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

Zachary J. Lee*, Daniel Johansson†, Steven H. Low*
*Division of Engineering & Applied Science, Caltech, Pasadena, CA
†Faculty of Engineering, Lund University, Lund, Sweden
zlee@caltech.edu

***Abstract*—Smart electric vehicle charging has recently garnered significant attention in the research community due to need to charge vast numbers of electric vehicles (EVs) economically, as well as the potential of providing grid services using EVs. However, research into practical online charging algorithms has been hampered by the lack of a widely available, realistic simulation environment in which to evaluate algorithms and test assumptions. To meet this need, we have developed ACN-Sim, a data-driven, open-source simulator based on our experience building and operating real-world charging systems. ACN-Sim provides a modular, extensible architecture which models the complexity of real charging systems, including battery charging behavior and unbalanced three-phase infrastructure. In addition, ACN-Sim integrates with a broader ecosystem of research tools for EV charging, including ACN-Data, an open dataset of EV charging sessions to provide realistic simulation scenarios, and ACN-Live, a framework for field-testing charging algorithms.**

I. Introduction

Electric vehicles (EVs) are growing in popularity globally, with the International Energy Agency (IEA) predicting over 125 million EVs worldwide by 2030 [1]. Likewise, McKinsey & Company estimates that EVs will consume 271 billion kWh of electricity in 2030, requiring 42 million EVSEs[1] (also know as charging ports) and necessitating nearly $50 billion in charging infrastructure investments [2].

Charging all of these EVs will also require new techniques for scheduling and control to avoid adverse effects on the electric grid while reducing capital and operating costs. The development of these techniques has garnered significant attention in the research community. However, transitioning these algorithms from theory to practice requires dealing with the complexities of practical systems which are often overlooked in our 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 implementation in the field.

This material is based upon work supported by NSF through grants CCF 1637598, ECCS 1619352, and CPS including grant 1739355, as well fellowship support through the NSF Graduate Research Fellowship Program 1745301 and Resnick Sustainability Institute. We would like to thank the team at PowerFlex, especially Cheng Jin, Ted Lee and George Lee, as well as Rand Lee, James Anderson, Umar Hashmi, and Jorn Reniers for providing data, expertise and ideas to this project.

[1]Electric Vehicle Supply Equipment

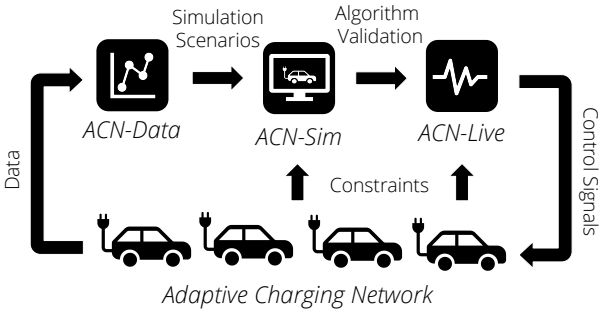

Fig. 1. The ACN Research Portal gives users many of the benefits of an EV charging testbed without needing to build one themselves. It includes data collected from real charging sessions (ACN-Data), a simulator to evaluate new ideas (ACN-Sim), and access to run on real hardware (ACN-Live).

In order to bridge this gap in our research, we developed 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 over 50 networked and controllable EV charging stations which allow us to collect data and field test algorithms with real hardware.

To give more researchers access to the benefits of a real EV charging testbed, we have developed and released the ACN Research Portal (ACN-Portal). The portal consists of three major components: (1) ACN-Data, a dataset of over 35,000 real EV charging sessions from the Caltech ACN and similar sites around California [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 shown in Fig. 1.

In this work, we will focus on ACN-Sim, which is open-source and available at [6]. We begin by comparing ACN-Sim with existing simulators in Section II. We then describe the modular architecture of ACN-Sim in Section III. In Section IV, we describe how algorithms are implemented for ACN-Sim and how they can easily be used to control real hardware with ACN-Live. We then explain the benefits of ACN-Sim in terms of simplifying researchers' workload in Section V and demonstrate these benefits through case studies in Section VI. Finally, we conclude this paper and look toward the future of ACN-Sim in Section VII.

II. Existing Simulators

Open-source tools and simulators have played an important role in smart grid research community. MATPOWER [7] and

978-1-5386-8099-5/19/$31.00 ©2019 IEEE

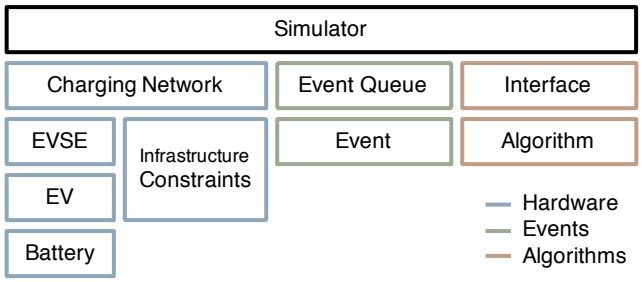

Fig. 2. Architecture of ACN-Sim. This simulator utilizes a modular, object-oriented architecture which closely resembles the components of a physical charging system. In this diagram color represents the four categories of modules in ACN-Sim. Depth in the diagram refers to composition i.e. an `EVSE` can contain an `EV` which contains a `Battery`.

Pandapower [8] have been extensively used by the community to study power flow related problems, while GridLab-D [9] and OpenDSS [10] have found widespread use for accurately modeling distribution systems. The importance of these open tools cannot be overstated, as they have allowed the community to work together to build more robust, feature-rich tools which can be modified as needed to support new research.

The research community has also developed simulators specific to EV charging. The most mature of these is V2G-Sim, a simulation environment developed at LBNL [11]. V2G-Sim has been used in many works to analyze the ability of EVs to meet drivers mobility needs in the context of: level-1 charging [12]; battery degradation [13]; and demand response [14]. More recently, V2G-Sim has also been used to examine grid-level effects of smart charging such as smoothing the duck curve [15]. Another recently proposed simulator is EVLibSim, which models many types of EV charging, including standard conductive charging, inductive charging, and battery swapping [16].

While both V2G-Sim and EVLibSim allow for precomputed charging schedules or simple control strategies, ACN-Sim is designed explicitly around evaluating online algorithms which adapt to changes in the system state over time. In this way, ACN-Sim is complementary to these existing simulators as they seek to address different questions. In addition, neither V2G-Sim or EVLibSim consider constrained infrastructure within a charging facility. This means these tools cannot be used to evaluate algorithms designed to allow infrastructure components to be oversubscribed, which is a crucial benefit of smart charging approaches.

III. Simulator Architecture

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

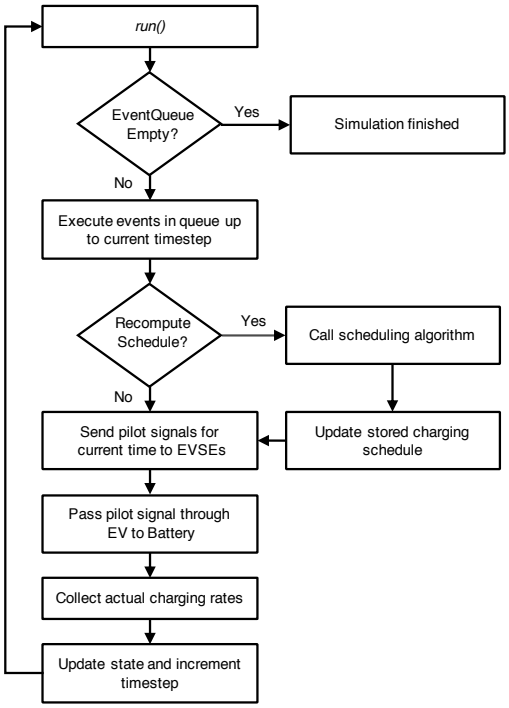

Fig. 3. Flow chart describing the simulator's `run()` function. Each timestep consists of a single iteration of this loop. The simulation ends when the last event from the `EventQueue` is executed at which time the user can analyze the results of the simulation.

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 as well as a queue of events which define when actions occur in the system. Figure 3 describes the operation of the simulator. 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, for later analysis.

B. Charging Network

Accounting for the electrical infrastructure of the charging system, including transformers, switch panels, and cables, is important to the design of practical EV charging algorithms. By considering these constraints, we can oversubscribe key pieces of infrastructure, reducing capital costs. However, this infrastructure is often three-phase and unbalanced.

To model electrical infrastructure, ACN-Sim uses the `ChargingNetwork` class. `ChargingNetwork` has two parts: a set of the `EVSE`s in the system and a set of constraints which represents infrastructure limits. Each constraint corresponds to a limit, R_j, on the magnitude of a current in the network, $I_j(t)$, and takes the form

$$|I_j(t)| = \left| \sum_{i=1}^{N} c_i r_i(t) \right| \leq R_j, \quad \forall t \in \mathcal{T}$$

where N is the total number of `EVSE`s in the system, $r_i(t)$ is the charging current of `EVSE` i at time t, and $\mathcal{T}$ is the set of all timesteps in the simulation. Note that $r_i(t)$ is a phasor

of the form $r_i(t) = |r_i(t)| \cdot e^{j\phi_i}$ where the phase angle, ϕ_i, is known based on how EVSE i is connected in the network. To find c_i we perform circuit analysis to calculate the relationship between $r_i(t)$ and $I_j(t)$.

Algorithms can interact with the constraints in two ways. One approach is to parse the constraints and include them directly in the algorithm, as is done with optimization-based algorithms such as those described in [4]. A second approach is to utilize the `is_feasible()` method, which can be used for algorithms that iteratively search for a feasible schedule, such as those described in section IV.

C. EVSE

EVSEs allow us to control the charging current of an EV for level-2 charging[2]. The EVSE communicates a pilot signal to the EV's on-board charger which is an upper limit on 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 of the pilot signal 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 [17]. In most current research, the additional constraints imposed by EVSEs without continuous control are neglected [18]. However, including these constraints is important for practical algorithms and is non-trivial.

ACN-Sim provides three EVSE models which cover most ideal and practical level-2 EVSEs:

- `EVSE` is the base class for all EVSE models and allows any pilot signal between an upper and lower bound. By default, `EVSE` allows any non-negative charging rate.
- `DeadbandEVSE` extends `EVSE` to exclude the J1772 standard deadband between 0 and 6 A.
- `FiniteRatesEVSE` only allows pilot signals within a finite set of allowable rates. `FiniteRatesEVSE` can be used to model many commercial EVSEs. For example, the EVSEs used in the Caltech ACN allow {6, 7, 8, ..., 31, 32} A or {8, 16, 24, 32} depending on the manufacturer.

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 is used to contain relevant information for a single charging session such as arrival time, departure time, estimated departure time, and requested energy. It also contains a `Battery` object. Note that 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].

[2]Level-2 EV charging utilizes the EV's on-board charger. Level-2 EVSEs provide between 3.3 - 19.2 kW at 208 - 240 V AC.

E. Battery

Most research in EV charging utilizes an ideal battery model, where EVs are assumed to follow the given pilot signal exactly. In practice, however, 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 [4], [18]. This can significantly increase the total time required to charge the battery and result in underutilization 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 as well as the vehicle's on-board charger, its 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, $\hat{r}(t)$, in this idealized model is described by

$$\hat{r}(t) := \min\{r(t),\ \bar{r},\ \hat{e}(t)\}$$

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

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

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

where `SoC` is the state-of-charge of the battery and `th` marks the transition from the *bulk* stage to the *absorption* stage of the charging process.

Figure 4 shows how these two models compare with one charging profile taken from ACN-Data. In general, we find that while the piecewise linear model is a good approximation, it does not capture some of the higher frequency dynamics of battery behavior that we observe in some charging sessions. Because of this, we plan to implement additional, higher fidelity battery models in future releases of ACN-Sim.

F. Event Queue

ACN-Sim uses events to describe actions in the simulation. There are three 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.

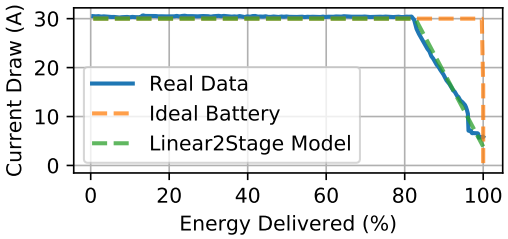

Fig. 4. Comparison of `Linear2Stage` and idealized `Battery` models with a real charging curve collected from the Caltech ACN when the pilot signal is not binding. We can see that the `Linear2Stage` model with appropriate parameters matches the battery behavior well in this case.

- `UnplugEvent` signals when an EV leaves the system at the end of its charging session.
- `RecomputeEvent` signals when the algorithm should be called regardless of if another event occurs.

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 and finally RecomputeEvents. At each timestep, the `Simulator` executes all events left in the queue which have timestamps on or before the current timestep. After any event, the scheduling algorithm is called so that it can adapt to the new system state. Users are free to create new events by extending the `Event` class.

IV. Charging Algorithms

A. Interface

We begin by describing the algorithm interface. This interface abstracts away the underlying infrastructure, whether that be 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 write a single `schedule()` function. This function takes in a list of EVs which are currently active, meaning they are plugged in and their energy demand has not been met, and returns a charging schedule for each. This schedule takes the form of a dictionary which maps station ids to a list of charging rates in amps. Each entry in the list 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 which can be used as benchmarks.

1) Uncontrolled Charging: Uncontrolled Charging is the most common "algorithm" used in charging systems today. It allows each EVSE to charge at its maximum allowable rate. This algorithm does not factor in infrastructure constraints.

2) Round Robin: Round Robin (RR) is a simple algorithm which attempts to share charging capacity among all active EVs equally. To do this, it first creates a queue of all active EVs then iterates over this queue. It then checks if it is feasible to increment the rate of the current EV by one unit. If it is, the algorithm increments the rate and adds the EV back to the end of the queue. If 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 which does not cause an infrastructure constraint to be violated and is less than the maximum charging rate of the EVSE.

3) Sorting Based Algorithms: There are three sorting based algorithms included in ACN-Sim: First-Come First-Served (FCFS), Earliest-Deadline First (EDF), and Least-Laxity First (LLF). Sorting based algorithms are commonly used in other deadline scheduling tasks such as job scheduling in servers due to their simplicity [19]. 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 given that the assignments to all previous EVs are fixed. This process continues until all EVs have been processed.

To determine if a given charging schedule is feasible, RR, FCFS, EDF, and LLF use the `is_feasible()` method included in the algorithm interface. This method accounts for the full three-phase constraints of a network model. In addition, all of these algorithms are compatible with the constraints of the three `EVSE` models included with ACN-Sim.

V. Simplified Research Pipeline

ACN-Sim is designed to simplify the process of evaluating new algorithms for smart EV charging. In this section, we demonstrate how ACN-Sim can simplify each step in a researcher's workflow as they evaluate the performance of their algorithm against others proposed in the literature.

A. Implementing an algorithm

As described in Section IV, ACN-Sim greatly simplifies the process turning an algorithm into code. In addition, since algorithms can easily be shared, researchers do not need to re-implement baseline algorithms to compare against.

B. Defining an experiment

Without ACN-Sim, defining an experiment can be a difficult process, as researchers need to model the charging system they want to use and generate a realistic set of scenarios. ACN-Sim simplifies this process by providing predefined charging networks, as well as integrating with ACN-Data to generate scenarios from real charging data. With these utilities, researchers only need to define the start and end times of the simulation, what site they want to simulate, and what algorithm(s) they want to test.

C. Constructing and running a simulation

After an experiment is defined, the process of constructing and running a simulation is as simple as constructing a `Simulator` object then invoking its `run()` method. ACN-Sim allows researchers to achieve this task with only two lines of code, versus the thousands necessary to build a realistic simulator from scratch.

D. Analyzing results

After a simulation, researchers can analyze the results using data stored in the `Simulator`. This data includes the pilot signals sent to each EVSE and the actual charging rate of the EV attached. The `Simulator` also has the option to save the results of each call to the scheduling algorithm for more in-depth analysis. To make analysis even easier, ACN-Sim includes tools to perform common calculations such as finding aggregate currents, measuring phase imbalances, and calculating the proportion of energy demands met.

E. Sharing experiment

Reproducibility is key to good science. However, when experiments are run on in-house simulators and closed datasets, it can be difficult for researchers to share their work and verify the results of others. With ACN-Data and ACN-Sim, all researchers have access to the same dataset and simulation environment, making it much easier to share experiments and benchmark algorithms. For example, once an algorithm has been implemented to work with the ACN-Sim framework, it can be easily shared, either by contributing it to the `algorithms` module or sharing the code separately. Other researchers can then directly test against the original authors' implementation of the algorithm.

To make this process even easier, ACN-Sim works with Google Colab, a free cloud-based service which hosts Jupyter notebooks. Using this service, the researchers can share links to their experiments which can be run in the cloud without downloading or installing software locally. Other researchers can then easily try the experiment themselves, even changing parameters, or trying different scenarios.

VI. Case Studies

In these case studies, we demonstrate the type of research which ACN-Sim enables. The code for these case studies is available as a Jupyter notebook at [20].

Experiment 1: Unbalanced three-phase infrastructure

A key feature of ACN-Sim is its ability to simulate the unbalanced three-phase electrical infrastructure common in large charging systems. Currently, most charging algorithms in the literature rely on constraints which assume single-phase or balanced three-phase operation. We can use ACN-Sim to demonstrate why these assumptions are insufficient for practical charging systems.

For this experiment, we consider two versions of the LLF algorithm. In the first, LLF uses a single-phase representation of the system and ensures that total charging power is less than the transformer's capacity. In the second, LLF uses the full three-phase system model. The results of this experiment are shown in Fig. 5. In both cases, we evaluate the algorithms using the complete three-phase network model.

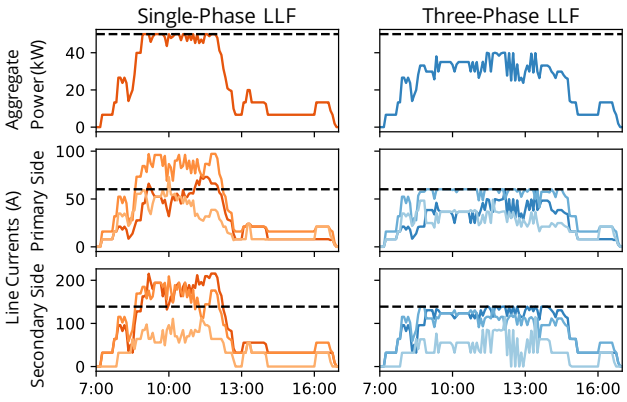

Fig. 5. Aggregate power draw and line-currents at the primary and secondary side of the transformer when running single-phase and three-phase LLF algorithms on the Caltech ACN with a 50 kW transformer capacity. Shading in the lower plots denote each phase while the black dotted line denotes the power/current limit. The experiment is based on on data from the Caltech ACN on March 5, 2019 and uses a 5 minute timestep.

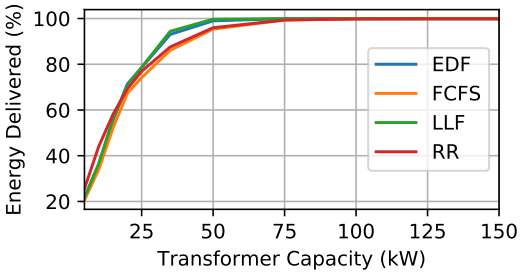

Fig. 6. Comparison of percentage of energy delivered as a function of transformer capacity. The simulation runs from March 1 through April 1, 2019 with a timestep of 5 minutes. To generate events we use ACN-Sim's integration with ACN-Data to get real charging sessions from the Caltech ACN. We also use the included Caltech ACN charging network model with its optional `transformer_cap` argument to limit the infrastructure capacity.

From Fig. 5, we can see that only considering single-phase constraints can lead to significant constraint violations in line currents. However, by designing an algorithm which considers the full three-phase model, we can respect these constraints. Note that because of phase unbalance, we are not able to make use of the full 50 kW transformer capacity while also respecting line limits.

Experiment 2: Constrained infrastructure comparison

ACN-Sim makes it easy to compare algorithms against one another. Since it would be out of scope to introduce a new algorithm, in this experiment, we will compare ACN-Sim's baseline algorithms at various infrastructure capacities.

The results of this experiment are shown in Fig. 7. From this plot, we can see that with sufficient infrastructure capacity, all algorithms were able to meet 100% of demand. However, we see that for moderately constrained infrastructure (20-75 kW), algorithms which account for user's deadlines and energy requests (EDF and LLF) outperform those that do not (RR and FCFS). Moreover, when using EDF or LLF, only 50 kW of capacity are needed to fully meet charging demands vs.

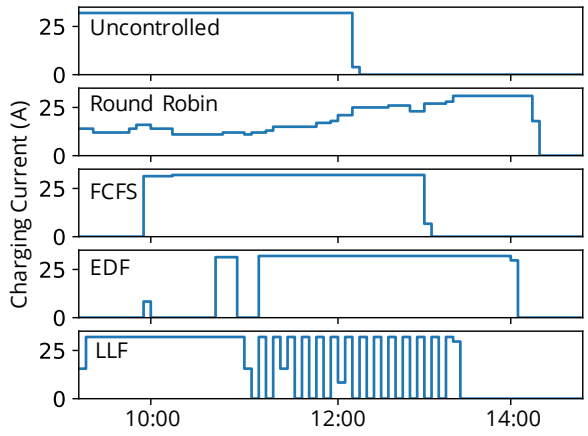

Fig. 7. Comparison of charging profiles for one EV on March 5, 2019 with a 50 kW transformer capacity.

75 kW for RR and FCFS. Interestingly, for extremely low infrastructure capacities, RR deliverers more energy than the other algorithms. This is because RR can better utilize the available capacity by balancing between phases.

ACN-Sim also allows us to examine the charging profile of individual EVs, as shown in Fig. 6. From this plot, we can see that FCFS behaves very similarly to Uncontrolled, but charging is delayed as the EV must wait its turn in the queue. In the case of EDF and LLF, we see that charging can be interrupted when EVs with earlier deadlines arrive (EDF) or when the laxity of the EV increases as it charges (LLF). As we can see from Fig. 6, LLF can cause oscillations in the charging profile. These oscillations are the result of an increase in laxity as the EV charges, which can decrease its standing in the queue. The behavior of RR is quite different, as the EV charges steadily, but at a rate below its maximum as congestion in the system necessitates sharing of charging capacity. We can also note that Uncontrolled charges the EV the fastest, followed by FCFS, LLF, EDF and finally RR.

VII. Conclusions and Future Work

In this work, we present ACN-Sim, a data-driven simulator designed to aid the development of practical online scheduling algorithms for EV charging. This tool significantly reduces the software engineering burden on researchers and helps to expose them to practical issues present in real charging systems. ACN-Sim also makes it easier for researchers to share their experiment code, helping to improve transparency and code reuse in the community. Finally, ACN-Sim integrates with the Adaptive Charging Network Research Portal, a larger suite of tools which includes a database of real charging sessions and a framework for field testing algorithms.

ACN-Sim will continue to grow to meet the needs of the community, including new models of systems components and charging networks. We also plan to develop a suite of test cases (include networks and scenarios) which can be used as a standard evaluation set for algorithms. Moreover, we plan to integrate ACN-Sim with other tools such as OpenAI Gym and GridLab-D. OpenAI Gym is a toolkit for developing and evaluating reinforcement learning (RL) algorithms [21]. By integrating this with ACN-Sim we will allow RL researchers to more easily apply their techniques to the problem of smart EV charging. In addition, integration with GridLab-D will let researchers explore grid-level effects of smart EV charging and evaluate algorithms for providing grid services.

References

[1] T. Bunsen, P. Cazzola, M. Gorner, L. Paoli, S. Scheffer, R. Schuitmaker, J. Tattini, and J. Teter, "Global EV Outlook 2018: Towards Cross-Modal Electrification," 2018.

[2] H. Engle, R. Hensley, S. Knupfer, and S. Sahdev, "Charging Ahead: Electric-Vehicle Infrastructure Demand." https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/charging-ahead-electric-vehicle-infrastructure-demand. Accessed: 2019-05-03.

[3] G. Lee, T. Lee, Z. Low, S. H. Low, and C. Ortega, "Adaptive Charging Network for Electric Vehicles," in *2016 IEEE Global Conference on Signal and Information Processing*, Dec. 2016.

[4] Z. J. Lee, D. Chang, C. Jin, G. S. Lee, R. Lee, T. Lee, and S. H. Low, "Large-Scale Adaptive Electric Vehicle Charging," in *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, Oct. 2018.

[5] Z. J. Lee, T. Li, and S. H. Low, "ACN-Data: Analysis and Applications of an Open EV Charging Dataset," in *Proceedings of the Tenth International Conference on Future Energy Systems*, e-Energy '19, June 2019.

[6] Z. J. Lee and D. Johansson, "acnportal." https://github.com/zach401/acnportal, Apr. 2019.

[7] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," vol. 26, no. 1, pp. 12–19.

[8] L. Thurner, A. Scheidler, F. Schfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "PandapowerAn Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems," vol. 33, no. 6, pp. 6510–6521.

[9] D. P. Chassin, J. C. Fuller, and N. Djilali, "GridLAB-D: An agent-based simulation framework for smart grids," *Journal of Applied Mathematics*, vol. 2014, 2014.

[10] D. Montenegro, R. Dugan, R. Henry, T. McDermott, and W. Sunderm, "Opendss - epri distribution system simulator." https://sourceforge.net/projects/electricdss/.

[11] "V2G-Sim," 2019. Available: http://v2gsim.lbl.gov/home.

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

[13] 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," *Journal of Power Sources*, vol. 282, pp. 265–276, 2015.

[14] 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," tech. rep., SAE Technical Paper, 2015.

[15] J. Coignard, S. Saxena, J. Greenblatt, and D. Wang, "Clean vehicles as an enabler for a clean electricity grid," *Environmental Research Letters*, vol. 13, no. 5, p. 054031, 2018.

[16] 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," *Simulation Modelling Practice and Theory*, vol. 87, pp. 99–119, Sept. 2018.

[17] SAE, "SAE Electric Vehicle and Plug in Hybrid Electric Vehicle Conductive Charge Coupler J1772_201710," 2017.

[18] Q. Wang, X. Liu, J. Du, and F. Kong, "Smart Charging for Electric Vehicles: A Survey From the Algorithmic Perspective," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1500–1517, 2016.

[19] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, *Deadline scheduling for real-time systems: EDF and related algorithms*, vol. 460. Springer Science & Business Media, 2012.

[20] Z. J. Lee, "ACN-Sim-Demo." https://github.com/zach401/ACN-Sim-Demo, May 2019.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.