EA: Managing Green Data Centers Using Deep Reinforcement Learning Without Discounting

Peijian Wang, Thu Nguyen
Rutgers University
New Brunswick, NJ, USA
wpeijian@gmail.com, tdnguyen@cs.rutgers.edu

Abstract

For the past two decades, sustainability and carbon reduction have emerged as critical factors for data center (DC) design and operation. A set of advances has been encapsulated in green DCs with the onsite generation of renewable energy and efficient cooling systems. In this paper, we study how to apply Deep Reinforcement Learning (DRL) to optimize green DC operation. Green DC management is typically an infinite-horizon problem with exogenous stochastic input processes. We propose EA, a framework that applies DRL to the typical infinite-horizon problem without discounting. EA approximates the infinite-horizon problem with a finite-horizon one. In this approach, it is important to avoid actions optimized for the end of the finite-horizon problem but inappropriate for the true infinite-horizon one. EA addresses this challenge by combining a stationary policy with the fact that green DC management has repeating patterns (e.g., daily temperature, solar energy generation, and workload). We apply EA to the management of a green DC with onsite solar energy generation and a hybrid cooling system that includes "free" cooling. Evaluation results show that EA successfully learns important principles such as delaying deferrable jobs to solar-rich times and gracefully maintaining inside temperature. Further, EA outperforms three state-of-the-art DRL algorithms, realizing the greatest benefits on days with high outside temperature and high solar generation. While we evaluate EA in the specific context of a green DC, we believe that EA is a promising approach for more general system management settings.

Introduction

The demand for data center (DC) computing is surging due to the rapidly increasing needs for applications such as cloud computing, data analytics, and artificial intelligence (AI). Given that DCs are energy intensive, electricity consumption is also skyrocketing. In 2018, global DC energy use was estimated to be 205TWh, accounting for 1% worldwide electricity consumption (Masanet et al. 2020). 33% of these DCs are located in the US, and their electricity use is expected to grow at a rapid pace from about 4% of the US electricity demand in 2022 to about 6% in 2026 (IEA). This huge electricity consumption leads not only to large operational expenses, but also to high carbon emissions, since a large fraction of electricity is produced by fossil fuels; e.g., 60% in the US (EIA).

To improve sustainability and reduce carbon footprint, both industrial and academic have started building green (or sustainable) DCs, equipped with onsite renewable energy generation and efficient cooling systems (Goiri et al. 2013; Li, Qouneh, and Li 2012; Apple 2018). However, managing state-of-the-art green DCs is complex, possibly involving consideration of many different factors such as workload scheduling to meet performance goals, performance vs. energy efficiency of heterogeneous computing equipment, sources and composition of energy, and cooling considerations. Most existing solutions use heuristic designs that involve intensive human efforts (Wu et al. 2024; Qiu et al. 2020), and/or rely on strong (sometimes unrealistic) assumptions (Chen et al. 2018; Liao et al. 2024).

Recent work is demonstrating that Deep Reinforcement Learning (DRL) can provide a robust systematic approach to building such complex management systems (Sarkar et al. 2024b; Zhang et al. 2022). However, it is non-trivial to appropriately apply DRL to green DC management. One important challenge is that we are typically seeking to optimize an infinite-horizon problem. The most successful DRL algorithms use discounting to address this challenge. However, the discounted objective biases the optimization problem, preferring the near-term reward to the long-term one, and is generally not appropriate to the problem that tries to optimize long-term behavior (Zhang and Ross 2021).

A more natural criteria is to maximize the average reward. However, this setting is commonly considered in the classical theory of dynamic programming, but less commonly in RL (Sutton and Barto 2018), especially in the context of DRL (Zhang and Ross 2021). In addition, most RL algorithms are only benchmarked using experimental environments (e.g., OpenAI Gym (Brockman et al. 2016)) that generally do not have realistic physical constraints or environmental shifts (Yeh et al. 2023). There exist fundamental challenges (Dulac-Arnold et al. 2021) to apply DRL in real-world systems. The realism gap limits the reliable deployment of existing DRL algorithms in real-world systems.

In this paper, we investigate how to apply DRL to green DC management in non-discounted settings. Specifically, we orchestrate workload scheduling (shifting to solar-rich time), power management (tuning on/off servers), and cool-

ing control (using different cooling equipment and/or at different speeds) to reduce carbon footprint. We develop a new framework called Episodic Approximation (EA) which could apply a lot of off-the-shelf RL algorithms without discounting. In this approach, we pose the green DC management problem as an infinite-horizon problem and approximate it with a finite-horizon one. This approach provides several benefits: 1) it could increase the diversity of the agent's experience and avoid the agent trapped in bad states by frequently resetting the environment; 2) it can use the Monte Carlo method rather than bootstrapping to avoid learning value functions which are hard to estimate in the average-reward setting. However, it is non-trivial to apply this idea in a DRL setting. A common mistake (Pardo et al. 2018) is to consider the approximation as a true finite-horizon problem. The artificial end of the finite-horizon will lead to greedy actions which are inappropriate to the original infinite-horizon problem. For example, in workload scheduling, if some jobs are deferrable, the solution policy may never schedule these jobs to avoid incurring energy costs in the finite horizon.

Our solution is based on an important intuition that, if the same/similar states can be visited multiple times in an episode, optimization for the common operational case (former visits) can be more beneficial than inappropriate optimization for the artificial end (later visits). Then, we address the above challenge by 1) using stationary policies, despite the fact that non-stationary policies often achieve the best performance for finite-horizon problems; 2) choosing an appropriate length for the finite horizon. As a result, the optimal solutions for both original infinite-horizon problem and its finite-horizon approximation are the same/similar. In green DC management, there commonly exist exogenous stochastic input processes, which usually exhibit repetitive patterns; e.g., daily outside temperature, renewable energy production, and workload. The repetitive patterns can ensure multiple visits of states, and explicitly recognizing it allows us to explore what time horizon is appropriate for the finite-horizon problem to be a good approximation of the infinite-horizon one. Specifically, the time horizon has to be long enough for the repetitive patterns to be observable. With respect to our example of deferrable jobs above, the policy cannot defer jobs past the horizon to avoid incurring energy expenses because it has to behave similarly throughout the horizon, when jobs have to be scheduled or otherwise penalties will be incurred.

We evaluate the EA-based management system using both simulation and experimentation run on a small green DC. Extensive evaluation results show that EA can successfully learn important scheduling principles for managing green DCs, e.g., delaying deferrable jobs to solar-rich time period, and maintaining the inside temperature under a threshold. Furthermore, experiments on real system show some interesting observations, e.g., mismatch between simulator and real system can cause the agent to learn bad behavior, real system is partially observable and must be handled carefully. We compare EA with 3 state-of-the-art DRL algorithms, showing that EA outperforms all of them.

In summary, our contributions include:

- Developing the EA framework, analyzing fundamental challenges and discussing techniques most appropriate for applying DRL to the green DC management problem;
- Extensive evaluation of the EA-based implementation to show that EA can learn important management principles and outperforms two state-of-the-art DRL algorithms;
- Implementing EA on a small DC and showing interesting observations not captured in pure simulation studies.

Although we apply EA in the specific context of green DC, we believe that it is a promising approach for more general system management settings, helping bridge the gap between DRL studies and applying it to real-world problems.

Related Work

Infinite-Horizon Deep Reinforcement Learning

The most successful DRL algorithms (Mnih et al. 2015; Schulman et al. 2017) have been designed for the discounted setting. Among all, one approach called PEB (Pardo et al. 2018) is related to our method. PEB also studies how to learn an infinite-horizon problem in an episodic manner. Unlike EA learning a finite approximation, PEB continues bootstrapping at the artificial termination of the episode. The average-reward receives much less attention in the DRL setting. An extensive review can be found in (Dewanto et al. 2021). Recently, some on-policy (Zhang and Ross 2021; Ma et al. 2021) and off-policy (Zhang et al. 2021; Saxena et al. 2023; Hisaki and Ono 2024) algorithms have been developed. However, all above methods are only evaluated using experimental environments (e.g., Gym), and do not consider specific problem properties in green DC management (e.g., exogenous input processes). It is worth noting that EA is different from Episodic Control (Pritzel et al. 2017; Hu et al. 2021), which records highly rewarding experiences to improve sample efficiency.

Green DC Management

There is extensive research on green DC management using traditional heuristics and/or mathematical programming. Managing renewable energy (Goiri et al. 2013; Li, Qouneh, and Li 2012) and cooling (Goiri, Nguyen, and Bianchini 2015; Desu et al. 2021) are two important aspects. (Liu et al. 2012) integrates renewable energy and multiple cooling techniques, and proposes a convex optimization based on a very strong assumption.

Recently, people started applying DRL in green DC management (Li et al. 2020; Wang et al. 2022; Sarkar et al. 2024a). The most related works are GreenDRL (Zhang et al. 2022) and DC-CFR (Sarkar et al. 2024b). However, they directly use off-the-shelf RL algorithms without any additional consideration of infinite-horizon challenges. Specifically, GreenDRL applies an algorithm for finite-horizon; on the other hand, DC-CFR uses the discounted setting.

Episodic Approximation

In this section, we present our approach Episodic Approximation (EA) as a general framework, since it is independent of the specific problem. In the next section, we will show

how to apply it to green DC management. We define the objective function using the average reward criteria:

$$\max_{\pi} \lim_{h\to\infty} \mathbb{E}_{\pi}\left[\frac{1}{h}\sum\nolimits_{t=1}^{h} r_t\right], \tag{1}$$

where π is the policy, r_t is the reward obtained in slot t.

Problem Approximation

The basic idea of EA is very simple, approximating the infinite-horizon problem with a finite-horizon (or episodic) problem: $\max_{\pi} \mathbb{E}_{\pi}\left[\frac{1}{T}\sum_{t=1}^{T} r_t\right]$, where T is the length of the episode. T is the only parameter that we can control while approximating the problem. It is very important and can significantly impact EA's performance. In the above average reward setting, T does not necessarily need to be a constant number, and can be a random variable that varies from episode to episode. However, we argue that the constant T is a better choice. Random T may cause unfair comparison among different episodes. Intuitively, longer episodes can generally better approximate the infinite-horizon problem. If we use different lengths T from episode to episode during training, the average rewards of different episodes may not truthfully reflect the impact of actions. Therefore, we will use the constant T in this paper. Then we can change the objective function from average to sum without impacting the optimal solution:

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum\nolimits_{t=1}^{T} r_t\right]. \tag{2}$$

This formulation is consistent with the definition of finite-horizon problems in most of the literature.

Challenge The challenge of this approach comes from the fact that optimal policies for the original problem (1) and its approximation (2) are generally quite different (Pardo et al. 2018). Specifically, the possible future rewards beyond the artificial termination of finite-horizon cannot be accounted, which will lead to greedy actions at artificial end of episode.

Design

Design choice: Monte Carlo or bootstrapping. Monte Carlo and bootstrapping (also called temporal-difference or TD) are two major learning methods. Bootstrapping can be applied to both finite- and infinite-horizon problems, whereas Monte Carlo methods can only be applied to the former. In addition, bootstrapping is used in the most successful RL algorithms for the average-reward setting. Thus, one might lean toward using bootstrapping in EA. However, it is non-trivial to apply bootstrapping in the EA setting. Bootstrapping is generally used to learn a value function. But accurately estimating the value is one of the key difficulties in the average-reward RL, e.g., instability and drifting problems. Actually, using Monte Carlo methods is one of the benefits of EA, as it avoids learning a value function.

Design choice: stationary or non-stationary policy. Infinite- and finite-horizon problems usually have different types of optimal policies, stationary and non-stationary, respectively. In EA, we are approximating an infinite-horizon problem with a finite-horizon one. Thus, the following analysis leads us to choose stationary policies.

A true finite-horizon problem needs a non-stationary policy, because the time impacts the optimal decision. For example, it only needs to consider present rewards and choose greedy actions at the end of episode. However, in the early stage, it is better to take into account the future rewards even if encountering the same state. But in EA, we don't want the agent to act differently no matter the remaining time within the episode, and always prefer non-greedy actions. As discussed later, stationary policy can help training always optimize for the common operational case rather than the artificial end of episode.

Repeating pattern and episode length. As discussed, the key challenge of EA is the difference of optimal policies between the infinite-horizon problem and its finite-horizon approximation. In addition, exogenous input processes should be carefully considered. We argue that the challenge can be smoothly addressed if the exogenous processes exhibit repeating patterns. Combined with stationary policy, repeating pattern makes it equivalent/similar to optimize for short- and long-term duration.

Here, we also use the example of deferrable jobs to explain why repeating patterns can address the challenge and how to choose a good value for episode length. Consider a time slot very close to the end of the episode. Assuming we are minimizing the cost (maximizing negative reward), all deferrable jobs in this slot tend to be delayed after episode termination, such that they do not need to be executed and no related cost will be accounted for. However, there should be another slot in the previous cycle that has similar/same state if the episode length is longer than a cycle. The agent needs to choose the same/similar action. Deferring too many jobs will incur a large amount of penalty (assuming the deadline of defferable jobs is less than a cycle). If penalty incurred in the first cycle exceeds the cost saving in the second cycle, then arbitrarily delaying jobs at end of episode will lower the total reward. In this case, episode longer than two cycles is usually enough to learn a good policy as penalty for deadline violation is commonly very large. For the case with smaller penalty or longer deadline, we may need longer episode to make sure penalty incurred from earlier cycles is larger than the cost saving from later ones. Usually, the episode with several cycles is long enough. For the positive-reward (maximizing utility) scenario, analysis and conclusion are similar.

Therefore, if the exogenous input processes have repeating patterns and the episode length is appropriately chosen, the benefit of making good decisions for common operational cases will prevent the policy from making greedy actions. Clearly, repeating pattern becomes an important assumption for our method. However, we argue that this pattern commonly exists in green DC management, e.g., renewable energy generation and outdoor temperature have diurnal patterns, workload arrival usually have diurnal and/or weekly pattern(Gmach et al. 2007; Shahrad et al. 2020; Atikoglu et al. 2012; Verma et al. 2009).

Note The key idea behind EA is that (frequently visited) states are visited multiple times in an episode. Thus, repeating patterns are sufficient but not necessary for EA.

Therefore, we argue that EA should also work for more general scenarios. For example, under commonly used assumptions of unichain or ergodic, the most important (or frequently visited) states are recurrent and can be visited multiple times.However, EA can still benefit from repeating patterns, which allow us to easily decide the appropriate episode length.

Policy-gradient and advantage estimation EA is only a framework that needs to incorporate an underlying RL algorithm. In this paper, we focus on policy-gradient (PG) methods, one of the most important classes of RL algorithms.The parameter θ of a policy π_θ can be updated by

$$\theta \leftarrow \theta + \alpha A_t \bigtriangledown_\theta ln\pi_\theta(a_t|s_t), \tag{3}$$

where α is the update step size, and A_t is called advantage.

There are two commonly used Monte Carlo advantage estimations, total episode reward $G = \sum_{\tau=1}^{T} r_\tau$ and reward-to-go $G_t = \sum_{\tau=t}^{T} r_\tau$. Reward-to-go is much more commonly used as a simple and effective way to reduce variance.

However, we argue that reward-to-go may be problematic for EA setting. The advantage A_t will be the weights of related gradients that contribute to the final policy gradient (3). The reward-to-gos for different time slots may have different orders of magnitude (the earlier the higher), which suggests that different slots contribute very differently to the policy gradient. This makes sense for true finite-horizon problems, as actions in earlier slots have more impact. However, we are targeting infinite-horizon problems and want actions to have the same impact no matter the time in the artificial episode. Therefore, we choose the total episode reward, although reward-to-go is used in most existing algorithms.

Variance reduction Reducing variance is very important for RL algorithms. There exist many techniques that can be incorporated. In this paper, we use a method called input-dependent baseline (Mao et al. 2019), which handles exogenous stochastic processes. The basic idea is simple. Our control decisions can do nothing about the input processes; however, they have a significant impact on the rewards. If we can subtract some baseline related to the input processes, the result will be a better estimate to evaluate the pure impact of actions. A simple but commonly used baseline is the average reward among different trajectories with the same inputs $b = \frac{1}{N}\sum_{i=1}^{N}\sum_{\tau=1}^{T} r_\tau^i$, where N is the number of trajectories collected in a single training iteration. Then we can use $G - b$ as the advantage estimation A_t in update (3).

Green Data Center Management

We now apply EA to green data center management. Specifically, our goal is to orchestrate workload scheduling, power management, and cooling control to minimize total grid power consumption subject to certain operational constraints, such as maintaining the inside temperature within a certain range and avoiding workload deadline violation.

Data Center and Workload

We consider a data center partially powered by onsite generation of renewable energy. The data center draws power from the grid when there is insufficient renewable energy. It is equipped with a hybrid cooling system that includes air conditioner (AC) and "free" cooling (directly blow outside cold air). AC has a large and stable cooling capacity but also more power consumption, whereas free cooling is more power efficient, but the cooling capacity is dependent on weather conditions. AC and free cooling are good representatives with a trade-off between cooling capacity and power consumption.

The data center hosts a set of servers. We currently assume a homogeneous cluster and will consider heterogeneity in future work. Each server can be put into a sleep state.

We consider CPU-intensive workload. Each job specifies the requirement for the number of CPU cores. The workload can be classified into two categories: non-deferrable and deferrable. The former should be dispatched as soon as possible. The latter specifies a deadline and has the flexibility to be delayed as long as dispatched before its deadline.

Problem Definition

Epochs: We consider epochs (time slots) with 5 minutes. We choose 5 minutes because we need it to be long enough to operate the cooling facilities, and short enough to react to inside temperature change and workload arrival.

States and transition probabilities: In model-free DRL, models of system transitions are not needed. In addition, inputs to the neural network are usually different from the states. So we omit this information due to space limitations.

Actions: We need to make decisions in 3 dimensions: workload scheduling, power management, and cooling control. For cooling, we need to decide between AC and free cooling, and if the latter, the fan speed. For power management, we need to choose the number of active servers (turning on/off servers). For scheduling, we define NDF_server_t as the number of servers dedicated to non-deferrable jobs to reserve some space for them.

Rewards: We define the reward as a negative sum of energy cost, temperature violation penalty, and penalties for delaying jobs: $r_t = -\left(c_t + p_t^{temp} + p_t^{df} + p_t^{ndf}\right)$. We do not have any assumptions about the electricity cost c_t. In this paper, we use the simplest setting with a constant price. We define the temperature penalty p_t^{temp} as a linear function of the inside temperature exceeding an upper or lower threshold (set as $30°C$ and $10°C$). For deferrable jobs, the penalty p_t^{df} is a linear function of how much the dispatching time exceeds the deadlines. For non-deferrable jobs, the penalty p_t^{ndf} is a linear function of the waiting time (dispatching after arrival).

System Design

For scheduling and dispatching (i.e., selecting a job to be executed and sending it to a machine), we use a simple heuristic. 1) Try to schedule non-deferrable jobs before deferrable ones, 2) For the same type of jobs, schedule with FIFO, 3) Consolidate jobs to servers (turn on fewer servers) to save energy, 4) Skip a job if no active servers can host it to avoid head-of-queue blocking. If the number of skips reaches a predefined threshold, then stop scheduling all the other jobs until the blocked job is dispatched (prevent starving).

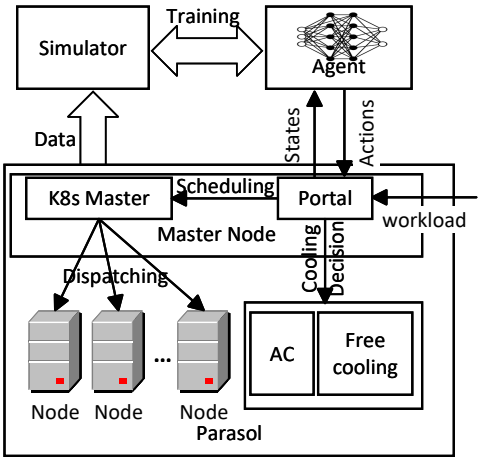

Figure 1: Implementation overview

We also use a simple heuristic for server sleep/wake-up. When selecting a server for sleeping, it always prefers the one with fewest running jobs. It first marks this server as unschedulable, and puts it into sleep state immediately after all running jobs are finished. For wake-up, it will first mark the unschedulable servers (if exist) to schedulable. If there still need more servers to be active, it will (randomly) choose the appropriate number of sleep servers to be waken up.

Implementation

We have implemented a prototype system on our real data center, Parasol (Goiri et al. 2013). It comprises a set of 16 solar panels that can generate up to 3.2kW of power, a RuppAir free-cooling unit, a Johnson Controls HVAC (DHR18NDB21S/DHR18CSB21S), an Automated Logic SE563sp controller, and two server racks which host more than 20 servers.

The overall structure is shown in Figure 1. We built a simulator with power and thermal models for both servers and cooling facilities, parameterized by more than 1 year of data collected from Parasol. The RL agent is trained on the simulator and then deployed on Parasol. We pack the jobs in containers (dockers), and use Kubernetes (K8s) (Burns et al. 2022) to manage them. We built a portal in front of K8s Master to implement additional logic, e.g., server sleep/wake-up, job scheduling, and cooling control. We use object detection jobs as workload, and choose two different object detection models as deferrable and non-deferrable jobs, respectively.

All major functions are implemented in the portal. 1. It keeps receiving the incoming job requests and maps them to K8s jobs. 2. It monitors system states, feeds them into the agent, and gets actions. 3. It manages worker nodes, including sleep/wake-up, communicating with K8s master to obtain/change the node states. 4. It monitors the lifetime of all jobs. 5. It controls cooling facilities.

The portal also works with K8s for job scheduling. 1. We define two non-preemption priority classes in K8s to distinguish the deferrable and non-deferrable jobs. K8s has its own back-off strategy, i.e., if jobs with higher priority cannot be scheduled (e.g., not enough resources), the scheduler will try other jobs. 2. We change K8s default dispatching strategy from load-balanced (called LeastRequested) to consolidation (MostRequested). 3. We use K8s taints and tolerations to reserve servers dedicated for non-deferrable jobs.

	GDRL	Mixed	EA	
Reward	-103.7	-84.9 ± 3.9	-82.7 ± 2.5	↑
Grid energy	96.4	83.2 ± 4.2	80.8 ± 3.7	↓
Server energy	182.9	182.4 ± 3.7	177.8 ± 3.3	↓
Cooling energy	37.5	31.6 ± 1.6	34.8 ± 2.4	↓
Green usage	124.0	130.8 ± 0.9	131.75 ± 2.0	↑
Temp penalty	6.4	0.5 ± 0.5	0.3 ± 0.4	↓
DF penalty	0.3	0.0 ± 0.0	0.5 ± 0.9	↓
NDF penalty	0.6	1.3 ± 0.9	1.2 ± 0.6	↓

Table 1: Comparison with GDRL. ↑ means higher is better.

	EA	PEB	APO
Reward	-84.6 ± 2.0	-97.9 ± 13.0	-117.4 ± 10.8

Table 2: Comparison with 2 baselines.

Evaluation

We evaluate EA applied to green DC management using both simulation and experiments on our real data center.

Evaluation Methodology

We use real-world trace (data) in our evaluation. For workload, we use Google trace (Reiss, Wilkes, and Hellerstein 2011) which contains the data from a cluster of about 12.5K servers over 29 days. For outdoor temperature and solar generation, we use historical data collected from Parasol. We also choose days with representative weather patterns to evaluate the policies.

We choose Proximal Policy Optimization (PPO) (Schulman et al. 2017) as the RL algorithm. We use Stable Baselines3 (SB3) (Raffin et al. 2021), a well-known set of reliable implementations of RL algorithms, modifying the advantage estimation from value function to Monte Carlo method. We train each policy 6 times with different seeds.

We evaluate EA from two aspects: 1) Performance comparisons with 3 representative state-of-the-art approaches, GreenDRL (Zhang et al. 2022) (our previous work on green DC management using DRL), PEB (Pardo et al. 2018) (a good way to apply RL in infinite-horizon problems with discounting), APO (Ma et al. 2021) (an algorithm for average-reward criteria also using PPO). 2) How does each component of our proposed method affect the performance.

Results on Simulator

Overall Performance We have renovated Parasol and built a new simulator after GreenDRL was published. We used the old simulator for comparison with GreenDRL and the new one for other experiments, and the agent (only one) reported in GreenDRL paper. To separate the impact of the EA framework and all the other factors, we also train another version of EA (referred to as Mixed), which follows

Episode length(days)	0.5	1	1.5	2	2.5	3
Reward	-7506.3 ± 12510.8	-112.0 ± 14.2	-89.0 ± 2.2	-86.7 ± 6.4	-85.6 ± 2.0	-84.6 ± 2.0

Table 3: Sensitive analysis on episode length.

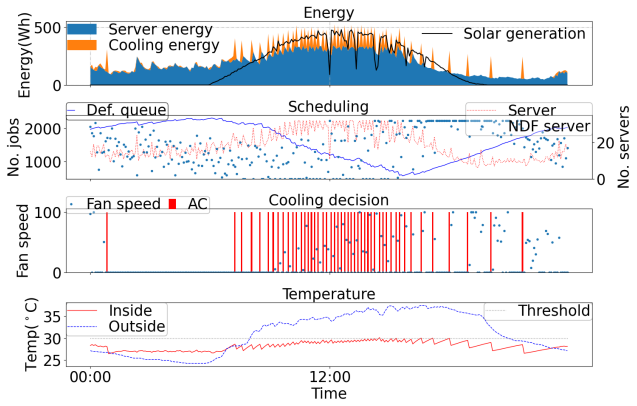

Figure 2: EA performance. Figures from top to bottom: 1) energy; 2) scheduling decisions and deferrable queue size; 3) cooling decisions; 4) inside and outside temperature.

the EA framework, but uses the NN inputs and actions of GreenDRL. The comparisons are summarized in Table 1.

Clearly, both Mixed and EA outperform GreenDRL (denoted as GDRL in the table) in almost all dimensions. EA has the highest reward, and the difference from Mixed is noticeable, but not too much. This confirms that the main performance improvement comes from the EA framework. Compared to GreenDRL, EA increases the reward by 20.3% and reduces the grid energy by 16.2%. Grid energy saving comes from both cooling (thereby total) energy reduction and better utilization of green energy. EA incurs a much smaller penalty for temperature violations. After checking the detailed behavior, we find that GreenDRL has 7 time slots with threshold violation and the maximum inside temperature reaches $30.8°C$. In contrast, EA5, which has the highest temperature penalty among all EAs, only has 4 violations with a maximum temperature of $30.2°C$. Both EA and GreenDRL have very low workload penalties, which suggest that they could schedule both types of jobs very well. EA has slightly larger penalties for both types of jobs.

For the other baselines (PEB and APO), we only compare the reward in Table 2, due to space limitations. Clearly, EA outperforms both. It is a little surprising that APO (average-reward) has much lower reward than PEB (discounting), showing that it is not easy to leverage DRL research advances in real-world problems like green DC management.

As a real-world problem, detailed behavior is also important. Figure 2 shows the EA performance for one day with high solar production and high outdoor temperature. From 8am to 5pm, when the outside temperature exceeds $30°C$ and rises to a peak, EA is able to gracefully keep the inside temperature below $30°C$ and only turns on AC when necessary. It also keeps all the servers active to fully utilize green energy. After 5pm, as solar generation decreases, EA decides to schedule fewer jobs and tries to use more free cooling. It keeps accumulating jobs for the next (potential) solar-rich period, until a certain point to avoid penalty. But it is a little aggressive to turn on AC and keep the inside temperature noticeably lower than $30°C$, thus losing some opportunity to use free cooling and recirculation. Generally, we conclude that EA can successfully learn to shift deferrable jobs to solar-rich periods, gracefully maintain the inside temperature under a threshold, and use free cooling to avoid running the expensive AC, although its performance still leaves room for further improvements.

Scenario	Deadline	Reward
Without repeating ($0.5 \rightarrow 3$ days)	36 hours	$(-3.4 \pm 2.3) \times 10^5$
	72 hours	$(-5.0 \pm 0.03) \times 10^7$
With repeating ($3 \rightarrow 0.5$ days)	2 hours	-103.3 ± 7.2
	4 hours	-85.7 ± 20.6

Table 4: Impact of repeating pattern.

Episode Length and Repeating Pattern In this section, we evaluate the impact of episode length and repeating pattern. We first perform a sensitive analysis on episode length and show the results in Table 3. Performance increases with the episode length. The rewards are very low for episodes of 0.5 and 1 day, and then stay stable, only increasing modestly for longer episodes. This confirms our analysis, as the episode with 1.5 days starts to have repeating patterns.

The above results confirm the importance of episode length. We need to separate the repeating pattern from the episode length to verify which is the key. We do experiments with synthetic traces that 1) long episode without repeating pattern; 2) short episode with repeating pattern. For 1), we expand the trace of half day to 3 days, repeating each time slot 6 times. For 2), we shrink the trace of 3 days to a half-day, sampling one slot from 6 consecutive slots. The (deferrable) deadlines should be changed proportionally. To give a more comprehensive study and understand the impact, we also try different deadlines. Table 4 shows the results.

As expected, rewards are good for short episodes with repeating patterns. The variance is large because short deadlines reduce the scheduling flexibility (job durations remain the same). By checking the detailed results, we can confirm that the agents can learn all important principles. In addition, a longer deadline (4 hours) leads to higher rewards, as it provides more flexibility.

The rewards are quite low for the scenarios without repeating pattern. For a deadline of 72 hours (equal to the episode length), all agents do not schedule any jobs, as delaying jobs will not incur any penalty during training. It is a little surprising that jobs are still delayed very aggressively

	Reward-to-go	Total episode reward
No baseline	-117.2 ± 14.2	-86.1 ± 4.3
State-value	-111.8 ± 16.4	-87.2 ± 2.3
Input-dependent	-85.9 ± 3.0	-84.6 ± 2.0

Table 5: Advantage and baseline.

(although not all) for a deadline of 36 hours (much shorter than the episode length). During training, the penalty for violation is observable, but the agents still fail to learn it. These experiments show the importance of repetitive patterns.

Advantage and Baseline In this section, we evaluate two design choices. 1. Advantage: total episode reward G versus reward-to-to G_t. 2. Baseline: no baseline, (Monte Carlo) state-value baseline, and input-dependent baseline. Table 5 shows the results. Total episode reward always outperforms reward-to-go. Without baseline, total episode reward is much better than reward-to-go. This confirms our intuition that reward-to-go is not a good choice for EA setting. Input-dependent baselines can significantly improve reward-to-go because the main part of different weights will be subtracted. However, baselines only give a marginal improvement on total episode reward, which implies that it is already a good choice for EA, leaving little room for further improvement.

Results on Parasol

After the EA agent was initially deployed on Parasol, it violated the temperature threshold (exceeds $32°C$ very often) in most slots during the hottest time. We found that the reason was the mismatch between the simulator and the real system. 1. We use regression to obtain the cooling models, which always have errors. 2. We define the state by temperature at the beginning of the slots. However, the maximum temperature may occur in the middle of slots as our AC has a 3-minute delay to turn on the compressor. These mismatches show that the real system commonly exhibits both stochasticity and partial observability compared to simulated environments. We solve this problem by adding a new (stochastic) model of maximum inside temperature and using it to calculate the penalty. It is worth noting that the maximum temperature is not included in the input to the NNs, and thereby is still not observable to the EA agent.

We ran the experiment over one month, but only show the most representative consecutive days (with different solar-temperature patterns) in Figure 3. Clearly, the inside temperature was successfully maintained below the threshold. When there was enough solar energy (e.g., after 12:00 on 8/28 and 8/29, and some time slots on 8/31), the agent might aggressively turn on the AC and keep the inside temperature noticeably lower than the threshold. But this incurred no cost. When the green energy was not enough, the agent turned on less AC and maintained the inside temperature smoothly under the threshold, e.g., later time on 8/28. But EA's decisions were not perfect. 1. It aggressively turned on free cooling to keep the inside temperature unnecessarily low, e.g., 8/30. It did not incur too much cost, as free cooling is cheap, but still leaving some space for improvement. 2. It occasionally turned on AC even if unnecessary, e.g., early morning on 8/30.

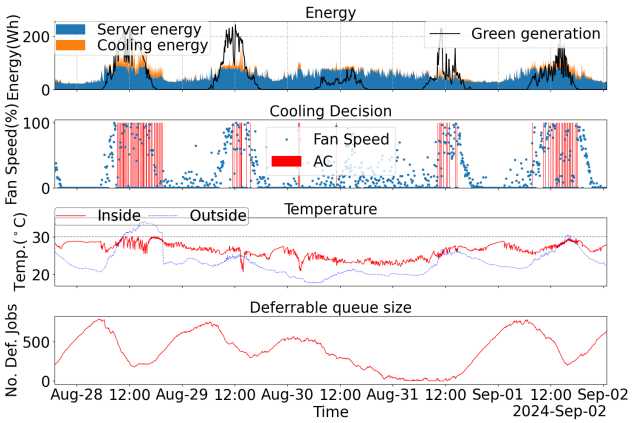

Figure 3: Experiment on Parasol. Figures from top to bottom: 1) energy consumption; 2) cooling decisions; 3) inside and outside temperature; 4) deferrable queue size.

The scheduling decisions are similar to those on the simulator. When there was insufficient green energy, the deferrable queue was accumulated fast until a certain level, and then kept relatively stable to avoid deadline violation. When there was enough solar energy, it turned on all servers and set small NDF_server to schedule more deferrable jobs.

It is worth noting that the agent still scheduled a lot of jobs starting early 8/30, even if there was no green energy and the queue was still at a low level. This is because our simulator and the real system have different dispatching strategies. As mentioned, the simulator blocks jobs and thereby prevents starving jobs. But K8s will continue to dispatch jobs even if starving occurs in real system. Starting early 8/30, one big deferrable job (requiring 8 CPU cores) never got its required resources. So the agent (trained on simulator) realized some job was violating its deadline and kept trying to schedule more deferrable jobs, expecting that job can be dispatched. This is good in our simulator, as deadline violations always mean that the queue is too long, and dispatching more jobs is a correct decision. However, this decision was not good for the real system, as it actually decreased the possibility of dispatching the big job. This interesting observation shows that our agent can learn a good behavior on our simulator, but may make bad decisions on the real system (due to the mismatch between the two).

Conclusion

This paper presents Episodic Approximation, a new method to solve infinite-horizon RL problems without discounting. It uses a simple idea of finite-horizon approximation. However, there exists a fundamental challenge while applying this idea. We find that repeating patterns and stationary policy can address this challenge. Then we apply this method to a green DC management problem and implement a prototype system. Extensive experiments on both the simulator and a real data center show the effectiveness of our method.

References

Apple. 2018. Apple now globally powered by 100 percent renewable energy. https://www.apple.com/newsroom/2018/04/apple-now-globally-powered-by-100-percent-renewable-energy/. Accessed: 2025-07-25.

Atikoglu, B.; Xu, Y.; Frachtenberg, E.; Jiang, S.; and Paleczny, M. 2012. Workload analysis of a large-scale key-value store. In *Proceedings of the 12th ACM SIGMETRIC-S/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, 53–64. New York, NY, USA: Association for Computing Machinery. ISBN 9781450310970.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:1606.01540.

Burns, B.; Beda, J.; Hightower, K.; and Evenson, L. 2022. *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc.".

Chen, L.; Lingys, J.; Chen, K.; and Liu, F. 2018. AuTO: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, 191–205. New York, NY, USA: Association for Computing Machinery. ISBN 9781450355674.

Desu, A.; Puvvadi, U.; Stachecki, T.; Vishwakarma, S.; Khalili, S.; Ghose, K.; and Sammakia, B. G. 2021. Latency-Aware Dynamic Server and Cooling Capacity Provisioner for Data Centers. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '21, 335–349. New York, NY, USA: Association for Computing Machinery. ISBN 9781450386388.

Dewanto, V.; Dunn, G.; Eshragh, A.; Gallagher, M.; and Roosta, F. 2021. Average-reward model-free reinforcement learning: a systematic review and literature mapping. arXiv:2010.08920.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9): 2419–2468.

(EIA), U. E. I. A. 2024. Monthly Energy Review December 2024. www.eia.gov/mer. Accessed: 2025-01-19.

Gmach, D.; Rolia, J.; Cherkasova, L.; and Kemper, A. 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. In *2007 IEEE 10th International Symposium on Workload Characterization*, 171–180.

Goiri, Í.; Nguyen, T.; and Bianchini, R. 2015. CoolAir: Temperature- and variation-aware management for free-cooled datacenters. In *ASPLOS 2015 - 20th International Conference on Architectural Support for Programming Languages and Operating Systems*, 253–265. Association for Computing Machinery.

Goiri, I. n.; Katsak, W.; Le, K.; Nguyen, T. D.; and Bianchini, R. 2013. Parasol and GreenSwitch: managing datacenters powered by renewable energy. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, 51–64. New York, NY, USA: Association for Computing Machinery. ISBN 9781450318709.

Hisaki, Y.; and Ono, I. 2024. RVI-SAC: average reward off-policy deep reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Hu, H.; Ye, J.; Zhu, G.; Ren, Z.; and Zhang, C. 2021. Generalizable Episodic Memory for Deep Reinforcement Learning. arXiv:2103.06469.

(IEA), I. E. A. 2024. Electricity 2024. Technical report, International Energy Agency, Paris. License: CC BY 4.0.

Li, C.; Qouneh, A.; and Li, T. 2012. iSwitch: Coordinating and optimizing renewable energy powered server clusters. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, 512–523.

Li, Y.; Wen, Y.; Tao, D.; and Guan, K. 2020. Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning. *IEEE Transactions on Cybernetics*, 50(5): 2002–2013.

Liao, X.; Tian, H.; Zeng, C.; Wan, X.; and Chen, K. 2024. Astraea: Towards Fair and Efficient Learning-based Congestion Control. In *Proceedings of the Nineteenth European Conference on Computer Systems*, EuroSys '24, 99–114. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704376.

Liu, Z.; Chen, Y.; Bash, C.; Wierman, A.; Gmach, D.; Wang, Z.; Marwah, M.; and Hyser, C. 2012. Renewable and cooling aware workload management for sustainable data centers. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, 175–186. New York, NY, USA: Association for Computing Machinery. ISBN 9781450310970.

Ma, X.; Tang, X.; Xia, L.; Yang, J.; and Zhao, Q. 2021. Average-Reward Reinforcement Learning with Trust Region Methods. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2797–2803. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Mao, H.; Venkatakrishnan, S. B.; Schwarzkopf, M.; and Alizadeh, M. 2019. Variance Reduction for Reinforcement Learning in Input-Driven Environments. In *International Conference on Learning Representations*.

Masanet, E.; Shehabi, A.; Lei, N.; Smith, S.; and Koomey, J. 2020. Recalibrating global data center energy-use estimates. *Science*, 367(6481): 984–986.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Pardo, F.; Tavakoli, A.; Levdik, V.; and Kormushev, P. 2018. Time limits in reinforcement learning. In *International Conference on Machine Learning*, 4045–4054. PMLR.

Pritzel, A.; Uria, B.; Srinivasan, S.; Badia, A. P.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural Episodic Control. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 2827–2836. PMLR.

Qiu, H.; Banerjee, S. S.; Jha, S.; Kalbarczyk, Z. T.; and Iyer, R. K. 2020. FIRM: An Intelligent Fine-grained Resource Management Framework for SLO-Oriented Microservices. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 805–825. USENIX Association. ISBN 978-1-939133-19-9.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Reiss, C.; Wilkes, J.; and Hellerstein, J. L. 2011. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, 1: 1–14.

Sarkar, S.; Naug, A.; Guillen, A.; Luna, R.; Gundecha, V.; Ramesh Babu, A.; and Mousavi, S. 2024a. Sustainability of Data Center Digital Twins with Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(21): 23832–23834.

Sarkar, S.; Naug, A.; Luna, R.; Guillen, A.; Gundecha, V.; Ghorbanpour, S.; Mousavi, S.; Markovikj, D.; and Ramesh Babu, A. 2024b. Carbon Footprint Reduction for Sustainable Data Centers in Real-Time. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20): 22322–22330.

Saxena, N.; Khastagir, S.; Kolathaya, S.; and Bhatnagar, S. 2023. Off-Policy Average Reward Actor-Critic with Deterministic Policy Search. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 30130–30203. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shahrad, M.; Fonseca, R.; Goiri, I.; Chaudhry, G.; Batum, P.; Cooke, J.; Laureano, E.; Tresness, C.; Russinovich, M.; and Bianchini, R. 2020. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 205–218. USENIX Association. ISBN 978-1-939133-14-4.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Verma, A.; Dasgupta, G.; Nayak, T. K.; De, P.; and Kothari, R. 2009. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, 28. USA: USENIX Association.

Wang, R.; Zhang, X.; Zhou, X.; Wen, Y.; and Tan, R. 2022. Toward Physics-Guided Safe Deep Reinforcement Learning for Green Data Center Cooling Control. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, 159–169.

Wu, D.; Wang, X.; Qiao, Y.; Wang, Z.; Jiang, J.; Cui, S.; and Wang, F. 2024. NetLLM: Adapting Large Language Models for Networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, 661–678. New York, NY, USA: Association for Computing Machinery. ISBN 9798400706141.

Yeh, C.; Li, V.; Datta, R.; Arroyo, J.; Christianson, N.; Zhang, C.; Chen, Y.; Hosseini, M. M.; Golmohammadi, A.; Shi, Y.; Yue, Y.; and Wierman, A. 2023. SustainGym: Reinforcement Learning Environments for Sustainable Energy Systems. In *Advances in Neural Information Processing Systems*, 59464–59476. Curran Associates, Inc.

Zhang, K.; Wang, P.; Gu, N.; and Nguyen, T. D. 2022. GreenDRL: managing green datacenters using deep reinforcement learning. In *Proceedings of the 13th Symposium on Cloud Computing*, 445–460.

Zhang, S.; Wan, Y.; Sutton, R. S.; and Whiteson, S. 2021. Average-Reward Off-Policy Policy Evaluation with Function Approximation. In *Proceedings of the 38th International Conference on Machine Learning*, 12578–12588. PMLR.

Zhang, Y.; and Ross, K. W. 2021. On-Policy Deep Reinforcement Learning for the Average-Reward Criterion. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 12535–12545. PMLR.