Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

INTEGRATING ELECTRIC VEHICLES INTO SMART GRID INFRASTRUCTURES A SIMULATION-BASED APPROACH THAT BECAME REALITY

Marco Lützenberger
Tobias Küster
Sahin Albayrak

Technische Universität Berlin, Distributed Artificial Intelligence Laboratory
Ernst-Reuter-Platz 7
10587 Berlin, GERMANY

ABSTRACT

The development of software that controls real life processes can be highly difficult and error prone. In the case that the destination test-bed does not fully exist, the situation becomes significantly more challenging. We developed a control software for charging processes of an electric vehicle fleet in a smart grid architecture. To accelerate the development and to ease the integration process, we used an agent-based approach and embedded the optimization software within a simulation environment. Later we enhanced this simulation environment to a consultant tool which can be used to assess the impact of structural extensions. In this paper we present both, the optimization mechanism as well as the simulation environment.

1 INTRODUCTION

It is commonly known that software development tends to be highly complex and error-prone. Whenever the purpose of this software is to control real-life processes, the situation becomes even more challenging. Within the project *Berlin Elektromobil 2.0*, or *BeMobility* (Freund et al. 2012), we developed a control software for an electric car sharing fleet, which was integrated into a smart grid infrastructure. This control software was supposed to determine charging periods that satisfy several conditions.

To start with, we dealt with a commercial car sharing fleet, provided by *Flinkster*, thus, our highest priority was to ensure that vehicles are fully charged when picked up by a customer. Secondly, it was our objective to exploit locally produced energy from renewable energy sources and thus to decrease the effective CO_2 fingerprint of the vehicle fleet. Another objective was to facilitate the electrical autarchy of the smart grid architecture and to minimize the amount of energy that is taken from the energy grid. This objective appears to be similar to the previous one, though, given that the installed electric vehicles support *Vehicle-to-Grid*, or *V2G* technology, the vehicles are capable to serve as energy buffers which can be used to store surpluses of energy from renewable energy sources. Later, when the smart grid's energy demand increases its production, these surpluses can be used to avoid grid procurement.

While it is already difficult to arrange charging processes with respect to these objectives (let alone that the above-presented list is only a selection of constraints that we had to account for—refer to Section 3.2 for a complete listing), the development of such software was additionally aggravated by the fact that the smart grid architecture was still under construction and only partially operational.

To counter this problem—and to avoid significant delays due to the late completion of the site—we developed our optimization within a simulation environment, which we present in this paper. The development of this simulation environment was started within the BeMobility project and later continued within the follow-up project *Forschungscampus EUREF: Energietechnische Voraussetzungen für dezentrale Erzeugung, Speicherung, Lastmanagement und Netzbetrieb, TU Berlin*, or *FC EUREF*. The aim was to design this architecture environment such that the same optimization system which is used within the simulated environment can later be used to control charging processes within the real test-site.

Furthermore, we aimed to design the same environment to act as a prototypical consultant tool, helping decision makers to assess the effects of embedding vehicle fleets into smart grid architectures as well as to estimate the profit in extending the elements of this promising combination.

Similar optimization systems were presented by Kamboj et al. (2010), Keiser et al. (2011), Markel et al. (2009), Gray and Francfort (2012), Kempton et al. (2008). Although most of these approaches were used to control the charging of real vehicles, none of them was combined with a simulation mode.

The remainder of this paper is structured as follows: In Section 2, we present the model that we use to formally conceptualize the problem domain. In Section 3, we explain our optimization software. This section comprises a part in which we explain how the elements of our domain model are interpreted and how their quality is assessed, as well as a part in which we explain the applied optimization mechanism. In Section 4, we proceed by explaining the system architecture of our simulation environment. To facilitate modularity and distribution, as well as to make the approach compatible with the real life system, we applied an agent-based approach and designed all relevant components of our system as *JIAC V* (Lützenberger et al. 2013) software agents. In Section 5, we present the evaluation of our simulation system by means of an exemplary car fleet and smart grid configuration. Finally, in Section 6, we conclude our work.

2 THE TEST-SITE

In this section we present the test-site, namely the *Europäisches Energieforum* (eng. *European Energy Forum*), or *EUREF*, where our optimization software is deployed. This section comprises two parts. First, we present devices that are installed at the EUREF and that are able to produce/consume energy. These devices are relevant for the optimization process. Secondly, we present the domain model used to formally conceptualize our test-site.

2.1 The European Energy Forum

The EUREF is an area located in a central part of Berlin, Germany, formerly used as a gas storage facility. In 2011, the entire site was completely re-developed in order to provide commercial space for innovative enterprises in the energy sector. Within the course of this redevelopment, the area was equipped with a sophisticated, state-of-the-art smart grid infrastructure, namely the *Micro Smart Grid*, or *MSG*.

The Micro Smart Grid architecture comprises a 3-phase system connected to a step up transformer (0.4 kV / 10 kV) that can be operated up to 630 kVA. Photovoltaic systems with a total installed capacity of 53.5 kW peak and wind turbines with a total installed capacity of 5 kW represent fluctuating energy sources. A hydrogen fuel cell and a Stirling motor—controllable micro generators with an installed capacity of 1 kW (electrical) each—are installed as well. In addition, a grid buffer battery allows balancing the injected power and the power demand. In total, there are 22 charging points whereby some of them can be used for Vehicle-to-Grid operations that allow electric vehicles to feed energy back into the grid. The car pool that was used for the BeMobility project comprised ten vehicles, all of which belonged to a car sharing company—meaning that estimated usages (e.g., booking data) was available.

2.2 Micro Smart Grid Domain Model

Our goal was to create a domain model specifically for representing and simulating electric vehicles (EV) in a micro smart grid, together with their respective charging infrastructure, and local energy consumers, producers, and storages.

The central element of the domain model is the `MSG`, aggregating a number of `ElectricVehicles`, `Storages`, `ChargeController` and `Prosumers`. Storages can be both local energy storages as well as EV batteries, and each one is associated to a charge controller, describing how that storage can be (dis-)charged. Electric vehicles add vehicle-specific information on top of that, depending on their type.

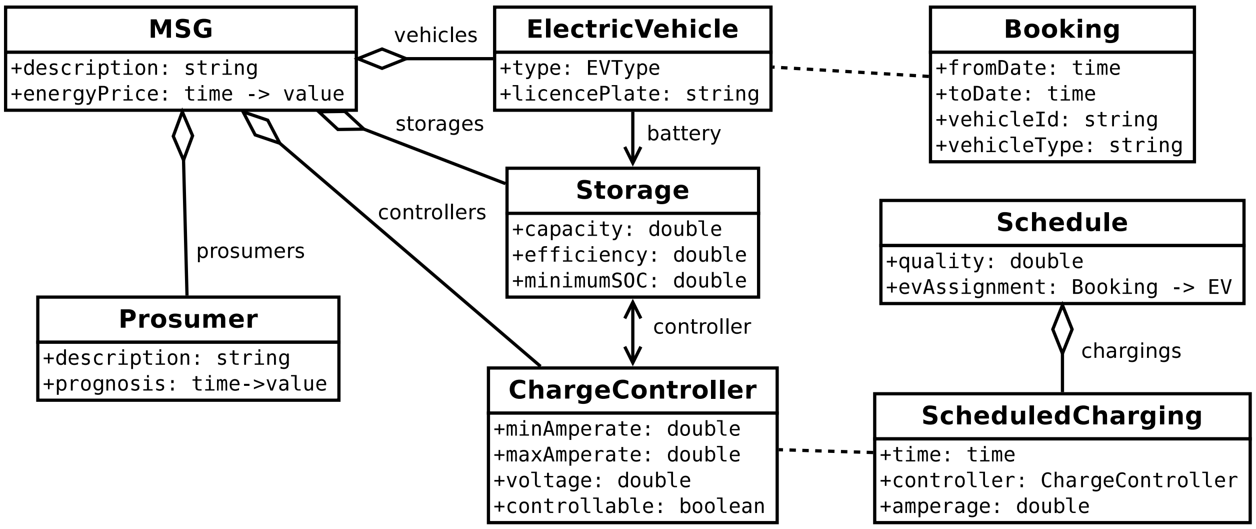

Figure 1: UML Class Diagram of Micro Smart Grid domain model, slightly simplified.

Prosumers combine both producers and consumers of electric energy, e.g., locally installed power plants (particularly wind and solar) and buildings.

Complementary to this architectural information, the model also holds prospective information. Each prosumer has a `prognosis` of its future energy production/consumption that is either derived from historical data or from auxiliary sources such as weather forecasts for wind and global radiation. Also, the projected *energy price* can be taken into account using, e.g., data provided by the energy spot marked. Last but not least, a list of `Booking`s indicates when which car needs to be charged. The actual *charging schedule* is represented as a `Schedule` object, basically a list of `ScheduledCharging`s, each being a tuple of a time, a charging station, and the amperage with which to charge.

In its current form, the domain model (see Figure 1) satisfied all the requirements of the project BeMobility 2.0 in which it was developed. For the future, we are planning to extend the domain model so it can be used in a broader range of e-mobility projects.

3 STOCHASTIC OPTIMIZATION

The idea behind the optimization is rather simple. In short, all controllable consumption and charging processes have to be arranged, such that the overall energy balance is optimal with respect to the constraints.

Yet, given that we are dealing with a discrete time interval and given that charging intervals can also be spitted into two, three, or more sub-intervals—including attempts to consume surpluses of energy, which are later used to cushion energy shortages—the complexity of the optimization problem becomes obvious. In fact, we are dealing with a problem class which is know to be NP-hard. Solutions that investigate the entire problem space of NP-hard problems may involve rather long response times. To counter this problem, we decided to use a stochastic approach, namely *evolution strategy* (*ES*) (Rechenberg 1973).

The implementation of the optimization algorithm used in this paper is based on previous work from an earlier project, *EnEffCo*, where it has already served well for optimizing schedules for production processes (Küster et al. 2013).

The operation principle of ES is quickly explained. The algorithm proposes a set of randomly arranged charging and feeding processes for all controllable consumers and producers (for a given time period, e.g., a day). These charging schedules are assessed with respect to their quality to fit the purpose. From all proposed schedules, the best ones are selected, randomly manipulated (mutated), and assessed again. The process iterates until some steady state is reached. In this section we explain how charging schedules are first interpreted and subsequently assessed. Finally, we present our evolution strategy in more detail.

3.1 Interpreting Charging Schedules

To determine the quality of a charging schedule, the schedule is interpreted using the given Micro Smart Grid model, tracking the state of charge of each individual electric vehicle and storage, the overall energy consumption, and whether the individual bookings could be fulfilled, or not.

For this purpose, the period to be interpreted can also be split into a number of discrete time steps, and for each of those time steps, the state of the MSG is calculated based on the previous step, starting with the current state of the MSG (which vehicles are connected to which charging stations, what is the state of charge of the individual storages, etc.).

While the interpretation is not physically correct in all aspects—for example, we do not take into account the differences in charging efficiency based on the storage's current state of charge—it fits purposes for determining whether a given booking can be fulfilled or to assess other factors that determine the charging schedule's quality.

3.2 Constraints and Quality Assessment

Once the charging schedule has been interpreted, the acquired metrics can be used for determining the schedule's quality (or, conversely, its "defect").

There are two kinds of metrics: hard constraints and soft optimization goals. The former includes aspects such as: Can all bookings be fulfilled, or is the car ever charged beyond its capacity? Those constraints should be satisfied by each charging schedule. The latter includes aspects like the overall energy consumption and cost, the load smoothness, or the total number of charging events.

Currently, we do not draw a clear distinction between constraints and soft goals. Instead, each criteria can be given a weight (the weights for constraints typically being much larger than those for soft goals), and the sum of all the weighted criteria makes up the charging schedule's quality. This way, defective solutions are not discarded up front, but are given a chance to be improved upon by the optimization algorithm.

The advantage of this is that it makes stochastic optimization such as evolutionary algorithms much simpler, and does not introduce the danger of biasing randomized creation of new individuals towards a certain class of solutions. However, the downside is that it requires some amount of fine tuning of the different weights to prevent defective charging schedules from appearing in the solution set.

3.3 Evolutionary Charging Schedules

As the name implies, evolutionary algorithms are nature-inspired stochastic optimization algorithm mimicking the principles of mutation, recombination and selection to iteratively improve on existing results and to converge to a close-to-optimal solution. In this work, we are using a variant of evolution strategy (Rechenberg 1973), having one or more populations of charging schedules, from which a number of "parents" are randomly selected, recombined and mutated. Finally, each individual is interpreted and those with the highest quality are selected for the next generation, until the process converges or for a fixed number of generations.

In contrast to *complete* search algorithms, stochastic approaches, and evolutionary approaches in particular, are far more efficient, while at the same time not being as restricted, or biased, as deterministic algorithms using some particular heuristic.

3.3.1 Recombination

For the recombination of multiple parents, one of two options is selected at random:

- The parent schedules are split up into time frames, each being taken from another parent.
- The charging events for each electric vehicle are taken from another parent.

Other than randomly selecting charging events from different parents, this way charging events that "co-evolved" for the same vehicle or for the same time remain together in the new charging schedule.

3.3.2 Mutation

Mutation is the main driver of evolution in our algorithm. There are a number of different mutation operations, one of which is selected at random:

- Insert a new random charging event.
- Remove a random existing charging event.
- Modify the amperage of an existing charging event.
- Move the charging events within a random time frame by some random time delta.

Charging schedules that do not satisfy all the constraints—e.g., whether all the bookings are fulfilled—are not discarded immediately but are given a chance for improvement in the next generations until they are displaced by schedules with a higher quality.

4 SYSTEM ARCHITECTURE

As mentioned in the introduction, we aimed to place the optimization software within a simulation environment. The purpose of this environment was to mimic the behavior of the real Micro Smart Grid, which—by the time the software has been developed—was still under construction.

Later, we deployed the same optimization system on the test-site in order to control charging processes for real.

The simulation environment was intended as a test-platform for the optimization system. Furthermore, we wanted to use it as a consultant tool, which is capable to visualize the effects of infrastructural improvements at the test-site (e.g., more powerful wind turbines or more effective solar panels, batteries, or charging stations). To meet these requirements, we decided to use an agent-based approach and designed all relevant system entities as software agents.

This section comprises two parts. First we present agent types that make up the simulation system. We go into detail regarding their tasks and explain how they cooperate in order to serve the purpose. Subsequently we present the real system as it is currently deployed at the EUREF test-site. In doing so, we put particular emphasis on adaptations that were needed to transfer the simulation approach into reality.

4.1 Simulation

As mentioned above, we used an agent-based approach in order to logically separate all relevant parts of the system. We selected the JIAC V agent framework (Lützenberger et al. 2013) for the implementation of the multi-agent system. The modular assembly of JIAC V multi-agent systems allowed us to implement all relevant instances as encapsulated, reusable modules which communicate transparently—even beyond network borders. This modularity significantly facilitated our objective to use the same optimization system for both, for simulation purposes as well as for the real test-site.

In total we identified four different agent types with different responsibilities. The assembly of these agents, that is, the way they interact, communicate, and cooperate, is illustrated in Figure 2. We explain the operation principle of all agents in more detail, below:

4.1.1 The WebApp Agent

The *WebApp Agent* constitutes the interface between the human operator and the multi-agent system. The WebApp Agent receives input from a web-based configuration front-end. Its purpose is to forward this input to the responsible agents in the multi-agent system.

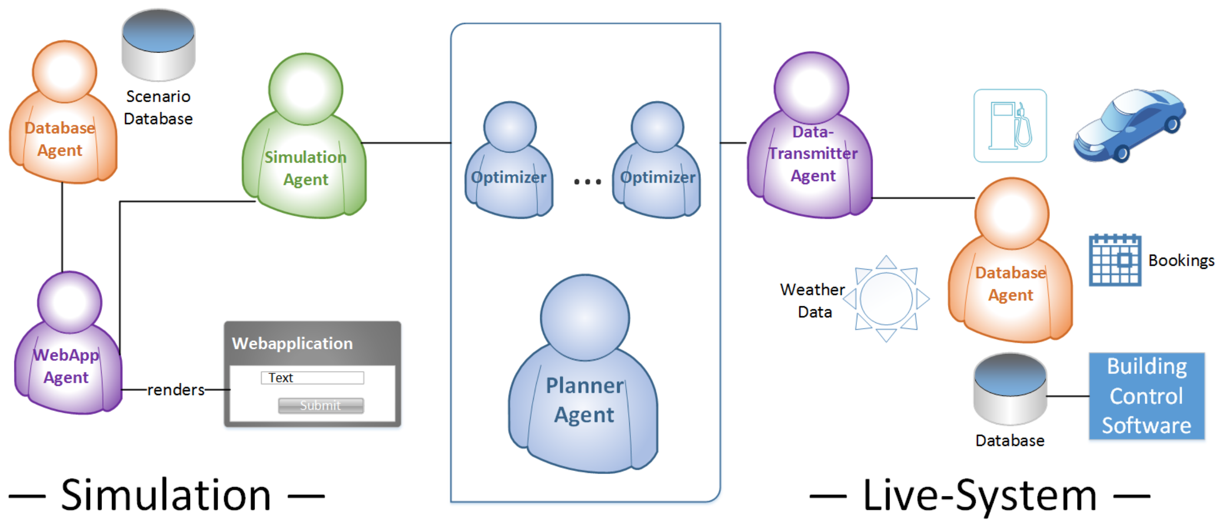

Figure 2: The system architecture. All relevant entities are modeled as JIAC agents.

Configuration options include infrastructure-specific settings, e.g., size and type of the solar panel, size and type of the wind turbine, installed batteries, installed vehicles, amount of available charging stations, charging station characteristics, but also secondary parameters, such as booking data or expected- and real weather situations. Furthermore, it is possible to specify simulation-specific settings, such as the selected period or the applied interval. Finally, the front-end allows the user to start a simulation and to monitor the simulation's progress. Features that help to analyze the simulation results are provided as well.

In order to exchange infrastructure-specific settings from the web-interface to the WebApp Agent and from the WebApp Agent to the multi-agent system, we use the `MSG` class type. Simulation-specific parameters are expressed by simple data types.

In order to allow users to choose highly realistic scenarios, we connected the user interface to a database in which contains a whole set of historic data from the test-site. Access to this database is provided by the *Database Agent*.

4.1.2 Database Agent

The Database Agent provides access to the simulation database, which contains data that can be used for simulation purposes. In total, three categories of data are stored. First, building-specific consumption data for the entire year 2013 is stored. This series of data can be used as realistic input value for the energy demand at the test-site. Secondly, the database includes the entire booking situation for the year 2013, including information about the entrance of the reservation, pickup and return times, mileage, etc. Finally, the database contains 24-hour weather forecasts as well as recorded weather data for each day of the year 2013.

Using the WebApp Agent, the web front-end is able to access the simulation database and to retrieve data that can be used for the configuration of simulation scenarios. The web front-end allows to use this data in its original form, but also to cut selected time periods (e.g., only one month, or a week) or to scale data series (e.g., twice the consumption, half the energy production).

4.1.3 Simulation Agent

Once the user configured a simulation scenario and initiates a simulation, the web front-end uses the WebApp Agent to send all relevant simulation parameters (in the form of an `MSG` object and a list of simple data types) to the *Simulation Agent*, which proceeds as follows:

First, the received simulation scenario is divided into a list of simulated days. The day-interval is used to exchange information with the optimization system, such that all available information (expected bookings, expected weather, expected consumption) for a given day is send to the optimization system, optimized, and subsequently simulated. While "expected data" was sent to the optimization system, the measurement data (also stored in the `MSG` object) is used for the simulation process.

First, the simulation divides the day interval into 1440 minute intervals. For each minute interval, the energy demand (from buildings), the amount of locally produced energy and the amount of energy that comes from feeding electric vehicles is determined. The simulation routine tries to cover the energy demand with the amount of available energy and respectively uses average CO_2 values for renewable energy sources (average values are 75 g/kWh for solar energy and 21 g/kWh for wind energy) as well as the current CO_2 fingerprint of feeding vehicles, to determine the overall emission efficiency. Remaining surpluses are used to cover the demand of electric vehicles—these are derived from the optimized charging schedule that was returned by the optimization system. Whenever surpluses of locally produced energy are used to charge local batteries (e.g., those of electric vehicles), the battery's emission fingerprint is recalculated (it is important to mention that it is not allowed to charge Flinkster vehicles outside the MSG). Remaining demands are covered by using energy from the urban power grid. For this type of energy we assume the average emission fingerprint for electricity in Germany, that is: 576 g/kWh. Using this mechanism the simulation processes each day in the simulation scenario.

Once the simulation has finished, the same scenario is simulated again. Yet, instead of using the optimization system, the second simulation uses an "intuitive approach" to charge the electric vehicles. This intuitive approach works as follows: i) whenever vehicles are returned, they are charged at a maximum amperage until the battery is full, ii) vehicles are not able to use V2G capability. The approach reflects "partial knowledge", which is typical for human-controlled processes where "the bigger picture" is missing (e.g., knowledge about currently available energy, or knowledge about expected future energy progression, etc.). We use the results of this second simulation to compare the capabilities of our optimization system to intuitive behavior. The results are returned to the web front-end.

4.1.4 The Planner Agent

The *Planner Agent* receives an optimization job in the form of an `MSG` object. The object describes the Micro Smart Grid, which is supposed to be optimized and contains all the information required for the optimization process, e.g., installed devices, expected weather, available vehicles, expected bookings, etc. Once the `MSG` object has been received, the Planner Agent broadcasts the optimization problem to all available Optimizer Agents and waits for their results. From all available results, the charging schedule with the best quality is selected.

4.1.5 The Optimizer Agent

While there is only one instance of the Planner Agent in our system, there can be several instances of the *Optimizer Agent* type. Optimizer Agents listen for optimization requests that are broadcasted by the Planner Agent. These requests are in the form of the same `MSG` object which the Planner Agent initially received. Once an Optimizer Agent detects an optimization request, it decides on whether to accept or to reject the request. A request can be rejected for several reasons, e.g., when the agent is currently occupied with another optimization task. An accepted request causes the agent to initiate the optimization procedure. This procedure is done in compliance with the stochastic optimization algorithm that we presented in the previous section.

The advantage in using multiple instances of the Optimizer Agent is twofold. First, the problem of stochastic optimization to get stuck in local optima is countered. Each agent derives its very own initial charging schedule (initial population), which serves as input for the evolution strategy. Due to the fact that random factors are used for this generation process, initial populations of all Optimizer Agents differ. The result of a stochastic optimization process, however, is significantly affected by the initial population. To put it in other terms, stochastic optimization procedures "evolve" their initial populations. Secondly, JIAC agents are truly multi-threaded, thus, multiple instances can run on different cores and do not affect the system's performance (at a reasonable agents per core ratio).

Once an Optimizer Agent found a (quasi-)optimal solution, the result is returned to the Planner Agent, which retrieves the best solution from all available proposals.

4.2 Reality

From the beginning, we engineered the optimization system to be applicable for both, simulation as well as reality. Thus, we use the same implementation for the Planner and the Optimizer Agents. We present all other system components in the following:

4.2.1 Database Agent

The implementation of the real *Database Agent* is similar to the simulation Database Agent. The agent stores all information about the MSG in a database and provides access to that data. In more detail, the database is connected to the MSG's building control, which permanently stores current consumption data in the database. Beyond that, the Database Agent polls information from two additional channels at a regular interval. First, weather data is retrieved. This data is provided by *MeteoGroup* (MeteoGroup website: http://www.meteogroup.com) and contains 24-hour forecast data for: temperature, wind, and global radiation. The data is tailored for the EUREF test-site. Secondly, available booking information is retrieved from Flinkster. Booking data contains information about the expected pickup and return time, as well as the expected mileage. The data is updated every 24-hours.

4.2.2 Data Transmitter Agent

The *Data Transmitter Agent* retrieves all data that is required for an optimization process from the Database Agent. The data is prepared, such that an `MSG` object is instantiated.

In more detail, we use the booking data provided by Flinkster as well as the forecasts provided by MeteoGroup. Yet, while these data series are highly precise, we currently have no valid forecast model for the expected energy demand. To counter this problem, we compute average values for each day of the week (to account for season-specific consumption patterns we only use three weeks) and use these values as forecast.

Readily configured optimization jobs are send to the Planner Agent, while the result (in the form of a charging schedule) is directly forwarded to the Database Agent, which stores it into the database. The building control software retrieves the charging schedule from the database and executes the charging commands, respectively.

We programmed the Data Transmitter Agent to initiate optimization processes every 15 minutes. Given that the most important input parameters, e.g., weather, demand, and booking forecast do not change for 24 hours, a 15 minutes planning interval appears to be superfluous, yet, to reflect electric control and charging processes in a computational model is highly difficult. The real result is affected by many additional parameters, e.g., power cable length, temperature, humidity, to name but a few. The bottom line is that the expected behavior, which is computed by the optimization system frequently deviates from the real behavior of the real Micro Smart Grid. In order to keep this derivation at a minimum we initiate planning processes every 15 minutes and respectively use the latest data as input for the Planning Agent.

It can be seen that the system architecture, as well as the use of the multi-agent paradigm (and JIAC in particular) facilitated the development of our system. The simulation environment helped us to test the system in each and every aspect and allowed us to deploy a functional version soon after the MSG's hardware was readily installed. We encountered only few problems when deploying the optimization system on the test-site. We still use the simulation environment to test critical improvements of our optimization system and further enhanced the simulation environment into a robust consultant tool.

5 EVALUATION

In order to evaluate our approach, we selected exemplary simulation scenarios. In this section, we explain these scenarios in more detail. Subsequently we present and discuss the simulation results.

5.1 Scenarios

In total we defined two simulation scenarios. We explain these below:

Scenario 1 Input parameters for the simulation tool can be grouped into four categories, namely: booking data, weather data, vehicle data, and infrastructure data. We used booking data for two days. In total, the booking set contains ten reservations with an average reservation time of 161 minutes. From these 161 minutes the vehicles were 152 in use, meaning that the discrepancy between the scheduled pickup and return time and the real pickup and return time was nine minutes in average. The average mileage for each reservation was 38.62 km. We used real data for the weather and selected a stormy day in summer with a rather high values for wind availability, temperature, and global radiation. For this scenario, we used four electric vehicles—one *MCC E-Smart*, two *Citroen C1 Electro*, and one *Citroen C-Zero* (as mentioned in Section 2.1, ten vehicles are installed at the test-site, though, only four of them are “controllable”). The selected car pool is the same as at the real test-site. Finally, we configured the Micro Smart Grid similar to the real one. We used a solar panel with a surface of 383.7 m^2, a wind turbine with a surface of 17.4 m^2, and a stationary grid buffer battery with a capacity of 24.7 kWh. For the MSG’s consumption, we picked a real energy profile from the year 2013. Consumption values range between 7.8 kW and 38.5 kW.

Scenario 2 In this scenario, we used the same values for booking, weather, and vehicle data. We also used the same infrastructure design as in scenario one, yet, we selected a different consumption profile. While in scenario one we used a real energy profile, we simply used no consumption for this second scenario.

5.2 Simulation Results

Due to the fact that our optimization is based on a stochastic mechanism, we simulated each scenario 1.000 times in order to compensate statistical outliers. Results presented here are average values. We respectively compared the performance of our optimization system to the performance of intuitive charging (see Section 4.1.3). In the following, we refer to intuitive charging as “simple” and to optimized charging procedures as “optimized”.

For the evaluation we used the analysis feature of our web front-end. Simulation results are processed and visually illustrated. An exemplary illustration of the analysis tool, showing two energy balances (intuitive and optimized), is presented in Figure 3.

In the first scenario, the simple charging caused a total consumption of 769.56 kWh. The value for optimized charging was slightly smaller, that is 744.57 kWh. A decrease by 3.2%. In the simple case, each kilowatt hour that has been consumed, was burdened by 454.86 grams CO_2. In the optimized case, this value dropped by .5% to 452.50 g/kWh. In the simple case, the energy mix that was used to cover the overall MSG consumption comprised: 7.6% wind energy, 15.7% solar energy, and 76.7% energy from the urban energy grid. In the optimized case, the energy mix comprised: 7.7% wind energy, 16.1% solar energy, 76.2% energy from the urban energy grid. Thus, compared to the simple case, the share of renewable energy sources was increased by .5% in the optimized one.

In the second scenario, the differences became more obvious. The simple charging caused an overall MSG consumption of 127.38 kWh. The consumption for the optimized charging dropped by 14.7% to 108.67 kWh. In the simple case, each kilowatt hour that has been consumed, was burdened by 436.94 grams CO_2. In the optimized case, this value dropped by 15.2% to 371.40 g/kWh. In the simple case, the

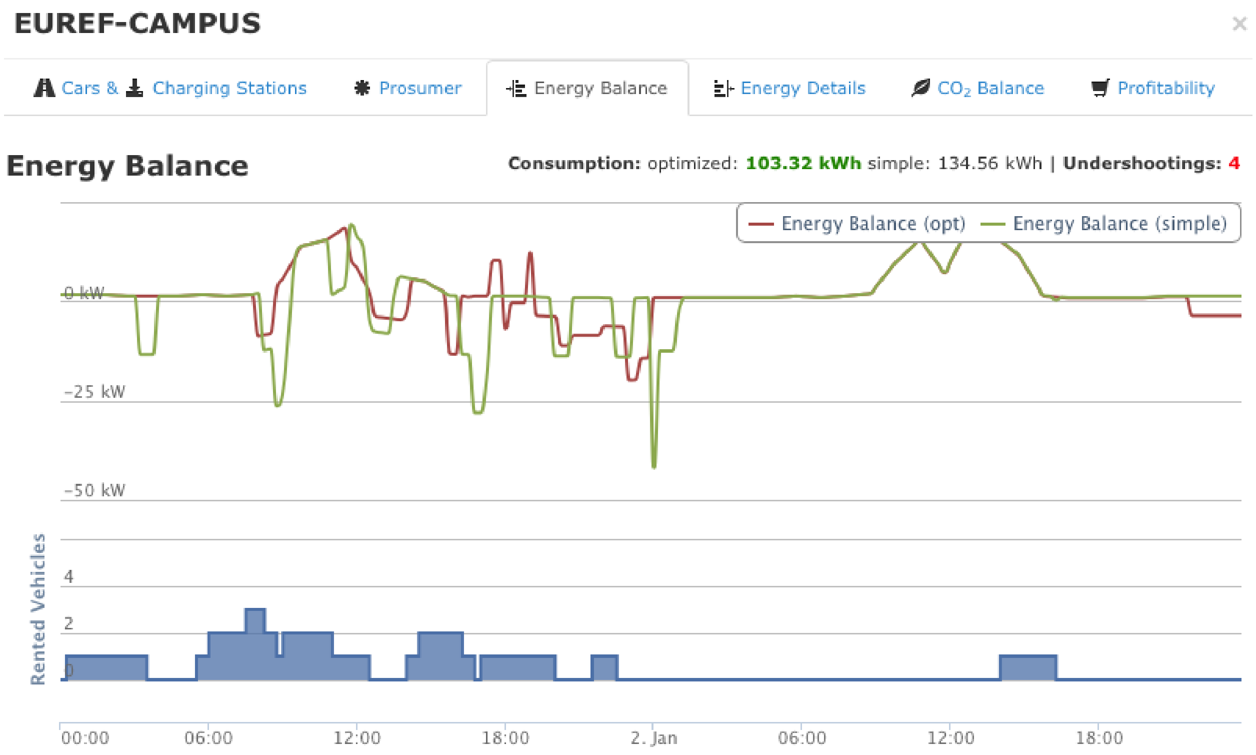

Figure 3: The web front-end comparing the results of intuitive and optimized charging. The tool illustrates the energy balance of both charging profiles (green=intuitive, red=optimized). Furthermore, the absence of electric vehicles is illustrated (blue line at the bottom). It can be seen that returned vehicles correspond with fluctuations in the intuitive energy balance (as a result to the immediate charging process). Absolute consumption values are displayed as well (top right corner).

energy mix that was used to cover the overall MSG consumption comprised: 9.9% wind energy, 16.8% solar energy, and 73.3% energy from the urban energy grid. In the optimized case, the energy mix comprised: 20.5% wind energy, 18.3% solar energy, 61.2% energy from the urban energy grid. Thus, compared to the simple case, the share of renewable energy sources was increased by the optimization, by 15.5%.

5.3 Discussion

Compared to intuitive charging, the optimization algorithm was able to increase the performance of the Micro Smart Grid in both scenarios. In the first scenario, this difference was not as obvious as in the second one. The reason for this is the little impact of locally produced energy on the overall MSG consumption. While the MSG consumption ranges between 7.8 kW and 38.5 kW, the production is mostly around 2–3 kW throughout the entire day, with only two short-term peaks where the production climbs up to 19.24 kW and 19.42 kW, respectively. As a consequence the available energy is completely used to cover the MSG demand and the optimization system is not able to explicitly utilize surpluses of CO_2 efficient energy. As a consequence, the efficiency of the optimized schedules is similar to those schedules that result from intuitive behavior. In the second scenario we emphasized the capabilities of our optimization system by focusing only on charging and feeding processes of electric vehicles. The difference became far more obvious, that is, all relevant values were improved by roughly 15%. While the installed energy producers were not able to cover the energy demand of the entire grid, they were certainly able to produce enough energy for the electric vehicles in the grid. Information about future reservation allowed the optimization system to schedule charging periods within CO_2 effective time periods, without risking that partially charged vehicles are given to customers.

Another result that should be discussed is the overall MSG consumption which varies between the intuitive and the optimized scenario. The difference can be explained with the knowledge of the optimization system. The system knows about future reservations and tries to recharge vehicles before the pickup date. In the case that a particular vehicle has no reservations left, the optimization system refuses to charge this vehicle unless a surplus of renewable energy is available, which would otherwise be wasted.

To sum up, both scenarios substantiate the capability of the optimization system. In the first scenario, the difference between optimized and intuitive charging was rather small and can be explained with the fact that locally produced energy had to be used to cover the MSG demand, thus, only few energy surpluses were available to cover the controllable charging processes of electric vehicles. In the second scenario, the capability became more obvious. The second scenario allowed the optimization system to use charging processes explicitly to utilize energy from renewable energy sources. Therefore, the result was more convincing.

6 CONCLUSION

In this paper we presented a simulation architecture that was used for the development of a charging optimization system. The problem in developing such system was the unavailability of the infrastructure, which made the development process difficult. We decided to place the system within a simulation environment and thus to benefit twice. First, it was possible to develop and to test the system without having a functional real-life test platform. Secondly, it was possible to enhance the simulation environment to a consultant tool, capable to explain the processes at the real test-site and to assess the effects of infrastructure extensions. We applied the multi-agent system approach and implemented all relevant components of the system as JIAC V agents. JIAC provides a reliable infrastructure as well as transparent communication between all involved entities, thus, it was possible to focus on the development of the system rather than putting efforts on fundamental requirements. We implemented our optimization in compliance with the evolution strategy paradigm and used multiple optimization instances in order to counter the problems of stochastic optimization as well as to exploit the hardware infrastructure. We designed the system in a way that allowed its usage in a simulation context as well as in the real context. This was done by implementing all components as JIAC agents and by using the same data-types for the communication between the agents. Currently we use the exact same code for the real as for the simulated optimization system. We evaluated our system by using two exemplary simulation scenarios. While both scenarios had the same values for booking, weather, and vehicle data, we slightly changed the infrastructure data. In more detail, we used the same infrastructure as in reality, but altered the (uncontrollable) energy consumption from the office building at the test-site. While we used original data for the first scenario, we set the consumption to zero for the entire second scenario. The effects were significant. While in the first scenario, the optimization system was not able to provide convincing results, the explicit consideration of vehicle charging processes in the second scenario resulted in a performance improvement of roughly 15%.

The results emphasize that available infrastructures hinder the vision of electric vehicles and smart grids, yet, the same results also emphasize the capabilities of such combination and further the enthusiasm to turn the vision into reality.

ACKNOWLEDGEMENTS

This work was partially funded by the *Federal Ministry of Education and Research* under the funding reference number 03FO16004D.

REFERENCES

Freund, D., A. F. Raab, T. Küster, S. Albayrak, and K. Strunz. 2012, October. "Agent-based Integration of an Electric Car Sharing Fleet into a Smart Distribution Feeder". In *Proceedings of the 3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies Europe (ISGT Europe), Berlin, Germany*, 1–8: IEEE.

Gray, T., and J. Francfort. 2012, February. "Bi-Directional Fast Charging Study Report". Technical report, U.S Department of Energy.

Kamboj, S., N. Pearre, W. Kempton, K. Decker, K. Trnka, and C. Kern. 2010. "Exploring the formation of Electric Vehicle Coalitions for Vehicle-To-Grid Power Regulation". In *Proceedings of the 1st International Workshop on Agent Technologies for Energy Systems (ATES 2010), Toronto, Canada*, 1–8.

Keiser, J., J. Glass, N. Masuch, M. Lützenberger, and S. Albayrak. 2011, October. "A Distributed Multi-Operator W2V2G Management Approach". In *Proceedings of the 2nd IEEE International Conference on Smart Grid Communications, Brussels, Belgium*, 291–296: IEEE.

Kempton, W., V. Udo, K. Huber, K. Komara, S. Letendre, S. Baker, D. Brunner, and N. Pearre. 2008, November. "A Test of Vehicle-to-Grid (V2G) for Energy Storage and Frequency Regulation in the PJM System". Technical report, University of Delaware.

Küster, T., M. Lützenberger, D. Freund, and S. Albayrak. 2013. "Distributed Evolutionary Optimisation for Electricity Price Responsive Manufacturing using Multi-Agent System Technology". *International Journal On Advances in Intelligent Systems* 7 (1&2): 27–40.

Lützenberger, M., T. Küster, T. Konnerth, A. Thiele, N. Masuch, A. Heßler, J. Keiser, M. Burkhardt, S. Kaiser, J. Tonn, M. Kaisers, and S. Albayrak. 2013. "A Multi-Agent Approach to Professional Software Engineering". In *Engineering Multi-Agent Systems — First International Workshop, EMAS 2013, St. Paul, MN, USA, May 6–7, 2013, Revised Selected Papers*, edited by M. Cossentino, A. E. F. Seghrouchni, and M. Winikoff, Volume 8245 of *Lecture Notes in Artificial Intelligence*, 158–177. Springer Berlin / Heidelberg.

Markel, T., K. Bennion, W. Kramer, J. Bryan, and J. Giedd. 2009, August. "Field Testing Plug-in Hybrid Electric Vehicles with Charge Control Technology in the Xcel Energy Territory". Technical report, National Renewable Energy Laboratory.

Rechenberg, I. 1973. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata. Stuttgart-Bad Cannstatt: Frommann-Holzboog.

AUTHOR BIOGRAPHIES

MARCO LÜTZENBERGER received a diploma in computer science in 2009 from the *Technische Universität Berlin*. From 2009 until now, he is part of the *Competence Center Agent Core Technologies* at the *DAI-Lab*, which he is directing since 2014. He is working on his doctoral dissertation in the field of multi-agent based traffic simulation models. His email address is marco.luetzenberger@dai-labor.de.

TOBIAS KÜSTER has obtained his diploma in computer science in 2007 at *Technische Universität Berlin* and is currently working on his doctoral thesis in the field of integrating business process design into agent oriented software engineering. His email address is tobias.kuester@dai-labor.de.

SAHIN ALBAYRAK is Professor of Agent Technologies in Business Applications and Telecommunication at Technische Universität Berlin. He is the founder and head of the *DAI-Lab*, currently employing about one hundred researchers and support staff. His email address is sahin.albayrak@dai-labor.de.