International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Algorithms to reduce Electrical Utility Costs using Machine Learning

Nambivel Raj and Pramod Patil

Engineering, DYPIET, Pune, India.

Received 10 Nov 2020, Accepted 10 Dec 2020, Available online 01 Feb 2021, **Special Issue-8 (Feb 2021)**

Abstract

In this paper we propose an energy Management framework based on Machine Learning algorithms and IOT to provide energy savings for buildings. We specifically consider buildings with energy storage systems attached which then allow multiple use cases for energy savings. We start with one particular application of peak shaving which helps by providing savings in energy cost by reducing peak demand charges. The algorithms help forecast and calculate key variables which are then leveraged by the IOT platform to translate the insight into actions for savings. The objective is to have an end to end framework to help save by reducing monthly consumption cost. We propose end to end framework and mechanism and also evaluate available machine learning algorithms that can be used. This framework is a combination of smart edge devices and a cloud data platform used for remote monitoring by customers which provides real time data and information about their consumption patterns and trends. Also the framework created is such that it can be extended for further applications like Frequency Regulation Service and Virtual Power Plant without modifications to architecture.

Keywords: *Battery Energy Storage Systems, peak shaving, smart edge device, IOT Gateway, data platform, Machine Learning, IOT, cloud platform, energy savings, Virtual Power Plant*

Introduction

Today almost all socio-economic activities revolve in and around built structures. This holds more true for developed economies of the modern world. However, the upkeep and maintenance of buildings requires high quality electrical energy and accounts for a significant percentage of the total global energy consumption. With pressing issues such as anthropogenic environmental and climate change taking center stage, a conscious shift towards reducing fossil fuel consumption, greenhouse emissions and, "going green" are the need of the hour. Keeping in mind the ever increasing global energy requirements, the use of renewable energy sources are expected to become major contributors to the future electricity system. However, the large scale use of such renewable energy sources are prohibitively expensive. Therefore a reward-based strategy wherein consumers are rewarded or penalized depending on their energy consumption behaviour becomes an efficient short-term consumer-oriented energy saving alternative. Practically, such a strategy can be expected to directly translate into tangible capital savings further reinforcing the energy saving behaviour making the use of Energy Storage Systems (ESSs) a viable and realizable option. We focus on the use of Battery Energy Storage Systems (BESSs), which capture energy and store it for use at a later time or date, to achieve energy savings. This stored energy can also be coupled with renewable energy sources in an attempt to balance energy production and consumption. Further, the BESS can also act as an inverter backup in the event of power failure. In order to demonstrate the applicability of the reward-based paradigm using a BESS, we consider a concrete example:

Typically, the electrical consumption of a large commercial building in the United States is billed partly for the energy used (kWh), and partly for the maximum power ("demand") drawn from the grid (kW) within any specific month. Billing costs can be reduced if a BESS is charged and discharged to reduce the monthly maximum grid demand. This process is very bene□cial, as it can reduce electricity costs, for a typical large commercial building, by an appreciable amount. With improving manufacturing processes expected to reduce battery costs, the return on initial investment is further expected to improve. However, in the meantime, maximizing the return requires the grid demand to be controlled to the lowest monthly limit possible with the available storage capacity. Such a maximization can be achieved through a charging and discharging strategy using a BESS. In Fig 1. we see a BESS being charged during the off peak periods of energy from the grid and discharged during peak hours to provide energy thereby "shaving" the demand peaks to reduce the peak demand charges for the month.

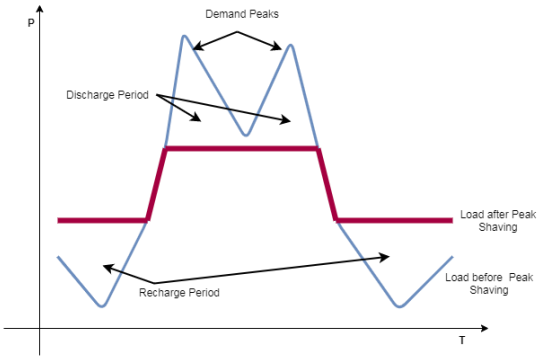

Fig.1. Concept of Peak Shaving

In Fig. 2. we depict how the power from the utility provider powers the building load as well as the BESS. The BESS is rigged to provide power to the building in case of a power failure. The graph on the top right shows typical incidents of peak demand during the month resulting in a large demand surge as compared to the average consumption of the building. This surge could have occurred in a one-off situation or for an extremely short duration of a few seconds. Despite this, the demand charges applied would be as per the peak contributing to an extremely large energy bill. As the penalty for creating the peak demand is steep. Thus the use of Peak Shaving could find significant use in the daily lives of average citizens.

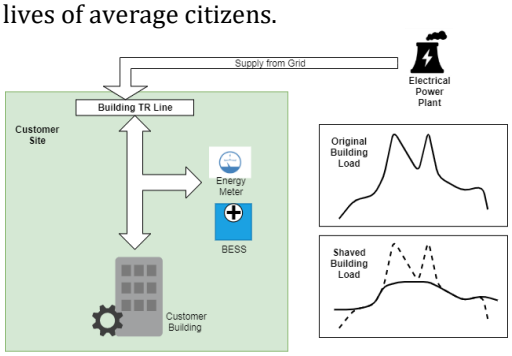

Fig. 2. Typical BESS system used for Peak Shaving

In this paper we describe a system developed to leverage the capabilities of the Internet of Things (IoT) and Cloud Computing technologies to achieve remote, real time realization of a Peak Shaving BESS. Additionally, we explore the applicability of Machine Learning strategies for understanding consumer behaviour to tailor Peak Shaving strategies for optimized energy savings. Of the many strategies we discuss two strategies which demonstrate the potential for incorporation in our existing system.

Problem Definition And Scope

A. Existing System

The buildings we consider have an inverter in the form of the BESS. The building and BESS is powered by power from the utility provider. We read key parameters from the building and the BESS and transfer them to the cloud to allow remote monitoring and visualization of the data and trends. A schematic of the end to end system architecture is shown in Fig. 3. The following sections describe key components of the existing system.

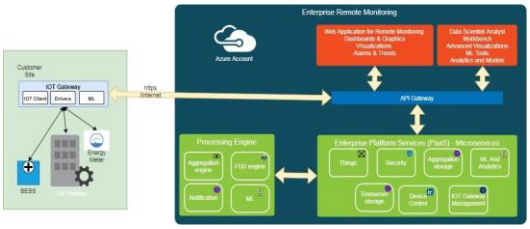

Fig 3. End to End System Architecture

a. *Automation Sub Systems at Site*

The automation system consisting of the sub systems like meters, PLCs, controllers and sensors allow them to integrate and function together. The primary purpose of the automation system is to ensure that the BESS is able to provide power to the building at the time of interrupted supply and ensures charging/discharging cycles at the appropriate time. It coordinates the different systems to work together with necessary built in triggers and interlocks. The key subsystems that our cloud platform communicates with are as below:

i. Energy Meter: The power from the utility provider is via the energy meter. This helps our system monitor the parameters like demand, energy, power factor, power, etc. for the electrical utility consumed by the building.

ii. BESS: This BESS further consists of multiple subsystems like inverter, batteries, controller, sensors, etc. The BESS routes information for its subsystems together, which allows us to monitor the parameters like State of Charge, State of Health, Capacity, etc for the BESS. It is also capable of reporting different alerts and alarms that are used in remote monitoring.

iii. Sensors: Various sensors like temperature, door open/close, system on/off, etc are also added for extra parameters not directly available. These are all further connected to PLC or controllers and join the automation system in different logic/monitoring required.

b. IOT Gateway

The automation system is primarily responsible for smooth functioning of the system at the site. It does not usually extend to capabilities like cloud connectivity, or integrating other essential system components. This is where the IOT gateway fits in. It is an embedded Linux box, which has an IOT client application running on it which connects with the cloud data platform via the API gateway. This gateway provides us with a secure mechanism of connecting the building/system to the cloud platform. It also enables building functionality like custom alerts, data cleansing, etc to be built onto it as required before pushing data to the cloud acting like a smart edge device. The IOT gateway comprised of the following:

i. IOT Client: The IOT client is an implementation of the cloud IOT framework which links to the microservices in the cloud to ensure security, encrypted data transfer, data transfer, cloud to device messages, etc functionality implementation. This client is tightly integrated with the cloud data platform and ensures efficient bidirectional communication with the cloud data platform.

ii. Local System Interface: The IOT gateway interfaces with the automation system using multiple physical interfaces like ethernet, serial rs232/485, etc. These physical interfaces further have different protocols used by the automation systems that need to be implemented to enable communication with them. These protocols are implemented in the form of drivers in the IOT gateway. Using these drivers the IOT gateway is able to connect and read/write the different sensor values and push the same to the cloud. The drivers needed are:

1. Modbus: This is common for most meters or PLCs/Controllers. It can be over serial or IP and supports both read as well as write.
2. BacNet: This is the building automation standard, and well implemented at the controller level of components.

c. Cloud Data Platform and Framework

The Data platform and framework is a set of components in the cloud that work together to provide the functionality of remote monitoring of the different sites. It includes a set of complex functions that work together to provide an extensible framework to build multiple applications consuming and reporting the data collected. The hosting in our case is in the Microsoft Azure cloud and leverages Microsoft .NET framework for most components, Microsoft Azure IOT framework and different Azure cloud components for hosting, compute and storage.

i. Microservices*:* All communication with the cloud data platform is via the API exposed. The API follows the microservices architecture, where each is unique and has no overlapping function. This architecture greatly helps in extensibility and ensuring a clean implementation that is easier to maintain. We have separated the required functionality into different microservices as seen below:

ii. Telemetry: All sensor data is managed by this api. It only deals with the sensor id, timestamp and associated value. Multiple ways of posting or retrieving this data as per requirement have been built into the methods.

iii. Things: All configuration and metadata related to systems, subsystems, relationships between systems, hierarchy, etc is maintained in this API.

iv. Device Control: This API helps send cloud to device messages and receive acknowledgements and heartbeat from the IOT client on the IOT gateway.

v. IOT Gateway Management: This API helps manage all IOT gateway functions like registration, setup state, configuration, updates, etc.

vi. Security: Each API is secured via the security API and use of tokens for roles and data access control.

vii. Notification: This API is for pushing notifications related to alerts, alarms and configuration changes in the system to the users monitoring applications.

viii. Storage: For the complex functionality exposed by the cloud data platform, there are multiple storage requirements that cannot be fulfilled by a single backend storage infrastructure. So there are multiple databases in use as per the desired functionality.

1. SQL: Used to store simple metadata and transactions that are low volume, but frequently read.
2. NOSQL: Used to store telemetry data and high volume transactions.
3. Memory Cache: Used to store key data used in visualizations and dashboards to report real time parameters in the monitoring application.

ix. Compute Engines: The cloud data platform has multiple compute requirements for different functions that are logically separated and built accordingly.

1. Aggregation: telemetry data is aggregated in different reporting intervals, with calculations as per analytical requirements
2. FDD: fault detection and diagnostics are run on the different subsystems to identify complex faults based on certain rules and conditions.
3. Notification: Notifications via email and text are generated here for different workflow scenarios

x. API Gateway: The microservices work well to separate the functions into different API for a clean architecture and easy maintenance. But, from an application perspective to use multiple API, certain functionalities in the UI can mean multiple calls to the backend slowing performance and increasing complexity. This is where we introduce the API gateway which acts as an intermediary between the application and the platform microservices, and does the necessary data aggregation from multiple microservices and provides to the UI applications via a single call. Since the multiple calls are all on the server side, with caching and other strategies also implemented, it boosts performance and simplifies implementation.

xi. Enterprise Web Application for Remote Monitoring: This is the HTML5 based web application used by the users for remote monitoring of the different sites and systems. The user can monitor any site around the globe from anywhere in the globe via an internet connection and a browser. The application communicates to the data platform via the API gateway implementing roles and user rights for data access as defined in the security API. The application provides real time data, visualizations, trending and alarm monitoring over the web making it a great tool.It helps

reduce the workforce required to manage multiple sites, improves response time and provides proactive actions as well greatly increasing efficiency and providing customer satisfaction.

Proposed System Architecture

The cloud data platform has access to all the sensors and systems and subsystems connected. These systems are bidirectional and allow input of values to work with their control functions as well. Using our cloud infrastructure and introducing Machine learning blocks we can extend the use of the system beyond remote monitoring to active energy savings. We can implement machine learning over the data collected in the past to build models and run the algorithms. These algorithms can then drive key decisions and actions via the framework to generate the necessary savings. The changes to be added to the platform in the form of extensions are as below:

d. Extension to IOT Gateway: ML Client

ML capability is added to the IOT gateway. It helps to use real time values in the ML models to predict/verify conditions to trigger actions that will drive the savings.

e. Inclusion of Compute Engines for ML

These are the ML engines in the cloud to help the data scientists create their models on the past accumulated data

f. Analyst Workbench with advanced visualizations- For Data Scientists

This is the HTML5 based application used by the scientists which provides access to all the sensor and system data with advanced visualizations and ML blocks. This helps them build and test the models for the different algorithms to be used. Based on these models and algorithms the different savings mechanisms are implemented for sites, and monitored in the application for further improvement and extension. At present, we have identified two ML algorithms namely which can be easily incorporated into our existing framework and briefly discuss them following sections.

Mathematical Model

Peak shaving is usually approached by calculating, at the beginning of each day, a forecast of building demand for every measurement interval, then calculating a single daily limit from this forecast, with the assumption that there is no need to change the limit during the day. However such a calculation is quite erroneous due to the monthly billing cycle. Thus a strategy wherein monthly peak demand can be adaptively predicted based on prior usage data would be more useful. [10] proposes an algorithm incorporating strategies based on linear regression and an extremely simple control strategy: use the historical maximum peak demand in case it is larger than the predicted demand. Specifically, it forecasts two parameters Zα and Zβ using different control strategies described below:

- Instead of focusing on the daily maximum demand implement an adaptive control method to minimize the monthly maximum demand.
- Instead of using a single forecast, the said adaptive control method will combine multiple demand forecasts.
- A control law denoted the “Ratchet Rule”, whereby grid demand is shaved no lower than the maximum prior value observed during the month

Forecast Zα

Zα(j, k)forecasts the optimal grid demand limit:
Y1(j, k) = pb(j, k)= actual building demand Y2(j, k) = pg,lim,opt,h (j, k)= optimal grid demand limit input X11 =Tmean(j)from a weather forecast input X12 = mean demand for the week one year earlier input X13 = mean demand for the same weekday input X14 = mean demand for intervals (k−4 : k−1)
Z1α forecasts Y1(j, k)as a linear regression, using inputs X11,
X12 , X13, X14
Zα(j, k) = F([Z1a(j, k :nk)] , Es(j, k))

Forecasts used for combined forecast Zβ

Forecast Zβ forecasts the optimal grid demand limit using four forecasts, i.e. Z3j, Z4j, Z4e, Z4g. Of these, Z3j and Z4j are calculated from sub-forecasts of actual demand pb(j, k) using function FP lim, but each other forecast or sub-forecast is a least-squares linear regression of its inputs:

input X1(j) = Tmean(j) input X2(j) = Tmean(j)2 input X3(j) = Y2(j−7) plim,error 4d(j)= error in forecast Z4d(j) input X4(j) = plim,error 4d(j−7) input X5(j, k) = M5(j, k) input X6(j, k) = M5(j, k)∗Tmean (j) input X7(j, k) = M5(j, k)∗Tmean (j)2 input X8(j, k) = M5(j, k)−M5(j−7, k)
Z3a forecasts Y1 using X5(j, k3), X1, X2, X3, X4 Z4a forecasts Y1 using X5(j, k4), X1, X6, X7 where k3, k4are optimized to minimize errors
Z3j(j, k) = F([Z3a(j, k :nk)] , Es(j, k))
Z4j(j, k) = F([Z4a(j, k :nk)] , Es(j, k))
Z4dforecasts Y2using X1, X2, X3
Z4eforecasts Y2using X1, X2, X4
Z4gforecasts Y2using X3, X5, X6, X7, X8

Control Strategy α

The forecast and control limit are calculated at intervals [6 12 32 40 48 56 60 64 68 72]. The control limit Cα(j, k)kW is calculated by adding a safety margin to Zα(j, k). The safety margin is equal to the standard deviation of the errors in forecast Zα(j, k), evaluated over the rst year of data. At each time interval, the battery is charged or discharged to ensure the grid demand equals the control limit Cα(j, k).

Control Strategy β

Cβ(j, k) = Zβ(j, k) At each time interval, the battery is charged or discharged to ensure the grid demand equals the control limit Cβ(j, k1).

Ratchet Rule

Control strategies α and β were evaluated with and without an additional rule denoted the "Ratchet Rule". Under this rule, if the control system calculates a control limit that is lower than the maximum grid demand pg, month, max(j, k)for the month so far, then pg, month, max(j, k)is used instead.

Conclusion

This paper proposes an energy Management framework based on Machine Learning algorithms and IOT to provide energy savings for buildings. We start with one particular application of simple peak shaving by providing forecasts and predicting key variables in the algorithms. The end goal being reduced energy consumption and thus increased savings, while having the underlying framework to extend to further applications as well. Presently, the algorithm described above is being implemented on the end-to-end framework as a pilot. However, in place of a regression based strategy, more involved supervised/unsupervised machine learning strategies are being studied to incorporate into the framework. Implementing the system with layers of abstraction allows extensibility and empowers the same platform to be used in the form of PaaS. This PaaS can then be used to monitor multiple front end applications by being a central data platform extending use cases further and making it simpler and with faster time to market. Further it is possible to have deviations in the nominal frequency in the grid due to irregular power generation from different sources and changing loads. The proposed architecture easily allows for the leveraging the BESS to restore the balance in the difference between the supplied and desired frequency. Also, the proposed architecture could be used as a virtual power plant that aggregates different types of distributed energy sources for the purpose of enhancing energy generation, and also towards specific commercial agreements with utility providers over the cloud based power plant. We can implement the concept of the VPP using multiple BESS distributed across multiple sites.

References

[1]. S. Aman, Y. Simmhan and V. K. Prasanna, "Energy management systems: State of the art and emerging trends," in *IEEE Communications Magazine*, vol. 51, no. 1, pp. 114-119, January 2013. doi: 10.1109/MCOM.2013.6400447

[2]. Chua, K., Lim, Y. and Morris. S (2016), "Energy storage system for peak shaving", International Journal of Energy Sector Management, Vol 10 No. 1, pp. 3-18. https://doi.org/10.1108/IJESM-0.1-2015-003

[3]. R. T. de Salis, A. Clarke, Z. Wang, J. Moyne and D. M. Tilbury, "Energy storage control for peak shaving in a single building," *2014 IEEE PES General Meeting | Conference & Exposition*, National Harbor, MD, 2014, pp. 1-5. doi: 10.1109/PESGM.2014.6938948

[4]. Y. Shi, B. Xu, D. Wang and B. Zhang, "Using Battery Storage for Peak Shaving and Frequency Regulation: Joint Optimization for Superlinear Gains," in *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2882-2894, May 2018. doi: 10.1109/TPWRS.2017.2749512

[5]. G. C. Lazaroiu, V. Dumbrava, S. Leva, M. Roscia and M. Teliceanu, "Virtual power plant with energy storage optimized in an electricity market approach," *2015 International Conference on Clean Electrical Power (ICCEP)*, Taormina, 2015, pp. 333-338. doi: 10.1109/ICCEP.2015.7177644

[6]. A. Barbato and G. Carpentieri, "Model and algorithms for the real time management of residential electricity demand," *2012 IEEE International Energy Conference and Exhibition (ENERGYCON)*, Florence, 2012, pp. 701-706. doi: 10.1109/EnergyCon.2012.6348242

[7]. B. Chai, Z. Yang and J. Chen, "Optimal residential load scheduling in smart grid: A comprehensive approach," *2013 9th Asian Control Conference (ASCC)*, Istanbul, 2013, pp. 1-6. doi: 10.1109/ASCC.2013.6606320

[8]. Y. Zong, L. Mihet-Popa, D. Kullmann, A. Thavlov, O.

[9]. Gehrke and H. W. Bindner, "Model Predictive Controller for Active Demand Side Management with PV self-consumption in an intelligent building," *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Berlin, 2012, pp. 1-8. doi: 10.1109/ISGTEurope.2012.6465618

[10]. Karmiris, Georgios, and Tomas Tengnér. "Peak shaving control method for energy storage." *Corporate Research Center, Vasterås, Sweden* (2013)..

[11]. Tull de Salis, Rupert & Clarke, A. & Wang, Zheng & Moyne, J. & Tilbury, D.M.. (2014). Energy storage control for peak shaving in a single building. IEEE Power and Energy Society General Meeting. 2014. 10.1109/PESGM.2014.6938948.