OPTIMUS: Discrete Event Simulator for Vehicle-to-Building Charging Optimization

Jose Paolo Talusan*, Rishav Sen*, Ava Pettet†, Aaron Kandel†,
Yoshinori Suzue†, Liam Pedersen†, Ayan Mukhopadhyay*, Abhishek Dubey*
*Vanderbilt University
†Nissan Advanced Technology Center - Silicon Valley

***Abstract*—The increasing popularity of electronic vehicles has spurred a demand for EV charging infrastructure. In the United States alone, over 160,000 public and private charging ports have been installed. This has stoked fear of potential grid issues in the future. Meanwhile, companies, specifically building owners are also seeing the opportunity to leverage EV batteries as energy stores to serve as buffers against the electric grid. The main idea is to influence and control charging behavior to provide a certain level of energy resiliency and demand responsiveness to the building from grid events while ensuring that they meet the demands of EV users. However, managing and co-optimizing energy requirements of EVs and cost-saving measures of building owners is a difficult task. First, user behavior and grid uncertainty contribute greatly to the potential effectiveness of different policies. Second, different charger configurations can have drastically different effects on the cost. Therefore, we propose a complete end-to-end discrete event simulator for vehicle-to-building charging optimization. This software is aimed at building owners and EV manufacturers such as Nissan, looking to deploy their charging stations with state-of-the-art optimization algorithms. We provide a complete solution that allows the owners to train, evaluate, introduce uncertainty, and benchmark policies on their datasets. Lastly, we discuss the potential for extending our work with other vehicle-to-grid deployments.**

***Index Terms*—Simulation, Optimization, EV charging**

I. Introduction

The increasing ubiquity of electronic vehicles (EVs) has increased the demand for EV charging stations. As of 2023, there are more than 140,000 public and 20,000 private charging ports across the USA[1]. EV asset managers such as fleet, charging asset managers, and EV manufacturers are now seeing the game-changing opportunity to leverage EVs as batteries which could be effectively used as buffers against the electric grid. This is also true for building owners with many EV chargers. They have begun offering charging at their parking lots and are now using this opportunity to avoid buying power from the grid during peak time-of-use periods, instead discharging from parked cars. The opposite is true during off-peak time-of-use periods when prices are low, where building owners can afford to charge all cars with minimal costs.

This transaction between EVs and smart building owners is defined as Vehicle-to-Building (V2B). The key idea behind V2B is the management and co-optimization of energy and cost requirements of EVs and smart buildings [2]. When done correctly, EVs can provide a certain level of energy resiliency and demand-responsiveness for buildings during high grid peak rates and sudden grid events [3]. Simultaneously, buildings can offer a level of guarantee to EVs that they will be charged by the time they leave. This is possible by actively controlling charger actions and behavior in response to stimulus from the grid, building, and requests from EV owners. However, effectively realizing this is non-trivial due to several endogenous and exogenous variables.

The ability to optimize the two objectives of cost and energy hinges on the configuration of chargers that would be used and the actions they take. EV chargers come in two particular types, unidirectional and bidirectional. While both types of chargers can store energy in the EV, only bidirectional chargers can harness this energy by discharging the vehicle The number and types of deployed chargers have a massive effect on the possible actions that a smart building can take to meet its objectives. Aside from these decisions, the majority of challenges in optimizing objectives lie in the inherent uncertainty present when dealing with multiple agents. The transient behavior of EVs makes it difficult to completely plan and optimize actions for the entire billing period. Charging EVs while avoiding adverse effects on the managers and the grid, requires new techniques and algorithms. The development of these techniques has attracted attention in the research community. However, the efforts required to make these algorithms into reality require addressing the complexities of practical systems that are often overlooked in theoretical models. Creating an impact in practice requires access to real-world data, the ability to generate new data, introduce uncertainty, and integrate it with existing systems.

In this work, we tackle the identified gaps by creating a Python-based, event-driven simulation platform, which we developed in collaboration with our industry partner, Nissan. The complete framework, called OPTIMUS, seamlessly unites EV charging policies, physics-based charging profiles, and grid events to facilitate easy, minimal-change policy development and testing. The platform consists of several reusable and configurable components, adaptable to various datasets and deployment scenarios. Noteworthy features include modules for data generation, policy execution, and policy training. Additionally, the platform offers a web application with multiple interfaces, specially designed to streamline testing processes for building owners.

The summary of our contributions is as follows: **(1)** We have developed OPTIMUS, a platform that utilizes real-world

data to train both learning and generative models to allow us to create realistic input data for testing policies. **(2)** Our framework includes several prebuilt policies for generating charging actions, which users can easily extend to meet their specific needs. Finally, **(3)** we provide a comprehensive end-to-end solution designed for policy evaluation on scenarios with uncertainty reflecting real-world usage, all within a ready-to-use web application.

II. Existing Simulators

Open-source tools and simulators are often used in smart grid research. GridLAB-D [4] is widely used to model power systems and flow computations accurately. Simulators specific to EV charging often use tools such as GridLAB-D to ensure accurate representations of their EV batteries and grid levels. V2G-Sim, developed at LBNL [5], is a mature simulation environment that has been used to meet drivers' mobility needs in the context of demand response and level-1 charging. On the other hand, Sarieddine et al. developed a hardware-in-the-loop (HIL) real-time simulator, while accurate, is costly and requires existing infrastructure [6]. ACN-Sim [7] is designed specifically around evaluating online algorithms that adapt to changes in the system over time while considering infrastructure constraints within a charging facility. EV-EcoSim [8] is a more recent co-simulation platform that uses several modules such as electric vehicle charging, battery systems, and control strategies, to perform cost quantification and analyze the impacts of EV charging on the grid. Both ACN-Sim and EV-EcoSim have been developed as modular architectures that model physical systems as closely as possible while making it easier to extend for new use cases.

While V2G-Sim allows for precomputed charging schedules and simple control strategies, it cannot evaluate online algorithms and adapt to changes over time. Both ACN-Sim and EV-EcoSim address this issue and can handle changes to the state over time. However, EV-EcoSim is not designed with ease of use in mind, requiring more knowledge to configure on own data. ACN-Sim allows researchers and users to share and reproduce experiments through Jupyter Notebooks hosted on Google Colab. However, it lacks the presence of a complete dashboard that allows users to generate data, select algorithms, and export and share results. We also allow users to inject uncertainty into the simulations through API requests. Additionally, we design OPTIMUS to interface with mobile devices and existing chargers to mimic sudden out-of-input distribution EV arrivals and departures. This is absent in all prior work.

III. Problem Statement

In this section, we define the costs and issues related to EV charging. Then, we define the problem faced by Nissan as the building owner and as an EV manufacturer.

A. EV Chargers

The building owner, Nissan Advanced Technology Center Silicon Valley (NATC-SV), uses two types of chargers to charge their Nissan LEAF EVs, unidirectional and bidirectional. Uncontrolled unidirectional chargers are more common and are typically limited to charging EVs as soon as they plug in and only stop when the EV battery is full. Others can be turned on or off, essentially changing between 0 and their maximum charging rate. This behavior is similar to any traditional battery charger. However, it is not the most cost-effective solution for avoiding high peak demand rates. On the other hand, bidirectional chargers can charge EVs and also discharge their batteries to power external loads. Bidirectional chargers may change their charging and discharging rates over time to match EV requirements. The desired behavior is still being investigated for bidirectional chargers, thus the need for a simulator.

B. Uncertainty in the Environment

Cars arriving and departing from the parking lot follow a certain distribution that can be influenced by weather, day of the week, and month of the year. However, within these distributions, there is always uncertainty. Users may arrive later and depart earlier than expected with different arrival SoCs and required SoCs upon departure each time. Uncertainty also extends to building power draw across the billing period, which can be affected by several factors such as weather, ongoing office projects, and sudden grid events such as the Emergency Load Reduction Programs (ELRPs) [9], which are demand response approaches to help avoid rotating outages during peak summer electricity usage periods from May through October. Thus, policies relying on expected behaviors to generate actions will often fail once users diverge from their particular schedules. The simulator handles this uncertainty by allowing the management to select and test different policies on set schedules with unexpected arrivals and departures injected during simulation run time.

C. Billing Costs

Electric utility companies typically offer a variety of standard billing arrangements. Although billing is usually conducted monthly, the rates charged can differ based on the quantity of electricity consumed by the users. For example, Silicon Valley Power (SVP) in California, USA, assigns consumers to one of several tiers based on their consumption behavior. It is divided into residential and commercial consumers, with the commercial consumers further divided into levels. Commercial and industrial customers are often subjected to **demand charge** and **energy charge**. The following are the different billing components:

- **Time-of-use** rates differ across time-of-day, with peak hours having higher rates than off-peak times.
- **Energy charge** is the cost incurred for every kWh of energy used. It is time-of-use dependent.
- **Demand charge** reflects the highest rate of electricity consumption during a specific time interval. Under SVP, if the energy use exceeds 8,000 kWh per month, and maximum electric demand does not exceed 4,000 kW, the category that NATC-SV falls into, they will accrue an additional cost —

the demand charge. For SVP it is composed of two parts: *Maximum Demand* and *Billing Demand*. The Maximum Demand is determined by the highest average kilowatt (kW) delivery over any 15-minute interval within that month. The Billing Demand is the average of the current month's Maximum Demand and the highest Maximum Demand in the past year, including the current month (billing period). For SVP, the Maximum Demand is only considered during the peak hours.

- **Total bill** is the sum of the energy and demand charges over the billing period. Applying the demand charge depends on the power utility's pricing policy. Otherwise, only the energy charge is considered.

Managing energy and demand charges makes it difficult to efficiently manage costs along with the charging behavior of the chargers in the building. Thus, the main problems faced by building owners and EV manufacturers are: **(1)** How to minimize the total costs charged to the building for each billing period? **(2)** How to ensure that the EV owner's demands are met, given that it is feasible to do so? **(3)** How to verify this effectively and efficiently?

IV. OPTIMUS Overview

OPTIMUS uses a modular, object-oriented architecture shown in Figure 1. Encapsulating different components and subcomponents that make up the entire framework makes it easier to extend for new use cases. We replicate the behavior of physical systems by utilizing their corresponding dynamic models, with their interactions being represented through data exchange at every discrete time interval. The figure also includes how data and charger owners interact with the framework. OPTIMUS is a complete system that includes both a backend solver and data generator and a frontend visualization tool that allows users to fully utilize the backend components. The system is meant to allow building owners the ability to train on their data, and then evaluate and benchmark proposed policies on their defined objectives.

Generative Models: OPTIMUS relies on several data sources to run. Data provided by owners are first collected and processed by a pipeline that trains generative models. Each of the six models: **(1)** building load prediction, **(2)** duration of stay, **(3)** demand charge forecasting, **(4)** EV arrival forecast, **(5)** Arrival SoC, and **(6)** Departure SoC, are used to build an input to the simulator. Each input, which can be as short as a single day or as long as an entire billing period, is defined as an *episode*. These episodes form the basis of what the simulator interprets as the real world.

Policies: These are algorithms or models trained to identify the state at each time step and return an action. States, actions, and policies will be discussed further in Section V.

Frontend UI: Interfaces expose backend components, which a frontend web application easily accesses. Models for data generation actively create episodes on the fly. Adjusting parameters in the UI influences the resulting episode. During policy execution, the system calls solvers. Owners select a policy in real time, and the corresponding solver generates actions based on the input data.

TABLE I: State Member Variables Description

Attribute	Tracked Parameters
Time	Current simulation time.
Cars	Car ID, arrival time, battery capacity, current soc, required soc, allowed SoC range.
Chargers	Charger ID, charge rate, connected car ID, availability, direction, and charge limits.
Building	Building power draw.
Grid	Grid peak type and energy rate.

API Interface: OPTIMUS offers an API interface that enables seamless interaction with devices such as EV chargers and mobile applications using REST APIs. When an EV plugs in, the system receives immediate notification, allowing users to submit additional information, such as estimated departure time, through mobile apps.

Integrating these components simplifies the process for owners to select and apply various policies, create episodes for policy evaluation, and integrate the system with actual EV charger infrastructure and proprietary mobile applications. The aim is to provide a straightforward and effective approach to assess policy performance in specific scenarios, considering uncertainties. We measure performance by the policies' effectiveness in minimizing total costs for a billing period.

Handling Uncertainty: Uncertainty in the real world is represented in OPTIMUS as event injections. Injection times correspond to sudden EV arrivals, departures, or changes in user requests. This may occur when a user plugs their EV to a charger or modifies requirements through the app. These events are not in the initial input episode and are thus exogenous to the simulator. There are two types of injections: (1) *arrival injections* and (2) *update injections*. These injections are triggered in OPTIMUS via REST APIs.

V. Discrete Event Simulator

In this section, we define the simulator that serves as the engine of OPTIMUS. We define the states and state transitions that occur within the simulator. Finally, we define the policies that interact with the simulator.

A. Events

Input files, known as episodes, feed into the simulator which then converts them into a list of events. Each event features a specific time that corresponds to its real-world occurrence. Four potential events can initiate state changes: (1) grid price changes, (2) building load readings, (3) car arrivals, and (4) car departures. An event queue holds all events. The system processes and removes events that fall within the current time from the queue.

B. State

The state is a representation of the current environment or the real world. The simulator is stateless, thus, each state is a snapshot of the real world only at the current time.

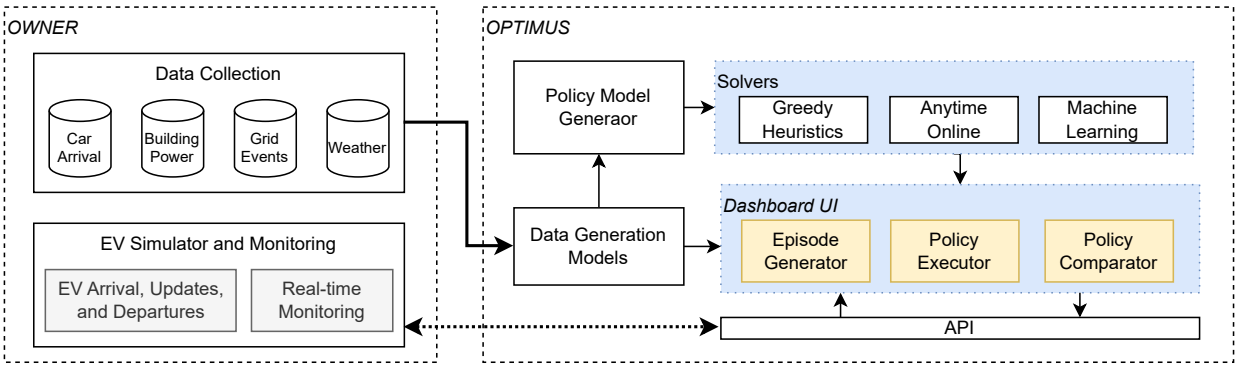


Fig. 1: OPTIMUS software design. It includes how data and charger owners interface with the framework.

This prevents any leakage of future information that could potentially skew their decisions. Each state includes only the current state of connected cars, current charging rates of active chargers, building power at that time, and current grid conditions. Table I lists all the states the simulator tracks.

C. State Transition Model Updates

The simulator is a discrete time-driven simulator that aggregates multiple events within each time step and processes them all at once. If there are multiple that take place in the same time step, they are sorted according to priority and processed in sequential order. The first two events simply update the state based on the updated price and power meter reading respectively. However, car arrival events are handled differently. EV arrivals are considered special events by any policy. Once a car has arrived at the parking lot, policies determine which available charger the EV should be connected to. Finally, car departure simply frees up a charger and flags it as available again.

At every elapsed time step, cars connected to a charger are charged via physics-driven battery profiles that consider the current input charger rate and the SoC constraints for the car battery. Building energy consumption is computed based on the last available power reading, assuming that the power draw was consistent between readings. The sum of all charger usage and building usage (kWh) multiplied with the energy charge ($\frac{\$}{kWh}$) is the energy cost ($) for this time step. Meanwhile, the instantaneous power draw (kW) for this time step is considered when computing the demand cost ($).

D. Policy Driven Charger Actions

Users of OPTIMUS can extend the `Policy` class by defining their custom `getAction()` function within it. This function, which processes the current state, outputs an array specifying the charging actions. Through the configuration interface, policies can access a wealth of information, including constraints and user-defined parameters.

By default, OPTIMUS is equipped with several predefined policies: an anytime online algorithm, a mixed integer linear programming approach, and a machine learning model. We provide a brief description of each below:

Greedy Heuristics: Similar to traditional chargers, these models are entirely myopic. A greedy charger initiates charging at the highest rate as soon as an EV connects and continues until the EV's state of charge (SoC) reaches either the established maximum or the departure requirement, for naive and informed greedy policies, respectively. We employ this approach as a benchmark in our experiments.

Online Algorithms: Utilizing tree-based search methods like Monte Carlo Tree Search (MCTS) [10], these anytime policies tailor actions to the current state. They yield non-myopic actions that effectively handle uncertainty. Despite this, their computational demands render online algorithms slower than their counterparts due to their intensive sampling techniques.

Mixed Integer Linear Programs: Knowing or assuming the complete trajectory for a given billing period allows one to formulate the problem as a linear program and then solve it. Thus, MILP's solutions are the upper bounds in terms of performance. However, they are not responsive to uncertainty, resulting in potentially non-optimal solutions given enough uncertainty.

Reinforcement Learning (RL): An RL agent is trained on an abstracted state using simulated environmental conditions. action is to adjust the charging rate of each charger. To minimize total costs over a billing period, we define the reward for each action based on the required SoC of each car before departure and the total bill at the end of each episode. We employ the Deep Deterministic Policy Gradient approach to train the RL model, considering the continuous action space in this context. During training, the RL-based policy continuously interacts with our simulator, which provides state transitions and features for determining action rewards.

Each of these policies can be queried at any time given the current state of the environment, which is a snapshot at the current time. This includes information regarding current cars plugged in, available chargers, current grid condition, and current building power draw. Each policy has its method of generating charging actions while considering uncertainty, ranging from completely myopic decisions from heuristics to utilizing some estimates of the possible future trajectories as in the machine learning models.

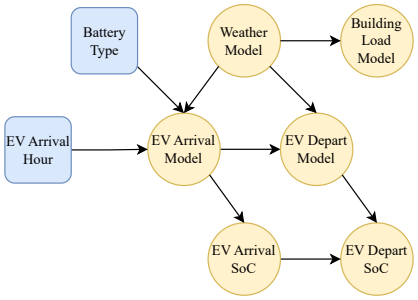


Fig. 2: Bayesian Network depicting the joint probability distributions across several models. Squares are static features, circles are the models, and arrows depict their relationships.

In summary, a policy when given any state, will respond with a set of charger actions for each charger in the system. This action persists until the next time the policy is again queried. The quality of a policy is therefore how effective it is in meeting the goals set in Section III-C.

VI. Evaluation of Policies

In this section, we discuss the datasets used to generate episodes and train the models for policies. We also provide more information on the different policies and how they are incorporated into the framework.

A. Datasets

OPTIMUS uses EV data, building power draw data, weather data, and potential grid events data to train different models.

- **EV data** includes EV arrival time, departure time, arrival SoC, and departure SoC for each charging event. Charging event data is obtained from real-world EV charging data collected from January 2021 to December 2022 from a California office building.
- **Building data** includes power draw meter readings for Nissan's California building from April 2023 to January 2024 with a frequency of $\sim$ 15min.
- **Weather data** primarily consists of the recorded temperature and precipitation matching the duration of the recorded building data.
- **Grid Events** are specific events triggered by utility companies that signal a change in electricity rates over a certain period. These can also limit power draw from the grid by incentivizing power-saving measures and penalizing overuse.

B. Data Generation

We train models using a Bayesian network that emphasizes the relationships between different datasets through a large joint distribution. Figure 2 shows the relationships between the different models. Battery types are sampled using a probability based on actual vehicle counts and EV arrival hours are the different hours in a day. EV Arrival models are Poisson models trained on weather patterns, arrival hours, and battery types. The results of arrival models are then fed into the arrival SoC and departure models. These in turn are fed into the departure SoC models.

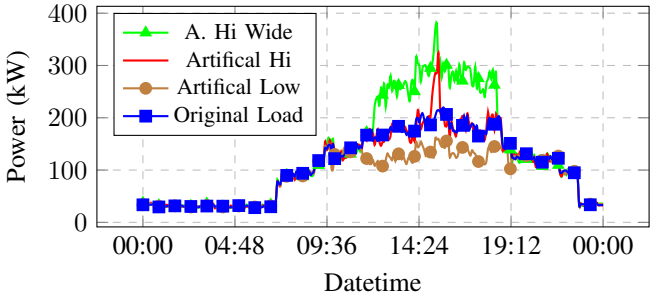


Fig. 3: Original power draw over time with artificial power draw from models to test uncertainty.

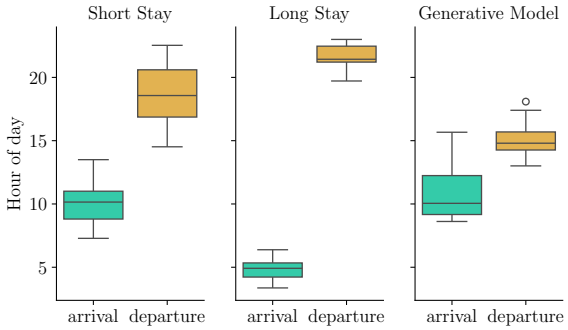


Fig. 4: Different distributions for parking duration based on selected configurations in the dashboard.

Using these models, OPTIMUS can generate input files representing the expected real-world state. Each input episode is made up of four files: (1) car schedules, (2) building load readings, (3) grid costs and schedules, and (4) charger configurations. Toggles on the application allow the owner control over how building and EV data is generated. Figure 3 shows different power consumption readings for the same building, on the same day while, Figure 4 shows different types of distributions of durations of stay and Figure 5 shows the arrival and departure distributions for the trained generative model. This opens up the possibility of testing policies across different scenarios.

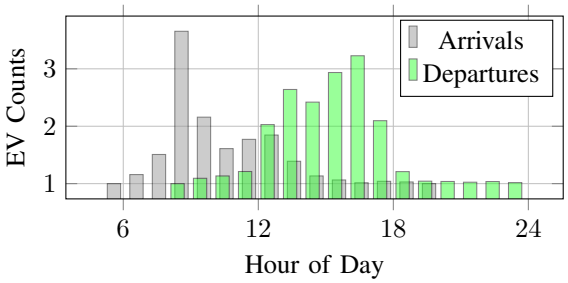


Fig. 5: EV arrivals and departures from generative models.

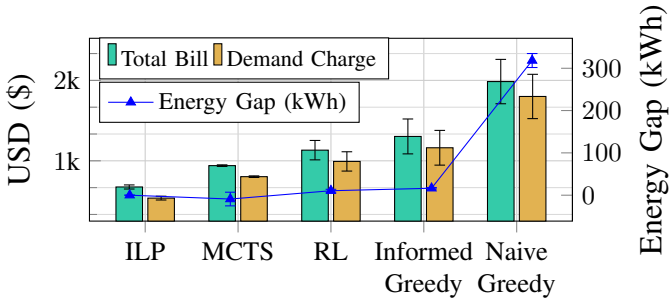


Fig. 6: Comparison of different policy performances (lower is better). An energy gap of < 0 means cars left without meeting SoC requirement, > 0 means cars left with more than required.

C. Policy Comparison

Each policy has its methods to generate the charging actions for any given state. We evaluate the policies by generating 20 samples for a billing period with a duration of a single day. We run this across multiple policies. Figure 6 shows each policy's performances. Ideally, the energy gap should be 0, where all EVs are charged to their desired level. Greedy heuristics, with their myopic view of the environment, will only give actions based on the current state, often resulting in the worst performance. Naive Greedy will charge cars until the maximum level, resulting in a large positive energy gap (excess). While Mixed Integer Linear Programming-based (MILP) policies will consider all present and future car arrivals and building meter readings and attempt to create a perfect solution for this single trajectory. Thus, MILP policies will serve as the upper bounds of performance. However, it relies on complete knowledge of this single trajectory, any deviation or injections of uncertainty will degrade its solution. Between these policies, other approaches such as MCTS and RL will often undercharge vehicles resulting in a lower bill and a negative energy gap. However, these policies can accommodate more uncertainty in EV behavior. The goal for the building owner is to find a policy that is non-myopic and will perform well under the uncertainty of EV arrival, departures, SoC requirements, and power grid fluctuations

VII. Real World Deployment

In this section, we describe how OPTIMUS is deployed as a web application and how it is integrated into EV chargers and mobile applications. We also show that our framework can be extended beyond the intended application of V2B.

A. Web Application

The application serves as an interface to the backend while allowing owners a clear overall view of the system. Figure 7 shows the main pages of the web app: Generator, Executor, and Comparator. Users can enter the **Generator page**, fig. 7a to generate episodes by selecting parameters shown in Section VI-B. They can view the generated episode which consists of the four main input files (cars, chargers, building, and grid). Once satisfied, the episode can be named and saved into the system. This ensures that all other pages can access it.

Once the episode is saved, a user can switch to the **Executor page**, fig. 7b, where they can select a policy, and then run the simulator on an episode. OPTIMUS will go through the entire episode's billing period, sending the current state to the policy and then receiving the policy's action for each time step. Once done, OPTIMUS will return control to the user then they can navigate across time by moving the slider or clicking on the main plot. Upon clicking, the application will jump to the corresponding time and update the different plots and indicators accordingly. At any point in time, they can view the current metrics for the performance evaluation. Users can also opt to play the entire day back in real time. Finally, the Executor page is also the main interface for injecting uncertainty as vehicle arrivals, departures, and updates.

Users can also switch to the **Comparator page** to select multiple policies and compare their performance for the given episode. For example, Figure 7c shows the result screen after selecting MILP and a greedy heuristic policy for an episode. The Comparator tab offers all the same plots as the Executor page (one set of plots for each policy and episode combination), with a few exceptions. The Comparator tab will display all the plots at once without playback ability, it is limited to showing only the plots without the gauges and indicators present in the Executor, and it does not support injections of sudden car arrivals and departures.

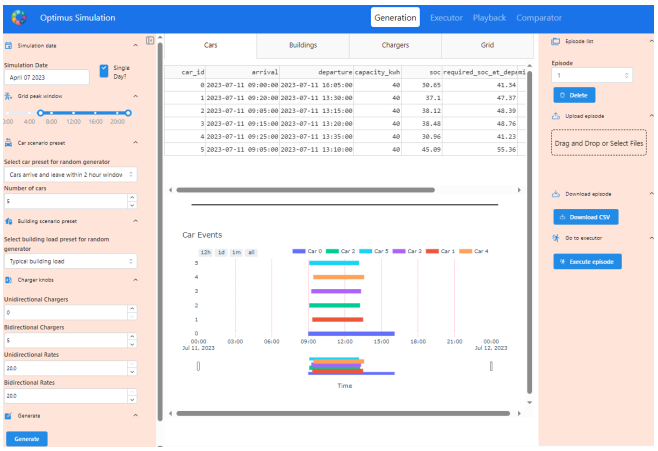
B. Injections and API Integration

Figure 7d shows how plugging a car into a charger automatically triggers an *arrival injection* and reloads the Executor page with the newly arrived car on the screen. Both mobile and web applications can be used to update battery requirements for a connected EV. This will trigger *update injections*. Injections need to pass through feasibility checks before they can be accepted and executed by OPTIMUS. For example, a car must arrive before any configuration is updated.

Upon receiving an injection, OPTIMUS will save the states from the start of the data up to the injected time and freeze it. It will then modify the input episode to introduce the newly injected event and run the Executor from this point in time to the end of the billing period. Locking the states before the injection ensures that OPTIMUS will not be able to make any retroactive changes that could improve its performance. Once processing is done, the injection will be saved as part of the episode allowing users to re-run the episode with any desired injection (without having to trigger any API calls from their end). Injections are sorted by time with arrival injections being triggered before any update injection. When an injection is selected in the episode drop-down box, it will trigger all injections before and up to the currently selected injection, based on their injection times.

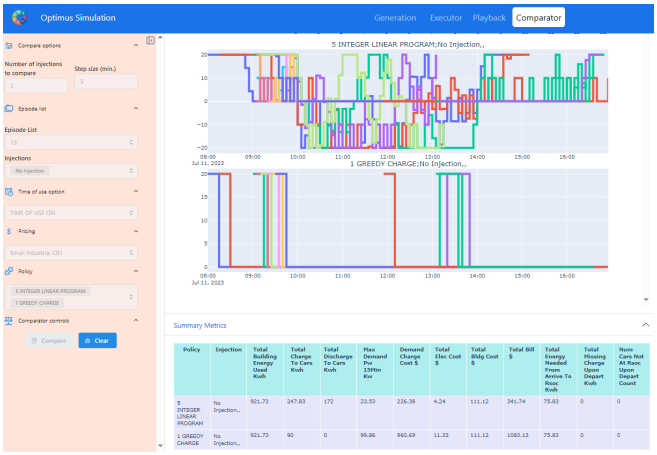
C. Visualization and Generated Plots

Upon execution of any episode, either through the *Executor* page or *Comparator* page, several plots will be generated. These plots are all navigable with more information provided on hover. In cases where episodes last longer than a single day,

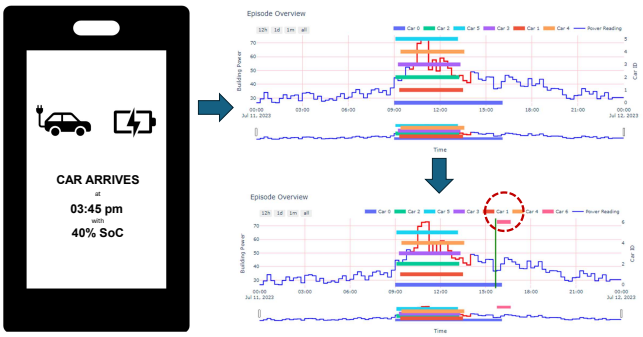


(a) Generator page with single and multi-day options.

(b) Executor page with episode summary and car status.



(c) Comparator page with performance metrics table.



(d) Injection prompted by actual vehicle arrival.

Fig. 7: Main components of the OPTIMUS web application: Generator, Executor, Comparator, and Injections.

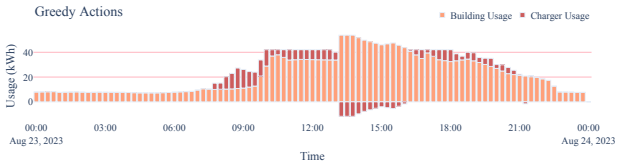


Fig. 8: Energy usage shows chargers and building use (kWh).

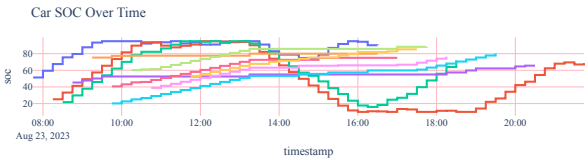


Fig. 10: EV SoC plot shows SoC (%) per EV over time.

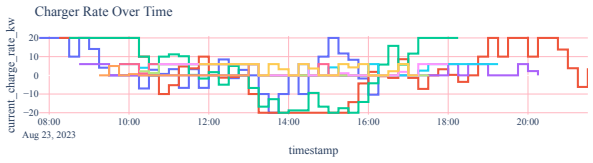


Fig. 9: Charger rates show actions per charger (kW).

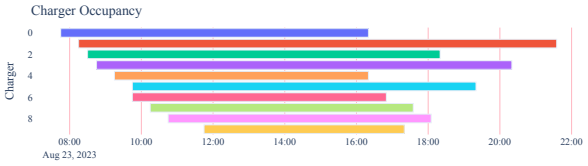


Fig. 11: Occupancy plots show charger availability.

the plots will be truncated to the last 24 hours. These plots are intended to provide a complete picture of the episode.

Figure 8 displays the overall power draw of the building, aggregating charger usage and building consumption in 15-minute intervals. Figure 9 depicts the actions of each charger in the lot, as dictated by the selected policy, thereby enabling owners to confirm policy adherence and to keep tabs on individual chargers. Figure 10 presents the EV battery levels, detailing the outcomes of charger activities. Figure 11

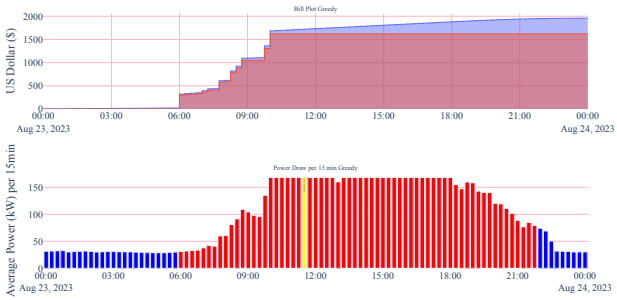


Fig. 12: Power demand (kW) and demand charge ($) plots show how demand charge is influenced by power demand.

indicates the duration each charger is occupied by an EV, thereby reflecting the efficacy of the charger assignment policy. Finally, fig. 12 offers a pair of plots that clearly illustrate the policy's impact on energy and demand costs, simplifying the task of pinpointing critical power draw times within the billing cycle, such as peak demand instances, and it elucidates the consequences of implementing peak shaving measures or the lack thereof. Together, these six plots provide owners with an exhaustive view of the system's various facets.

D. V2X Extension

While it is primarily designed with V2B in mind, OPTIMUS can easily be modified or extended to handle other V2X deployments such as vehicle-to-home (V2H), vehicle-to-community (V2C), and vehicle-to-grid (V2G) [11]. V2H has a single site (home charger) where only a few vehicles connect to it regularly. By maintaining a record and tracking vehicles by a unique vehicle ID number (VIN), OPTIMUS can form a profile for each vehicle and essentially give personalized actions. V2C can be addressed by aggregating multiple buildings and their associated chargers in a locality. Finally, V2G is a direct extension where EV aggregators can directly sell stored energy in EVs back to the grid.

E. Dataset Extension

While the models and dataset used in this paper are proprietary Nissan data, OPTIMUS is data agnostic. As long as several key features are present in the data, such as battery model, SoC, charger types, and duration of stay, models can still be trained on any EV data. Open source datasets of EV charging sessions such as EVWatts [12] and ACN-Data [13] can be used with OPTIMUS with minimal modifications.

VIII. Conclusion and Outlook

In this work, we present OPTIMUS, a complete end-to-end discrete event simulator and monitoring tool for vehicle-to-building EV charging optimization. This work was done in close collaboration with Nissan as a platform that would allow them to use their data to train charger behavior policies, create generative models, and evaluate and benchmark different configurations before their deployment. Going forward, we will be extending this work to incorporate contract negotiations with users and to handle other V2X implementations and integrate it with Nissan's planned test bed.

Acknowledgments

This material is based upon work sponsored by the National Science Foundation (NSF) under Award Numbers 1952011 and 2238815 and by Nissan Advanced Technology Center - Silicon Valley. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or Nissan. Results presented in this paper were obtained using the Chameleon Testbed supported by the NSF.

References

[1] A. Brown, J. Cappellucci, A. Heinrich, and E. Cost, "Electric vehicle charging infrastructure trends from the alternative fueling station locator: 3rd quarter 2023."

[2] K. Tanguy, M. R. Dubois, K. L. Lopez, and C. Gagné, "Optimization model and economic assessment of collaborative charging using vehicle-to-building," *Sustainable Cities and Society*, 2016.

[3] C. Heymans, S. B. Walker, S. B. Young, and M. Fowler, "Economic analysis of second use electric vehicle batteries for residential energy storage and load-levelling," *Energy Policy*, 2014.

[4] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, 2008.

[5] S. Saxena, "Vehicle-to-grid simulator," 2013.

[6] K. Sarieddine, M. A. Sayed, D. Jafarigiv, R. Atallah, M. Debbabi, and C. Assi, "A real-time cosimulation testbed for electric vehicle charging and smart grid security," *IEEE Security and Privacy*, 2023.

[7] Z. J. Lee, S. Sharma, D. Johansson, and S. H. Low, "Acnsim: An open-source simulator for data-driven electric vehicle charging research," 2021.

[8] E. Balogun, E. Buechler, S. Bhela, S. Onori, and R. Rajagopal, "Ev-ecosim a grid-aware co-simulation platform for the design and optimization of electric vehicle charging infrastructure," *IEEE Transactions on Smart Grid*, 2023.

[9] J. Aghaei, M.-I. Alizadeh, P. Siano, and A. Heidari, "Contribution of emergency demand response programs in power system reliability," *Energy*, 2016.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[11] C. Liu, K. T. Chau, D. Wu, and S. Gao, "Opportunities and challenges of vehicle-to-home, vehicle-to-vehicle, and vehicle-to-grid technologies," *Proceedings of the IEEE*, 2013.

[12] Y. Pavuluri, "Ev watts public database," 2024.

[13] Z. J. Lee, T. Li, and S. H. Low, "Acn-data: Analysis and applications of an open ev charging dataset," in *Proceedings of the 10th ACM International Conference on Future Energy Systems*, 2019.