Les Cahiers du GERAD ISSN: 0711–2440

Pleiad: An open-source modeling package for exploring residential flexibility in the smart grid

H. Souchard de Lavoreille,
J. A. Gómez-Herrera, M. F. Anjos

G–2020–61

November 2020

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2020
– Bibliothèque et Archives Canada, 2020

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2020
– Library and Archives Canada, 2020

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Pleiad: An open-source modeling package for exploring residential flexibility in the smart grid

Hugues Souchard de Lavoreille [a]

Juan A. Gómez-Herrera [b]

Miguel F. Anjos [c,d]

[a] *Department of Mathematics and Systems, MINES ParisTech, PSL University, 75006 Paris, France*

[b] *ExPretio Technologies, Montréal (Québec), Canada, H2T 2N8*

[c] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[d] *School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom, EH9 3FD*

hugues.souchard_de_lavoreille@mines-paristech.fr
juan.gomez@polymtl.ca
anjos@stanfordalumni.org

November 2020
Les Cahiers du GERAD
G–2020–61

Abstract: Demand response (DR) has been increasingly growing in significance among the solutions to tackle climate change, along with the development of intermittent renewable energy sources in the smart grid. Many models based on mathematical optimization were developed to address the challenge of making residential customers provide flexibility services to the grid. However, comparing and applying those models is not always straightforward because of particular data handling or specific assumptions. In this work, we take advantage of the common aspects of DR models to build a metamodel, and hence an open source Python library that aims to unify the concepts and the data streaming in and out of the underlying mathematical optimization models. We demonstrate the effectiveness of the metamodel and of the Python library by using it to implement a task scheduler and to optimize the energy consumption for two dwellings.

Keywords: Residential flexibility, optimization, open source, smart grid

Résumé : La réponse à la demande est aujourd'hui considérée comme un levier majeur parmi les solutions possibles pour faire face au changement climatique, si elle est combinée avec l'intégration de ressources renouvelables intermittentes dans les nouveaux réseaux électriques intelligents. De nombreux modèles d'optimisation mathématique ont été développés pour répondre au défi consistant à permettre aux particuliers de proposer des services de flexibilité au réseau. Cependant, la comparaison et l'utilisation de ces modèles n'est pas toujours aisée car ils se restreignent en général à des cas particuliers ou gèrent leurs données de façon spécifique. Dans ce document, nous proposons de combiner les points communs de différents modèles de gestion de la flexibilité résidentielle pour proposer un méta-modèle et une bibliothèque Python unifiant les concepts et les flux de données associés à ces modèles. Nous montrons l'utilité de ce méta-modèle et l'efficacité de la bibliothèque Python en les utilisant pour optimiser la consommation électrique de deux foyers.

Mots clés : Flexibilité résidentielle, optimisation, logiciel libre, réseaux électriques intelligents

Acknowledgments: This work was supported by the NSERC Energy Storage Technologies Network (NESTNet).

1 Introduction

The development of new renewable energy sources and the trend to electrify energy uses put energy systems under strong pressure. On the one hand, power systems have to operate with a growing share of intermittent sources, and on the other hand, they need to accommodate new demanding electricity uses. This issue can be addressed by increasing the flexibility of power systems: according to the European Network of Transmission System Operators for Electricity (ENTSO-E) [1], flexibility is one of the five big innovation clusters which need to be addressed in the coming years. Innovations can be made either by integrating new storage systems (batteries, hydrogen storage, etc.) or by encouraging consumers to change their electricity use patterns. The latter resource is often called demand response (DR).

Residential consumption is a strong lever of DR. A review of residential DR optimization techniques was carried out in [2]. They often rely on Home Energy Management Systems (HEMSs) intended to help consumers to modify their consumption in order to minimize an objective such as their energy cost or their discomfort. A review of HEMSs was done in [3].

In practice, several tools have been developed to analyze and optimize residential energy consumption, and a recent review of these was carried out in [4]. However, to the best of our knowledge, they mostly cover specific aspects of home energy management and no existing platforms enable the study of residential flexibility by considering at the same time several appliance models, user aggregation or user discomfort in a customizable framework. In this paper, we propose a flexible and open source Python package dedicated to the optimization of residential flexibility and able to tackle many aspects of this question.

This paper is organized as follows. In Section 2, we extract common features and characteristics of residential DR models, which we use in Section 3 to introduce the Pleiad framework, designed to provide modeling tools encompassing several models. In Section 4 we explain how Pleiad can effectively be used by building an optimization model derived from a previous work [5]. Finally in Section 5, we show the capabilities and the results provided by the Pleiad library on a case study built using the model defined in the previous section.

2 The Pleiad metamodel

In this section, we derive common characteristics of several residential DR models to introduce the Pleiad metamodel, which is an abstract model designed to encompass several models. Metamodelling aims at building a common ground for several models, which simplifies their comparison and enables interchangeability whenever possible.

2.1 Optimization problem

All the methods and algorithms developed for DR have in common the fact that they are actually instances of mathematical optimization problems: whatever the model and the optimization methods used, they are composed of variables, constraints and one or several objective functions which should be minimized. We choose to only deal with discrete-time problems and we work with a time step Δ_t. The involved variables will for example be the energy consumed between t and $t + \Delta_t$.

2.2 Network structure and flows

DR problems also have in common an inherent network structure which is inherited from the organization of the power grid as an energy transit network to which appliances, power sources and storage capacities are connected. One may generalize and call these three entities *connectables*, as they can be connected to the network. One may then group connectables together when they have the same owner; we refer to such groupings as *nodes*. For instance, a dwelling containing all its appliances as well as

the photovoltaic (PV) panels placed on its roof is a node. Finally, several nodes may exchange power according to predefined rules. For instance, a dwelling (node 1) may buy power from an electricity provider (node 2) according to some contract. We call such connections *exchange nodes*. An example of such a network of connectables, nodes and exchange nodes is shown in Figure 1.

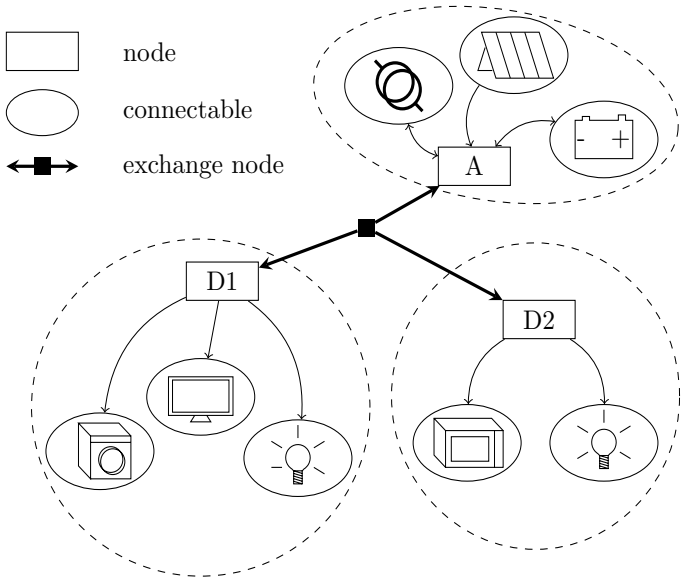

Figure 1: Network example with 8 connectables; 3 nodes (1 aggregator [A], 2 dwellings [D1] [D2]), and 1 exchange node.

To each connectable c and to each time step t, we associate 3 variables that we call *metavariables* because they are common to all objects in the metamodel:

- The energy E_t^c consumed by connectable c between t and $t + \Delta_t$. It may be positive (if power is actually being used) or negative (if it provides power, as is the case for a discharging battery or PV panels);
- The cost C_t^c induced by the use of connectable c between t and $t + \Delta_t$. This cost may account for fuel cost, for the amortization cost of the connectable, etc. It may simply be zero.
- The utility cost U_t^c induced by the use of connectable c between t and $t + \Delta_t$. This cost is a way to account for user preferences, and measures the user's discomfort. For example, if the use of a dryer is particularly unpleasant at night, the associated utility cost will be high.

Within each node, logical conservation and summation rules apply. We call them *metaconstraints*:

- The sum of the signed consumed energies of all connectables in a node (including energy flows through exchange nodes) must be identically zero at all times;
- At any given time, the instant cost for a node is the sum of the costs of all its connectables (including the signed cost of power through exchange nodes);
- At any given time, the instant utility cost (discomfort) for a node is the sum of the utility costs of all its connectables.

2.3 Inside connectables: the origin of flexibility

Each connectable has its own behavior defined by internal variables and constraints. For instance, a battery has a current load state Q_1 at time t_1 as well as a current load state Q_2 at time t_2. Between these times, it may exchange a signed energy flow E_{12}^c with other connectables by respecting

$$E_{12}^c = Q_2 - Q_1,$$
$$|E_{12}^c| \leq p_{\max} \Delta_t,$$

where $p_{\max}$ is a parameter establishing the maximum instant power that can flow in and out of the battery. Such variables and constraints depend on the particular type of object or load that is being considered. Two appliances having the same function but built by different manufacturers may function differently and have different internal constraints.

In general, the solver optimizes an objective function within the admissible space defined by the variables and all the constraints of the connectables. The latter are constrained by their dynamics but could operate differently: this is where flexibility comes from. For instance, a dishwasher does not necessarily have to start right after the end of lunch but it must have run before dinner. The maximum ending time is a hard constraint which must be ensured, but other constraints (such as the starting time) may be relaxed, possibly along with the introduction of a utility cost accounting for discomfort.

3 The Pleiad environment

We now introduce a framework to unify the concepts we identified in the previous section. We call this environment *Pleiad (PLanner for Electrical and Intelligent Appliances in Dwellings)*.

3.1 An open-source package

Pleiad is being developed in Python. It is released under an open source license and made freely available on a gitlab repository. The package can therefore be downloaded, used, and anyone can improve it or propose new features. We aim at improving reproducibility in the field of DR research by providing a common ground for modeling residential DR, and by encouraging users and researchers to publish their code using the same open source license. Other recent developments have also been released under an open-source license, such as PANDAPOWER [6] for the analysis and optimization of power systems. This license allowed many users to write code for the project, with 56 contributors registered on github and more than 4,500 commits during the four years after the first release of the library in 2016.

The up-to-date documentation of Pleiad is available on the gitlab repository [7].

3.2 A modeling package

Pleiad is a modeling tool: it helps to define a model-independent structure and then to instantiate a model for this structure.

3.2.1 Model-independent network structure

Before modeling, the Pleiad package requires the definition of a network composed of connectables, grouped in nodes, which are in turn connected via exchange nodes as explained in the previous section. These objects and the network structure define the physical properties and constraints of the network.

3.2.2 Model instantiation

After the network structure is defined, the user must choose a time step Δ_t, a planning horizon, and a model to represent the internal dynamics, constraints and flexibility potentials of each connectable using variables and equations. These mathematical objects are associated to connectables, and removing one of them automatically drops the related variables and constraints.

3.3 Typing in Pleiad

Pleiad creates mathematical optimization models converting real-world connectables into variables and constraints. For this purpose, variables and quantities are typed and can be combined in a very natural way into typed mathematical constraints.

3.3.1 Typed quantities

In Pleiad, physical dimensions become *intrinsic* properties of quantities. Any physical quantity like energy can be given as input with its unit. For example,

```
e = Energy(kWh = 5),                    (1)
dt = timedelta(hours = 5).              (2)
```

All operations preserve types and are performed transparently, so that

```
p = e / dt
```

is a power-typed quantity from which the user can obtain the value in any possible unit: `p.W` returns 1000, which is the value (in Watts) of the associated power.

3.3.2 Typed variables and constraints

Strict quantity typing also enables to type variables and to check the physical dimensionality of equations. If the engine detects that power is compared to energy, a warning will immediately be triggered. Finally, variables and constraints can be written in a natural way, by using symbols like `+`,`-`,`==`. For instance, if `p` is a "power"-typed variable, and if `dt` and `e` are given by (1) and (2), the following constraint will be valid

```
constraint = (dt * p == e)
```

and it will be the exact translation of the following numerical equality constraint (in SI units):

$$(5 \times 3600) \times p = (5 \times 1000 \times 3600).$$

After the optimization phase, `p` will be a power object equivalent to 1 kW.

3.4 Data visualization

As we saw it in Section 2, all models implemented in Pleiad must define 3 metavariables for each connectable at each time step: energy flow, cost and utility cost. This makes it possible to define, for example, the total cost for a node or the total energy drawn from all its appliances during some time period.

Those three metavariables are automatically computed and made available in a PANDAS results data frame as illustrated in Table 1.

Table 1: Structure of the results data frame.

Time	Dishwasher		
	energy_drawn	cost	utility_cost
2017-01-01T12:00	1kWh	0$	0.1$
2017-01-01T13:00	0.5kWh	0$	0$
2017-01-01T14:00	1kWh	0$	0$

Pleiad also provides a data visualization interface able to generate six types of graphs: power versus time, cost versus time, utility cost versus time, battery state of charge versus time, power retail price versus time, schedule chart.

4 Pleiad instances

In this section, we illustrate how Pleiad can be instantiated and used to model actual residential appliance scheduling problems (RASPs). We developed a scheduler named "base scheduler" that automatically transforms a Pleiad network into a mixed-integer linear optimization problem (MILP). We also propose a library of realistic connectables to build Pleiad networks. We emphasize that the model and the library of connectables we propose here are only one way to use the Pleiad framework, and we encourage using the latter with other models and other real-world data.

4.1 A model: the MILP scheduler

In this paper, we formulate a Pleiad network of connectables as a set of continuous or integer variables and linear constraints. Such a formulation can be solved efficiently to compute appliance operating patterns, and is a good compromise between model complexity and computation time because there exist several high-quality optimization solvers able to solve mixed-integer linear optimization problems.

4.1.1 From connectables to MILPs: modeling rules

In the MILP scheduler, we automatically transform a network of connectables into a MILP based on a subdivision of home appliances into three categories. The classification of loads into several types is a modeling choice and even though it has been done previously in the literature, there appears to be no general consensus on this question. For instance, the authors of [3] classify loads into six types whereas the authors of [2] use a noticeably different classification of loads into three types. In our model, we chose to use the three categories (regular, activity-based, flexible) presented in [5] as they can encompass most of the load types of the aforementioned works.

In addition to appliances, we model batteries as shown in Figure 2, with a self-discharge rate δ_{self}, charge and discharge efficiencies (η_c, η_d), an energy capacity $E_{\max}$, and a parameter $p_{\max}$ defined in paragraph 2.3. Finally, we treat power sources either as uncontrollable loads consuming a negative power or as unlimited power sources providing electricity at a given cost.

In the MILP scheduler, we also allow to process several nodes at a time. Even if each node has its own objective (minimize a combination of its cost and its discomfort), we choose to reduce the multi-objective problem to a single-objective one by summing all objectives with weights chosen by the user. For example, if all the dwellings in a building gather to reduce together their power cost, the overall power cost will be optimized and can be attributed to each user *ex-post*. This way to group several nodes together enables to prevent situations when all of them individually shift their consumption and simply move their peak together to another time period.

4.1.2 From the MILP to the solution: optimization solvers

Once the connectables and the network have been transformed into an MILP, the latter is solved after being translated into the formalism of commercial optimization solvers. Interfaces were developed with IBM ILOG CPLEX Optimization Studio and Gurobi, which are easy to use in Python.

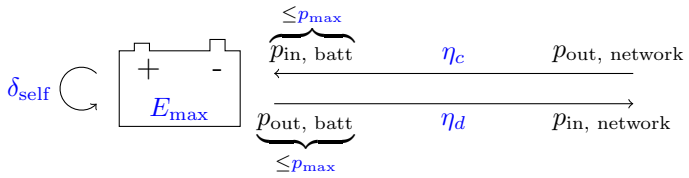

Figure 2: Storage model. Parameters are written in blue, variables are in black. The four variables p are powers flowing in and out the battery.

4.2 A library of connectables built on data

In this paragraph, we now study how networks can be built using the Pleiad appliance library, which provides realistic connectables. Building such a library requires collecting data so as to reflect the behavior of the objects included in the model and to make it as realistic as possible. This step is done in all research works, but using different methodologies. In [8], a survey was conducted to collect data about appliance use and characteristics. The authors of [2] cite three data sources providing statistics about appliance ownership or loads. Similarly, solar PV generation data can be found in the PVOutput database [9]. Individual load measurements can also be found in the literature in [10] and [11]. Both papers developed experimental protocols to record the consumption profile of several home appliances.

4.2.1 Appliances

We used load data from [10] and [11] and noticed that most of the studied appliances have (almost) piecewise constant power consumptions and that their operation can be divided in phases. We chose for simplicity to keep the classification of appliances of the previous paragraph and, for each appliance, we arbitrarily determined the most suitable type before adjusting its parameters according to the recorded power consumption and duration of each phase.

4.2.2 PV generation

We use numerical weather prediction (NWP) models to determine the PV power output. Pleiad was interfaced with PVLIB [12] which natively features NWP data fetching using the Global Forecast System (GFS). We linearly interpolate these data minute-wise in the clear-sky index domain, which is proven to be more accurate for resampling [13]. Once irradiance has been obtained, PVLIB extracts the characteristics of the chosen PV module from the California Energy Commission database [14] and computes its PV output.

4.2.3 Power tariffs

Several tariffs from grid operators were included in the library, such as the power price of Ontario's Independent Electricity System Operator or the French regulated electricity price. Those prices vary regularly and should be updated accordingly in the library.

5 Case study

To illustrate the effectiveness of Pleiad, we run a case study using the MILP scheduler presented above on a network of two dwellings and an aggregator as shown in Figure 1. The aggregator possesses a battery, PV modules and is connected to the main power grid. Each of the dwellings is equipped with a water heater, an electric space heater, and a dishwasher. One of them has in addition an oven, a stove, and a fridge, while the other has a TV and a garage heater.

The optimization is performed on three consecutive days (12^{th}, 13^{th} and 14^{th} of November 2020) and the PV production is automatically estimated using GFS weather forecasts.

We consider two pricing scenarios and study the behavior of the total power demand (power bought by the aggregator from the grid). In scenario (a), the price of grid power is constant and therefore, the power cost cannot be optimized with demand response. The objective of the optimization problem is to minimize user discomfort in both dwellings. In scenario (b), the grid adopts a Time and Level Of Use (TLOU) pricing scheme [15]. The lower level is taken as Ontario's Independent Electricity System Operator (IESO) tariff in Winter 2020. The higher level is a simple shift of the lower level, as shown in Figure 3. The consumption of power from the grid and from the PV modules in both scenarios is shown in Figure 4.

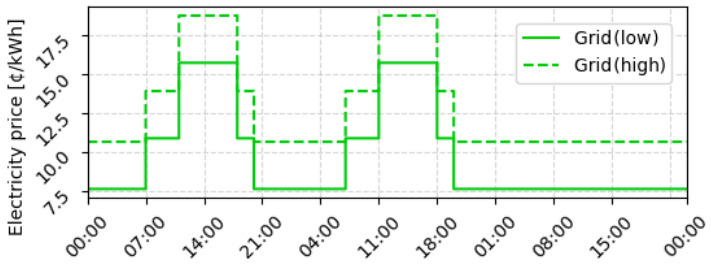

Figure 3: Level of the TLOU grid electricity price in scenario (b) as a function of time, from Thursday 12th of Nov. to Saturday 14th of Nov. 2020.

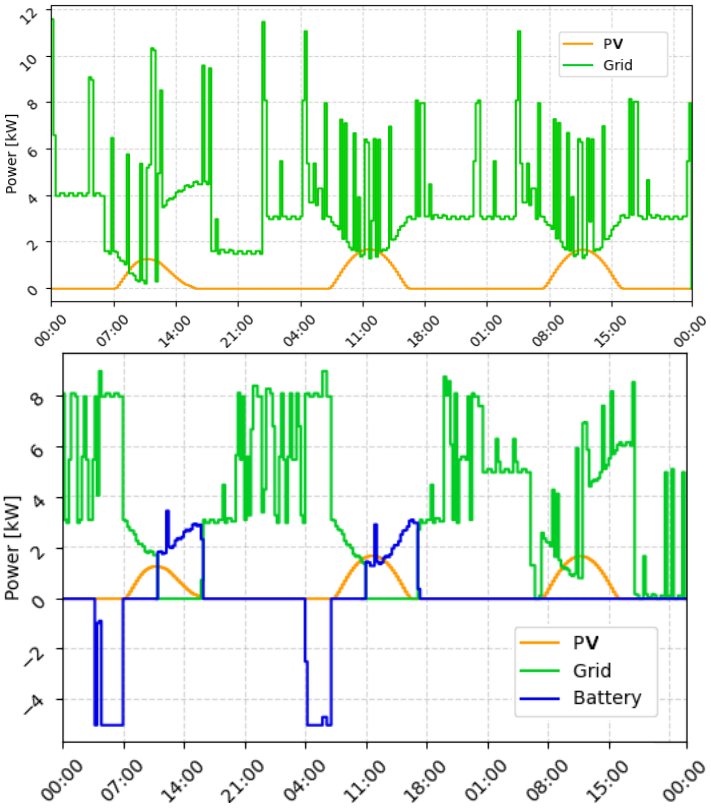

Figure 4: Signed instant power consumption from the grid, solar panels and from the battery (negative values mean that the battery is charging) as a function of time, in scenario (a) [top] and scenario (b) [bottom], between Thursday, Nov. 12th and Saturday, Nov. 14th 2020.

The use of the MILP scheduler optimizing both cost and user discomfort, together with a TLOU pricing structure, makes it possible to efficiently shift power consumption outside of the peak hours when the battery and the PV modules provide enough power for both dwellings. Moreover, the power demand does not exceed a certain amount (9 kW in this case) because of the TLOU price structure.

Using Pleiad, it is straightforward to change appliances, tune the objective function or fetch real-world data. Here, solar production forecast data are automatically processed by PVLIB using up-to-date weather prediction data obtained online. In Figure 4, one may observe that the PV production is lower on the first day (Nov. 12th) as this day is forecast to be more cloudy than both following days. Figures 3 and 4 were automatically generated using the Pleiad framework.

6 Conclusion

In this paper, we presented a metamodel for residential DR models aiming at providing a common ground for several models. This metamodel comes with a Python modeling framework called Pleiad, which is released as an open source project on a git repository. Pleiad's capabilities were illustrated through the implementation of a mixed-integer linear optimization scheduler based on previous research and with a case study carried out with realistic appliances. In this setting, Pleiad was able to connect to real-world streaming PV data, and subsequently to reduce the power consumption of two dwellings during peak hours.

We propose Pleiad as a common framework for future work on residential DR and as a means to make fair comparisons between different models and enhance reproducibility of research. We encourage the community to contribute to Pleiad by using it, by contributing their models to it, and by adding new features.

In the future, Pleiad could be used to efficiently analyze and automatically optimize the consumption of smart homes using streaming consumption and production data. Future work could also include the study of flexibility resources shared by several users, as well as the associated economic models.

References

[1] ENTSO-E. Power in Transition, R&I Roadmap 2017-2026, page 39. ENTSO-E, Avenue de Cortenbergh 100, 1000 Brussels, Belgium, 2017.

[2] Amit Shewale, Anil Mokhade, Nitesh Funde, and Neeraj Dhanraj Bokde. An overview of demand response in smart grid and optimization techniques for efficient residential appliance scheduling problem. Energies, 13(16), 2020. doi: 10.3390/en13164266.

[3] Marc Beaudin and Hamidreza Zareipour. Home energy management systems: A review of modelling and complexity. Renewable and Sustainable Energy Reviews, 45:318–335, 2015. doi: https://doi.org/10.1016/j.rser.2015.01.046.

[4] Khizir Mahmud, Uzma Amin, M.J. Hossain, and Jayashri Ravishankar. Computational tools for design, analysis, and management of residential energy systems. Applied Energy, 221:535–556, 2018. doi: https://doi.org/10.1016/j.apenergy.2018.03.111.

[5] Miguel F. Anjos, Luce Brotcorne, Martine Labbé, and Maria I. Restrepo. Load Scheduling for Residential Demand Response on Smart Grids. unpublished, Dec 2017.

[6] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Transactions on Power Systems, 33(6):6510–6521, Nov 2018. doi: 10.1109/TPWRS.2018.2829021.

[7] Pleiad documentation. URL https://gitlab.com/h_sdl/pleiad/-/blob/master/docs/documentation.pdf.

[8] M. M. Hossain, K. R. Zafreen, A. Rahman, M. A. Zamee, and T. Aziz. An effective algorithm for demand side management in smart grid for residential load. In 2017 4th International Conference on Advances in Electrical Engineering (ICAEE), pages 336–340, Sep. 2017. doi: 10.1109/ICAEE.2017.8255377.

[9] Pvoutput. URL https://pvoutput.org/.

[10] M. Pipattanasomporn, M. Kuzlu, S. Rahman, and Y. Teklu. Load profiles of selected major household appliances and their demand response opportunities. IEEE Transactions on Smart Grid, 5(2):742–750, March 2014. doi: 10.1109/TSG.2013.2268664.

[11] Fatih Issi and Orhan Kaplan. The determination of load profiles and power consumptions of home appliances. Energies, 11(3), 2018. doi: 10.3390/en11030607.

[12] William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. pvlib python: a python package for modeling solar energy systems. Journal of Open Source Software, 3(29):884, 9 2018. doi: 10.21105/joss.00884.

[13] Philippe Blanc and Lucien Wald. On the intraday resampling of time-integrated values of solar radiation. In 10th EMS Annual Meeting (European Meteorological Society), Zurich, Switzerland, September 2010.

[14] California energy commission - pv modules and inverters database. URL https://www.energy.ca.gov/programs-and-topics/topics/renewable-energy/solar-equipment-lists.

[15] Juan A. Gomez-Herrera and Miguel F. Anjos. Optimization-based estimation of power capacity profiles for activity-based residential loads. International Journal of Electrical Power & Energy Systems, 104: 664–672, 2019. doi: https://doi.org/10.1016/j.ijepes.2018.07.023.