ACN-Sim: An Open-Source Simulator for Data-Driven Electric Vehicle Charging Research

Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low, *Fellow, IEEE*

***Abstract*—ACN-Sim is a data-driven, open-source simulation environment designed to accelerate research in the field of smart electric vehicle (EV) charging. It fills the need in this community for a widely available, realistic simulation environment in which researchers can evaluate algorithms and test assumptions. ACN-Sim provides a modular, extensible architecture, which models the complexity of real charging systems, including battery charging behavior and unbalanced three-phase infrastructure. It also integrates with a broader ecosystem of research tools. These include ACN-Data, an open dataset of EV charging sessions, which provides realistic simulation scenarios, and ACN-Live, a framework for field-testing charging algorithms. It also integrates with grid simulators like MATPOWER, PandaPower and OpenDSS, and OpenAI Gym for training reinforcement learning agents.**

***Index Terms*—Electric vehicles, computer simulation, charging stations, distributed energy resources, open-source software, cyber-physical systems.**

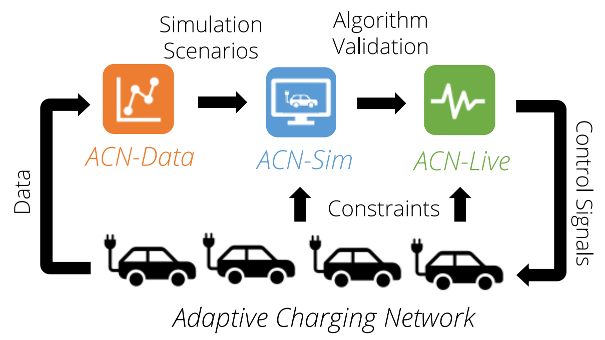

Fig. 1. The ACN Research Portal gives users many of the benefits of an EV charging testbed without needing to build one themselves. It includes data collected from real charging sessions (ACN-Data), a simulator to evaluate new ideas (ACN-Sim), and access to run on real hardware (ACN-Live).

I. Introduction

With millions of electric vehicles (EVs) expected to enter service in the next decade, generating gigawatt-hours of additional energy demand, engineers must work quickly to develop new algorithms to provide safe and affordable charging at scale. This need has resulted in a large body of research in managed or smart charging algorithms, outlined in [1], [2]. However, transitioning these algorithms from theory to practice requires dealing with the complexities of practical systems, which are often overlooked in simplified theoretical models. While these simpler models can make analysis tractable, they can also lead to a sizable gap between theoretical results and robust, high-performance implementations of algorithms. Bridging this gap is critical to making an impact in practice, but doing so requires (1) access to real-world data, (2) detailed simulations driven by realistic models, and (3) the ability to test an algorithm in the field.

We began to bridge this gap in 2016 by developing the Caltech Adaptive Charging Network (ACN), a first-of-its-kind testbed for large-scale, high-density EV charging research [3], [4]. This testbed consists of 126 networked and controllable EV charging stations, which allow us to collect data and field test algorithms with real hardware. The ACN Research Portal (ACN-Portal) is designed to give more researchers access to the benefits of this testbed. The portal has three major components: (1) ACN-Data, a dataset of over 80,000 real EV charging sessions from ACNs like the one at Caltech [5]; (2) ACN-Sim, an open-source, data-driven simulation environment; and (3) ACN-Live, a framework for researchers to field test algorithms on the Caltech ACN. The interaction between these tools and the physical ACN infrastructure is summarized in Fig. 1.

In this work, we will focus on ACN-Sim, which is open-source and available at [6]. ACN-Sim has previously been described in [7], [8]. However, in this work, we extend this description with new features and applications. Our contributions are as follows.

1) We describe the architecture and models of ACN-Sim and its integration with MATPOWER, PandaPower, OpenDSS, and OpenAI Gym.
2) We describe how algorithms are implemented using ACN-Sim, including a suite of baseline deadline-scheduling algorithms and a new module for model-predictive control, making it easy to implement algorithms like those proposed in [4].

Manuscript received November 4, 2020; revised March 28, 2021 and July 3, 2021; accepted July 6, 2021. Date of publication August 9, 2021; date of current version October 21, 2021. This work was supported in part by the National Science Foundation under the Graduate Research Fellowship Program under Grant 1745301, in part by NSF ECCS under Grant 1932611 and Grant 1619352, in part by NSF CCF under Grant 1637598, and in part by NSF CPS under Grant 1739355. It also received support under the Resnick Sustainability Institute Graduate Fellowship. *(Corresponding author: Zachary J. Lee.)*

Zachary J. Lee, Sunash Sharma, and Steven H. Low are with the Division of Engineering and Applied Science, Caltech, Pasadena, CA 91125 USA (e-mail: zlee@caltech.edu).

Daniel Johansson is with the Faculty of Engineering, Lund University, 221 00 Lund, Sweden.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2021.3103156.

Digital Object Identifier 10.1109/TSG.2021.3103156

3) We demonstrate, using ACN-Sim, that managed EV charging allows charging systems to safely operate with significantly smaller transformers/interconnections and at lower costs than conventional uncontrolled systems.
4) We compare algorithm performance using ACN-Sim, including the effect of unbalanced three-phase infrastructure, which has not been considered in most smart charging research.
5) We evaluate the effect that managed EV charging and on-site solar generation will have on a distribution feeder, using ACN-Sim and its integration with OpenDSS.

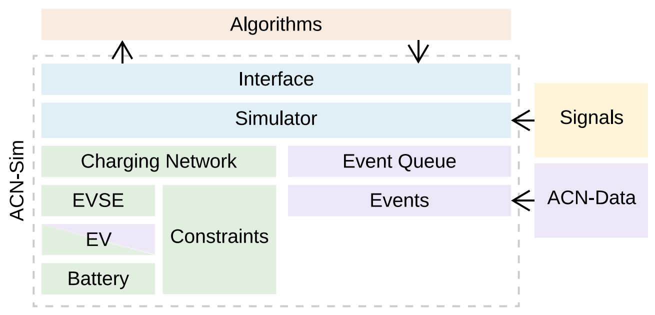

Fig. 2. Architecture of ACN-Sim along with related sub-modules Signals, Algorithms, and ACN-Data. Note that `EV` models both the physical vehicle and session information, such as departure time and energy requested.

II. Existing Simulators

Open-source tools and simulators have a long history of supporting smart grid research. MATPOWER [9] makes it easy to solve power flow and optimal power flow problems in MATLAB. It has inspired projects in other languages, including PandaPower [10] in Python and PowerModels.jl in Julia [11]. Other important simulators include OpenDSS [12] and GridLab-D [13], which enable large-scale studies of the distribution system. These tools have demonstrated the importance and impact of open tools within the smart grid community. ACN-Sim integrates with many of these, including MATPOWER, PandaPower, and OpenDSS, to enable studies of the grid impacts of EV charging.

ACN-Sim is not the first simulator specific to EV charging. V2G-Sim was developed at Lawrence Berkeley National Laboratory and used to evaluate EVs' ability to meet drivers' mobility needs in the context of level-1 charging [14]; battery degradation [15]; and demand response [16]. V2G-Sim has also been used to examine grid-level effects of smart charging, such as smoothing the duck curve[17]. EVLib and EVLibSim were developed at Aristotle University of Thessaloniki to model many types of EV charging, including standard conductive charging, inductive charging, and battery swapping [18]. These simulators address a different problem space from ACN-Sim. While ACN-Sim is designed to evaluate online and closed-loop control strategies, these simulators only allow precomputed schedules or simple controls. ACN-Sim is also unique in modeling unbalanced, behind-the-meter electrical infrastructure, allowing it to evaluate algorithms that support oversubscribed local infrastructure.

More recently, the Open Platform for Energy Networks (OPEN) from Oxford was released to facilitate simulation and optimization of smart local energy systems, including electric vehicle charging [19]. OPEN supports model predictive control algorithms at the distribution feeder level and unbalanced three-phase infrastructure. It also allows for control of other distributed energy resources such as stationary storage and building loads. However, it has not been used to consider the electrical infrastructure behind the meter.

Despite these open-source tools, many researchers still utilize custom simulators, which are often simple MATLAB or Python scripts. Building a custom simulator takes time and distracts researchers from focusing on their research questions. The history of MATPOWER and other open-source projects has shown that researchers can accomplish more by using open-source tools. Moreover, one-off simulators can be error-prone. ACN-Sim has over 12,000 unit and integration tests, which help ensure that bugs are caught before they affect research results. Finally, reproducibility is key in modern research. Using an open-source simulator like ACN-Sim, researchers can easily share code and use original implementations as baselines to compare against.

Overall, ACN-Sim's realistic models and data taken from real charging systems, along with its simple interfaces for defining new control algorithms and a suite of baseline algorithms, set it apart from existing open-source and custom simulators, making it a useful addition to the suite of tools available to researchers.

III. Simulator Architecture and Models

ACN-Sim utilizes a modular, object-oriented architecture which is shown in Fig. 2. This design models physical systems as closely as possible and makes it easier to extend the simulator for new use cases. Each box in Fig. 2 refers to a base class that can be extended to model new behavior or add functionality. While ACN-Sim includes several models of each component, users are free to customize the simulator to meet their needs. We encourage researchers to contribute extensions back to the project so that others can utilize them.

A. *Simulator*

A `Simulator` object forms the base of any ACN-Sim simulation. This `Simulator` holds models of the hardware components in the simulated environment and a queue of events that define when actions occur in the system. ACN-Sim is based on a discrete-time, event-based simulation model. Figure 3 describes its operation. During a simulation, the `Simulator` stores relevant data, such as the event history, EV history, and time series for the pilot signal and charging current for each EVSE,[1] for later analysis.

Mathematically speaking, we denote each time step of the simulator as k in $\mathcal{K} := \{1, 2, 3, \ldots, \}$. We denote the set of EVs in the simulation by $\hat{\mathcal{V}}_{all}$, the set of EVs currently plugged in by $\hat{\mathcal{V}}_k$, and the set of active EVs which are still charging by $\mathcal{V}_k$. The state of each EV at time k is described by a tuple $(a_i,\ e_i(k),\ d_i(k),\ \bar{r}_i(k), s_i)$ where a_i is the arrival time of the

[1]EVSE stands for Electric Vehicle Supply Equipment, they are more commonly known as charging stations or charging ports.

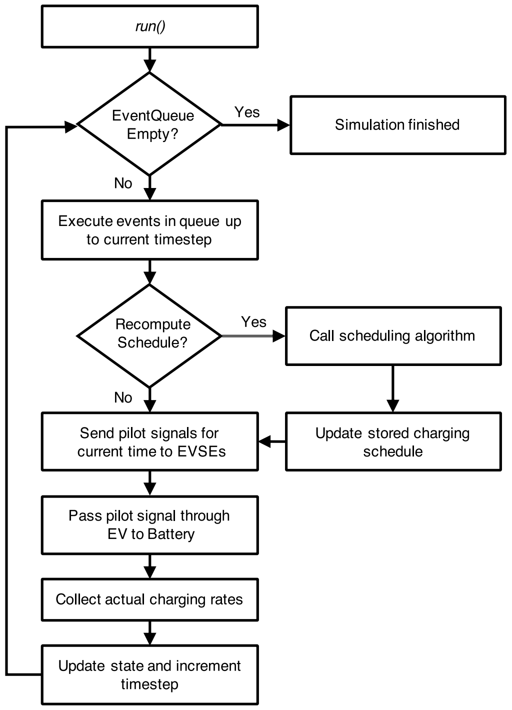
run()
EventQueue Empty?
Yes
Simulation finished
No
Execute events in queue up to current timestep
Recompute Schedule?
Yes
Call scheduling algorithm
No
Send pilot signals for current time to EVSEs
Update stored charging schedule
Pass pilot signal through EV to Battery
Collect actual charging rates
Update state and increment timestep

Fig. 3. Flow chart describing the simulator's `run()` function. Each timestep consists of a single iteration of this loop. The simulation ends when the last event from the `EventQueue` is executed, at which time the user can analyze the simulation results.

EV, $e_i(k)$ is its remaining energy demand at the beginning of the period, $d_i(k)$ is the remaining duration of the session, $\bar{r}_i(k)$ is the maximum charging rates for EV i, and s_i is the EVSE where the EV chooses to charge. We generally treat the charging network as stateless, so the state of the simulator at time k is simply the concatenation of the states of all currently charging EVs, i.e., $i \in \mathcal{V}_k$.

The simulator's action space is the pilot signal for each EV, which we denote $r_i(k)$. This is an upper bound on the charging rate of the EV. This action space is constrained by the charging network's infrastructure, the EVSE's limits, the EV's maximum charging rate $\bar{r}_i(k)$, and energy requested by the EV.

After taking an action, we can observe from the environment the actual charging rate of the EV, $\hat{r}_i(k)$ and the actual energy delivered to the EV $\hat{e}_i(k)$.

The state is updated according to the rule:

$$d_i(k+1) := d_i(k) - 1 \tag{1}$$
$$e_i(k+1) := e_i(k) - \hat{e}_i(k) \tag{2}$$

The maximum charging rate of the EV, $\bar{r}_i(k)$ is not observable, but can be estimated based on $\hat{r}_i(k)$. EVs can also arrive and depart at the beginning of each timestep.

B. Charging Network

1) Electrical Infrastructure: ACN-Sim uses the `ChargingNetwork` class to model the electrical infrastructure of the charging system, including EVSEs, transformers, switch panels, and cables. Each `ChargingNetwork` instance contains a set of `EVSE` objects, as well as a set of constraints.

We model constraints by limiting the current through each bottleneck component in the network. Because charging systems are radial networks and electrical codes specify ampacity limits that keep voltages within specifications, it is sufficient to model only constraints on current magnitudes. Using Kirchhoff's Current Law, we can express these constraints by

$$|I_l(k)| = \left| \sum_{i \in \mathcal{V}_k} A_{li} r_i(k) e^{j\phi_i} \right| \leq c_l \quad \forall k \in \mathcal{K} \tag{3}$$

where $I_l(k)$ is the current through the bottleneck, c_l is the limit on the current magnitude, $r_i(k)$ is the charging current of `EV` i at time k. The parameter ϕ_i is the phase angle of the current phasor, which can be calculated based on how EVSE s_i is connected in the network. For simplicity, we assume ϕ_i is fixed, and voltages in the network are nominal. A_{li} can be found via circuit analysis, as shown in [20] for a subset of the Caltech ACN. ACN-Sim will allow (3) to be violated in the simulation but will raise a warning at run-time to alert the user this schedule would not be valid on a real system. This allows the user to evaluate the severity of such an overload.

To incorporate these constraints, algorithms can either parse the constraints and include them directly in the algorithm, as is done in model predictive control, or use the built-in `is_feasible()` method, which returns if the proposed charging rates are allowable under the given network model.

2) Stochastic Space Assignment: `ChargingNetwork` assumes that each EV is preassigned to a specific EVSE, and no two EVs are ever assigned to the same EVSE at the same time. This holds when applying a workload from ACN-Data to its corresponding network model. However, in some cases, such as when generating events from a statistical model or applying a real workload to a new network configuration, it can be helpful to allow for non-deterministic space assignments. ACN-Sim accomplishes this through the `StochasticNetwork` class (which is a subclass of `ChargingNetwork`). Using this network model, EVs are assigned to a random open EVSE when they arrive instead of using a predefined `station_id` (s_i in the mathematical model) for assignment. Since it is possible for no EVSEs to be available when a new EV arrives, `StochasticNetwork` also includes a waiting queue for EVs which arrive while all EVSEs are in use. When an EV leaves the system, the first EV in the queue takes its place. By default, we assume that the presence of EVs in the waiting queue does not affect drivers' departure times. However, with the `early_departure` option, drivers swap places with the first EV in the queue as soon as they finish charging. This is a common practice in many offices that have more EV drivers than EVSEs.

3) Included Site Models: While users are free to develop their own charging networks, ACN-Sim includes functions to generate network models that match the physical infrastructure of the three sites currently included in ACN-Data (Caltech, JPL, and Office001). In addition, the `auto_acn` function allows users to quickly build simple single-phase and three-phase networks by providing just a list of station ids and a transformer capacity. In these `auto_acn` networks, it is assumed that the transformer is the only source of constraints. These functions work with both `ChargingNetwork` and `StochasticNetwork`, which can be set as a parameter.

C. EVSE

EVSEs, short for Electric Vehicle Supply Equipment, are the devices EVs plug into to charge. The EVSE communicates a pilot signal to the EV's onboard charger, which is an upper limit on the current the EV is allowed to draw from the EVSE. The granularity of this pilot is dependent on the particular EVSE. Some EVSEs provide continuous control, while others offer only a discrete number of set-points. In addition, according to the J1772 standard, no pilot signals are allowed between 0 to 6 A [21]. In most current research, the additional constraints imposed by EVSEs without continuous control are neglected [1]. However, including these constraints is important for practical algorithms and is non-trivial.

ACN-Sim provides three EVSE models that cover most ideal and practical level-2 EVSEs.

- `EVSE` allows any pilot signal between an upper and lower bound. By default, `EVSE` allows any non-negative charging rate.
- `DeadbandEVSE` also allows continuous pilots but excludes 0 - 6 A as required by the J1772 standard.
- `FiniteRatesEVSE` only allows pilot signals within a finite set, accurately modeling most commercial EVSEs. For example, many of the EVSEs used in the Caltech ACN allow $\{6, 7, \ldots, 31, 32\}$ or $\{8, 16, 24, 32\}$ amps.

Within ACN-Sim, `EVSE` is also the interface between the charging network and an `EV`. When an `EV` plugs into the system, a reference to that `EV` is added to the corresponding `EVSE`. When it is time to update the pilot to an `EV`, the `Simulator` first passes the pilot to the `EVSE`, which in turn passes it on the `EV` and eventually the `Battery`. This mimics the flow of information in a real charging system. Similarly, when an `EV` leaves the system, the reference to that `EV` is removed from the `EVSE`.

D. EV

The `EV` object contains relevant information for a single charging session, such as arrival time, departure time, estimated departure time, and requested energy. The estimated departure time may differ from the actual departure time. Likewise, it may be infeasible to deliver the requested energy in the allotted time due to maximum charging rate restrictions, system congestion, or insufficient battery capacity. By allowing this, ACN-Sim models the case where user inputs or predictions are inaccurate, which is common in practice [5].

E. Battery

Most EV charging research utilizes an ideal battery model, where EVs are assumed to follow the given pilot signal exactly. However, in practice, we see that the charging rate of an EV is often strictly lower than the pilot signal and decays as the battery approaches 100% state-of-charge [1], [4]. This can significantly increase the total time required to charge the battery and results in under-utilization of infrastructure capacity.

ACN-Sim jointly models the vehicle's battery and battery management system. The battery's actual charging rate depends on the pilot signal and the vehicle's onboard charger, state-of-charge, and other environmental factors. ACN-Sim currently includes two battery models.

The `Battery` class is an idealized model and serves as the base for all other battery models. The actual charging rate of the battery, $\hat{r}(k)$, in this idealized model is described by

$$\hat{r}_i(k) := \min\{r_i(k),\ \bar{r}(k)_i,\ \bar{e}_i(k)\}$$

where $r_i(k)$ is the pilot signal passed to the battery, $\bar{r}_i(k)$ is the maximum charging rate of the on-board charger, and $\bar{e}_i(k)$ is the difference between the capacity of the battery and the energy stored in it at time k in the units of A·periods. We do not consider discharging batteries, so all rates are positive.

`Linear2StageBattery` is an extension of `Battery` that approximates the roughly piecewise linear charging process used for lithium-ion batteries, often referred to as Constant Current - Constant Voltage (CC-CV) charging. The first stage, referred to as *bulk charging*, typically lasts from 0% to between 70 to 90% state-of-charge. During this stage, the current draw, neglecting changes in the pilot, is nearly constant. In the second stage, called *absorption*, the battery's voltage is held constant while the charging current decreases roughly linearly. The actual charging rate of the `Linear2StageBattery` is given by

$$\hat{r}_i(k) := \begin{cases} \min\{r_i(k),\ \bar{r}_i(k),\ \bar{e}_i(k)\} & \text{if } \mathtt{SoC}_i \leq \mathtt{th} \\ \min\left\{(1 - \mathtt{SoC}_i)\frac{\bar{r}_i(k)}{1-\mathtt{th}},\ r_i(k)\right\} & \text{otherwise} \end{cases}$$

where $\mathtt{SoC}_i$ is the state-of-charge of the battery and $\mathtt{th}$ marks the transition from the *bulk* stage to the *absorption* stage of the charging process. Figure 4 shows how these two models compare for two charging profiles taken from ACN-Data.

We find that while the piecewise linear model is a good approximation, it does not capture all the battery/BMS behaviors we observe in practice (as in the right panel of Fig. 4). For many experiments, the Linear2Stage model is sufficient. However, for evaluating the interactions between battery management systems and smart charging or studying the effect of smart charging on battery aging, more advanced models may be needed. ACN-Sim's modular architecture allows new battery/BMS models to be easily implemented as subclasses of `Battery`. By doing this, a researcher can implement various battery/BMS systems such as electrochemical models [22], electro-thermal and aging models [23], or manufacturer specification (spec) based models [24], [25].

F. Event Queue/Events

ACN-Sim uses events to describe actions in the simulation. There are two types of events currently supported.

- `PluginEvent` signals when a new EV arrives at the system. A `PluginEvent` also contains a reference to the `EV` object which represents the new session.

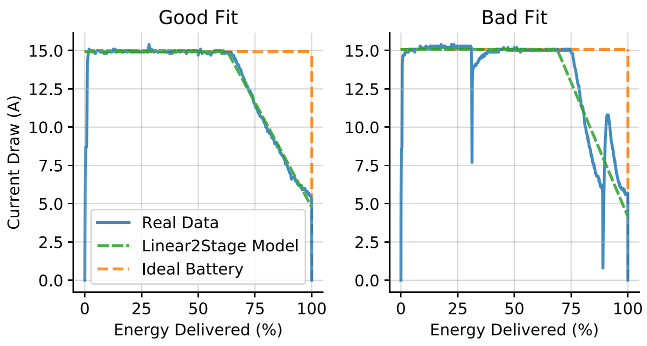

Fig. 4. Comparison of `Linear2Stage` and idealized `Battery` models with a real charging curve collected from two distinct users of the Caltech ACN when the pilot signal is not binding. We can see that the `Linear2Stage` model with appropriate parameters matches the battery behavior well in the first case, but in the second case, there are dynamics in the joint battery/battery management system that the `Linear2Stage` model does not capture; namely the double-tail behavior (which is recurring for this user).

- `UnplugEvent` signals when an EV leaves the system at the end of its charging session.

Each event has a timestamp describing when the event should occur. Events are stored in a queue sorted by their timestamp. Since multiple events could occur at the same timestep, we further sort by event type, first executing `UnplugEvents`, then `PluginEvents`. At each timestep, the `Simulator` executes all events left in the queue with timestamps on or before the current timestep. After any event, the scheduling algorithm is called to adapt to the new system state. Users are free to create new events by extending the `Event` class.

To generate events, users can either get real event sequences from ACN-Data, generate event sequences from statistical models, or manually create events to investigate edge cases. To make accessing ACN-Data simpler for users, ACN-Sim provides direct integration with the ACN-Data API. This allows the user to specify a site and date range, and ACN-Sim will gather the actual workload from that ACN and generate the appropriate `PluginEvents` and `UnplugEvents`. ACN-Sim also provides utilities for learning statistical models such as Gaussian Mixture Models, directly from data using tools from `scikit-learn` as described in [5].

Researchers could also generate events through co-simulation of the transportation or power network. For example, a `PluginEvent` might be generated by a transportation model that calculates user's arrival time and energy demands based on their travel patterns taken from GPS data or travel surveys. ACN-Sim can also be used to address routing/station assignment problems. To do this, the decisions of the assignment problem can be used with models of vehicle energy use and traffic to generate plugin and unplug events. Depending on the particular problem setting, events could be precomputed or generated during the ACN-Sim simulation. Likewise, a power network simulator might generate a demand response event when a distribution transformer's loading becomes too high. These are not included in the current release of ACN-Sim but are planned as future work.

The above models for generating `PluginEvents` and `UnplugEvents` assume that driver behavior is fixed. In reality, charging operators interact with drivers who may strategically respond to scheduling algorithms or pricing schemes by adjusting their arrival time, departure time, or energy request. While not currently included, we plan to incorporate strategic driver models in a future release of ACN-Sim.

G. *Signals*

The signals sub-module allows ACN-Sim to integrate with external signal sources, which can be an important part of EV charging systems such as 1) utility tariffs, 2) solar generation curves, 3) external loads.

1) Utility Tariffs: Operating costs are an important concern for EV charging facilities. To support utility tariffs, ACN-Sim includes the `TimeOfUseTariff` class, which supports time-varying and seasonal tariff schedules with or without demand charges. To make integration easier for users, ACN-Sim includes several utility tariff schedules. Users can define new schedules in a simple JSON format. This functionality allows users to investigate cost minimization strategies and accurately estimate the operating costs of charging system designs under different tariff structures. In Section V-A, we use these tariff schedules to calculate operating costs, and in Section V-B, we provide the tariff as an input to the minimum cost objective for the MPC algorithm.

2) Solar Generation: As many sites with EV charging also have on-site solar generation, studying the behavior of an EV charging facility that takes solar generation into account is an important use case. ACN-Sim allows users to input a solar generation signal as a CSV file to the Simulator. Solar data can be user-generated, downloaded from an external source, or generated by an external solar generation simulator such as NREL's system advisor model (SAM) [26]. Such functionality allows users to study the effects of on-site solar on cost, energy demands met, grid loading, and other metrics associated with large-scale EV charging.

3) External Load: EV charging facilities often share a meter with other loads, such as the buildings on a university or corporate campus. To reduce demand charge and stress on the grid, it can be advantageous to consider these other loads when scheduling EV charging. To facilitate the study of algorithms that do this, ACN-Sim allows users to input an external load profile as a CSV file. External load data can be user-generated or downloaded from an external source. Section V-D provides an example of an experiment using external loads and on-site solar generation.

4) Other Signals: Other signals such as pollution indexes or demand response profiles can be loaded into the simulator using the `signals` dictionary within the `Simulator` constructor or passed directly to the control algorithm.

H. *Co-Simulation With Grid Simulators*

ACN-Sim also provides co-simulation with popular grid-level simulators, including MATPOWER, PandaPower, and OpenDSS. This allows researchers to investigate Vehicle-Grid Integration (VGI) problems such as algorithms to alleviate voltage and overload issues in the local distribution system or aggregation approaches to bid into markets. In the current version, simulations are run sequentially, with the output of the ACN-Sim experiment serving as an input to the grid simulator. In future releases, we plan to support feedback from the grid simulation into ACN-Sim.

I. OpenAI Gym Integration

Reinforcement learning (RL) has long been applied to problems in scheduling and resource allocation [27]. Smart EV charging is a particularly interesting application for RL as it involves a complex and uncertain environment with large state and action spaces and safety-critical constraints. To help researchers apply new and existing RL algorithms to problems in EV charging, we have integrated ACN-Sim with OpenAI's Gym package [28]. Gym uses a standard `Environment` interface to make it easy for researchers to apply RL algorithms to problems ranging from video games to robotics.

To integrate ACN-Sim with Gym we implement this `Environment` interface to wrap an ACN-Sim `Simulator`. The `gym-acnportal` package also provides an expanded `Interface` object (see Section IV-A), which allows the simulation to proceed one time-step at a time during training. At each step, the agent receives a state, which is a partial observation of the environment. This consists of the concatenation of each EV's parameters, as described in Section III-A, along with the index of the timestep, k, and network constraints C, where

$$C := (A_{li}, \phi_i, c_l) \quad \forall l \in \mathcal{L}, \quad \forall i \in \mathcal{V}$$

The agent also receives a reward from its previous action. This reward can be customized to the particular objective of the agent. For example, a simple reward for an agent whose only objective is to charge vehicles without violating infrastructure constraints might be:

$$R(k) := u^{ED} - \eta u^{CV} \tag{4a}$$

where

$$u^{ED}(k) := \sum_{i \in \mathcal{V}} e_i(k) - e_i(k-1) \tag{4b}$$

$$u^{CV}(r, k) := \max\left(0, \sum_{l \in \mathcal{L}} \left| \sum_{i \in \mathcal{V}} A_{li} r_i(k) e^{j\phi_i} \right| - c_l\right)^2 \tag{4c}$$

Here (4b) rewards the agent for the energy it delivered in time step k, while (4c) penalizes the agent for any constraint violation. We use the coefficient η to adjust the magnitude of this penalty.

We also provide a vector of the individual constraint violations (i.e., each term of the exterior summation in (4c)) as an entry in the `info` dict returned by the `step()` function to aid those researchers studying algorithms for constrained Markov decision processes (CMDPs) [29].

After receiving the observation, reward, and constraint violation costs, the agent then calculates an action, which here is a charging rate for each EV. To keep the size of the state and action spaces consistent, we pad the state with additional triplets of 0's for each EVSE without an EV plugged in, such that the number of $(e_i(k), d_i(k), \overline{r}_i(k))$ triplets is always equal to the number of EVSEs in the network. We also pad schedules (actions) with 0's for each EVSE without an EV. This action is then passed to the step function, and the process repeats. The reset function allows the environment to return to a known state after a training episode. After training an RL agent, the `gym-acnportal` package also provides a wrapper that allows the agent to be deployed as an ACN-Sim algorithm using the standard `Interface.`

In addition to customizing the reward and constraint violation functions, researchers can also adjust the environment's state and action space. For example, in [30] the action space is a set of parameters for an additive-increase, multiplicative-decrease (AIMD) algorithm. Additions to the state space might include congestion metrics, prices, or renewable generation forecasts.

IV. Charging Algorithms

A. Interface

To make algorithm implementations more flexible, we introduce an interface that abstracts away the underlying infrastructure, whether simulated or real, allowing us to use the same algorithm implementation with both ACN-Sim and ACN-Live. This means that algorithms can be thoroughly tested with ACN-Sim before they are used on physical hardware. It also means algorithms developed to work with ACN-Sim can work with other platforms simply by extending the `Interface` class.

B. Defining an Algorithm

To define an algorithm in ACN-Sim users only need to extend the `BaseAlgorithm` class and define the `schedule()` function. This function takes in a list of active sessions, meaning that the EV is plugged in and its energy demand has not been met and returns a charging schedule for each. This schedule is a dictionary that maps `station_id` to a list of charging rates in amps. Each entry in the schedule is valid for one timestep beginning at the current time. Algorithms have access to additional information about the simulation through the `Interface` class, such as the current timestep, infrastructure constraints, and allowable pilot signals for each EVSE.

C. Included Algorithms

ACN-Sim is packaged with many common online scheduling algorithms that can be used as benchmarks.

Uncontrolled Charging: Most charging systems today do not manage charging. With Uncontrolled Charging, each EV charges at its maximum allowable rate. This algorithm does not factor in infrastructure constraints, so they may be violated.

Round Robin: Round Robin (RR) is a simple algorithm that attempts to share charging capacity equally. It creates a queue of all active EVs. For each EV in the queue, it checks if it is feasible to increment its charging rate by one unit. If it is, it increments the rate and replaces the EV at the end of the queue. If it is not, the charging rate of the EV is fixed, and the algorithm does not return the EV to the queue. This continues until the queue of EVs is empty. In this context, a feasible charging rate is one that does not cause an infrastructure constraint to be violated and is less than the maximum charging rate, $\bar{r}_i(k)$, and the energy demand, $e_i(k)$, of the EV.

Sorting Based Algorithms: Sorting based algorithms are commonly used in other deadline scheduling tasks such as job scheduling in servers due to their simplicity [31]. ACN-Sim includes several of these algorithms, including First-Come First-Served (FCFS), Last-Come First-Served (LCFS), Earliest-Deadline First (EDF), Longest Remaining Processing Time (LRPT), and Least-Laxity First (LLF). These algorithms work by first sorting the active EVs by the given metric, then processing them in order. Each EV is assigned its maximum feasible charging rate, which is calculated using a bisection algorithm, given that the assignments to all previous EVs are fixed. This process continues until all EVs have been processed. For any algorithm which uses departure time, e.g., EDF and LLF, estimated departure time is used. The researcher is left to decide the accuracy of the estimated departure time.

Model Predictive Control: Many approaches to the EV scheduling problem rely on model predictive control (MPC). In [4] we present the Adaptive Scheduling Algorithm (ASA), which is one example of an MPC algorithm for managed EV charging applications. Every time period, k, we solve an optimization problem in the form:

$$\max_{r} \quad U(r) \tag{5a}$$

$$\text{s.t.} \quad 0 \leq \; r_i(t) \; \leq \; \bar{r}_i(k) \qquad t \leq \tilde{d}_i(k), i \in \mathcal{V} \tag{5b}$$

$$r_i(t) = 0 \qquad t > \tilde{d}_i(k), \; i \in \mathcal{V} \tag{5c}$$

$$\sum_{t \in \mathcal{T}} r_i(t) \leq e_i(k) \qquad i \in \mathcal{V} \tag{5d}$$

$$\left| \sum_{i \in \mathcal{V}} A_{li} r_i(t) e^{j\phi_i} \right| \leq c_l \quad t \in \mathcal{T}, l \in \mathcal{L} \tag{5e}$$

where $t \in \mathcal{T}$ are time-steps within the optimization horizon and is the $\tilde{d}_i(k)$ is the estimated remaining session duration for EV i.

An open-source implementation of the ASA algorithm is available in the adacharge package [32], which is based on CVXPY [33], [34], and makes it easy to use variations on ASA with ACN-Sim. With this library, users can easily choose from existing objective functions and constraints or create their own. The general framework for these MPC algorithms is outlined in [4].

V. Use Cases

ACN-Sim has been used to explore many research questions. In this section, we provide examples, including evaluating (1) possible infrastructure solutions, (2) the effect of unbalance on oversubscribing infrastructure, (3) time-series of EV charging profiles, and (4) the effect of large-scale EV charging on a distribution feeder. In addition, ACN-Sim has been used to design dynamic pricing schemes and cost-optimal scheduling [35], train reinforcement learning agents for EV charging systems [3], and examine the effect of non-ideal batteries and EVSE pilot quantization on model predictive control and baseline algorithms [4]. The code for all case studies presented here is available at [36].

A. System Planning

In this section, we demonstrate how the simulator can be used to aid in system planning and design. We consider a site host who would like to install an EV charging solution at an office building. The host estimates that the system will charge approximately 100 EVs per day. There are several ways to meet this demand.

1) Install 102 level-1 EVSEs with a maximum charging power of 1.9 kW with a 200 kW transformer.
2) Install 102 level-2 EVSEs with a maximum charging power of 6.6 kW with a 680 kW transformer.
3) Install 30 level-2 EVSEs with the 200 kW transformer.
4) Install 102 level-2 EVSEs with the 200 kW transformer and use smart charging algorithms to avoid overloading the transformer.

We can use ACN-Sim to guide this site host. We assume that the office will have a usage pattern similar to that of JPL.[2] As such, we train a Gaussian Mixture Model based on the data collected from weekday usage at JPL, as described in [5]. We assume the site will not allow usage on weekends. We then use ACN-Sim's `GaussianMixtureEvents` tool to create a queue of events from this generative model, assuming 100 arrivals on weekdays and 0 on weekends. We also create models of the charging networks described in each proposal. Since EVs are generated, we use the `StocasticNetwork`, which randomly assigns EVs to EVSEs when they arrive. For proposals 1, 2, and 3, we use the built-in `Uncontrolled` charging algorithm. For proposal 4, we consider an MPC based algorithm based on (5) with cost minimization objective

$$U^{\text{CostMin}} := u^{EC} + u^{DC} + 10^{-6} u^{QC} + 10^{-12} u^{ES} \tag{6a}$$

where

$$u^{EC}(r) := \pi \sum_{\substack{t \in \mathcal{T} \\ i \in \mathcal{V}}} r_i(t) - \sum_{\substack{t \in \mathcal{T} \\ i \in \mathcal{V}}} \kappa(t) r_i(t) \tag{6b}$$

$$u^{DC}(r) \; := -\hat{P} \cdot \max\left(\max_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}} r(t), \; q_0, q' \right) \tag{6c}$$

$$u^{QC}(r) := \sum_{t \in \mathcal{T}} \frac{T - t + 1}{T} \sum_{i \in \mathcal{V}} r_i(t) \tag{6d}$$

$$u^{ES}(r) := -\sum_{\substack{t \in \mathcal{T} \\ i \in \mathcal{V}}} r_i(t)^2 \tag{6e}$$

Here (6b) contains the value and cost of energy. We use $\pi = 1000$ to ensure that the algorithm delivers as much energy as possible. $\kappa(t)$ is the time-varying cost of energy, for which we use summer rates from the `sce_tou_ev_4_march_2019` tariff schedule included in ACN-Sim. Equation (6c) is a demand charge proxy that prorates the demand over the

[2]JPL users pay a fixed price of \$0.10/kWh, so we will assume that this site will charge a similar (subsidized) price.

TABLE I
INFRASTRUCTURE SOLUTION EVALUATION (100 EV/DAY)

EVSEs (#)	EVSE Type	Alg	Transformer Capacity	Swaps (#/month)	Demand Met	Cost ($/kWh)
102	Level 1	Unctrl	200	0	75.4%	0.278
102	Level 2	Unctrl	680	0	99.9%	0.351
30	Level 2	Unctrl	200	1,103.5	99.6%	0.256
102	Level 2	MPC	200	0	99.8%	0.234

TABLE II
INFRASTRUCTURE SOLUTION EVALUATION (200 EV/DAY)

EVSEs (#)	EVSE Type	Alg	Transformer Capacity	Swaps (#/month)	Demand Met	Cost ($/kWh)
102	Level 1	Unctrl	200	1,174.5	73.2%	0.244
102	Level 2	Unctrl	680	1,081.5	99.8%	0.327
30	Level 2	Unctrl	200	2,973.9	91.6%	0.233
102	Level 2	MPC	200	1,441.9	87.1%	0.223
201	Level 2	MPC	200	0	98.4%	0.227

remaining days in the month. Equations (6d) and (6e) are regularizers that promote charging as quickly as possible and equal sharing of capacity, respectively. The coefficients for these regularizers are selected so that their impact on the objective value is small while still promoting desirable properties in the final schedule. For more information about this objective function, see [4]. For this, and all other experiments in this paper, we will assume perfect estimates of the session duration, e.g., $\tilde{d}_i(k) = d_i(k)$.

We evaluate the scenarios on four criteria: 1) transformer capacity required, 2) percentage of total energy requested that was delivered, 3) number of times drivers need to swap spaces to allow others to charge after they finish, and 4) the operating cost of the system. We repeat these experiments for ten months of generated data, with mean results shown in Table I. Note that the standard deviation between months was less than 3.5% for each metric in each case.

From Table I, we can see that while installing 100 level-1 EVSEs might be the simplest solution, these slow chargers can only meet 75.4% of demand because they cannot support users with large energy needs and short deadlines. However, the alternative of installing a 680 kW transformer and associated service upgrade would be cost-prohibitive for most sites, and installing only 30 level-2 EVSEs requires over 1,100 swaps per month, leading to lost productivity and poor user experience. In this case, the smart charging solution with model predictive control has clear advantages in both capital cost (only requiring a 200 kW transformer), user satisfaction (no swaps are necessary while nearly all user demands are met), and operating costs (having the lowest cost per kWh). This illustrates the benefits of smart charging systems, which we can quantify with ACN-Sim.

The benefit of smart charging approaches is amplified as EV adoption grows, and charging infrastructure must scale accordingly. In this scenario, we consider how the system will scale to 200 charging sessions per day. The results are shown in Table II. Intuitively the systems designed for 100 EVs per day require far more swaps with increased demand, and similarly, the percent of demand met decreases. This is also true for the smart charging (MPC) case. However, while scaling the number of EVSEs in traditional uncontrolled charging systems would require a corresponding scaling of the transformer capacity to ensure safety, the smart charging approach allows us to add new EVSEs without increasing the transformer capacity. To enable scalability, we can leave an open space beside each of the originals and install a second EVSE using the same cable. We then use the charging algorithm to ensure the capacity of this cable is not exceeded. In this experiment, we assume the cable was sized for a single EVSE (32 A). However, if the scale-out was planned, a larger cable could have been installed initially. Thus, we can easily scale the number of EVSEs without increasing transformer or interconnection capacity.

Interestingly, as the number of EVs served by the system increases, the effective cost per kWh decreases for all systems. This indicates the economies of scale associated with demand charge. With more usage, it is possible to spread the demand charge over more energy delivered, decreasing the price per kWh. This decrease in demand charge is greater than the increase in energy price, which results from needing to charge users in more expensive TOU periods, leading to a net decrease in per-unit costs.

B. Importance of Three-Phase Models

As we have seen in Section V-A, smart charging algorithms can lead to significant savings in terms of both capital investment and operating costs. However, despite significant work, there are still relatively few algorithms proposed in the literature which can be directly applied in practice. To develop more practical algorithms, ACN-Sim provides a platform to evaluate them in as realistic a setting as possible. A key feature of ACN-Sim is its ability to simulate the unbalanced three-phase electrical infrastructure common in large charging systems. Most charging algorithms in the literature rely on constraints that implicitly assume single-phase or balanced three-phase operation.

To see why these assumptions are insufficient for practical systems, we consider two versions of the LLF algorithm. In the first, LLF only ensures that the total power draw is less than the transformer's capacity, which is sufficient for a single-phase or balanced system. In the second, LLF uses the full three-phase system model that includes individual line constraints. This experiment's results are shown in Fig. 5, where we can see that only considering maximum power draw leads to significant constraint violations in line currents. However, by using an algorithm that considers the full three-phase model, we ensure these line constraints are not violated at the cost of not fully utilizing the 70 kW transformer's capacity due to unbalance.

This motivates us to consider algorithms that incorporate unbalanced three-phase constraints. These constraints are necessary to ensure safety and can significantly impact the performance of an algorithm. To see this, we will consider the percentage of user energy demands met when infrastructure constraints are binding. We use this metric to evaluate six algorithms over a range of possible transformer capacities based on the real charging workload of the Caltech

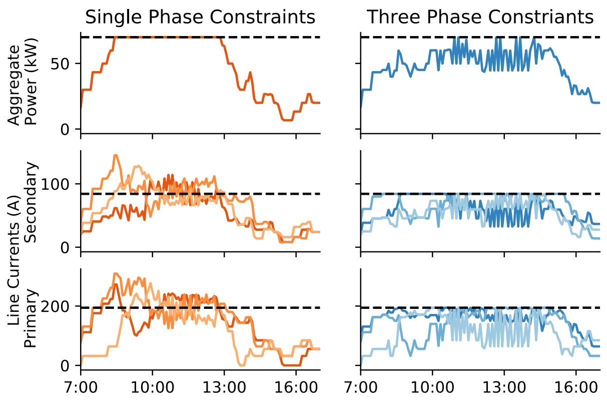

Fig. 5. Aggregate power draw and line-currents at the primary and secondary side of the transformer when running single-phase and three-phase LLF algorithms on the Caltech ACN with a 70 kW transformer capacity. Shading in the lower plots denote each phase while the black dotted line denotes the power/current limit. The experiment is based on data from the Caltech ACN on September 5, 2018 and uses a 5 minute timestep.

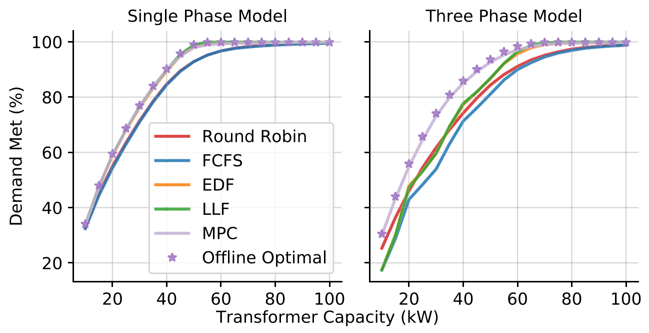

Fig. 6. Comparison of percentage of energy delivered as a function of transformer capacity for single-phase (left) and three-phase (right) systems. Stars represent the offline optimal, which is an upper bound based on perfect future information. The simulation runs from Sept. 1 through Oct. 1, 2018, with a timestep of 5 minutes. To generate events, we use ACN-Sim's integration with ACN-Data to get real charging sessions from the Caltech ACN, assuming the ideal battery model. We also use the included Caltech ACN charging network model with ideal EVSEs and use its optional `transformer_cap` argument to limit the infrastructure capacity. In the left plot, MPC, EDF, and LLF are nearly coincident, as are Round Robin and FCFS. Similarly, in the right plot, EDF and LLF overlap in most cases.

ACN from September 2018. To demonstrate the effect of infrastructure models, we conduct this experiment with single-phase and three-phase models, as shown in Fig. 6. Here we can see that in the single-phase case, EDF, LLF, and MPC (with objective (6a)) all perform near optimally,[3] exceeding the performance of Round Robin and FCFS by up to 8.6%. However, the subplot on the right tells a different story. Here we see that the MPC algorithm can match the offline optimal performance as before, while EDF and LLF both underperform. In fact, in the highly constrained regime, Round Robin outperforms EDF and LLF despite having less information about the workload. We attribute these results to the importance of phase-balancing in three-phase systems, which has been historically under-appreciated in the managed charging literature.

In addition to comparing algorithms, the curves in Fig. 6 can also inform charging systems' design when accounting for the online algorithm used. For example, we can see that if a host wants to deliver >99% of charging demand using MPC, a 70 kW transformer would be sufficient, assuming an unbalanced three-phase system. Alternatively, if an existing transformer can only support 40 kW of additional demand, a host could expect to meet approximately 85% of demands without an upgrade.

C. Time Series Inspection

ACN-Sim also allows us to examine the charging profile of individual EVs, as shown in Fig. 7. Here we can see a qualitative difference between the algorithms. For example, FCFS behaves similarly to Uncontrolled charging but is delayed as the EV must wait its turn in the queue. For EDF and LLF, charging can be interrupted when EVs with earlier deadlines arrive or as an EV's laxity evolves over time. Oscillations in the LLF plot result from an increase in laxity as the EV charges, which can decrease its standing in the queue, causing

[3]Here optimally is defined as the maximum amount of energy which could be delivered subject to constraints. It is found by solving (5) with perfect foresight for all EVs in the simulation. We use $U(r) = \sum_{i \in \hat{\mathcal{V}}_{all}, t \in \mathcal{T}} r_i(t)$.

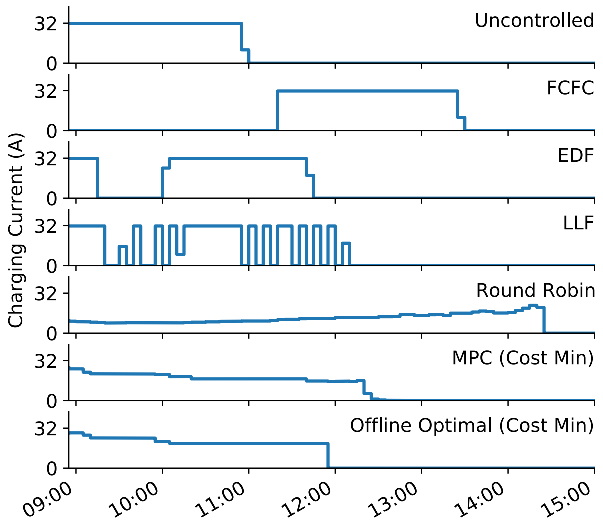

Fig. 7. Comparison of charging profiles for one EV on September 13, 2018 with a 70 kW transformer capacity.

it to stop charging temporarily. These oscillations are generally bad for user experience, preventing LLF from being used widely for smart charging. The smoothed LLF algorithm proposed in [37] adapts the LFF algorithm to prevent these oscillations. Round Robin, MPC, and the offline optimal are quite different. Each EV charges steadily but at a rate below its maximum as congestion in the system necessitates sharing charging capacity. Here both MPC and the offline optimal use objective (6a). With this tariff schedule, on-peak rates run from 12 - 6 pm. Offline Optimal finishes charging this user before this peak period. Meanwhile, MPC charges briefly in during the peak hours.

D. Grid Integration

The load profiles generated by ACN-Sim can also be used to evaluate the impact that large-scale EV charging has on

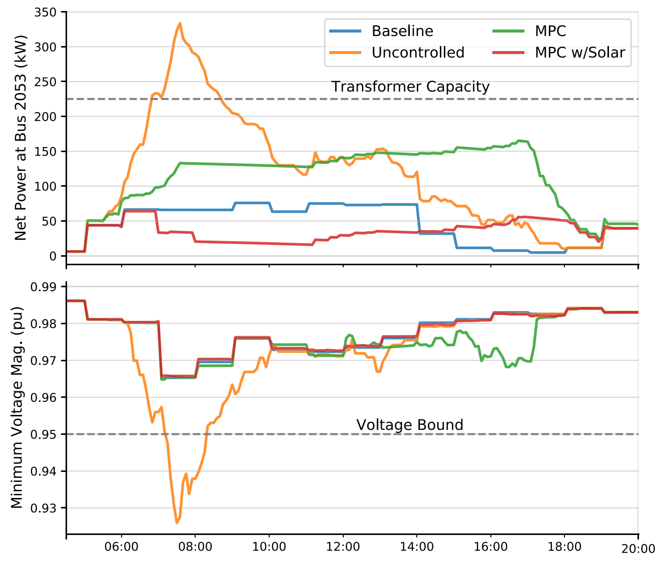

Fig. 8. Comparing the effect of charging scenarios on net power draw at Bus 2053 (top) and minimum system voltage (bottom). EV data taken from September 6, 2018 at the JPL ACN. Background load is taken from smart meter data from September 6, 2017. Solar PV production is estimated using NREL's SAM tool for a 225 kW AC PV array in Des Moines, Iowa.

a distribution feeder. In this case, we use OpenDSS [12] to add EV charging to one bus of a 240-bus distribution feeder located in the Midwest United States [38].[4]

We add an EV charging facility as an unbalanced three-phase load to bus 2053, which has a transformer capacity of 225 kVA. We use the JPL charging network with workload data from Sept. 5, 2019. Background load is from smart meter data collected Sept. 5, 2017 (both days were weekdays). To minimize the effect of EV charging on the grid, we will seek to flatten the load at the EV site. This prevents large spikes in demand which could cause voltage issues on the distribution feeder. While we only consider local load flattening in this example, ACN-Sim could also be used to investigate global load flattening based on the aggregate load on the feeder. We consider four cases, a baseline with no EV charging, uncontrolled charging, MPC with a load flattening objective, and MPC with load flattening and on-site solar. The load flattening objective is given by

$$U^{\text{LoadFlat}} := u^{LF} + 100u^{NC} + 10^{-3}u^{ES} \tag{7a}$$

where

$$u^{LF}(r) := \sum_{t\in\mathcal{T}}\left(\sum_{i\in\mathcal{V}} r_i(t) + N(t)\right)^2 \tag{7b}$$

$$u^{NC}(r) := \sum_{i\in\mathcal{V}}\left|\sum_{t\in\mathcal{T}} r_i(t) - e_i\right| \tag{7c}$$

Here $N(t)$ denotes the bus's net background load after subtracting on-site generation. The u^{NC} term is a non-completion penalty for failing to deliver all the energy requested by EVs. u^{NC} has a coefficient of 100 to encourage delivering all the demanded energy before flattening the load. This coefficient is selected empirically to ensure that $> 99.5\%$ of energy demands are met. We add the equal sharing term to ensure a unique solution to the optimization.

[4]This feeder includes a voltage regulator at the substation.

Because none of these algorithms use direct feedback from the grid simulator, we first run the ACN simulation for the full 24-hour horizon, then use this power draw as an input to OpenDSS. The results of these experiments are shown in Fig. 8. Uncontrolled charging results in an unacceptable minimum voltage of under 0.93 p.u. and overloads the transformer at bus 2053. This indicates that the grid as designed could not support uncontrolled charging at this scale at bus 2053.

To prevent voltage issues, we can schedule charging during periods of low background load by using MPC with objective (7a). We provide the actual building load as an input to the algorithm and ensure that the total load is constrained to be below the transformer's capacity. From Fig. 8, we can see that this improves the minimum system voltage to 0.965 p.u., which matches the system-wide minimum from the baseline.

Since many EV charging systems are co-located with solar PV, we also consider adding a 270 kW DC (225 kW AC) PV array at bus 2053. The solar data was generated from NREL's SAM tool for Des Moines, Iowa, in a typical meteorological year (TMY) for Sept. 5. We use the same MPC algorithm but now set the background load to the net load after subtracting solar. We see in Fig. 8 this roughly recovers the same grid-wide minimal voltage as before we added an ACN. This indicates that smart charging and solar PV could enable widespread adoption of EV charging without adverse grid impacts. These case studies assume perfect knowledge of background load and generation, as forecasting methods are beyond this study's scope. However, no knowledge of future EV arrivals is used.

VI. Conclusion

In this work, we present ACN-Sim, a data-driven simulator designed to aid in developing practical online scheduling algorithms for EV charging. This tool significantly reduces the software engineering burden on researchers and exposes them to practical issues present in real charging systems. ACN-Sim also makes it easier for researchers to share their experiment code, improving transparency and code reuse in the community. Finally, ACN-Sim integrates with the Adaptive Charging Network Research Portal, a larger suite of tools that includes a database of real charging sessions and a framework for field testing algorithms. ACN-Sim will continue to grow to meet the community's needs, including new models of systems components and charging networks.

Currently, ACN-Sim provides a realistic simulation environment for evaluating algorithms for controlling level-2 charging. Based on requests from users, we are currently working on extensions of ACN-Sim to model DC Fast Charging (DCFC) and stationary storage. We are also investigating models to generate `PlugIn` and `Unplug` events from vehicle GPS data or route maps.

Moving forward, we plan to continue to develop the simulator with the help of users. The long-term evolution of the tool will depend on what research questions users want to answer and how they modify the simulator to meet their needs. We are particularly interested in pursuing tighter integration with grid simulators to allow for grid-aware control schemes. Another interesting area is modeling strategic user behavior. Currently, our simulations assume that users will not change their behavior in response to charging algorithms or prices. This is likely not the case in practice. Instead, we should model users as strategic agents. This could open up entirely new lines of research where smart charging algorithms must account for this strategic behavior in users. We can also utilize ACN-Live to evaluate if our user models accurately represent the behavior we see in practice.

In the longer term, we plan to introduce physics-based battery models which capture battery degradation, add support for bi-directional, vehicle-to-grid modeling, and release a suite of representative test-cases to standardize algorithm evaluation.

Acknowledgment

The authors would like to thank the team at PowerFlex, especially Cheng Jin, Ted Lee, and George Lee, as well as Rand Lee, James Anderson, and Jorn Reniers, for providing data, expertise, and ideas to this project.

References

[1] Q. Wang, X. Liu, J. Du, and F. Kong, "Smart charging for electric vehicles: A survey from the algorithmic perspective," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1500–1517, 2nd Quart., 2016.

[2] J. C. Mukherjee and A. Gupta, "A review of charge scheduling of electric vehicles in smart grid," *IEEE Syst. J.*, vol. 9, no. 4, pp. 1541–1553, Dec. 2015.

[3] G. Lee, T. Lee, Z. Low, S. H. Low, and C. Ortega, "Adaptive charging network for electric vehicles," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Washington, DC, USA, Dec. 2016, pp. 891–895.

[4] Z. J. Lee *et al.*, "Adaptive charging networks: A framework for smart electric vehicle charging," *IEEE Trans. Smart Grid*, early access, Apr. 20, 2021, doi: 10.1109/TSG.2021.3074437.

[5] Z. J. Lee, T. Li, and S. H. Low, "ACN-data: Analysis and applications of an open EV charging dataset," in *Proc. 10th ACM Int. Conf. Future Energy Syst.*, 2019, pp. 139–149.

[6] Z. J. Lee, S. Sharma, and D. Johansson. (Nov. 2020). *Acnportal*. [Online]. Available: https://github.com/zach401/acnportal

[7] Z. Lee, D. Johansson, and S. H. Low, "ACN-sim: An open-source simulator for data-driven electric vehicle charging research," in *Proc. 10th ACM Int. Conf. Future Energy Syst.*, 2019, pp. 411–412.

[8] Z. J. Lee, D. Johansson, and S. H. Low, "ACN-sim: An open-source simulator for data-driven electric vehicle charging research," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids (SmartGridComm)*, Beijing, China, Oct. 2019, pp. 1–6.

[9] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

[10] L. Thurner *et al.*, "Pandapower—An open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 6510–6521, Nov. 2018.

[11] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "PowerModels.jl: An open-source framework for exploring power flow formulations," in *Proc. Power Syst. Comput. Conf. (PSCC)*, Jun. 2018, pp. 1–8.

[12] D. Montenegro, R. Dugan, R. Henry, T. McDermott, and W. Sunderm. *OpenDSS—EPRI Distribution System Simulator*. Accessed: Oct. 10, 2019. [Online]. Available: https://sourceforge.net/projects/electricdss/

[13] D. P. Chassin, J. C. Fuller, and N. Djilali, "GridLAB-D: An agent-based simulation framework for smart grids," *J. Appl. Math.*, vol. 2014, Jun. 2014, Art. no. 492320. [Online]. Available: https://www.hindawi.com/journals/jam/2014/492320/?utm_source=google&utm_medium=cpc&utm_campaign=HDW_MRKT_GBL_SUB_ADWO_PAI_DYNA_JOUR_X_PJ_GROUP4_Geostrategy&gclid=CjwKCAjwgviIBhBkEiwA10D2jwKhxngvRUzsUN8hQWtESBCr3jvAS-eOfCdw8KMc1Go_NU5xMQJpuxoC29oQAvD_BwE.

[14] S. Saxena, J. MacDonald, and S. Moura, "Charging ahead on the transition to electric vehicles with standard 120 v wall outlets," *Appl. Energy*, vol. 157, pp. 720–728, Nov. 2015.

[15] S. Saxena, C. Le Floch, J. MacDonald, and S. Moura, "Quantifying EV battery end-of-life through analysis of travel needs with vehicle powertrain models," *J. Power Sources*, vol. 282, pp. 265–276, May 2015.

[16] S. Saxena, J. MacDonald, D. Black, and S. Kiliccote, "Quantifying the flexibility for electric vehicles to offer demand response to reduce grid impacts without compromising individual driver mobility needs," SAE Techn. Paper, Warrendale, PA, USA, Rep. 2015-01-0304, 2015. [Online]. Available: https://doi.org/10.4271/2015-01-0304

[17] J. Coignard, S. Saxena, J. Greenblatt, and D. Wang, "Clean vehicles as an enabler for a clean electricity grid," *Environ. Res. Lett.*, vol. 13, no. 5, 2018, Art. no. 054031.

[18] E. S. Rigas, S. Karapostolakis, N. Bassiliades, and S. D. Ramchurn, "EVLibSim: A tool for the simulation of electric vehicles' charging stations using the EVLib library," *Simulat. Model. Pract. Theory*, vol. 87, pp. 99–119, Sep. 2018.

[19] T. Morstyn *et al.*, "OPEN: An open-source platform for developing smart local energy system applications," *Appl. Energy*, vol. 275, Oct. 2020, Art. no. 115397.

[20] Z. J. Lee *et al.*, "Large-scale adaptive electric vehicle charging," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids*, Anaheim, CA, USA, Oct. 2018, pp. 1–7.

[21] *SAE Electric Vehicle and Plug in Hybrid Electric Vehicle Conductive Charge Coupler J1772_201710*, SAE Standard J1772_201210, 2017.

[22] N. A. Chaturvedi, K. Reinhardt, J. Christensen, J. Ahmed, and A. Kojic, "Algorithms for advanced battery-management systems," *IEEE Syst. Mag.*, vol. 30, no. 3, pp. 49–68, Jun. 2010.

[23] H. E. Perez, X. Hu, S. Dey, and S. J. Moura, "Optimal charging of Li-ion batteries with coupled electro-thermal-aging dynamics," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 7761–7770, Sep. 2017.

[24] F. Kazhamiaka, S. Keshav, C. Rosenberg, and K.-H. Pettinger, "Simple spec-based modeling of lithium-ion batteries," *IEEE Trans. Energy Convers.*, vol. 33, no. 4, pp. 1757–1765, Dec. 2018.

[25] F. Kazhamiaka, C. Rosenberg, and S. Keshav, "Tractable lithium-ion storage models for optimizing energy systems," *Energy Informat.*, vol. 2, p. 4, May 2019.

[26] N. Blair *et al.*, *System Advisor Model (SAM) General Description (Version 2017.9. 5)*, Nat. Renew. Energy Lab., Golden, CO, USA, 2018.

[27] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, Jan. 1996.

[28] G. Brockman *et al.*, "OpenAI gym," 2016. [Online]. Available: arXiv:1606.01540.

[29] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Boca Raton, FL, USA: CRC Press, 1999.

[30] A. Al Zishan, M. M. Haji, and O. Ardakanian, "Adaptive control of plug-in electric vehicle charging with reinforcement learning," in *Proc. 11th ACM Int. Conf. Future Energy Syst.*, Jun. 2020, pp. 116–120.

[31] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, *Deadline Scheduling For Real-Time Systems: EDF and Related Algorithms*, vol. 460. New York, NY, USA: Springer, 2012.

[32] Z. J. Lee and S. Sharma. (Nov. 2020). *Adacharge*. [Online]. Available: https://github.com/caltech-netlab/adacharge

[33] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2909–2913, 2016.

[34] A. S. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *J. Control Decis.*, vol. 5, no. 1, pp. 42–60, 2018.

[35] Z. J. Lee, J. Z. F. Pang, and S. H. Low, "Pricing EV charging service with demand charge," *Elect. Power Syst. Res.*, vol. 189, Dec. 2020, Art. no. 106694.

[36] Z. J. Lee and S. Sharma. (Nov. 2020). *Acnportal-Experiments*. [Online]. Available: https://github.com/caltech-netlab/acnportal-experiments

[37] Y. Nakahira, N. Chen, L. Chen, and S. H. Low, "Smoothed least-laxity-first algorithm for EV charging," in *Proc. 80th Int. Conf. Future Energy Syst.*, 2017, pp. 242–251.

[38] F. Bu, Y. Yuan, Z. Wang, K. Dehghanpour, and A. Kimber, "A time-series distribution test system based on real utility data," in *Proc. North Amer. Power Symp. (NAPS)*, 2019, pp. 1–6.