D4.7 - Final Cooperative EV charging station control algorithms

Deliverable ID	**D4.7**
Deliverable Title	**Final Cooperative EV charging station control algorithms**
Work Package	**WP4**

Dissemination Level	**PUBLIC**

Version	**1.0**
Date	**31/08/2019**
Status	**Final**
Type	**Prototype**

Lead Editor	**Fraunhofer FIT**
Main Contributors	**FRAUNHOFER FIT (Gustavo Aragón)**

Published by the S4G Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731155.

Document History

Version	Date	Author(s)	Description
0.1	2019-07-25	FIT	First draft.
0.2	2019-08-12	FIT	Corrections after the review
1.0	2019-08-19	FIT	Final version

Internal Review History

Review Date	Reviewer	Summary of Comments
2019-08-12 (v0.1)	Vasco Delgado-Gomes (UNINOVA)	Approved: • Minor corrections to improve the document readability. • Acronyms list needs to be completed.
2019-07-29 (v0.1)	(LIBAL)	Approved: • Minor corrections • Some questions to the algorithm

Table of Contents

Executive Summary

This deliverable presents the developed algorithms implementing optimal control of electric vehicles' (EVs) charging in a car park taking uncertainties into consideration. The algorithms are implemented into a software tool named *Professional Realtime Optimization Framework for Electric Vehicles* (PROFEV) which was developed within Storage4Grid (S4G).

PROFEV is a software component running stochastic optimization algorithms for the control of charging of EVs and the charging/discharging of energy storage systems (ESS). It receives in quasi real time all available information from local devices (e.g., ESS systems, PV energy meters, load-side energy meters, EV chargers, grid connection). In S4G, PROFEV works also together with Grid-side Energy Storage System Control (GESSCon), a global ESS controller service, linking the charging/discharging profiles for the next 24-hours sent by GESSCon with the internal optimal control model of PROFEV.

For clarification, PROFEV differs from PROFESS presented in D4.3 [S4G-D4.3], in the fact that it runs stochastic optimization models for taking into consideration the EVs uncertainty (e.g. plugin time of an EV to the charging station or the EV's battery consumption) in the optimal control mechanism.

This is the final deliverable handling the development of the "Cooperative EV charging station control algorithms". The algorithm will be further evaluated in the test sites in D6.12 – "Phase 3 Evaluation Report".

1 Introduction

D4.7 describes the "Final Cooperative EV charging station control algorithms" prototype, developed within the S4G project. The optimal control algorithm uses the concept of virtual capacity (VAC) for modelling the system, where VAC is the summation of each single EV capacity in the car park. The PROFEV tool uses stochastic dynamic programming for optimizing the charging of electric vehicles (EVs) as well as the charging and discharging of energy storage systems (ESS). The uncertainty concerning the presence of EVs is modelled using Markov chains. The PROFEV tool is also responsible for implementing the algorithms in a real-time optimal control system.

The PROFEV prototypes and how they work are presented in detail in this deliverable. PROFEV also exchanges data with the cloud components, namely the Grid-side Energy Storage System Control (GESSCon) and the Decision Support Framework (DSF). More detailed information can be retrieved from related documents summarized in section 1.2.

1.1 Scope

This prototype deliverable has been developed in Task T4.3 – Cooperative EV charging station control. It is an updated version of D4.6 [S4G-D4.6] and no further updates are expected.

1.2 Related documents

ID	Title	Reference	Version	Date
D2.2	Final Storage Scenarios and Use Cases	[S4G-D2.2]	1.0	2018-07-31
D3.2	Updated S4G Components, Interfaces and Architecture Specification	[S4G-D3.2]	1.0	2018-08-30
D4.3	Final User-side ESS control system	[S4G-D4.3]	1.0	2019-06-13
D4.5	Final Grid-side ESS Control System	[S4G-D4.5]	1.0	2018-07-31
D4.9	Updated USM Extensions for Storage Systems	[S4G-D4.9]	1.0	2018-08-31
D4.6	Initial Cooperative EV charging station control algorithms	[S4G-D4.6]	1.0	2018-08-14
D5.2	Final DSF Hybrid Simulation Engine	[S4G-D5.2]	0.5	2019-08-09
D6.2	Phase 2 Test Site Plans	[S4G-D6.2]	1.0	2018-12-28

2 User-side ESS Control System Prototype Overview

The overall high-level structure of D4.7 prototype – "Final Cooperative EV charging station control algorithms" is shown in Figure 1.

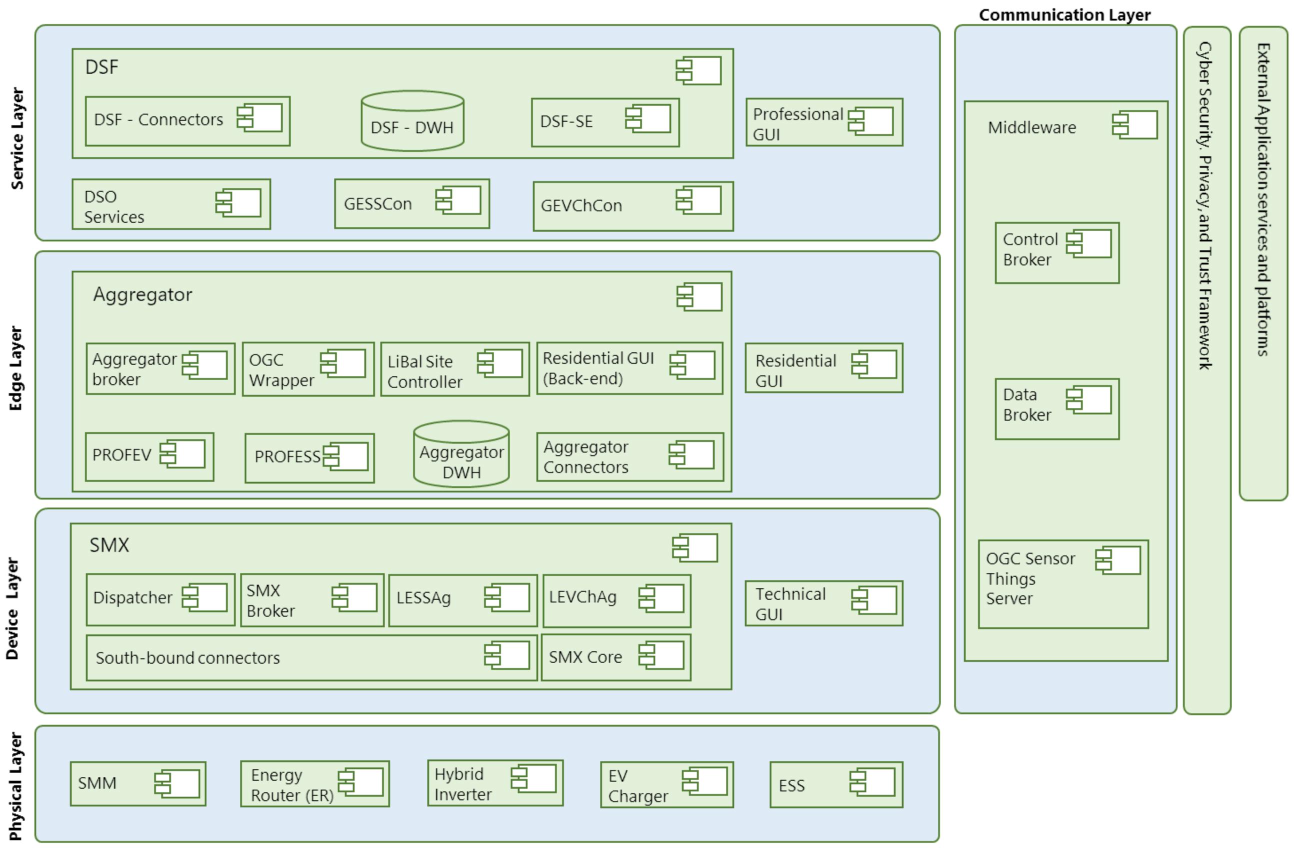


Figure 1. Prototype diagram [S4G-D3.2]-Figure 7.

Figure 1 taken from D3.2 [S4G-D3.2] shows the functional layer where the PROFEV System is located in S4G, i.e. at the edge layer.

2.1 PROFEV

This is the first deliverable where the functionality of PROFEV is presented. Nevertheless, PROFEV is an extension of the Professional Realtime Optimization Framework for Energy Storage Systems (PROFESS) tool (presented in D4.3 [S4G-D4.3]) for allowing the use of optimization models while integrating uncertainty into them. It works not only as user-side EV charging and ESS controller, but it is also designed and implemented to work together with the Decision Support Framework Simulation Engine (DSF-SE).
PROFEV uses Model Predictive Control (MPC) in a real-time application[i] including uncertainties brought by some parameters such as the plugin/unplug time of EVs. It means reading in real-time inputs, calculating an optimization problem and delivering outputs for the next time step as set-points to the different south-bound connectors of the S4G system. It is important to mention that some of these inputs have to be predicted for the optimization horizon being used. Examples of these predictions constitute individual electric consumption (load) or Photovoltaic (PV) generation for the next 24-hours. Predictions are calculated by PROFEV, exactly as it is done by PROFESS, to guarantee data privacy (the data does not leave the test site where it has been generated) and to fulfil the requirements of an optimization problem.

2.1.1 Architecture

The PROFEV architecture is shown in Figure 2. Functionalities of some modules were already described in D4.3 [S4G-D4.3].

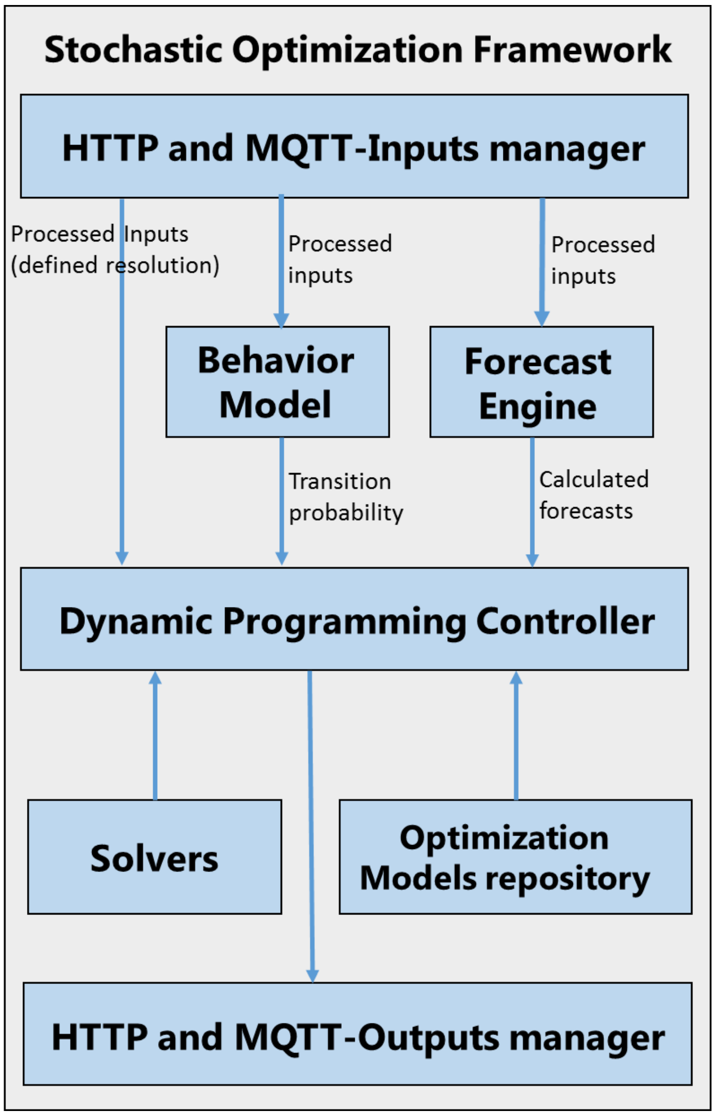

Stochastic Optimization Framework
HTTP and MQTT-Inputs manager
Processed Inputs (defined resolution)
Processed inputs
Processed inputs
Behavior Model
Forecast Engine
Transition probability
Calculated forecasts
Dynamic Programming Controller
Solvers
Optimization Models repository
HTTP and MQTT-Outputs manager

Figure 2. PROFEV architecture.

2.1.1.1 *HTTP and MQTT-Inputs and -Outputs Manager*

HTTP and MQTT- Inputs and -Outputs Managers were designed to use Sensor Measurement Lists (SenML) data format as standard. In this way, the information being delivered by sensors and the set-points calculated by PROFEV can be easily linked to other components inside the S4G project. One example is the integration with the EV charging station connector developed by LINKS, as well as the integration of PROFEV with the DSF-SE. For other components such as Smart Meter eXtension (SMX), a dedicated connector was developed. The SenML connector is fully detailed in D4.3 [D4.3].

2.1.1.2 Forecast Engine

PROFEV needs the forecast of some inputs for the calculation of the optimization problem, e.g., the load forecast. Because the optimization problem inside PROFEV analyses the future of the system in a determined time horizon, the inputs to the optimization model have to be defined in that specific horizon, such as the load. For this reason, PROFEV embeds the calculation of the forecasts. The information is then processed and delivered to the dynamic programming controller (DPC) exactly in the format it is needed.

2.1.1.3 Behaviour Model

The *behaviour model (BM)* module describes the uncertainty brought either by the plugin and unplug time of the EVs to the charging stations or by the presence of a number of EVs at a certain time. The modelling of the uncertainty was already presented in D4.6 [S4G-D4.6]. It makes use of an internal Montecarlo simulation module that simulates a number of scenarios taking into consideration mean value and standard deviation of the parameters. As an example, the plugin time of the EV is simulated using its mean value and standard deviation, which are entered through the PROFEV Application Program Interface (API). This simulation will deliver the necessary data for calculating the transition matrices, which are used by the calculation of the stochastic optimization problem.

2.1.1.4 Dynamic Programming Controller

The *Dynamic Programming Controller (DPC)* was included into PROFEV in order to allow the use of uncertainties into the optimization model. DPC was already explained in D4.6 [S4G-D4.6]. It divides the whole optimization problem into sub-optimization problems to allow the calculation of costs of a specific decision at a specific time step. For this, the DPC iterates from the final time step to the first one analysing all discrete states describing the output variables.
DPC uses Pyomo[iii] as core tool for understanding the optimization models entered by the user. Based on the models, DPC can understand which are the data inputs and outputs needed for calculating the optimization problem and wait for them. In fact, the controller will start the optimization calculation only when all sets of inputs are present as real-time data.

2.1.1.5 Optimization Models Repository

The *optimization models repository (OMR)* module enables the storing of different optimization models to the system. The user can use the models' endpoint from the PROFEV API[ii] to load any optimization model that he or she developed using the mathematical framework Pyomo[iii]. The user can also give a name to this model, which will be used to store it into the OMR.
Once the user wants to run a model, he or she has to specifically link it using the optimization/start endpoint under the flag <model_name>.

2.1.1.6 Mathematical Solvers

Compared to D4.6 [S4G-D4.6], solvers inside PROFEV were upgraded to the most recent versions. It includes GNU Linear Programming Kit (GLPK)[iv], Interior Point OPTimizer (Ipopt)[v] and Basic Open-source Nonlinear Mixed INteger (Bonmin)[vi]. Moreover, a new solver for Mix Integer Problems (MIP) was added. Its name is Coin-or Branch and Cut (CBC)[vii]. The Gurobi[viii] solver was also added to the system and can be linked only in Advanced Micro Devices (AMD) infrastructures using a floating license running on a server.

2.2 Optimization algorithms

In this section, the different optimization algorithms used for the residential and commercial Bolzano use cases will be presented.

2.2.1 Bolzano commercial test site use case

The general structure of the Bolzano commercial use case is presented in Figure 3. There are a number of electric vehicles that need to be charged. The system also has renewable generation in form of photovoltaics (PV) panels and an energy storage system (ESS). Energy can be taken from or fed into the grid. A more detailed diagram of the use case is presented in D6.2 [S4G-D6.2].

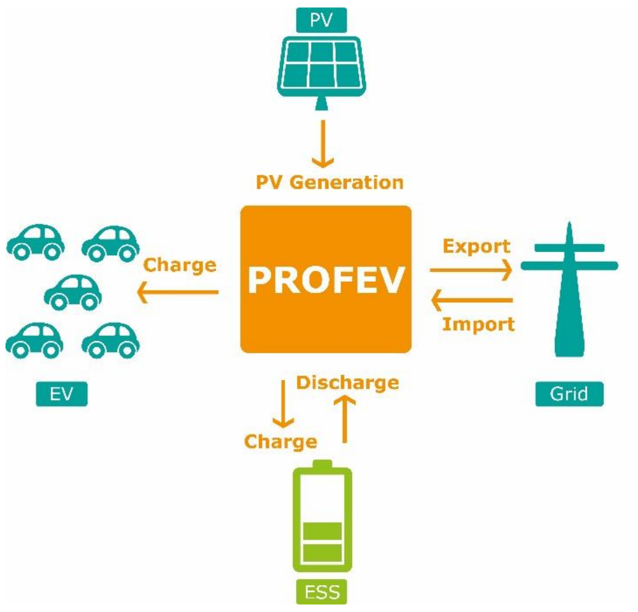


Figure 3. Bolzano commercial use case.

2.2.1.1 *Inputs for the optimization problem*

The optimization models require the inputs summarized in Table 1.

Table 1. Inputs example for the optimization model in Bolzano commercial.

Input_name	Value	MQTT parameters		Description
ESS_Min_SoC	0.2			Minimum SoC that the ESS can get
ESS_Max_SoC	1			Maximum SoC that the ESS can get
SoC_Value		url	172.17.0.1	SoC value in real-time being read from the ESS. It is always between 0 and 1
		topic	Input/SoC	
		QoS	1	
		predict	false	

ESS_Capacity	70			ESS capacity in Kilowatt hour given by the manufacturer
ESS_Max_Charge_Power	33			Maximum power in Kilowatt for charging the ESS given by the ESS inverter
ESS_Max_Discharge_Power	33			Maximum power in Kilowatt for discharging the ESS given by the ESS inverter
P_Grid_Max_Export_Power	150			Maximum power in Kilowatt to export to the grid
P_PV_50		url	172.17.0.1	PV power in Watt in real-time. It is always positive
		topic	input/P_PV_50	
		QoS	1	
		predict		
PV_Inv_Max_Power	50			Maximum PV power in Kilowatt that the PV can generate
P_PV_90		url	172.17.0.1	PV power in Watt in real-time. It is always positive
		topic	input/P_PV_90	
		QoS	1	
		predict		
PV_90_Inv_Max_Power	90			Maximum PV power in Kilowatt that the PV can generate
P_Load		url	172.17.0.1	Consumption power in Watt in real-time. It is always positive
		topic	input/P_Load	
		QoS	1	
		predict	true	

Some of them are constant values and others are real-time values read from the SenML connector or from the GESSCon connector and received as MQTT messages.

The inputs for EVs and charging stations are summarized in Table 2.

Table 2. Inputs example of EV and charging station information

Input_name	**Value**	**MQTT parameters**	**Description**

Max_Charging_Power_kW	7 or 22			Maximum charging power in Kilowatt for each charging station.
Battery_Capacity_kWh	18.7			EV battery capacity for each EV.
EV_SoC_Value		url	172.17.0.1	SoC value in real-time being read from the EV.
		topic	Input/EV_SoC	
		QoS	1	
		predict	false	
Start_Stop_Charging		url	172.17.0.1	1: Start of the recharging session. 0: Stop of the recharging session.
		topic	input/Start_Stop	
		QoS	1	
		predict		

The inputs for the uncertainty are summarized in Table 3.

Table 3. Inputs example for uncertainty definition.

Input_name	**Value**	**Description**
ESS_States	10	Steps between each ESS state for the DP calculation.
VAC_States	2.5	Steps between each VAC state for the DP calculation.
Plug_time	18.76 ±1.3 h	Mean and standard deviation of the plug-in time of the EVs.
Unplug_time	7.32 ±0.78 h	Mean and standard deviation of the unplug time of the EVs.

2.2.1.2 Optimization outputs

The optimization model provides the outputs summarized in Table 4. Outputs are delivered in different MQTT topics.

Table 4. Outputs example for the optimization model in Bolzano commercial.

Output_name	**MQTT parameters**		**Description**
P_Grid_Output	url	172.17.0.1	Power to be sent to the grid or supplied by the grid. If positive, the power is supplied by the grid.
	topic	output/P_Grid_Output	
	QoS	1	
	unit	Watt	

	horizon_steps	false	
P_PV_Output	url	172.17.0.1	PV power to be delivered to the internal electric system. It is always positive
	topic	output/P_PV_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	
P_ESS_Output	url	172.17.0.1	Charging or discharging power to or from the ESS. If positive, the ESS is discharging.
	topic	output/P_ESS_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	
P_EV_Output	url	172.17.0.1	PV power to be delivered to the specific EV in a specific charging station. It is always positive
	topic	output/P_EV1_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	

2.2.1.3 Start configuration

For starting PROFEV, the configuration values of the optimization are required (Table 5).

Table 5. Configuration values of the optimization.

Configurations	Value	Description
control_frequency	1800	Time step in seconds, in which the optimization calculation is going to be repeated
horizon_in_steps	48	Number of steps for the optimization horizon.
dT_in_seconds	1800	Time in seconds between each horizon step.
repetition	-1	Number of repetitions of the MPC. -1 means infinitely
solver	cbc	Name of the solver to be used. Possibilities: GLPK, Ipopt, Bonmin, CBC and Gurobi
optimization_type	stochastic	Defines the type of optimization to be calculated inside the platform. PROFEV uses the stochastic option for the Bolzano commercial use case.
model_name	CarParkModel	Model name of the optimization model to be used as it is stored inside the OMR.

Taking into consideration the number of steps of the optimization horizon (horizon_in_steps) of 48 and the time between steps (dT_in_seconds) of 1800, the optimization inside PROFEV will be calculated throughout the next 24-hours with a resolution of 30 minutes.

2.2.1.4 Optimization definition

The definition of uncertainty, the concept of a virtual capacity (VAC), and the definition of the stochastic equation for the optimization problem were already presented in section 3 of D4.6 [S4G-D4.6]. This section focus on defining the final objective function with and without the integration of GESSCon.

As explained in D4.6 [S4G-D4.6], the objective function takes the general stochastic form as in equation 2.1.

$$\min c_t(x_t) + \sum_{w=1}^{W} \xi_t{}^w v_{t+1}(s_{t+1}) \qquad 2.1$$

Translated to the cooperative charging station problem, equation 2.2 maximizes the utilization of the renewable potential, in this case the PV generation.

$$min\, P_{pv,max}^{t} - x_{PV} + \sum_{w=1}^{W} p(w\colon s^{t+1}|s^{t}) v_{t+1}(s_{\text{ess\_}SoC}^{t+1}, s_{\text{ev\_}SoC}^{t+1}, s_{plugged}^{t+1}) \qquad 2.2$$

Where x_{PV} is the output power to the PV and $P_{pv,max}^{t}$ is the power being delivered by the PV at that instant of time. The value at each state is calculated with equation 2.3.

$$v_t(s_{\text{ess}_{SoC}}^{t+1}, s_{\text{ev}_{SoC}}^{t+1}, s_{plugged}^{t+1}) = P_{pv,max}^{t} - x_{PV}^{*} + v_{t+1}(s_{\text{ess\_}SoC}^{t+1}, s_{\text{ev\_}SoC}^{t+1}, s_{plugged}^{t+1}) \qquad 2.3$$

Where x_{PV}^{*} is the optimal value of the PV power output variable at each state and every horizon step of the stochastic problem. The variables that describe the system state are the SoC of ESS (stationary battery) and SoC of VAC so that the system switches between the combinations of these states:

$$\begin{gathered} s_{ESS}^{t} \rightarrow s_{ESS}^{t+1} \\ s_{VAC}^{t} \rightarrow s_{VAC}^{t+1} \end{gathered} \qquad 2.4$$

In the case of GESSCon, the deviation between the GESSCon schedule ($x_{GESSCon}$) and the ESS power (x_{ESS}) is as follows:

$$dev = x_{GESSCon} - x_{ESS} \qquad 2.5$$

Inserting this term into the objective function (equation 2.1), equation 2.6 is obtained.

$$min\, w_1 * (P_{pv,max}^{t} - x_{PV})^2 + w_2 * dev^2 + \sum_{w=1}^{W} p(w\colon s^{t+1}|s^{t}) v_{t+1}(s_{\text{ess\_}SoC}^{t+1}, s_{\text{ev\_}SoC}^{t+1}, s_{plugged}^{t+1}) \qquad 2.6$$

Where w_1 and w_2 are two different weights that prioritize the local or global action.

The value at each state is calculated with:

$$v_t(s^{t+1}_{\text{ess}_{SoC}}, s^{t+1}_{\text{ev}_{SoC}}, s^{t+1}_{plugged}) = w_1 * (P^t_{pv,max} - x^*_{PV})^2 + w_2 * (dev^*_t)^2 + v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \quad 2.7$$

Where x^*_{PV} and dev^*_t are the optimal values of the PV power output and deviation variables at each state and every horizon step of the stochastic problem.

2.2.2 Bolzano residential test site use case

The general structure of the Bolzano's residential use case is presented in Figure 4. As observed, there is one EV that needs to be charged. The system also has renewable generation in form of PV panels and an ESS. Energy can be taken from or fed into the grid. A more detailed diagram of the use case is presented in D6.2 [S4G-D6.2].

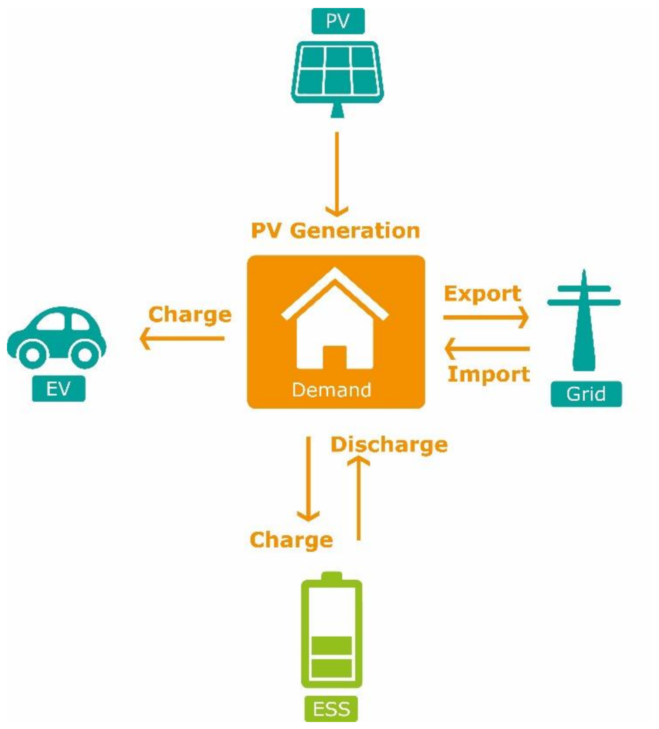


Figure 4. Bolzano residential use case.

2.2.2.1 Inputs for the optimization problem

The optimization models require the inputs summarized in Table 6.

Table 6. Inputs example for the optimization model in Bolzano residential.

Input_name	Value	MQTT parameters		Description
ESS_Min_SoC	0.2			Minimum SoC that the ESS can get
ESS_Max_SoC	1			Maximum SoC that the ESS can get

SoC_Value		url	172.17.0.1	SoC value in real-time being read from the ESS. It is always between 0 and 1
		topic	Input/SoC	
		QoS	1	
		predict	false	
ESS_Capacity	9.6			ESS capacity in Kilowatt hour given by the manufacturer
ESS_Max_Charge_Power	6.4			Maximum power in Kilowatt for charging the ESS given by the ESS inverter
ESS_Max_Discharge_Power	6.4			Maximum power in Kilowatt for discharging the ESS given by the ESS inverter
P_Grid_Max_Export_Power	6			Maximum power in Kilowatt to export to the grid
P_PV		url	172.17.0.1	PV power in Watt in real-time. It is always positive
		topic	input/P_PV_50	
		QoS	1	
		predict		
PV_Inv_Max_Power	7.6			Maximum PV power in Kilowatt that the PV can generate
P_Load		url	172.17.0.1	Consumption power in Watt in real-time. It is always positive
		topic	input/P_Load	
		QoS	1	
		predict	true	

Some of the inputs are constant values and others are real-time values read from the SenML connector or from the GESSCon connector as MQTT messages.

The inputs for EVs and charging stations are summarized in Table 7.

Table 7. Inputs example for EV information

Input_name	**Value**	**MQTT parameters**		**Description**
Max_Charging_Power_kW	3			Maximum charging power in Kilowatt hour

				for each charging station.
Battery_Capacity_kWh	30			EV battery capacity for each EV.
EV_SoC_Value		url	172.17.0.1	SoC value in real-time being read from the EV.
		topic	Input/EV_SoC	
		QoS	1	
		predict	false	
Start_Stop_Charging		url	172.17.0.1	1: Start of the recharging session. 0: Stop of the recharging session.
		topic	input/Start_Stop	
		QoS	1	
		predict		

The inputs for the uncertainty are summarized in Table 8.

Table 8. Inputs example for uncertainty definition.

Input_name	**Value**	**Description**
ESS_States	10	Steps between each ESS state for the DP calculation.
VAC_States	2.5	Steps between each VAC state for the DP calculation.
Plug_time	18.76 ±1.3 h	Mean and standard deviation of the plug time of the EVs.
Unplug_time	7.32 ±0.78 h	Mean and standard deviation of the unplug time of the EVs.

2.2.2.2 Optimization outputs

The optimization model provides the outputs summarized in Table 9. Outputs are delivered in different MQTT topics.

Table 9. Outputs example for the optimization model in Bolzano commercial.

Output_name	**MQTT parameters**		**Description**
P_Grid_Output	url	172.17.0.1	Power to be sent to the grid or supplied by the grid. If positive, the power is supplied by the grid.
	topic	output/P_Grid_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	
P_PV_Output	url	172.17.0.1	

	topic	output/P_PV_Output	PV power to be delivered to the internal electric system. It is always positive
	QoS	1	
	unit	Watt	
	horizon_steps	false	
P_ESS_Output	url	172.17.0.1	Charging or discharging power to or from the ESS. If positive, the ESS is discharging.
	topic	output/P_ESS_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	
P_EV_Output	url	172.17.0.1	PV power to be delivered to the specific EV in a specific charging station. It is always positive
	topic	output/P_EV1_Output	
	QoS	1	
	unit	Watt	
	horizon_steps	false	

2.2.2.3 Start configuration

For starting PROFEV, the configuration values of the optimization are required (Table 10).

Table 10. Configuration values of the optimization.

Configurations	Value	Description
control_frequency	900	Time step in seconds, in which the optimization calculation is going to be repeated
horizon_in_steps	96	Number of steps for the optimization horizon.
dT_in_seconds	900	Time in seconds between each horizon step.
repetition	-1	Number of repetitions of the MPC. -1 means infinitely
solver	CBC	Name of the solver to be used. Possibilities: GLPK, Ipopt, Bonmin, CBC and Gurobi
optimization_type	Stochastic_residential	Defines the type of optimization to be calculated inside the platform. PROFEV uses the stochastic option for the residential case.
model_name	BolzanoResidential	Model name of the optimization model to be used as it is stored inside the OMR.

Taking into consideration the number of steps of the optimization horizon (horizon_in_steps) of 96 and the time between steps (dT_in_seconds) of 900, the optimization inside PROFEV will be calculated throughout the next 24-hours with a resolution of 15 minutes.

2.2.2.4 Optimization definition

The outputs or controllable variables of the optimization model for the Bolzano residential case are the following:

$$x = \begin{vmatrix} x_{ESS} \\ x_{EV} \\ x_{PV} \\ x_{GRID} \end{vmatrix} \qquad 2.8$$

where x_{ESS} is the ESS charge/discharge power, x_{EV} the EV battery charge power, x_{PV} the PV generation power, and x_{GRID} the grid power, which takes positive or negative values in case of energy import or export, respectively.
As a difference to the car park model of section 2.2.1, the variable s_{pos}^{t} (position of the EV: home or away) is introduced in this stochastic optimization model. Therefore, the three variables that describe the system state are SoC of ESS, SoC of EV's battery and position of EV:

$$s^t = \begin{vmatrix} s_{SoC,ESS}^{t} \\ s_{SoC,EV}^{t} \\ s_{pos}^{t} \end{vmatrix} \qquad 2.9$$

where s_{pos}^{t} is a binary, i.e. 1 when the car is at home, 0 when the car is away. On the contrary, ESS and EV SoC are continuous physical variables, which have to be discretized in the mathematical model of Stochastic dynamic programming (SDP).

The transition between ESS SoC values is deterministic and has to meet the following constraint:

$$s_{SoC,ESS}^{t+1} = s_{SoC,ESS}^{t} + x_{ESS} \frac{\Delta T}{3600} \frac{1}{E_{ESS,cap}} \qquad 2.10$$

Evolution of EV battery's SoC is deterministic only when the car is present at home. Otherwise, it is quasi-deterministic based on the assumption presented in D4.6 [S4G-D4.6] while modelling the uncertainty.

$$s_{SoC,EV}^{t+1} = s_{SoC,EV}^{t} + \frac{\Delta T}{3600} \begin{cases} x_{EV} & , s_{pos}^{t} = 1 \\ -P_{EV}, & s_{pos}^{t} = 0 \end{cases} \qquad 2.11$$

In the previous two equations, the term $\frac{\Delta T}{3600}$ corresponds to the discretization step (in hours) that allows switching from energy to power domain while battery capacities are expressed in Kilowatt hour.

As already mentioned in D4.6 [S4G-D4.6], the stochastic relationship between position states in successive stages is suitable to model with Markov chains. Therefore, the transition probabilities from home to away state $p_{10}(t)$ and from home to home state $p_{11}(t)$ are expressed as show in equation 2.12.

$$\begin{aligned} p_{10}(t) &= p\left(s_{pos}^{t+1} = 0 \middle| s_{pos}^{t} = 1\right) \\ p_{11}(t) &= p\left(s_{pos}^{t+1} = 1 \middle| s_{pos}^{t} = 1\right) \end{aligned} \qquad 2.12$$

$$p_{10}(t) + p_{11}(t) = 1$$

PV output depends on the weather; and is thus constrained by the maximum power point of the time interval, as expressed in equation 2.13. EV charging power term is constrained by the charger's power limitation $P^t_{evCh,max}$ and can take non-zero value only when the EV is present at home, as show in equation 2.14. Equation 2.15 represents the electric power balance that assures meeting the household demand while charging the EV.

$$x_{PV} \leq P^t_{pv,max} \tag{2.13}$$

$$x_{EV} \leq P^t_{evCh,max} \tag{2.14}$$

$$P^t_{dem} + x_{EV} = x_{PV} + x_{ESS} + x_{Grid} \tag{2.15}$$

As explained in D4.3 [S4G-D4.3], SDP solves an optimization function that has two terms: incurred immediate cost of taken action and future cost of taken decision (Equation 2.1). As dynamic programs are solved backwards, the calculation starts from the final stages. Zero is assigned as cost V^T of being at any state in final stage. In equation 2.16 s stands for a specific ESS SoC-EV SoC state combination, and S for all possible combinations:

$$V^T = 0 \quad \forall s \in S \tag{2.16}$$

The following equations show the optimization objectives for the SDP in the Bolzano residential case.

$$min\, P^t_{pv,max} - x_{PV} + \sum_{w=1}^{W} p(w\colon s^{t+1}|s^t) v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \tag{2.17}$$

$$min\, cx_{GRID} + \sum_{w=1}^{W} p(w\colon s^{t+1}|s^t) v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \tag{2.18}$$

$$min\, {x_{GRID}}^2 + \sum_{w=1}^{W} p(w\colon s^{t+1}|s^t) v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \tag{2.19}$$

Equation 2.17 maximizes the PV utilization, Equation 2.18 minimizes the power bill, and equation 2.19 minimizes the power exchange with the grid.
The value of a state is the value of the objective function when the optimal decision is calculated in a state. The following equations show the calculation of the value for the respective objective function according to the optimization result-

$$v_t(s^{t+1}_{\text{ess}_{SoC}}, s^{t+1}_{\text{ev}_{SoC}}, s^{t+1}_{plugged}) = P^t_{pv,max} - x^*_{PV} + v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \tag{2.20}$$

$$v_t(s^{t+1}_{\text{ess}_{SoC}}, s^{t+1}_{\text{ev}_{SoC}}, s^{t+1}_{plugged}) = cx^*_{GRID} + v_{t+1}(s^{t+1}_{\text{ess\_}SoC}, s^{t+1}_{\text{ev\_}SoC}, s^{t+1}_{plugged}) \tag{2.21}$$

$$v_t(s_{\mathrm{ess}_{SoC}}^{t+1}, s_{\mathrm{ev}_{SoC}}^{t+1}, s_{plugged}^{t+1}) = {x_{GRID}^{*}}^{2} + v_{t+1}(s_{\mathrm{ess}\_SoC}^{t+1}, s_{\mathrm{ev}\_SoC}^{t+1}, s_{plugged}^{t+1}) \qquad 2.22$$

3 PROFEV - Installation/Deployment instructions

PROFEV uses a container architecture which simplifies its deployment. Images of the software are uploaded into an open space of Docker Hub[ix]. The user has to download the images and make them run into the final platform (AMD or ARM). A complete installation guide is presented under the Linksmart-Optimization Framework web site[x].

In summary, the installation consists in downloading the docker-compose file from the repository, pulling the images and starting the container. The current docker-compose file is the following for AMD platforms:

```
version: '3.2'
services:
  redis:
    image: redis:latest
    container_name: "redis_S4G"
    command: redis-server
    ports:
      - "6379:6379"
    restart: "always"
  ofw:
    image: garagon/optimization:aamd
    container_name: "ofw"
    ports:
      - "8080:8080"
    depends_on:
      - redis
    volumes:
      - ./optimization-framework/prediction/resources:/usr/src/app/prediction/resources
      - ./optimization-framework/optimization/resources:/usr/src/app/optimization/resources
      - ./optimization-framework/utils:/usr/src/app/utils
      - ./optimization-
framework/utils/gurobi/license:/usr/src/app/share/gurobi811/license/gurobi.lic
    command: ["python3", "-u", "ofw.py"]
    restart: "always"
  ml:
    image: garagon/optimization:arm_training
    container_name: "ml"
    depends_on:
      - redis
    volumes:
      - ./optimization-framework/prediction/resources:/usr/src/app/prediction/resources
    command: ["python3", "-u", "mlTraining.py"]
    restart: "always"
```

The command for pulling the image and starting the code is:

```
docker-compose -f {docker_compose_file_name} up –d
```

After this step, PROFEV is ready to run. The registry of inputs and outputs and the start command can be entered through HTTP commands using the PROFEV API.

4 PROFEV - Software dependencies and requirements

No special hardware requirements are needed to implement the final version of PROFEV.

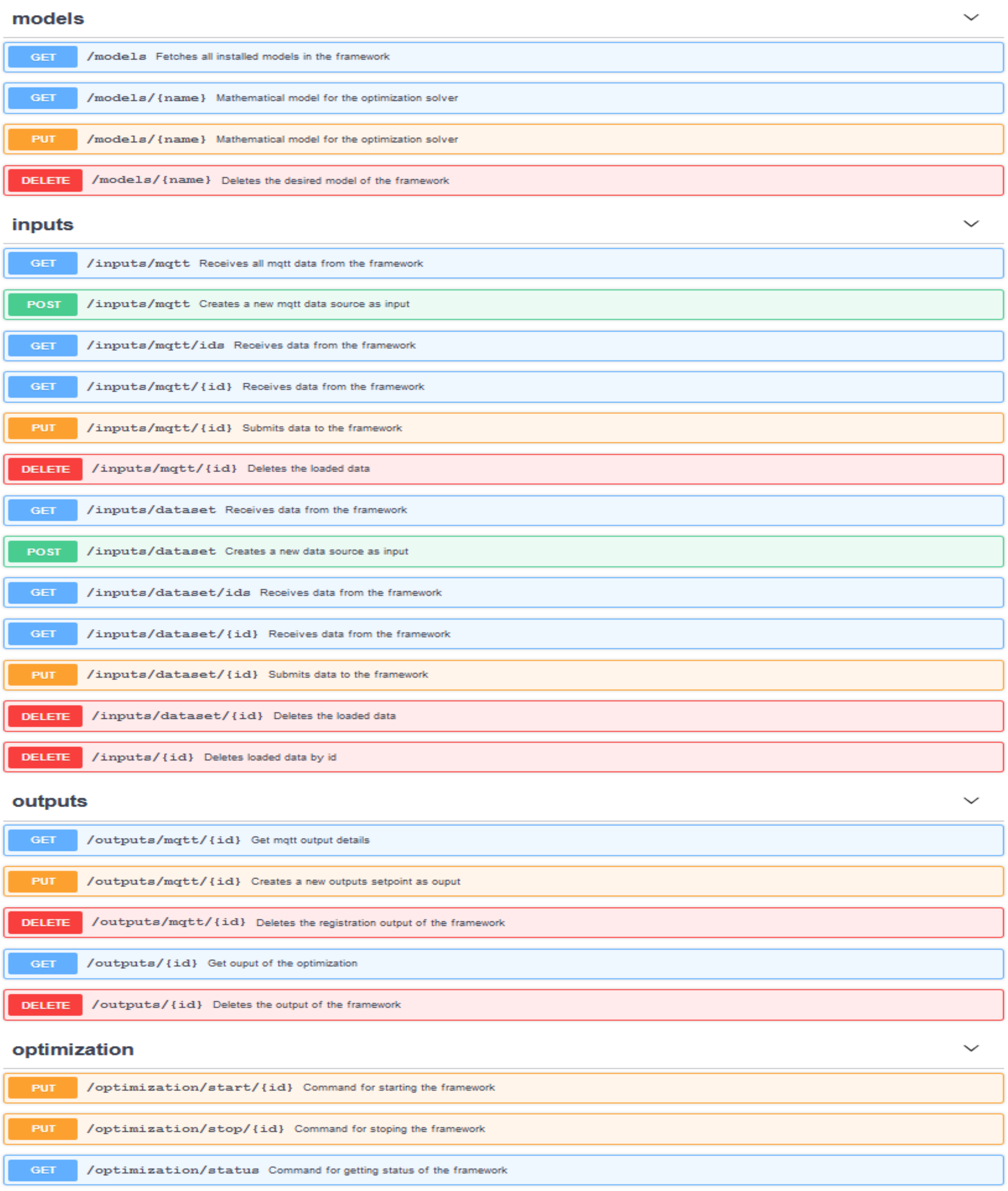
The PROFEV prototype has the software dependencies described in Table 11.

Table 11. PROFEV software dependencies.

Dependency	License	Role
Docker and docker-compose for Raspbian	Apache License 2.0	Docker is used to facilitate the PROFEV installation.

5 PROFEV - API Reference

PROFEV has been further developed in terms of functionalities and usability. Essential changes were made after D4.6 [S4G-D4.6] to increase its robustness and to allow the integration with other programs and services such as the residential Graphical User Interface (GUI) and the DSF-SE. Figure 5 shows the current PROFEV API.



models

GET /models Fetches all installed models in the framework
GET /models/{name} Mathematical model for the optimization solver
PUT /models/{name} Mathematical model for the optimization solver
DELETE /models/{name} Deletes the desired model of the framework

inputs

GET /inputs/mqtt Receives all mqtt data from the framework
POST /inputs/mqtt Creates a new mqtt data source as input
GET /inputs/mqtt/ids Receives data from the framework
GET /inputs/mqtt/{id} Receives data from the framework
PUT /inputs/mqtt/{id} Submits data to the framework
DELETE /inputs/mqtt/{id} Deletes the loaded data
GET /inputs/dataset Receives data from the framework
POST /inputs/dataset Creates a new data source as input
GET /inputs/dataset/ids Receives data from the framework
GET /inputs/dataset/{id} Receives data from the framework
PUT /inputs/dataset/{id} Submits data to the framework
DELETE /inputs/dataset/{id} Deletes the loaded data
DELETE /inputs/{id} Deletes loaded data by id

outputs

GET /outputs/mqtt/{id} Get mqtt output details
PUT /outputs/mqtt/{id} Creates a new outputs setpoint as ouput
DELETE /outputs/mqtt/{id} Deletes the registration output of the framework
GET /outputs/{id} Get ouput of the optimization
DELETE /outputs/{id} Deletes the output of the framework

optimization

PUT /optimization/start/{id} Command for starting the framework
PUT /optimization/stop/{id} Command for stoping the framework
GET /optimization/status Command for getting status of the framework

Figure 5. PROFEV API.

6 Conclusions

This document presents all the necessary information regarding the D4.7 - "Final Cooperative EV charging station control algorithms" prototype (PROFEV), developed by the S4G project. Optimization models used in the stochastic dynamic programming platform were presented. No further updates of this deliverable are expected. However, minor modifications and improvements might still need to be implemented/development during phase 3 integration and deployment activities.

Acronyms

Acronym	Explanation
AMD	Advanced Micro Devices
API	Application Program Interface
ARM	
BM	Behaviour Model
Bonmin	Basic Open-source Nonlinear Mixed INteger
CBC	Coin-or Branch and Cut
DP	Dynamic Programming
DPC	Dynamic Programming Controller
DSF	Decision Support Framework
DSF-SE	Decision Support Framework Simulation Engine
ESS	Energy Storage System
EV	Electric Vehicle
GESSCon	Grid-side Energy Storage System Control
GLPK	GNU Linear Programming Kit
GPL	General Public License
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
Ipopt	Interior Point OPTimizer
JSON	JavaScript Object Notation
MIP	Mix Integer Problems
MPC	Model Predictive Control
MQTT	Message Queuing Telemetry Transport
OMR	Optimization Models Repository
PROFESS	Professional Realtime Optimization Framework for Energy Storage Systems
PROFEV	Professional Realtime Optimization Framework for Electric Vehicles
PV	Photovoltaics
QoS	Quality of Service
S4G	S4G
SB	South-bound
SDP	Stochastic Dynamic Programming

Acronym	Explanation
SenML	Sensor Measurement Lists
SMX	Smart Meter eXtension
SoC	State of Charge
SOFW	
URL	Uniform Resource Locator
USM	Unbundled Smart Meter
VAC	Virtual Capacity

List of figures

List of tables

References

[i] Seborg et al., "Process Dynamics and Control: Model Predictive Control," pp. 414–438, 2011.
[ii] Optimization Framework, API description, https://docs.linksmart.eu/display/LOF/API+Description, accessed 23 May 2019.
[iii] Pyomo, python-based, open-source optimization modeling language, http://www.pyomo.org/, accessed 23 May 2019.
[iv] GNU Linear Programming Kit, https://www.gnu.org/software/glpk/, accessed 13 May 2019.
[v] Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT, https://www.coin-or.org/Ipopt/documentation/, accessed 13 May 2019.
[vi] Basic Open-source Nonlinear Mixed INteger programming, https://projects.coin-or.org/Bonmin, accessed 13 May 2019.
[vii] Computational Optimization Infrastructure for Operations Research, COIN-OR Branch-and-Cut solver, https://github.com/coin-or/Cbc, accessed 13 May 2019.
[viii] Gurobi, The fastest mathematical programming solver, http://www.gurobi.com/, accessed 13 May 2019.
[ix] Docker Hub, Build and ship any application anywhere, https://hub.docker.com/, accessed 23 May 2019.
[x] Optimization Framework, Installation tutorial, https://docs.linksmart.eu/display/LOF/Installation, accessed 13 May 2019.
[xvii] Kohonen, T. (2001). Self-Organizing Maps. Third, Extended Edition. Springer Series in Information Sciences vol. 30, Berlin, Germany: Springer-Verlag, ISBN 978-3-540-67921-9