Online Decision-Making Under Uncertainty for Vehicle-to-Building Systems

Abstract

Vehicle-to-building (V2B) systems combine physical infrastructure such as smart buildings and electric vehicles (EVs) connected to chargers at the building, with digital control mechanisms to manage energy use. By utilizing EVs as flexible energy reservoirs, buildings can dynamically charge and discharge EVs to effectively manage energy usage, and reduce costs under time-variable pricing and demand charge policies. This setup leads to the V2B optimization problem, where buildings coordinate EV charging and discharging to minimize total electricity costs while meeting users' charging requirements. However, the V2B optimization problem is difficult due to: 1) fluctuating electricity pricing, which includes both energy charges ($/kWh) and demand charges ($/kW); 2) long planning horizons (usually over 30 days); 3) heterogeneous chargers with differing charging rates, controllability, and directionality (unidirectional or bidirectional); and 4) user-specific battery levels at departure to ensure user requirements are met. While existing approaches often model this setting as a single-shot combinatorial optimization problem, we highlight critical limitations in prior work and instead model the V2B optimization problem as a Markov decision process, i.e., a stochastic control process. Solving the resulting MDP is challenging due to the large state and action spaces. To address the challenges of the large state space, we leverage online search, and we counter the action space by using domain-specific heuristics to prune unpromising actions. We validate our approach in collaboration with an EV manufacturer and a smart building operator in California, United States, showing that the proposed framework significantly outperforms state-of-the-art methods.

ACM Reference Format:

. . Online Decision-Making Under Uncertainty for Vehicle-to-Building Systems. In *Proceedings of* . ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Electric vehicles (EVs) are at the frontier of the global energy landscape's shift towards more sustainable energy solutions [19]. The management of the EVs' energy requirements presents interesting opportunities and challenges; we focus on one such opportunity—vehicle-to-building charging (V2B)—that involves co-optimizing the energy management of EVs and smart buildings. The key idea behind V2B charging exploits the ability to control the rates of charging and leverages the use of bidirectional EVs as energy storage facilities to add resilience and demand-response capabilities to smart buildings [34]. For example, EVs can be charged when energy is available at lower costs and discharged to supply energy to buildings when energy costs are high, thereby promoting optimal energy use, and reducing peak power demands (highest instantaneous power usage in the billing period), substantially decreasing energy and demand costs for the building [29]. A part of the savings can then be shared with EV owners, either directly or through discounted charging, thereby producing a *win-win* framework for both the building and the EV owners.

Despite the apparent simplicity of the V2B framework, operationalizing it presents several challenges. While EV owners can be incentivized to participate in such programs by offering charges at low (or zero) cost, strategies that solely optimize energy costs can result in arbitrarily low EV charges when the owner leaves the smart building, thereby causing inconvenience to the building owners. Therefore, a V2B optimization framework must ensure that EV owners depart with a pre-specified level of charge at their departure time while dealing with the exogenous uncertainties of fluctuating building load, EV arrivals (and departures), and energy prices. Given that many EVs arrive throughout the day, the underlying optimization problem is extremely challenging—charging configurations vary across EVs, and modern buildings have heterogeneous EV chargers (of different makes, models, rates of charging, and directionality), thereby presenting a complex sequential decision-making problem under uncertainty. Additionally, the V2B framework must be able to accommodate complex pricing policies of power companies, e.g., time-of-use energy charges (dollar per kilowatt-hours (kWh)) and demand charges (dollar per kilowatts (kW)); crucially, the demand cost is computed over a relatively longer temporal horizon, e.g., for most American power companies, this involves computing the maximum instantaneous power utilization over a month.

The underlying optimization problem for the V2B problem has been explored in prior work; e.g., Tanguy et al. [32] present one of the most comprehensive optimization frameworks for this problem setting. Their model is elegant and extremely tractable—the decision problem (i.e., deciding *when* and by *how much* each vehicle is charged or discharged) is modeled as a linear program that can be efficiently solved in polynomial time, providing scalability by design. However, the inherent scalability comes at the cost of several assumptions that are not valid in practice: 1) the *single-shot* optimization approach assumes that car arrivals and departures are known apriori, and computing all vehicle-to-charger assignments at once prohibits adaptability to dynamic changes, e.g., varying energy prices or building loads; 2) the linearity of the formulation implicitly relies on homogeneous charger configurations; and 3) the formulation captures only energy cost but cannot accommodate demand cost, which is an important determinant of V2B policies in practice. We address these limitations by formulating the V2B problem as a stochastic control process and modeling it as a Markov decision process (MDP). Our formulation is motivated by the observation that, in practice, decisions about charging and

https://doi.org/XXXXXXX.XXXXXXX

discharging vehicles (and by how much) must be taken sequentially under exogenous sources of uncertainty.

However, computing an optimal policy for the MDP is very challenging in our problem setting, particularly due to *complex credit assignment* and the *curse of dimensionality* [31]. Specifically, given that the decisions must be computed at a relatively high frequency (e.g., every 15 minutes) and the demand cost is only observed at the end of a billing period (usually a month), the decision maker's actions have long-term consequences, but the rewards or feedback may only be received much later. Moreover, as the planning horizon increases, the number of possible states and actions grow exponentially. This growth leads to a combinatorial explosion, making it computationally infeasible to consider all possible sequences of actions and states. To tackle these challenges, we propose an online approach that focuses on computing near-optimal actions for the current state of the system instead of seeking to learn a policy. Our approach is based on Monte Carlo tree search (MCTS) [10], a general-purpose online algorithm for stochastic control processes.

While MCTS tackles some of the challenges of the V2B setting by using a bandit-based strategy for exploring promising trajectories in the decision space, it suffers from the curse of dimensionality from the action space, making it infeasible for our problem setting (as we show later). To address these challenges, we present *DG-MCTS* (domain-knowledge guided MCTS), which uses an action-generation framework that exploits domain knowledge and the underlying structure of the V2B optimization problem. This action-generation framework massively truncates the decision space. We draw from well-established heuristics and augment the set with randomly generated actions to ensure that the search tree is not limited to this truncated set. Collaborating with a major EV manufacturer, we conduct extensive simulations on real-world data, showing that our approach outperforms existing approaches. To further ensure that our approach can scale to extremely large problem instances, we also propose a decentralized search algorithm that sacrifices performance (by a *small* degree) to save computation time. In summary, our contributions are:

- We present a decision-theoretic formulation for the V2B problem that models the interaction between electric vehicles and smart buildings as a stochastic control process.
- We show how MCTS, augmented with an action-generator module based on domain-specific heuristics and a randomized augmentation step, can compute near-optimal actions for the MDP.
- We also present a decentralized version of the algorithm that is significantly faster, albeit at the cost of a small deterioration in performance.
- We use real-world data in collaboration with a large EV manufacturer and show that the proposed approach outperforms competitive baselines.

2 Related Work

We highlight four major challenges of solving the V2B problem, namely: 1) the uncertainty due to EVs' arrival and departure times; 2) the dynamic energy costs, building loads under Time-of-Use (TOU) pricing; 3) demand charges and long-term rewards; and 4) the heterogeneity of chargers;

Inherent uncertainty: Traditional optimization methods like Linear programming (LP) and Mixed Integer Linear Programming (MILP) are popular techniques for optimizing energy distribution in V2B systems, often applied to minimize operational costs while meeting various constraints [12, 32]. However, these methods typically assume knowledge of power usage and EV user schedules, assumptions that are difficult to predict and satisfy in real-world applications accurately. To address this, recent research has explored stochastic optimization [22, 35] and robust control methods to handle this inherent uncertainty[18]. Research on probabilistic forecasting of EV behavior, utilizing models such as Recurrent Neural Networks (RNN) can further help mitigate these challenges by capturing temporal dependencies and uncertainties [37]. Reinforcement Learning (RL) has also been applied to learn adaptive charging strategies [2, 27]. Despite these advances, managing uncertainty, especially over longer time horizons remains significantly challenging. Deep learning models may struggle with long-term dependencies, and RL algorithms can suffer from issues like sample inefficiency and convergence difficulties in complex environments[1]. The lack of exposure to diverse scenarios such as extreme weather conditions, sudden changes in power utility rates, or sudden shifts in EV behavior can limit the model's ability to adapt to new, unforeseen situations, potentially leading to highly suboptimal solutions.

Time-Of-Use (TOU) rates: The variability of energy prices under TOU rates introduces additional complexity to the V2B optimization, requiring charging schedules that adapt to dynamic electricity costs, to minimize expenses [7]. The use of heuristics and machine learning algorithms, along with Deep Reinforcement Learning (DRL) has been useful in learning optimal charging policies responsive to fluctuating TOU rates [14]. However, these methods face challenges due to difficulties in credit assignment and non-stationary patterns in energy usage, by both the smart building and the EV users [13].

Demand charges and long-term rewards: Optimizing V2B is hard due to the mismatch between the prediction horizon and billing period. Existing work on demand charge prediction using model predictive control (MPC) is limited by the computational complexity of the long billing period. Thus, they resort to using shorter prediction horizons of a single day, resulting in suboptimal decisions, especially when the peak demand occurs outside the prediction horizon [3, 16, 21].

Charger heterogeneity: Smart buildings typically implement EV charging infrastructure incrementally, resulting in a diverse array of chargers with varying power ratings and directionality. This further complicates the action space for optimization. Some prior work addresses heterogeneity [8, 17]. However, they sacrifice long-term rewards by limiting planning to a single day or disregarding the presence of demand charges altogether.

Role of Monte Carlo Tree Search (MCTS): Search-based algorithms like Monte Carlo Tree Search (MCTS) offer a promising alternative to handle inherent uncertainty in V2B systems. MCTS can manage stochastic environments by simulating numerous possible future scenarios, making it well-suited for planning under uncertainty [4, 20]. Unlike traditional RL methods, MCTS can effectively plan over long horizons by building a search tree that considers future states and rewards [23]. This capability allows it to handle long-term dependencies and adapt to sudden changes in

EV behavior by continuously updating the search tree with new information [15].

3 Problem Description

We begin by describing our problem setting. Recall that in our setting, the EV owners follow a regular pattern of arrivals and departures with a predictable battery usage profile. While the building's energy usage is typically not known ahead of time, we assume access to a predictive model that uses historical data to predict the building's energy usage, which can be done very reliably [28, 33].

3.1 Specifications

Time slots: The total time under consideration (billing period) is divided into a finite set of discrete time slots. A time slot $t \in \mathcal{T}$ begins at t^{start} and ends at t^{end}. The duration of each time slot is $\delta_t = t^{\text{end}} - t^{\text{start}}$, measured in hour. Depending on the pricing policy of the power utility company, the hours of the day can be divided into *peak* and *off-peak* hours. The power utility usually computes the demand cost (explained later) during the peak hours $\mathcal{T}^{peak} \subset \mathcal{T}$, based on an aggregate of time slots of length τ, i.e., $t_j^\tau = \{t_i, t_{i+1}, ..., t_{i+\tau}\}$ for the j^{th} aggregate slot, and i is an index to the corresponding time slot. The duration of each aggregate time slot is $\delta_a = \delta_t \cdot \tau$. The peak hours are then divided into Ω aggregated time slots $\{t_1^\tau, t_2^\tau, \cdots, t_\Omega^\tau\}$. We define all such sets of $t_i \in t_j^\tau$ using the notation $\mathcal{T}_j^\tau$. For example, if the peak hours are from 12 am to 1 am, each time slot is 5 minutes, and τ is 15 minutes, then the 1st aggregate time slot ($j = 0$), which is from 12 am to 12:15 am, is represented as $t_0^\tau = \{t_0, t_1, t_2\}$, then $\mathcal{T}_0^\tau = \{0, 1, 2\}$.

Time-of-Use Price: Power companies usually compute cost based on two components or charges—an energy charge and a demand charge. Some companies have variable energy charge rates, where a time-of-use pricing model is used. The energy charge (\$/kWh) can vary between the peak and off-peak hours and is denoted by the set $\mathcal{W}$ where w_t^e is the time of use price for time slot $t \in \mathcal{T}$.

Demand Charge: The other part of cost computation is the demand charge w^d (\$/kW). This is a fee based on the highest rate of electricity usage during a specific time period within a billing period. Often only usage during peak hours is considered.

Vehicles: We denote the set of EVs by $\mathcal{V}$. We assume that a vehicle $v \in \mathcal{V}$ has a battery capacity of $\left[e_{min}^v, e_{max}^v\right]$, with its SoC at time t denoted as e_t^v. Each vehicle is available to charge between their arrival and departure time slots $\left[t_{start}^v, t_{end}^v\right]$; where t_{end}^v is unknown to the solver, due to our stochastic problem formulation. Instead, the EV user provides a window of departure, $\mathcal{T}_{end}^v$. A user arrives with a state of charge (SoC) of $e_{t_{start}^v}^v$ and requires a SoC of e_{req}^v at departure, out of which only $e_{t_{start}^v}^v$ is known at the time of arrival. The SoC a user departs with may differ from their requested value, e_{req}^v, and is denoted as $e_{t_{end}^v}^v$. We later show how we compensate the users if their requirements are not met. Our EV partner's vehicles support bidirectional operations (charging and discharging), and the problem can be modified to serve only unidirectional (charging only) settings as well. The charging rate of an EV is represented as c_t^v. The power used to charge the EV over an aggregate time slot is represented as $c_{t_j^\pi}^v = \sum_{i=1}^{\tau} c_{t_i}^v$

Building's power consumption: We denote the building's energy consumption using $\mathcal{B}^e = \{b_1^e, b_2^e, \cdots, b_{|\mathcal{T}|}^e\}$ in kWh. It is used to compute the energy charge by the power company. The power draw during peak hours is represented by $\mathcal{B}^p = \{b_0^p, b_1^p, \cdots, b_\Omega^p\}$, at each aggregate time slot t_j^τ, which covers $t_i, \cdots, t_{i+\tau}$ time slots, $b_j^p = 1/\tau \cdot \sum_{i=1}^{\tau} (b_i^e / \delta_t)$ and is used to compute the demand cost.

Charging: We incorporate heterogeneity of chargers, i.e., chargers can have different control modes (i.e., *controlled* or *uncontrolled*), directionality (i.e., *unidirectional* or *bidirectional*), and maximum rate (e.g., 10 kWh, 20 kWh). Controlled chargers can start and stop charging at any time if instructed to do so, while uncontrolled chargers must charge continuously when in use until the battery is full. Unidirectional chargers can only provide charge to a vehicle, and bidirectional chargers can both charge and discharge, facilitating V2B charging. Any combination of these attributes defines a specific type of charger (e.g., a 20kWh controlled unidirectional charger). We represent the set of all available charger types by the set $\mathcal{K}$, with a total of N chargers. For each charger type $k \in \mathcal{K}$, there are $|k|$ chargers available, i.e., $\sum |k| = N$. We assume that each charger uniformly charges an EV v during each time slot t, based on a linear charging profile, which as shown by Sundström and Binding [30] does not affect optimization procedures considerably. Later, we perform an ablation experiment where we consider a piecewise charging curve, and compare against it.

Charging rate: We define charging rate as the amount of power delivered to the EVs at each time slot, measured in kWh. We assume that the EVs can charge and discharge at the maximum rate supported by the charger, which ranges from $[q_k^{min}, q_k^{max}]\ \forall k \in \mathcal{K}$, where q_k^{min} can be negative if the charger is bidirectional, denoting the discharging of a connected EV. A charger's efficiency is denoted by η, which is applicable to both charging and discharging. We also maintain an EV to charger type occupancy function $\zeta : \mathcal{V} \times \mathcal{T} \rightarrow C$, where $\zeta^v(t) = k_i$, representing the connection of EV v to a charger of type k_i at time t. Once a charger is assigned to an EV, it cannot switch chargers until departure.

Peak power use: We denote the peak power use over an aggregate time slot t_j^τ by π_j. Recall that an aggregated slot t_j^τ consists of smaller slots $\{t_i, t_{i+1}, ..., t_{i+\tau}\}$; the peak power is computed by averaging the power draw (sum of power used by the building and the chargers) across each aggregate time slot t_j^τ, $\pi_j = b_j^p + \sum_{i=1}^{N} c_{t_j^\pi}^v$. Thus, peak (maximum) power used during the peak hours is $P^{max} = \max(\pi_j)$.

Demand Cost: Demand cost is levied by electricity suppliers based on a customer's peak rate of electricity consumption (power), typically measured in kilowatts (kW), over a specific period, usually a month. This charge is separate from the energy cost, which is based on the total amount of energy used (kWh). It is the product of the peak power in an aggregate time slot and the demand charge, and is computed as $w^d \cdot P^{max}$. It is difficult for models to optimize and plan for demand costs over longer billing periods due to the uncertainty and inflexibility of its assessment. Thus, demand costs need to be calculated non-myopically.

Energy cost: The total energy used in time slot s is the sum of the energy used in recharging the vehicles and meeting the building's energy requirement. We denote the energy cost for the time slot using $g_t = w_t \cdot \sum_{v \in \mathcal{V}} (c_t^v + b_t^e)$. By summing this across all slots over the billing period, we get the total energy usage cost. During time slots with high electricity prices, we can effectively reduce the overall bill while meeting the building's energy needs by discharging connected EVs. These EVs are then charged at a later time when the electricity price is lower.

Missing SoC cost: The energy shortfall between required and actual SoC at departure for each EV is a key metric in the V2B problem. In our setting, since the departure of each EV is unknown, this metric is important to understand how well the user's requirements are met. We assign a monetary value to the missing energy, at the rate of w^s, in $/missing kWh.

Total Bill: The total bill over the billing period is the sum of demand cost and energy cost. The intuitive strategy is to minimize the peak power over the billing period, which reduces the demand cost and ensures that the energy cost during peak hours remains the minimum required to meet the constraints of charging the EVs to the required SoC level. It is represented as:

$$\sum_{t \in \mathcal{T}} w_t^e \cdot (b_t^e + c_t^v) + w^d \cdot P^{max} + \sum_{v \in \mathcal{V}} w^s \cdot |e^v_{t^v_{end}} - e^v_{req}| \quad (1)$$

Solution Space: We define a solution with an assignment of chargers to cars and the corresponding charging/discharging schedule. Let $\mathcal{H}$ be the set of solutions, where for each charging assignment (k, t), exactly one EV, $v \in \mathcal{V}$ is assigned, such that $\langle v, k, t \rangle \in \mathcal{H}$, as is the case in practical usage, where one EV is attached to only charger and not switched around. For each assignment $h_{v,k,t} \in \mathcal{H}$, we assign a charging rate $c_t^v \in C$. We consistently use the assumption that the EV remains at the charger for the entire duration of its stay. We assign cars on a *first-come first-serve basis*, where we prioritize assigning cars based on their time of arrival and chargers by their rate of charging, directionality, and controllability. For example, one priority list could be assigning cars to 20kWh bidirectional, 20kWh unidirectional (all controlled), and then the uncontrolled chargers. We can accommodate only as many cars as available chargers. Excess cars that arrive when there are no vacant chargers, will not charge for the remainder of their stay.

3.2 Markov Decision Process (MDP)

We model the V2B problem as a Markov Decision Process (MDP), building on prior work by Shi and Wong [24]. An MDP is characterized by a 4-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho\}$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of possible actions, $\mathcal{P}$ is a state-action transition model, and ρ is a reward function which captures the agent's utility (or cost) [9].

Decision Epoch: A decision epoch occurs at every discrete decision-making event, $t \in \mathcal{T}$. Between events, the environment moves in continuous time where chargers charge or discharge EVs. At each decision epoch, the decision-maker takes an action that moves the state from a pre-decision state to a post-decision state and is then given an immediate reward.

State: We denote the set of states as $\mathcal{S} \in \{\{b_t^e\}, \hat{p}^{max}, \{q_k^{min}, q_k^{max}\} \forall k \in \mathcal{K}, \{e_t^v, t_{start}^v, \hat{t^v}_{end}, e_{min}^v, e_{max}^v\} \forall v \in \mathcal{V}\}$ and a state at time slot t is represented as s_t at pre-decision time. It includes the current building load (b_t^e), all charger rates (q_k^{min}, q_k^{max}), EV details ($e_t^v, t_{start}^v, \hat{t^v}_{end}, e_{min}^v, e_{max}^v$), where $\hat{t^v}_{end}$ is the estimated departure time from user's departure time window $\mathcal{T}_{end}^v$. We also include an estimate of the peak power $\hat{p}^{max}$ to assist in making better decisions.

Actions: We denote the set of all feasible actions at time slot t by $\mathcal{A}_t$. An action in $\mathcal{A}_t$ corresponds to a combination of charging rates for all chargers $\mathcal{K}$. Charging rates are limited by the total power consumed by the building, including the power provided for charging the vehicles, i.e., the EVs cannot be discharged more than the building's current power usage. They are limited to discrete values between the maximum and minimum allowed charging rates.

State Transitions: At time slot t, the decision-maker can take an action that results in a transition from a state s to s'. In this transition, the system evolves through many different stochastic processes. First, EVs can arrive or depart out of their regular arrival and departure times, and are governed by some duration of stay distribution. Second, electricity prices may change with minimal warning, as may happen during *emergency load reduction programs* (ELRP) events wherein consumers receive financial incentives for reducing their energy consumption. Finally, the building's power draw may change depending on the time of day, month of year, or even weather. While these follow a predicted distribution, they still introduce stochasticity in state transitions. We omit detailed discussion of the mathematical model and expressions for temporal transitions and state transition probabilities, as our framework relies solely on a generative world model rather than explicit estimates of these transitions.

Rewards: Rewards in an MDP often have two components: a lump sum immediate reward for taking actions and a continuous time reward as the process evolves, and are highly domain-dependent. For charger optimization, we are concerned about minimizing the overall energy usage cost while meeting a certain quality of service for users. We denote the reward function by $\rho(s, a)$, for taking action a at state s. The reward is both intermediate r_t^i and episodic r_t^e. We use episodic rewards during the rollouts. The intermediate reward is $r_t^i = g_t + w^s \cdot \sum_{v \in \mathcal{V}} \left| e^v_{t^v_{end}} - e^v_{t^v} \right|$ if $t = t^v_{end}$, else, $r_t^i = g_t$. The episodic reward r_t^e is calculated according to equation (1), with a minor modification to the demand cost calculation which uses an estimate of the peak power, $\hat{p}^{max}$. It is as follows:

$$r_t^e = \sum_{t \in \mathcal{T}} w_t^e \cdot (b_t^e + c_t^v) + w^d \cdot \hat{p}^{max} + \sum_{v \in \mathcal{V}} w^s \cdot |e^v_{t^v_{end}} - e^v_{req}| \quad (2)$$

4 Online Approach

We propose an online approach for managing charging controls. First, we use a heuristic EV assignment policy, such as *first-come, first-serve* and bidirectional-first policies, to assign EVs to chargers while considering fairness and the peak shaving capability of bidirectional chargers. To address uncertainty in EV arrival and departure, we estimate future system states by sampling data and utilizing an offline solver to derive optimal actions and establish upper bounds on performance metrics.

To adapt to environmental uncertainty, we employ an online MCTS search for dynamic and robust decision-making. Recognizing the challenge of MCTS runtime in real-time scenarios, we incorporate domain-knowledge guidance (DG-MCTS) to shrink and adjust the action space using heuristic actions and demand charge predictions, improving exploration efficiency. Our online DG-MCTS solver constructs a forward-looking search tree using episodic data provided by our EV partner partners. To reduce computational complexity, we decompose long billing periods (monthly) into shorter, manageable daily planning horizons. Algorithm 1 provides an overview of the workflow of our DG-MCTS approach. In the following section, we detail each component of the approach, including EV assignment, future state estimation, and the integration of domain-knowledge-guided exploration into MCTS.

4.1 Episode Sampling and Handling Uncertainty

Episode samples: We collected real-world data from EV partner's research laboratory and we applied a Poisson distribution based on historical data.

Car to charger assignment: We assign EVs to chargers based on a *first-come-first-serve* policy based on the arrival time. If multiple cars arrive in the same time slot, we break ties by ordering them according to e^v_{req}. EVs are assigned to chargers based on the charger's directionality, maximum rate, and controllability.

Uncertainty in departure times: Given that each user only provides a departure window $\mathcal{T}^v_{end}$, our goal is to estimate the departure time within this interval. Since $\mathcal{T}^v_{end}$ represents a range of possible departure times rather than a fixed point, we model this uncertainty by treating the departure time as a random variable within $\mathcal{T}^v_{end}$. We draw samples from a uniform distribution over the interval $\mathcal{T}^v_{end}$ to approximate a representative departure time.

Let $\hat{t}^v_{end}$ denote the estimated departure time for user v as a realization of a uniformly distributed random variable over $\mathcal{T}^v_{end}$. We sample $\hat{t}^v_{end} \sim \mathcal{U}(\min(\mathcal{T}^v_{end}), \max(\mathcal{T}^v_{end}))$ where $\min(\mathcal{T}^v_{end})$ and $\max(\mathcal{T}^v_{end})$ are the lower and upper bounds of the departure window $\mathcal{T}^v_{end}$, respectively. Sampling in this manner allows us to select a feasible departure time that is unbiased with respect to any specific point within the interval, ensuring a fair estimation across the entire window. This serves as a proxy for user departure behavior.

Estimating peak power threshold: For an episode with *fixed* trajectory of EV arrivals and departures, we determine the optimal charging decisions by solving a MILP. This optimization yields a sequence of charging actions that minimizes costs while meeting the EV charging requirements within the given arrival and departure times. We use this same process to estimate the peak power threshold needed for each month. We solve a set of sampled episodes, which are distinct from the evaluation data, and calculate the peak power draw based on the optimal actions. We take the 99% confidence level of the peak power across these samples and we obtain a robust estimate of the peak power demand for that month. Using this confidence-based peak power estimate allows us to integrate demand charge considerations effectively within the sampling process, providing a realistic target that aligns with optimal charging outcomes observed in the MILP solutions.

4.2 Handling Exponential Action Space

A feasible action in our problem corresponds to a set of charger rates for all chargers, given that chargers can be heterogeneous with the possibility of turning off, charging, and discharging a connected EV. Additionally, the sum of charger rates must be constrained to the current building load at that time. As a result, for a building with a large number of chargers with varying configurations, the possible actions for a given state are combinatorially large, N^p if we consider p discrete actions for each charger. Such an action space is infeasible to explore in an online setting. To address this challenge, we introduce a heuristic that enables us to identify promising actions from the set of feasible actions.

Least Laxity First heuristic: One of our goals is to charge EVs such that they leave with their desired SoC level. To guide the charging decisions, we utilize the least laxity first (LLF) heuristic [36]. This heuristic prioritizes EVs with the least remaining time until departure, ensuring that vehicles close to their departure window receive priority in charging allocation. At each decision epoch, we compute the available power gap at the current state s and time t as power gap = $\hat{P}^{\max} - b^p_t$, where $\hat{P}^{\max}$ is the estimated peak power threshold and b^p_t is the current building load. The trickle charging rate for each EV v is defined as trickle rate = $(e^v_{\text{req}} - e^v_t)/(\hat{t}^v_{\text{end}} - t)$, where e^v_{req} is the required energy, e^v_t is the current energy, and $\hat{t}^v_{\text{end}}$ is the estimated departure time. We then calculate the sum of trickle rates for all EVs at the current time slot. If this sum is less than the available power gap, we have the capacity for overcharging. In such cases, we set each EV's charging rate to its trickle rate and assign additional charging to EVs with bidirectional chargers, following a reverse order of laxity (least laxity first) until the power gap is fully utilized. Conversely, if the sum of trickle rates exceeds the power gap (indicating limited capacity), we discharge EVs connected to bidirectional chargers, also based on reverse laxity (time remaining to estimated departure), to fill the negative gap before resuming trickle charging.

We improve the promising actions by introducing intuitive actions based on two parameters, the power gap and the bounds of feasible actions. If the available power gap is *positive*, indicating surplus capacity, we add discrete incremental charging actions to explore overcharging options. Otherwise, if the power gap is negative (where the building load exceeds the estimated peak power), additional discharging actions are included to mitigate the excess demand, we refer to these additional actions as β. Finally, we add an action between the minimum chosen action and the minimum feasible charging rate, $q^{\min}_k$, ensuring that it does not go below $e^v_{\min}$, the minimum state of charge required by the EV. Similarly, we add another action between the maximum chosen action, and the maximum feasible charging rate, $q^{\max}_k$, ensuring that it does not exceed $e^v_{\max}$, the EV's maximum allowable state of charge, we refer to this change in action-space as "offset".

This LLF-guided action serves a dual purpose: it enhances traceability by narrowing the search space and accelerating decision-making. This keeps the search process computational fast. By prioritizing actions based on current states, including the urgency of each EV, the LLF heuristic enables us to adapt charging rates quickly, focusing exploration on the most relevant actions.

Temporal decomposition: We decompose the long billing period (typically a month) into shorter, computationally manageable planning horizons, such as a day. However, this decomposition poses a significant challenge: the demand charge can only be accurately calculated at the end of the full billing period, i.e., over the longer planning horizon. To address this, we leverage two essential properties of the V2B Markov Decision Process (MDP).

First, note that the arrival and departure times of EVs are independent of the agents' charging actions—EVs arrive and depart from

Algorithm 1 Domain-knowledge Guided MCTS (DG-MCTS)

Input: Current State $\mathcal{S}_t$, iterations $\mathcal{I}$, exploration range β
Output: Best action $[\mathcal{A}^*_k$ for $k \in K]$,

$D \leftarrow$ EstimateDemandCharge$(\mathcal{S}_t)$ `// Estimate demand charge`
$[\mathcal{A}'_k$ for $k \in K] \leftarrow$ LLFHeuristic$(\mathcal{S}_t, D)$ `// Get heuristic actions`
`// Action Pruning`
if $BuildingLoad < D$ **then**
  foreach $k \in K$ **do** `// Encourage Charging`
    $Space(\mathcal{A}_k) \leftarrow$ GetNeighbors$([\mathcal{A}'_k - \beta, \mathcal{A}'_k + \beta + \text{offset}] \cup \text{NearBoundaryActions})$
else
  foreach $k \in K$ **do** `// Encourage Discharging`
    $Space(\mathcal{A}_k) \leftarrow$ GetNeighbors$([\mathcal{A}'_k - \beta - \text{offset}, \mathcal{A}'_k + \beta] \cup \text{NearBoundaryActions})$
$ActionSpace \leftarrow [Space(\mathcal{A}_K).Clip[q_k^{min}, q_k^{max}]$ for $k \in K]$
for $n \leftarrow 1$ *to* $\mathcal{I}$ **do** `// Establish tree`
  $Sample \leftarrow$ GenerateSample(EVDep, EVArr, BuildingLoad)
  $\mathcal{S}'_{t+1} \leftarrow$ SelectFrom$(\mathcal{S}_t, ActionSpace, Sample)$
  $\mathcal{S}'_{t+2} \leftarrow$ Expand$(\mathcal{S}'_{t+1}, Sample)$ `// Add new child node`
  $v \leftarrow$ Simulate$(\mathcal{S}'_{t+2}, Sample, \text{TrickleRate})$ `// Rollout`
  Backpropagate$(\mathcal{S}'_{t+2}, v)$ `// Update tree stats`
$\mathcal{A}^* \leftarrow$ BestAction$(\mathcal{S}_t)$ `// Select action with highest value`

the building regardless of how existing vehicles are charged. This independence allows us to *pre-sample* trajectories of EV arrivals and departures using a generative model based on historical data, without conditioning the sampling on the actions taken within the search tree.

Second, we leverage the estimated peak power threshold, as described earlier. Equipped with this peak power estimate, we introduce the core concept enabling temporal decomposition: divide the full planning horizon into smaller periods, with the added constraint that the total power consumed in each period remains below the estimated demand charge for a fixed sampled trajectory.

At the beginning of each day (the smaller decomposed planning horizon), we sample multiple trajectories of EV arrivals and departures from the generative model. For each trajectory, the trained model f estimates the demand charge over the longer planning horizon (e.g., a month). This estimated charge is then integrated and used as part of the episodic reward, within the search tree, which operates on the shorter daily planning horizon.

4.3 Monte Carlo Tree Search (MCTS) Evaluation

Offline approaches to solving the V2B problem such as MILP, fail to consider the stochasticisity present in the real world. Instead, they rely on complete knowledge of the system to optimally select the best actions. This motivates us to use MCTS, an anytime algorithm that has been widely used in game-playing scenarios [25].

MCTS models planning as a tree with states as nodes and actions as edges. It explores the tree asymmetrically, favoring promising actions with the Upper Confidence bounds applied to Trees (UCT) algorithm [10] by balancing exploitation and exploration. Node values are estimated through *rollouts* using a simple default policy, often random action selection. As the search progresses, node value estimates improve. This approach enables efficient exploration of large action spaces. MCTS requires a generative environment model, a tree policy for navigation, and a default policy for node value estimation.

We use the collected historical data to sample new episodes as the tree is built into the future. We use the standard Upper Confidence bound for Trees (UCT) [11] to navigate the search tree and decide which nodes to expand. When expanding a node we sample promising actions for the given state. When working outside the MCTS tree to estimate the value of an action during rollout, we rely on a default policy. This policy is simulated up to a time horizon and the utility is propagated up the tree. Our default policy is a trickle charging rate policy – which charges each car with the required energy to meet the required SoC by the estimated departure time.

Root parallelization: Given that EV arrivals and departures and building power draw, even when following a known distribution, are highly uncertain in time, sampling one episode may not represent actual future EV behavior. We handle this using *root parallelization*, which involves sampling many episodes, and instantiating a new MCTS tree for each with their EV arrivals/departures and building power draw as the root node. Each tree is explored in parallel, and after execution, the score for each of the actions from the common root node is averaged across the trees. The action with the highest average score across all trees is then the selected action.

4.3.1 Centralized MCTS. In a centralized multi-agent approach, a single search tree represents the combined decision-making space of all agents in the V2B system. Rather than each agent operating independently, this unified tree integrates the actions of all agents, allowing for joint optimization. Such an approach is advantageous in V2B settings, where decisions made by individual EVs impact overall building load and demand charges, requiring coordination to minimize costs. However, the centralized MCTS also faces computational challenges due to the high dimensionality of the action space and the extended planning horizon, as it must simultaneously consider all agents' actions at each decision point. We show the process in Algorithm 1.

Centralized MCTS explicitly considers the interactions among agents at each step, which is particularly advantageous in scenarios where joint coordination is necessary. For example, centralized MCTS can directly incorporate the influence of one EV's charging actions on the demand charge for all EVs and the building. However, centralized MCTS must address two primary challenges: **(1)** efficiently navigating the large action space within a single tree structure and **(2)** managing the long temporal horizon associated with the V2B optimization problem. We tackle these challenges by leveraging domain-specific heuristics, which streamline the exploration of the action space and introduce strategic planning over shorter horizons within the MCTS framework, and name it Domain-knowledge Guided MCTS (DG-MCTS).

4.3.2 Decentralized MCTS. While the MCTS algorithm can directly tackle the large state space, it still suffers from the dimensionality of the action space and long time horizon. Thus, a standard MCTS-based approach may not always be suitable, especially as the number of cars grow. This, we also introduce a decentralized approach to multi-agent MCTS [5, 20]. The key idea of decentralization in multi-agent MCTS is simple—instead of developing a monolithic tree for all the agents, each agent develops its own search tree to compute a near-optimal action for itself. Decentralization massively reduces the search space within the tree, thereby providing scalability. We present the Algorithm 2 in the Appendix.

Operationalizing decentralized multi-agent MCTS (dMCTS) poses two main challenges: **(1)** As an agent explores its decision space, it must account for the actions of other agents. Certain approaches learn a computationally cheap proxy for the behavior of other agents and invoke the proxy repeatedly within the search tree [20]. The search tree for each agent can then be generated in parallel. However, this approach does not work for our setting as using a proxy for other agents' actions can lead to potentially very high demand costs. **(2)** Decentralization of the action space does not inherently solve the long horizon problem. Thus, we propose a second heuristic that leverages the structure of the V2B problem to tackle these challenges:

Criticality score: We order the agents by computing the criticality of the decision faced by each agent. Consider a fixed amount of energy that the cars can draw collectively to minimize energy costs in the long run; we argue that it is natural to prioritize agents that must meet higher charge requirements within the least amount of time. Specifically, we compute a criticality score using Least Laxity First [36], $c_{score} = (t^v_{end} - t) - (e^v_{req} - e^v_t)/q^{max}_{\zeta^v(t)}$ for each connected EV $v \in \mathcal{V}$, where, $\zeta^v(t)$ is the maximum charging rate of the charger v is connected to. The EV parameters are the required soc e^v_{req}, the current soc e^v_t, the expected time slot to leave t^v_{end}, and the current time slot t. Once the most critical agent's (EV) action is computed, the consumed power is simply added to the total power usage, and this process continues for all the agents in the order of priority.

5 Experiments and Analysis

5.1 Setting

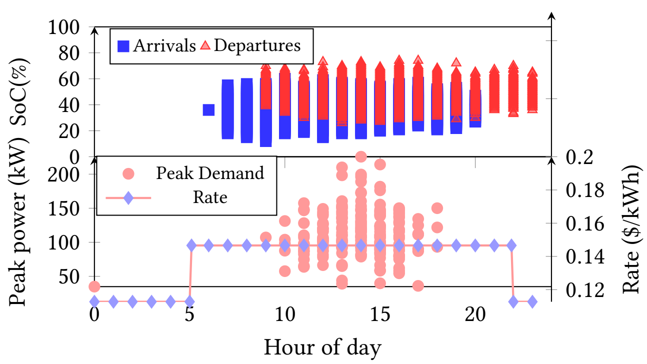

Figure 1: (Top) Distribution of EV arrival and departure hours against arrival and required SoC upon departure across 8 months. (Bottom) Distribution of peak building power draw against the hours of day it was drawn and the TOU rates across the day.

Data Collection To assess the effectiveness of our proposed method, we utilize data from our EV partner research laboratory. We compare our approach with various baseline models, focusing on total billing costs and peak shaving (demand charge reductions), along with minimizing missing SoC. We focus on only optimizing weekdays since there are almost no employees working during the weekends. Additionally, power utilities tend to keep weekends out of the demand charge calculation. Silicon Valley Power in California does not consider Sunday for demand charge calculations. In our setting, we have 10, 20 kWh unidirectional chargers, and 5 bidirectional chargers that support charging at 20 kWh as well as discharging at 20 kWh (-20, kWh to 20 kWh). All the EVs considered have bidirectional charging capability.

We collected real-world data from EV partner's research laboratory in Santa Clara, California, covering building power consumption, EV charger usage, and EV telemetry over a nine-month period from May 2023 to January 2024. To model distributions of EV arrivals, SoC requirements, and building power fluctuations, we applied a Poisson distribution based on historical data. The number of EVs arriving at the office on weekdays fluctuates daily, highlighting inherent uncertainties. Figure 1 shows the arrival and departure times in relation to SoC, along with the distribution of peak power demand and corresponding hours. We sampled 110 monthly billing episodes for each month from May 2023 to December 2023.

Estimated Peak Power. To improve action effectiveness, we account for varying weekday conditions by incorporating a monthly peak power estimate for each episode, based on optimal action sequences generated by the MILP solver. We use the lower bound of the 99% confidence interval from the MILP data as a conservative estimate of the demand charge. This input feature is further optimized during RL training.

Hyperparameter Tuning To optimize the performance of our DG-MCTS framework, we utilized a state-of-the-art hyperparameter optimization library, Optuna, which employs efficient sampling strategies to explore the hyperparameter space. Optuna's objective was to minimize a custom-defined score, calculated according to equation (1). The hyperparameter search space included key parameters such as the number of iterations, maximum depth, penalties for unmet SoC requirements and exceeding power gaps, and rewards for achieving specific SoC targets. Additionally, we explored regularization parameters like C and reward discounting factor γ, and a tolerance parameter for estimated peak power (ϵ). Optuna's trial-based approach generated a hyperparameter importance graph as shown in Appendix in Figure 3, revealing the most influential factors affecting performance, and guiding subsequent model refinement. We set the parameters as shown in Table 8 in the Appendix.

Baseline approaches. We evaluate the performance of our online approach by comparing it against various methods including a real-world charging procedure (baseline approach), several proposed heuristic approaches, and a reinforcement learning-based policy. We provide a brief description of the baselines here.

Hardware used. All the experiments were performed and timed on a 32-core 4.5 GHz machine with 128 GB of RAM.

- **MaxCharge**: This approach simulates current real-world charging procedures, charging all connected EVs at the fastest rate to e^v_{max}.
- **ReqCharge**: Similar to *MaxCharge*, however only charges all connected EVs as quickly as possible to e^v_{req}.
- **Least Laxity First (LLF)**: Uses the same heuristic policy used in Section 4.2. It charges beyond e^v_{req} if possible and then leverages

Table 1: Monthly Total Cost (lower is better).

Policy	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
DG-MCTS	5466.96 ± 24.4	**6032.16 ± 88.8**	6021.98 ± 42.8	**8512.24 ± 81.4**	**6357.79** ± 36.8	**6744.54 ± 75.7**	**5806.73 ± 79.4**	**5195.53 ± 127.1**
dMCTS	5534.54 ± 84.2	6050.26 ± 60.0	6123.93 ± 73.1	8550.6 ± 53.3	6467.52 ± 75.9	6819.85 ± 62.6	5849.69 ± 56.9	5317.49 ± 157.0
RL	**5416.3 ± 32.8**	6067.6 ± 152.1	**5913.83 ± 21.4**	8571.87 ± 87.9	6403.82 ± 105.3	6852.93 ± 129.3	5898.55 ± 71.9	5432.92 ± 135.0
LLF	5515.86 ± 30.7	6068.35 ± 45.0	6045.75 ± 37.1	8637.07 ± 44.9	6364.37 ± 37.0	6802.74 ± 48.8	5831.15 ± 33.1	5245.1 ± 140.6
EDF	5521.67 ± 38.32	6076.35 ± 58.8	6047.83 ± 37.6	8637.67 ± 45.2	6369.2 ± 41.1	6810.84 ± 51.8	5831.53 ± 33.4	5292.71 ± 152.6
ReqCharge	5577.22 ± 43.8	6147.51 ± 47.1	6123.04 ± 35.0	8743.39 ± 53.7	6465.7 ± 46.0	6852.65 ± 56.9	5939.52 ± 42.2	5254.94 ± 65.5
MaxCharge	6827.5 ± 188.6	7710.79 ± 228.91	7577.06 ± 207.8	9402.73 ± 144.4	8259.19 ± 249.7	8348.78 ± 204.7	7081.51 ± 184.7	7888.98 ± 291.8

Table 2: Missing SoC by Policy (lower is better).

Policy	DG-MCTS	dMCTS	RL	LLF	EDF	ReqCharge	MaxCharge
Mean ± Std	113.62 ± 83.86	164.6 ± 174.24	141.92 ± 58.17	59.66 ± 84.13	58.86 ± 82.29	84.87 ± 80.73	**27.56 ± 60.55**

Table 3: Cars with Missing SoC by Policy (lower is better).

Policy	DG-MCTS	dMCTS	RL	LLF	EDF	ReqCharge	MaxCharge
Mean ± Std	43.82 ± 30.98	73.16 ± 52.88	207.34 ± 60.29	40.47 ± 30.67	41.04 ± 31.36	80.57 ± 37.85	**4.61 ± 10.22**

Table 4: Peak Shaving across all months (lower is better)

Policy	DG-MCTS	dMCTS	RL	LLF	EDF	ReqCharge	MaxCharge
Mean ± Std	-5.17 ± 13.67	3.14 ± 11.80	18.16 ± 22.81	1.77 ± 9.02	2.77 ± 10.52	11.22 ± 6.15	94.02 ± 46.60

excess energy to reduce peak power demand. Otherwise, it uses trickle charging to charge EVs to e^v_{req}.

- **Early Deadline First (EDF)**: EDF prioritizes EVs which are estimated to depart soon. It follows the Early Deadline First approach, which was originally designed as a dynamic scheduling algorithm for real-time systems [26]. Similar to LLF, this policy will try to charge to excess in anticipation of high building demand. Given that it can discharge the EV prior to departure. Otherwise, it use trickle charging to e^v_{req}.
- **Reinforcement Learning (RL)**: We apply the Deep Deterministic Policy Gradient (DDPG) algorithm to train a model that manages charging actions. The state representation includes features such as current time, building load, charging status, the current SoC of EVs, and their expected departure times. The reward function is defined as the sum of the demand cost, energy cost, and a penalty for deviations from the required SoC after each action. The DDPG model employs a two-layer Multi-Layer Perceptron (MLP) architecture for both the actor and critic networks, with 96 neurons per layer. Action masking refines policy outputs by setting charging rates to zero when no EV is connected, ensuring charging meets the required SoC before the estimated departure time, and discharging when the SoC exceeds the required level. During training, a policy guidance approach incorporates optimal actions generated by an MILP solver into the training data buffer to improve policy performance. A separate model is trained for each month, using 60 simulated samples per month. The detailed hyperparameters of the RL model are provided in Table 7 in the appendix.

5.2 Results

We evaluate all approaches using four metrics:

(1) **Peak Shaving**: It is the difference in demand charge between (i) the building's power usage (without any charging) and (ii) by adding charging the EVs under the respective policies. Positive values indicate the policy reduced the demand charge by controlling the charging actions.
(2) **Missing SoC**: The energy deficit cars have when they depart, compared to their required SoC. Naturally, a deficit in the required state of charge results in customer dissatisfaction. For our experiments, we set this penalty to be 20 cents per kWh, a value that is 42% higher than the cost of buying the same volume of energy from the grid. This penalty means that the building can not naively exploit vehicles to save buying energy from the grid.
(3) **Cars under required SoC**: The number of cars with missing SoC helps us understand if the model is trying to achieve the required SoC for all cars, or prioritizing some over the rest.
(4) **Total cost**: The sum of electricity cost, demand charge, and missing SoC cost over the billing period, computed by Eq. (1).

We assess the performance of our online approach on data from May 2023 to December 2023. We randomly selected 10 episodes

Table 5: Ablation Study Results (lower is better)

Month	DG-MCTS	MCTS/P	MCTS/C	MCTS/H
Aug	**8512.24 ± 81.42**	8690.19 ± 86.53	8521.80 ± 85.13	8705.76 ± 63.60

Table 6: Total Cost of policies under unexpected increases in daily vehicle arrivals (lower is better)

Month	DG-MCTS	dMCTS	LLF	EDF	ReqCharge	MaxCharge
Aug	**8589.0 ± 74.19**	8650.75 ± 64.98	8691.76 ± 53.1	8696.32 ± 57.44	10345 ± 100	11395.53 ± 432.54

as our test set and then used the remaining 100 episodes as exploration samples during root parallelization. All policies were not aware of the actual departure time of any vehicle, instead relying on a window of potential departure times provided by each user. Table 1 compares the monthly bill over eight months across different policies. Centralized MCTS (MCTS) outperforms all other heuristics and the decentralized MCTS (dMCTS) approach in six of the eight months. It is consistently cheaper compared to the two existing baseline approaches, ReqCharge and MaxCharge, by a significant margin. Our approach also beats smart heuristics in all of the months. Additionally, dMCTS takes 15.38 seconds on average per decision, while DG-MCTS takes 23.75 seconds per decision.

RL-based policies were able to outperform MCTS in two out of the eight months. However, this is due to the policy not meeting the SoC requirements of the EVs upon departure. This discrepancy in SoC requirements is shown in Table 3 and Table 2. RL is unable to meet the EV SoC requirements for 207 cars on average, due to its inability to correctly select actions under uncertainty. While our approach misses more SoC across several episodes, shown in Table 2, when compared to the heuristics, it performs similarly in the total number of EVs that failed to be charged. The results highlight the capability of our approach to generate optimal actions under uncertainty. While the heuristics show significant improvements over the baselines, they lack the ability to estimate potential events in the future. Thus, limiting their performance. Note that even when simply charging the cars at a max rate upon connection, there are still instances of missing SoC due to earlier departure than expected.

We show also in Table 2, that while our approach has comparable performance to heuristics in terms of missed SoC, it can do more peak shaving compared to all other approaches.

Robustness to uncertainty. Finally, we show that our approach outperforms all heuristics when we further increase the degree of uncertainty by: **1)** Increasing the potential departure window as shown in Table 5 and **2)** Handling an unexpected increase in the number of daily EV arrivals, where we increase the number of daily cars to be around 25 per day, as demonstrated in Table 6. MCTS tackles changes in the environment better than the rest of the policies. Under the increased action space, dMCTS achieves results much faster, taking an average of 25.46 seconds per decision, while DG-MCTS takes 48.97 seconds. This highlights the capability of dMCTS to scale much better while making better decisions than other methods.

5.3 Ablation Study

We evaluate the contribution of key techniques in our approach through ablation. For ablation, we only selected the month with the highest building peak, August 2023, and tested the performance on the total bill. The ablations explored are: **1) MCTS/P**, solving episodes without using predicted peak demand, instead using only the current peak demand throughout the day. 2) **MCTS/H**, not leveraging any heuristic to narrow the feasible action space. **3) MCTS/C**, using piecewise linear battery profiles instead of linear assumptions. We present the average total bill for August 2023 in Table 5. Next, we discuss the significance of each ablated feature.

Peak demand prediction. We examine **MCTS/P**, where peak demand prediction is removed in lieu of using the current power demand as the threshold, and the performance drops drastically. This shows the importance of having an accurate model for predicting peak demand for any length of the planning horizon.

Heuristics for action pruning. The **MCTS/H** approach, which removes action pruning via heuristics, results in decreased performance. This highlights the dependency of MCTS on the quality of the generated trees. By removing action pruning, the action space increases and the makes it difficult to choose "good" actions. Increasing iterations may improve solutions, while also increasing computation time.

Piecewise linear charging profiles Executing charger actions on piecewise linear battery profiles, **MCTS/C**, compared to linear charging profiles, offer practically identical solutions at the cost of increased complexity. Additionally, most EV manufacturers do not release their battery charging profiles, making it challenging to understand the SoC curve. While piecewise linear profiles can potentially offer increased accuracy, not having it does not detract from the correctness of the approach [30].

6 Conclusion

This paper proposes a Domain-Knowledge Guided Monte Carlo Tree Search (DG-MCTS) approach to address V2B challenges in smart buildings by optimizing charging control. In online decision-making scenarios, we encounter challenges such as the stochastic nature of EV arrivals and departures and limited decision-making time. To address these challenges, DG-MCTS leverages domain-specific heuristics to guide action selection and prune the action space, enabling it to outperform traditional heuristic algorithms in terms of cost reduction and demand charge optimization within a reasonable computation time. While DG-MCTS demonstrates superior performance, we acknowledge that decentralized MCTS (dMCTS) scales more efficiently in scenarios with a larger number of chargers. We evaluate both approaches using simulated V2B scenarios with real-world data from an EV manufacturer and smart building with 15 chargers. Results show that DG-MCTS effectively handles the uncertainty in EV departure times while achieving the lowest monthly total costs and meeting SoC requirements, demonstrating its capability in managing online EV charging under dynamic conditions. Meanwhile, dMCTS provides a scalable alternative that performs well in high-dimensional multi-agent settings.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul W. Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. 2020. A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges. *Inf. Fusion* 76 (2020), 243–297. https://api.semanticscholar.org/CorpusID:226307260

[2] Heba M. Abdullah, Adel Gastli, and Lazhar Ben-Brahim. 2021. Reinforcement Learning Based EV Charging Management Systems–A Review. *IEEE Access* 9 (2021), 41506–41531. https://api.semanticscholar.org/CorpusID:232374491

[3] Omid Ardakanian, Catherine Rosenberg, and S. Keshav. 2013. Distributed control of electric vehicle charging. In *Proceedings of the Fourth International Conference on Future Energy Systems* (Berkeley, California, USA) *(e-Energy '13)*. Association for Computing Machinery, New York, NY, USA, 101–112. https://doi.org/10.1145/2487166.2487178

[4] Aijun Bai, Feng Wu, and Xiaoping Chen. 2013. Bayesian Mixture Modelling and Inference based Thompson Sampling in Monte-Carlo Tree Search. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:27805544

[5] Daniel Claes, Frans Oliehoek, Hendrik Baier, and Karl Tuyls. 2017. Decentralised online planning for multi-robot warehouse commissioning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Vol. 1. ACM, 492–500.

[6] IBM ILOG Cplex. 2009. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.

[7] Leonardo de A. Bitencourt, Bruno S.M.C. Borba, Renan Silva Maciel, Márcio Zamboti Fortes, and Vitor Hugo Ferreira. 2017. Optimal EV charging and discharging control considering dynamic pricing. *2017 IEEE Manchester PowerTech* (2017), 1–6. https://api.semanticscholar.org/CorpusID:42059425

[8] Mahdi Ghafoori, Moatassem Abdallah, and Serena Kim. 2023. Electricity peak shaving for commercial buildings using machine learning and vehicle to building (V2B) system. *Applied Energy* 340 (2023), 121052.

[9] Mykel J. Kochenderfer. 2015. *Decision Making Under Uncertainty: Theory and Application.* The MIT Press. https://doi.org/10.7551/mitpress/10187.001.0001

[10] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[11] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning* (Berlin, Germany) *(ECML'06)*. Springer-Verlag, Berlin, Heidelberg, 282–293. https://doi.org/10.1007/11871842_29

[12] Yanqing Kuang, Mengqi Hu, Rui Dai, and Dong Yang. 2017. A collaborative decision model for electric vehicle to building integration. *Energy Procedia* 105 (2017), 2077–2082.

[13] Jae Hyun Lee, Eunjung Lee, and Jinho Kim. 2020. Electric Vehicle Charging and Discharging Algorithm Based on Reinforcement Learning with Data-Driven Approach in Dynamic Pricing Scheme. *Energies* (2020). https://api.semanticscholar.org/CorpusID:218814040

[14] Karol Lina López, Christian Gagné, and Marc-André Gardner. 2019. Demand-Side Management Using Deep Learning for Smart Charging of Electric Vehicles. *IEEE Transactions on Smart Grid* 10 (2019), 2683–2691. https://api.semanticscholar.org/CorpusID:117695642

[15] Simon M. M. Lucas, Spyridon Samothrakis, and Diego Perez Liebana. 2014. Fast Evolutionary Adaptation for Monte Carlo Tree Search. In *EvoApplications*. https://api.semanticscholar.org/CorpusID:18556700

[16] Jianing Luo, Yanping Yuan, Mahmood Mastani Joybari, and Xiaoling Cao. 2024. Development of a prediction-based scheduling control strategy with V2B mode for PV-building-EV integrated systems. *Renewable Energy* 224, C (2024). https://doi.org/10.1016/j.renene.2024.120

[17] Ajay Narayanan, Srinarayana Nagarathinam, Prasant Misra, and Arunchandar Vasan. 2024. Multi-agent Reinforcement Learning for Joint Control of EV-HVAC System with Vehicle-to-Building Supply. In *Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD)* (Bangalore, India) *(CODS-COMAD '24)*. Association for Computing Machinery, New York, NY, USA, 332–341. https://doi.org/10.1145/3632410.3632421

[18] Hoang Tien Nguyen and Dae-Hyun Choi. 2023. Distributionally Robust Model Predictive Control for Smart Electric Vehicle Charging Station With V2G/V2V Capability. *IEEE Transactions on Smart Grid* 14 (2023), 4621–4633. https://api.semanticscholar.org/CorpusID:257898708

[19] Björn Nykvist and Måns Nilsson. 2015. Rapidly falling costs of battery packs for electric vehicles. *Nature Climate Change* 5, 4 (2015), 329–332.

[20] Geoffrey Pettet, Ayan Mukhopadhyay, Mykel Kochenderfer, Yevgeniy Vorobeychik, and Abhishek Dubey. 2020. On algorithmic decision procedures in emergency response systems in smart and connected communities. *arXiv preprint arXiv:2001.07362* (2020).

[21] Michael J Risbeck and James B Rawlings. 2019. Economic model predictive control for time-varying cost and peak demand charge optimization. *IEEE Trans. Automat. Control* 65, 7 (2019), 2957–2968.

[22] Tetsuya Sato, Takayuki Shiina, and Ryunosuke Hamada. 2022. Optimization of EV bus charging schedule by stochastic programming. *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)* (2022), 627–632. https://api.semanticscholar.org/CorpusID:252469548

[23] Liam Schramm and Abdeslam Boularias. 2024. Provably Efficient Long-Horizon Exploration in Monte Carlo Tree Search through State Occupancy Regularization. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 43828–43861. https://proceedings.mlr.press/v235/schramm24a.html

[24] Wenbo Shi and Vincent WS Wong. 2011. Real-time vehicle-to-grid control algorithm under price uncertainty. In *2011 IEEE international conference on smart grid communications (SmartGridComm)*. IEEE, 261–266.

[25] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[26] John A. Stankovic, Krithi Ramamritham, and Marco Spuri. 1998. *Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms.* Kluwer Academic Publishers, USA.

[27] SJ Sultanuddin, R Vibin, A Rajesh Kumar, Nihar Ranjan Behera, M Jahir Pasha, and KK Baseer. 2023. Development of improved reinforcement learning smart charging strategy for electric vehicle fleet. *Journal of Energy Storage* 64 (2023), 106987.

[28] Ying Sun, Fariborz Haghighat, and Benjamin CM Fung. 2020. A review of the-state-of-the-art in data-driven approaches for building energy prediction. *Energy and Buildings* 221 (2020), 110022.

[29] Yongjun Sun, Shengwei Wang, Fu Xiao, and Diance Gao. 2013. Peak load shifting control using different cold thermal energy storage facilities in commercial buildings: A review. *Energy conversion and management* 71 (2013), 101–114.

[30] Olle Sundström and Carl Binding. 2010. Optimization methods to plan the charging of electric vehicle fleets. In *Proceedings of the international conference on control, communication and power engineering*. Citeseer, 28–29.

[31] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[32] Kevin Tanguy, Maxime R Dubois, Karol Lina Lopez, and Christian Gagné. 2016. Optimization model and economic assessment of collaborative charging using Vehicle-to-Building. *Sustainable Cities and Society* 26 (2016), 496–506.

[33] Hao Tu, Srdjan Lukic, Abhishek Dubey, and Gabor Karsai. 2020. An LSTM-Based Online Prediction Method for Building Electric Load During COVID-19. In *Annual Conference of the PHM Society.*

[34] U.S. Department of Energy. 2023. Bidirectional Charging and Electric Vehicles for Mobile Storage. https://www.energy.gov/femp/bidirectional-charging-and-electric-vehicles-mobile-storage

[35] Zongfei Wang, Patrick Jochem, and Wolfgang Fichtner. 2020. A scenario-based stochastic optimization model for charging scheduling of electric vehicles under uncertainties of vehicle availability and charging demand. *Journal of Cleaner Production* 254 (2020), 119886. https://api.semanticscholar.org/CorpusID:213912494

[36] Yunjian Xu, Feng Pan, and Lang Tong. 2016. Dynamic scheduling for charging electric vehicles: A priority rule. *IEEE Trans. Automat. Control* 61, 12 (2016), 4094–4099.

[37] Xian Zhang, Ka Wing Chan, Hairong Li, Huaizhi Wang, Jing Qiu, and Guibin Wang. 2020. Deep-Learning-Based Probabilistic Forecasting of Electric Vehicle Charging Load With a Novel Queuing Model. *IEEE Transactions on Cybernetics* 51 (2020), 3157–3170. https://api.semanticscholar.org/CorpusID:214808666

A Offline Approach

For any given *episode*, which is a trajectory of EV arrivals and departures for a single billing period, we can use an offline optimization program to solve the V2B problem and obtain an exact demand cost for that period. Thus, we formulate a mixed integer linear program (MILP) that can compute the optimal demand cost for any episode.

A.1 Exact Solution

The objective of the MILP is *minimizing* the multi-objective weighted sum of the total rewards in Equation 1, while meeting the constraints and requirements of the V2B problem. These include ensuring each EV is assigned to a single charger throughout its stay and keeping the action within the charger's limits. We use CPLEX [6] to solve the MILP.

Decision Variables: We use the same set of variables defined in Section 3.1. Additionally, z_v is used to represent the gap between the car's required SoC and its SoC at departure.

Constraints: To match the car to the charger assignment policy of MCTS (which follows a *first-come first-serve* assignment policy), we pre-compute the car to charger assignments. When $h_{v,k,t} = 1$, it denotes that the assignment of vehicle v to charger k at time t, the vehicle stays assigned for the duration of its stay.

We couple the assignment and the charging variables as follows.

$$\forall v \in \mathcal{V}, \forall t \in \mathcal{T} : c_t^v \leq \sum_k q_k^{max} \cdot \eta \cdot h_{v,k,t} \tag{3}$$

$$\forall v \in \mathcal{V}, \forall t \in \mathcal{T} : c_t^v \geq \sum_k q_k^{min} \cdot \eta \cdot h_{v,k,t} \tag{4}$$

We keep track of the EV's SoC before it leaves the charging station, and track if there is any gap to the required SoC.

$$\forall v \in \mathcal{V} : z_v = \left| e_{req}^v - e_{t_{end}^v}^v \right| \tag{5}$$

Furthermore, for the i^{th} time slot t_i, for an EV, we can find the amount of energy remaining using $e_{t_{n+1}}^v$:

$$\forall v \in \mathcal{V}, t \in \mathcal{T}, \text{ if } i = 0 : \quad e_{t_i}^v = e_0^v + c_{t_i}^v, \tag{6}$$

$$\text{otherwise: } e_{t_i}^v = e_{t_{i-1}}^v + c_{t_i}^v \tag{7}$$

where e_0^v is the initial charge the EV arrives with.

The demand cost can be computed as (recall that Ω is the number of aggregate time slots):

$$\forall j \in t_j^\tau, j \in \{0, 1, \cdots, \Omega\} : \pi_j = \sum_{v \in \mathcal{V}} \sum_{\forall i \in \mathcal{T}^\tau} c_{t_j}^v + b_j^p \tag{8}$$

$$P_{max} \geq \pi_j \tag{9}$$

We need to ensure that a vehicle retains the same charger throughout its stay and do so by maintaining continuous assignment of the car to its charger for its duration of stay:

$$\forall v \in \mathcal{V}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T} \setminus t_{end} : a_{v,k,t} - a_{v,k,t+1} = 0 \tag{10}$$

Objective We want to find the minimum energy cost and demand cost, while reducing the gap between the SoC required at departure and the actual departure SoC, given the TOU electricity prices.

$$\min \sum_{t \in \mathcal{T}} w_t \cdot \sum_{v \in \mathcal{V}} \left(c_t^v + b_t^e \right) + P_{max} \cdot w^d + w^s \cdot \sum_{v \in \mathcal{V}} z_v \tag{11}$$

Table 7: RL Hyperparameters and selected values.

Parameter	Description	Range
Actor network	Number of units at each layer	[96, 96]
Critic network	Number of units at each layer	[96, 96]
Γ	Discount factor for future reward	1
Actor&Critic learning rate	Learning rate for updating actor and critic networks	10^{-5}, 10^{-3}
bufferSize	Batch size for fetching transitions from replay buffer	64
batchSize	Size of the replay buffer	10^6
actionNoise	Noise added during action exploration	normal(0,0.2)
policyGuidaRate	Probability to introduce policy guidance	0.5 or 0.7
$\lambda_S, \lambda_B, \lambda_D$	Penalty coefficients for SoC requirement, bill cost, and demand charge	1,1, 3
Random seed	Random seed for actor and critic network initialization	0-5
trainStep, *updateStep*	Training steps and steps per update of target networks	5, 5

B Decentralized MCTS Algorithms

In this section, we introduce the algorithms designed for decentralized Monte Carlo Tree Search (MCTS) in the context of our V2B framework. As described earlier, decentralized MCTS (dMCTS) enables individual chargers to make decisions independently by constructing their own search trees, rather than relying on a single centralized decision-making process. This approach significantly reduces the computational complexity, allowing each agent to explore its action space in parallel, which enhances scalability and responsiveness in multi-agent settings. We also show in Algorithm 3 how we implement sorting the cars based on *criticality score*.

We detail the algorithmic steps involved in dMCTS, including each charger's process for action selection, rollout, and backpropagation, while considering the impact of other agents' actions. The algorithm provides the pseudocode and procedural breakdown for implementing dMCTS, along with action space modifications to aid in the large action space exploration.

C Hyperparameter search for MCTS

Optuna is a modern hyperparameter optimization framework that uses an efficient sampling-based approach to identify the best combination of hyperparameters for a given objective function. It supports algorithms such as Tree-structured Parzen Estimators (TPE) to adaptively sample promising regions of the hyperparameter space while discarding less effective configurations. Each trial in Optuna represents a unique combination of hyperparameters, which is evaluated based on the objective function, and results are stored in a study database for further analysis. Optuna also provides visualization tools, such as hyperparameter importance graphs, to highlight which parameters have the most significant impact on performance.

In our work, we utilized Optuna to optimize the performance of DG-MCTS by minimizing a custom-defined score (equation (1)). The search space included critical parameters like penalties for unmet SoC, rewards for SoC milestones, and tolerance for estimated peak power. Optuna's trial-based optimization revealed key insights into

Table 8: Hyperparameters for MCTS Configuration

Hyperparameter	Value
Iterations (Simulation Steps)	200
Maximum Tree Depth	70
Exploration Coefficient (C)	1.414
Discount Factor (γ)	1
Penalty for Missing SOC	0.5
Penalty for Exceeding Power Gap	5
Reward for Meeting Required SOC	0.1
Reward for Maximizing SOC	0.01
Estimated Peak Power Adjustment (ϵ)	0
Exploration Samples	10

hyperparameter importance (Figure 3), guiding the selection of optimal values for our final implementation.

Algorithm 2 Decentralized Monte Carlo Tree Search (dMCTS)

Input: Current State $\mathcal{S}_t$, iteration number $\mathcal{N}$, exploration range β
Output: Best actions $[\mathcal{A}_k^* \text{ for } k \in K]$ for all chargers

$D \leftarrow \text{EstimateDemandCharge}(\mathcal{S}_t)$ `// Estimate demand charge`
foreach $k \in K$ *sorted by least laxity first* **do**
  $\mathcal{A}'_k \leftarrow \text{LLFPolicy}(\mathcal{S}_t^k, D)$ `// Get heuristic action`
  $Space(\mathcal{A}_k) \leftarrow$ GetNeighbors$([-\beta + \mathcal{A}'_k + \text{Noise}, \beta + \mathcal{A}'_k + \text{Noise}])$ **for** $n \leftarrow 1$ *to* $\mathcal{N}$ **do** `// Establish tree for each charger`
    $Sample \leftarrow \text{GenSample}(\text{EVDep}, \text{EVArr}, \text{BLoad})$.
    $\mathcal{S}_t^k \leftarrow \mathcal{S}_t$
    $\mathcal{S}_{t+1}^k \leftarrow \text{SelectFrom}; (\mathcal{S}_t^k, Space(\mathcal{A}_k), Sample)$
    $\mathcal{S}_{t+2}^k \leftarrow \text{Expand}(\mathcal{S}_{t+1}^k, Sample)$ `// Add new child nod`
    $v \leftarrow \text{Simulate}(\mathcal{S}_{t+2}^k, Sample, LLFPolicy)$ `// Rollout`
    $\text{Backpropagate}(\mathcal{S}_{t+2}^k, v^k)$ `// Update tree state for k`
  $\mathcal{A}_k^* \leftarrow \text{BestAction}(\mathcal{S}_t)$ `// Select the best action for k`
  $\mathcal{S}_t \leftarrow \text{Update}(\mathcal{S}_t, \mathcal{A}_k^*)$ `// Update state for the next charger`
return $[\mathcal{A}_k^* \text{ for } k \in K]$

Algorithm 3 Sort Cars by Criticality

Input: Set of cars $\mathcal{V}$
Output: Sorted list of cars $\mathcal{V}'$ based on criticality

$\mathcal{V}^{critical} \leftarrow \emptyset$ `// Initialize list of critical scores`
foreach $v \in \mathcal{V}$ **do**
  $c_v \leftarrow \text{ComputeCriticality}(v)$ `// Compute critical score`
  $\mathcal{V}^{critical} \leftarrow \mathcal{V}^{critical} \cup \{(v, c_v)\}$ `// Append car v and its criticality score`
$\mathcal{V}' \leftarrow \text{Sort}(\mathcal{V}^{critical}, \text{by } c_v \text{ descending})$ `// Sort cars by criticality score`
return $\mathcal{V}'$ `// Return sorted list of cars`

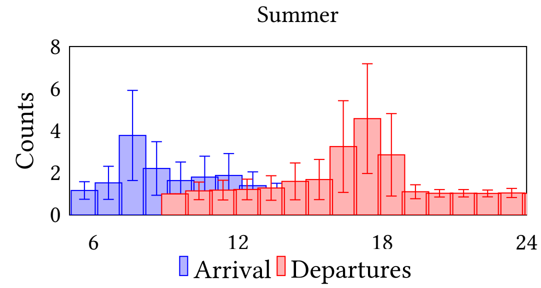

Figure 2: Universal Dataset distribution of EV arrivals and departures based on 1 month from each season. Distribution patterns change over the seasons, but the overall variation remains low.

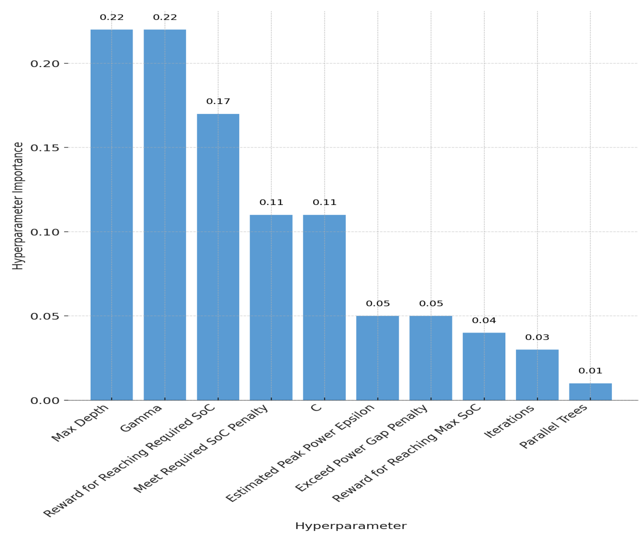

Figure 3: Importance of each factor in hyperparameter exploration