, and only recharged 1 unit at time period 14. This is because the Nectar workloads during time periods 0 to 13 are higher than Wikipedia workloads. Thus the green energy is consumed completely, and the battery has no chance to get recharged.

6   Conclusions and Future Work

We modeled the green-aware microservices management problem as a finite Markov Decision Process to reduce brown energy usage while provisioning resources for microservices. In our model, we consider brown energy, green energy (solar power) and a chargeable battery as the energy sources to power the data centers. To optimise system performance, our proposed MDP-based approach

called GMDP controls system actions and decides the number of microservices that must be executed out of the incoming workload and how much battery must be consumed in each time slot. We used real traces derived from Wikipedia and Nectar and Solar irradiance data from the Australian government Bureau of Meteorology to evaluate our system performance. Experimental results show that the proposed approach can efficiently balance the trade-off between the number of microservices and brown energy usage. In future, we plan to design and develop a prototype system incorporating the proposed algorithm. We will extend our model to consider more complex scenarios including battery self-discharge, variable grid electricity prices, and net metering. We will use a reinforcement learning approach to solve Markov decision processes.

References

1. Buyya, R., Srirama, S.N., et al.: A manifesto for future generation cloud computing: research directions for the next decade. ACM Comput. Surv. **51**(5), 105:1–105:38 (2018)
2. Cianfrani, A., Eramo, V., Listanti, M., Polverini, M., Vasilakos, A.V.: An OSPF-integrated routing strategy for QoS-aware energy saving in IP backbone networks. IEEE Trans. Netw. Service Manag. **9**(3), 254–267 (2012)
3. Farahnakian, F., Pahikkala, T., Liljeberg, P., Plosila, J., Hieu, N.T., Tenhunen, H.: Energy-aware VM consolidation in cloud data centers using utilization prediction model. IEEE Trans. Cloud Comput. **7**(2), 524–536 (2019). https://doi.org/10.1109/TCC.2016.2617374
4. Goiri, Í., Katsak, W., Le, K., Nguyen, T.D., Bianchini, R.: Parasol and GreenSwitch: managing datacenters powered by renewable energy. In: ACM SIGARCH Computer Architecture News, vol. 41, pp. 51–64. ACM (2013)
5. Han, Z., Tan, H., Chen, G., Wang, R., Chen, Y., Lau, F.C.M.: Dynamic virtual machine management via approximate Markov decision process. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM), pp. 1–9 (2016)
6. Jiang, D., Xu, Z., Liu, J., Zhao, W.: An optimization-based robust routing algorithm to energy-efficient networks for cloud computing. Telecommun. Syst. **63**(1), 89–98 (2016)
7. Liu, H., et al.: Thermal-aware and DVFS-enabled big data task scheduling for data centers. IEEE Trans. Big Data **4**(2), 177–190 (2018)
8. Liu, Z., et al.: Renewable and cooling aware workload management for sustainable data centers. In: ACM SIGMETRICS Performance Evaluation Review, vol. 40, pp. 175–186. ACM (2012)
9. Shaw, R., Howley, E., Barrett, E.: A predictive anti-correlated virtual machine placement algorithm for green cloud computing. In: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing, pp. 267–276. IEEE (2018)
10. Shen, H., Chen, L.: Distributed autonomous virtual resource management in datacenters using finite-Markov decision process. IEEE/ACM Trans. Netw. **25**(6), 3836–3849 (2017)

11. Terefe, M.B., Lee, H., Heo, N., Fox, G.C., Oh, S.: Energy-efficient multisite offloading policy using Markov decision process for mobile cloud computing. Pervasive Mob. Comput. **27**, 75–89 (2016)
12. Toosi, A.N., Qu, C., de Assunção, M.D., Buyya, R.: Renewable-aware geographical load balancing of web applications for sustainable data centers. J. Netw. Comput. Appl. **83**, 155–168 (2017)
13. Xu, M., Buyya, R.: Energy efficient scheduling of application components via brownout and approximate Markov decision process. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 206–220. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_14
14. Xu, M., Buyya, R.: Brownout approach for adaptive management of resources and applications in cloud computing systems: a taxonomy and future directions. ACM Comput. Surv. **52**(1), 8:1–82:7 (2019)
15. Zhang, Y., Wang, Y., Wang, X.: GreenWare: greening cloud-scale data centers to maximize the use of renewable energy. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 143–164. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25821-3_8