An Affordable and Portable Technology for Real-Time Scheduling of Appliances in Smart Homes

A. Raman, R. Sowers and R. S. Sreenivas
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
raman12@illinos.edu, r-sowers@illinois.edu, rsree@illinois.edu

Abstract

The objective of the paper is to motivate the readers towards finding implementable Internet of Things based solutions to existing well-researched engineering problems. Towards this end, this paper describes a portable and easy to use technology to enable consumers to shift their electricity consumption to times when the prices are low. This technology can be implemented on a platform that consists of a *Belkin WeMo Smart Plug* (WeMo), which can be autonomously turned on/off using an *Intel Edison* (Edison) through a WiFi connection. We use a *simple* Markov Model for predicting the real-time prices. Based on the predicted prices, a threshold-based policy is used for scheduling of devices. The choice of model and the scheduling algorithm is heavily influenced by the processing- and storage-capacity of this platform. Empirical analysis on real-data from *Commonwealth Edison* (ComEd), which is the largest electric utility in Illinois, shows that 90% of times, the customer paid no more than 1.5-1.6 times the omniscient-minimum electricity bill for a particular device usage requirement.

Keywords
Smart Homes, Internet of Things, Electricity pricing, Intel Edison and Markov Model.

1. Introduction
The concept of *Internet of Things* (IoT) is an extension of the concept of the internet in which a large number of devices, sensors and everyday objects can interact with each other and with the internet. IoT has applications in manufacturing, healthcare, agriculture, transportation etc. The objective of this paper is to motivate the readers towards finding elegant IoT based solutions to already well-researched problems. In this paper, we focus on a particular application of IoT for smart homes to achieve this end goal. This paper describes a portable and easy to use technology to enable consumers to shift their electricity consumption to times when the prices are low.

We first briefly describe the well-researched topic of electricity price prediction. Under *Real-Time Pricing* (RTP) the cost per kWh of electricity varies with time as a consequence of the changing supply and demand conditions in the market. Consumers can reduce their costs by shifting their electricity consumption to times when cost is low. This results in better utilization of generated-power, lower-costs to the consumer, and is also said to have a positive impact on the environment. The ability to forecast real-time prices, and a means to react to these forecasts are two equally important requirements for an optimal consumer-response to dynamic pricing. We take a consumer-centric approach for realizing the potential of RTP programs by aiming for a cheap and accessible technology. The approach presented in this paper uses the pricing scheme of *Commonwealth Edison* (ComEd), which is the largest electric utility in Illinois, serving the Chicago and Northern Illinois area.

The present work is motivated by the problem of scheduling electricity usage in a home, based on real-time prices. For example: Tesla's Charge-at-Home (see Tesla web-link in references) recommends an hour of charge every 29 miles of range. It is estimated that a Tesla user would require about four hours of charge daily. The charging-times do not have to be contiguous. The cost to the user would be determined by the price offered by the electricity provider during the charging periods. Suppose we want to charge for a total of 4 hours in the next 12 hours. The solution would be straightforward if by some algorithm, the precise hourly prices are known beforehand. The

Omniscient Optimal Strategy (OOS) would use this knowledge of precise hourly prices and pick the four cheapest hours and charge the battery during these times. Obviously, we cannot know the exact hourly prices beforehand, and hence OOS is not practically implementable. But the performance of OOS is the best that can be achieved and it serves as a benchmark. The performance of all other strategies is compared to that of the OOS.

A scenario where the price for each hour is known at the beginning of the hour can soften the requirement of omniscience introduced in the previous paragraph. If we were to charge the battery of a Tesla car, we are to find the four hours with the lowest-price during the time (say twelve-hours) when the car is at the consumer's premise. This is akin to the problem of hiring the *4*-best candidates among a total of 12 candidates that appear for the interview. This plays into the narrative of the literature on the *secretary*- and *hiring-problem* (Henke, (1976) Berezovskiy (1983), Preater (1994)). Kleinberg (2005), Henke (1973), Berezovskiy et. al. (1986) and Broder et. al. (2010) present approaches to solve this problem which we refrain from presenting here in the interest of space.

The main issue with applying these techniques to the scheduling problem for devices is that the precise hourly price is *only know after the hour has passed*. ComEd offers a 5-minute price, and an hourly-price, which is the real-time average of the current hour's twelve 5-minute prices (cf. ComEd 5-min prices web-link in references). The *Real-Time Hourly Price* (RTHP) for the hour is the average of these twelve 5-minute prices (cf. ComEd live prices web-link in references). Consequently, the RTHP for an hour is known only after the hour has passed; and the decision to turn on/off an end-users device is always made when the price is uncertain. This makes the problem equivalent to the multiple-secretary problem with noisy measurements. Unfortunately, the results of the classical secretary problem are not applicable in the presence of noise (cf. Krieger (2012)).

In light of the mechanism by which the RTHP is revealed, we are left with no choice but to use the predicted real-time prices in determining the periods of consumption for various devices and appliances. We limit the scope of this paper to scheduling devices for which the consumption-periods need not be contiguous, as would be the case with charging the battery of an electric vehicle. Such devices can be scheduled using a *preempt-then-resume* discipline, where consumption can be preempted at any point, to be resumed later on, and the disparate consumption-periods can be conceptually coalesced as one. This cannot be done with tasks like turning on a washer or a dryer at a customer premise. Some tasks have precedence constraints that have to be followed, as well. For example, the washing-task has to precede the drying-task. We leave the scheduling of devices with such constraints as future work; with a side-comment that there is a lot of literature on the optimal scheduling of home appliances using a *Mixed Integer Linear Programming* (MILP) formalism: Qayyum (2015), Antonopoulos (2012) and Sou (2011) among others. But in all of these instances, it is assumed that the hourly price of electricity is known with certainty beforehand (as by the OOS). As discussed earlier, this is not a true reflection of reality.

While we explore a solution to this problem, we would like to reiterate that the objective is to find a consumer-centric solution. For an optimal consumer-response the forecast of real-time prices and a means (a platform/device) to react to these forecasts are equally important. Such a consideration limits the scope of algorithms and strategies that can be used for predicting the prices because of the computation power and cost considerations. In this paper, we build on the work done in Ramavarapu (2017) on the design and construction of a smart outlet that uses an *Intel Edison* (Edison), *Belkin WeMo Mini Smart Plug* (WeMo) (see Edison and Belkin web-links in references), and (easily extended) JavaScript code. We develop the algorithm from the perspective of implementing it on devices with similar cost, computational power and memory constraints.

Next we discuss the possible consequences that such a technology can have. The *U.S. Energy Information Administration* (USEIA) reports that in 2016, the annual average price of electricity in the United States was 10.28c per kilowatt-hour (kWh). Residential customers paid 12.55c per kWh; commercial customers paid 10.37c per kWh; Industrial customers paid 6.75c per kWh; and Transportation customers paid 9.48c per kWh, respectively (see USEIA web-link in references). A striking feature of this data is that Industrial customers pay almost half of what Residential customers pay. A network of several small devices can potentially connect the devices and appliances in the homes of a large cooperative of people. The cooperative can enter the bidding market as a single large player and bargain for cheaper prices of electricity. This would be helpful for the suppliers as well, as they would have guaranteed consumption, and hence can manage their resources (generators etc.) more efficiently. While this is an oversimplified discussion of a complex system, it does suggest a future research direction with significant impact.

With all the above discussions in mind, we can broadly divide the technology in three components:

1. An affordable and portable platform.
2. A model, compatible with the platform, for price forecasting.
3. An algorithm, compatible with the platform, for scheduling.

The paper is organized as follows. Section 2 presents a short description of the platform developed in Ramavarapu (2017). The model development is discussed in detail in Section 3. Section 4 presents the information structure and Section 5 describes a tractable scheduling algorithm that can be implemented on a platform with limited processing and storage capacity, which can seamlessly adapt to various consumer preferences. Section 6 presents the results. We conclude the paper with Section 7 where we discuss some of the assumptions, and present future directions of research.

2. A Smart Power Outlet that Reacts to Real-Time Pricing

The *Intel Edison* (cf. figure 1) (See Balani web-link in references for more details) is a 22 nm Intel *System on Chip* (SoC) that includes a 500 MHz dual core, dual threaded Intel Atom CPU with 4 GB of flash memory with pre-installed *Yocto Linux* (see Yocto web-link in references). It also has a 100 MHz 32-bit Intel Quark micro-controller, and a Broadcom BCM43340 chip that provides standard dual-band 2.4 GHz and 5 GHz IEEE 802.11 a/b/h/n connectivity. The WiFi connection can be protected using WPA and WPA2 authentication. It supports 40 *General-Purpose I/O* (GPIO) connections. It can be programmed using Intel's Integrated Development Environment (IDE) -- Intel XDK -- which can be used develop JavaScript-based *Node.js* (see Node.js web-link in references) applications for the Edison.

Figure 1: The Intel Edison with a Mini Breakout Board from Adafruit.com.

Figure 2: The Belkin Wemo Mini Smart Plug (WeMo)

The Edison can connect via Bluetooth or WiFi to devices like smart-phones and outlets like the *Belkin WeMo Mini Smart Plug* (WeMo) (cf. Figure 2 of Belkin web-link given in references), which is a switched-outlet that can be controlled via IP networks. It comes with an iOS/Android App that can turn on/off the plug (and any device that is connected to it) remotely. The *wemo-client* is a low-level client library that can be used to control WeMo devices within *Node.js* applications. The details of the *wemo-client* API can be found in WeMo web-link given in the references. It can be installed on the Edison using the *npm install wemo-client* command via the SSH Terminal on the Intel XDK.

Ramavarapu (2017) implements a basic algorithm which turns on the device whenever the electricity price is negative. Figures 8 and 9 of Ramavarapu (2017) present JavaScript code that turns on (resp. off) the WeMo when the Current-hour-5-Minute-Price is negative (resp. positive). Since WeMo devices can be temporarily lost and the smart outlet of Ramavarapu (2017) includes the WeMo device-discovery process. It also includes details as to how the Edison's clock can be synchronized to local-time. The Markov Model of the next section can be easily implemented in this platform.

3. The Model

The objective is to find a reasonably accurate and simple model of the system. Since the model has to be compatible with the Intel Edison-based platform, its size, and the computation-time of any associated algorithm should be kept as low as possible. To this end we use the *Day-Ahead Hourly Price* (DAHP), that is available in addition to the *Real-Time Hourly Price* (RTHP), from providers like ComEd. It will be apparent that our strategy will give the exact same results as the OOS when the day-ahead prices reflect real time prices accurately. But that is not always the case, as shown in Figure 3. Suppose we have to use an electric appliance for 3 hours in the interval from 12:00PM to 6:00PM. Based on DAHP, we would have chosen to charge it from 12:00-3:00PM. For a 100kW/h consumption, we would be charged $26.5 for electricity. Now had we had a reasonable estimate of real time prices, we would have chosen to use the appliance from 2:00-3:00PM and 4:00-6:00PM, with an electricity bill of \$12.9. Therefore, by following a DA schedule, we ended up paying more than twice the cheapest amount.

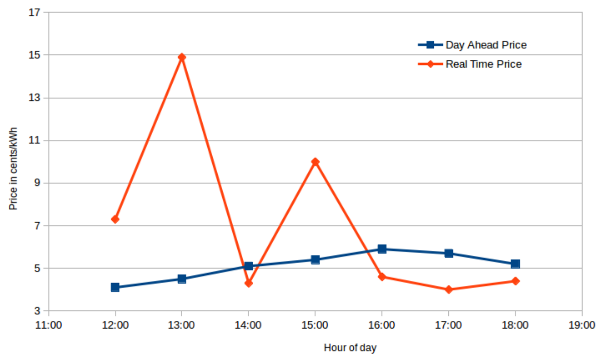

Figure 3. The plot shows the variation between DAHP and RTHP from 1200 to 1800 hours on July 26, 2017.

We adopt the following teleology when it comes to the DAHP and the RTHP: variations due to factors like weather, load, congestion etc. have been estimated accurately, and are accounted in the DAHP. The difference between RTHP and DAHP is due to factors that explicate themselves as the day progresses that could not have been predicted/estimated earlier by any means. In a sense, we assume the process of estimating the "slow-dynamics" of DAHP is as good as it can get, and the RTHP is the result of "faster-dynamics" being added to the DAHP. Another interpretation is that RTHP is the DAHP with some perturbation added to it. For ease of exposition, let us call the factors resulting in the faster dynamics (perturbations) as *fast-factors*. The best indicator of fast-factors are the 5-minute prices within each hour. We assume that the fast-factors are independent of each other, and act independently on each 5-minute price that they can influence. Further, we assume that the effect of a change in their values over price lasts for 5 minutes (like a finite impulse response of a linear system); that is, each fast-factor can influence at most two 5-minute prices. In other words, the present 5-minute price depends only on the fast-factors which start affecting the prices in this 5-minute interval, and which affected the prices of the previous 5-minute interval. With these considerations, a naturally arising model is a Markov Model (MM). That is we assume that the slot prices follow Markov property. Keeping in mind the constraints of the platform, to simplify the algorithm and to reduce the size of the state-space, we assume that the probability distribution of prices is stationary. The stationarity assumption may not valid in practice if we have to estimate the prices. However, since we only have to select the cheapest hours, any model would work as long it predicts the *relative ranks* of the hourly prices accurately. This gives us some leeway in making assumptions.

4. The Information-Structure of Strategies based on the Real-Time Hourly Price

Taking the lead from ComEd's API, we will assume the mechanism/information structure for real-time pricing is as follows:

1. Each day is split into 24-many semi-open intervals $\{(t-1,t]\}_{t=1}^{24}$.
2. Within each semi-open interval $(t-1,t]$, there are 12-many, 5-minute prices, $\{p_t^i\}_{i=1}^{12}$ that are revealed to the end-user. More specifically, the 5-minute price, p_t^i, is revealed to the end-user at the "$(t-1)$-hour-and $(i\times 5)$" minute on the clock.
3. The hourly-price for the interval $(t-1,t]$ is known *with certainty* at time "t-hours" onwards, and it is $\frac{1}{12}\left(\sum_{i=1}^{12} p_t^i\right)$. That is, the hourly-price for the interval $(t-1,t]$ is known only after the hour has passed.

We restrict our attention only to *minimum-cost strategies*. That is, we are interested in developing strategies that decide when to power-up/power-down devices attached to the smart outlet with an eye towards minimizing the cost of use to the end-user. We consider tasks like charging an electric-vehicle, where the charging periods do not have to be contiguous, and there are no dependencies among the appliances. Suppose we are to charge the Tesla's battery for a total of four hours during the night, the *Omniscient Optimal Strategy* (OOS) would be to pick the four cheapest hours and charge the battery during these times. The OOS is our performance benchmark. Under the above mentioned mechanism/information-structure, a non-OOS strategy has to estimate the hourly-price for the interval

$(t-1,t]$ after observing a few of the 5-minute prices. The variance of this estimate will (automatically) be zero at the *t*-th hour onwards, but by this time this precise hourly-price would be too late to be useful in decision-making.

5. Algorithm

The first step in developing the Markov Model is converting the prices into discrete states. Following this, the state-transition probabilities are estimated using historic data.

5.1 State Classification

The resolution of electricity prices is 0.1 cents. We convert the prices into discrete state by multiplying them by 10. To make the state space finite, we define two constants $minT$ and $maxT$. Any price less (resp. greater) than $minT$ (resp. $maxT$) is lumped to the state corresponding to $minT$ (resp. $maxT$). Intuitively, this corresponds to the consumer specific threshold. If the price is less (resp. more) than $minT$ (resp. $maxT$), the consumer will definitely turn the device on (resp. off). The total number of states that we have in our space is: $|S| = 10 \times (maxT - minT) + 2$. In our analysis, we take $minT = 0$ and $maxT = 10$ resulting in a total of 102 states. Prices less than or equal to 0 cents\kWh are lumped to state 1, and prices greater than 10 cents\kWh are lumped to state 102.

```
Algorithm 1: StateClassification(price, $maxT$, $minT$)
  if price ∈ $[minT, maxT]$ then
    state(price) = 10 × price+1
  else if price > $maxT$ then
    state(price)= $|S|$
  else state(price)=1
  end if
```

5.2 Estimating the Transition Probability Matrix

After the prices are classified into states, we use historical data to estimate the transition probability matrix A of the Markov Model. Let the maximum likelihood estimate of the transition matrix be denoted by $\hat{A}$. $\hat{A}_{ij}$ is the element in row i and column j of $\hat{A}$ which denotes the probability of transitioning from state i to state j. N is the size of the data set which we use for estimation. $z_n = s_k$ means that the system is in state s_k at instant *n*. $\mathbf{1}\{\cdot\}$ denotes the indicator function, which is equal to unity if its argument is true, and zero otherwise.

$$\hat{A}_{ij} = \frac{\sum_{n=1}^{N} \mathbf{1}\{z_{n-1} = s_i, z_n = s_j\}}{\sum_{n=1}^{N} \mathbf{1}\{z_{n-1} = s_i\}}$$

Basically, the maximum likelihood probability of transitioning from state i to state j is the ratio of the number of times we transition from state i to state j and the number of times we visit state i.

5.3 Algorithm

There are 12-many, 5-minute prices $\{p_t^i\}_{\{i=1\}}^{\{i=12\}}$ within an hour-long semi-open interval $(t-1,t]$. Let $\{\hat{p}_t^i\}_{\{i=1\}}^{\{i=12\}}$ denote the 12-many, estimated, 5-minute prices based on the presented input, we compute (a revised-estimate) of the real-time hourly price for the $(t-1,t]$-interval to be

$$\hat{C}_t^k = \frac{1}{12}\left(\sum_{i=1}^{k} p_t^i + \sum_{i=k+1}^{12} \hat{p}_t^i\right)$$

That is, $\hat{C}_t^k$ is the predicted hourly price for the $(t-1,t]$-interval at the "$(t-1)$ hour-and-($(k+1)\times 5$)th-minute" time-instant. This estimate stays relevant for five minutes, after which the next 5-minute price is known and the estimate is revised accordingly. In Algorithm 2, the *expected value of state* obtained by taking the product of the transition probability matrix with $[1 \ldots |S|]^T$ is the predicted state for price in a slot $\hat{p}_t^i$. $[1 \ldots |S|]^T$ denotes a column vector with entries 1 to $|S|$. Then, $\hat{p}_t^{i+1}$ is calculated by interpreting $\hat{p}_t^i$ as the actual price. This is carried on till we have predicted all 5-minute prices: $\{\hat{p}_t^i\}_{\{i=1\}}^{\{i=12\}}$, following which we have an estimate of the hourly price $\hat{C}_t^k$. $\hat{A}_{(i):}$ denotes the *i*-th row of $\hat{A}$. h is the number of hours for which we want to use the device.

Algorithm 2: PredictAndSchedule($\{p_t^i\}_{\{i=1\}}^{\{i=k\}}$, $\hat{A}$)

```
$\{\hat{p}_t^i\}_{\{i=1\}}^{\{i=k\}} = \{p_t^i\}_{\{i=1\}}^{\{i=k\}}$
**for** $i = k + 1$ to 12 **do**
    $\hat{p}_t^i = \hat{A}_{(i-1):} \times [1 \dots |S|]^T$
**endfor**
$\hat{C}_t^k = \frac{1}{12}\left(\sum_{i=1}^{12} \hat{p}_t^i\right)$
**if** $\hat{C}_t^k \leq lB \times \max\{D_p\}$ and $c \leq h$ **then**
    Turn ON
**elseif** $\hat{C}_t^k \in [lB, uB) \times \max\{D_p\}$ and $D_t(t) = 1$ and $c \leq h$ **then**
    Turn **OFF**
**endif**
```

Once we have predicted the hourly price, the next task is to device a policy for turning on and off of devices. If the policy is too strict on prices, then we might not fulfill the quota of the required hours of charging. On the other hand, if the policy is too relaxed, then the electricity bill might be high. Besides, we have to consider the fact that the "value" of electricity is different for different consumers. What seems cheap to one consumer might be expensive for others. On the other hand, based on the urgency of the task, the "value" of electricity might be different at different times for the same consumer. At this point, it is tempting to pose this as an optimization problem for minimizing the cost due to the electricity price and hours lost. Having a variable for denoting the "value" of electricity would complete the formulation. But if the technology is to be used by the consumer, then such a formulation (with a complicated interpretation of "value") would not make it consumer-friendly. With these restrictions in mind, we opt for a threshold-based policy. To keep it simple, the thresholds are prices themselves, which now directly represent the subjective "value" of electricity to the consumer. The algorithm develops in the following steps:

1. The day ahead prices are released at the beginning of each day. The first step is to schedule the usage based on the day-ahead prices. This scheduling would give us an idea about the subjective "value" of electricity for a particular consumer at a particular time. This step also, rather than redoing everything from scratch, makes use of the back-end machinery that predicts the day ahead prices. This helps in keeping the computations tractable for the platform.
2. The second step is to determine the thresholds. Let D_t be a vector of 0's and 1's which represents the DA schedule of electricity usage. The size of D_t represents the horizon of the hours over which the electricity is to be used. 0 and 1 as entries of D_t denote the turning ON and OFF of device respectively, for respective hours. Let D_p denote the vector of prices of hours for which $D_t(\cdot) = 1$. We use the maximum over D_p as an indicator of the subjective "value" of electricity for the consumer.
3. The third step is to insure the consumer against drastic departures of real time prices from the day ahead prices (Fig. 3). The idea is to change the schedule if the real time price is cheaper or costlier than the day ahead prediction. Here, again, the terms "cheap" and "expensive" are subjective. The parameters lB and uB are used to encode these preferences for fine-tuning the "value" of electricity for a consumer. $uB \times \max\{D_p\}$ denotes the maximum value that the consumer is willing to pay. $lB \times \max\{D_p\}$ takes care of the case when the real time prices are less than predicted prices.
4. The prices are broken into three bands: *definitely-ON*, *definitely-OFF*, and *fuzzy*. Predicted prices less than $lB \times \max\{D_p\}$ and greater than $uB \times \max\{D_p\}$ are the *definitely-ON* and *definitely-OFF* bands respectively. For prices lying in between these bands, we turn ON or turn OFF the devices depending on the day-ahead schedule.

6. Results

We present a qualitative analysis of the algorithm before discussing the plots. The consumers are more likely to miss the required hours of charging if they use a lower value of uB. A large of value of uB would indicate a consumer's willingness to pay more, and a large value of $(uB - lB)$ would mean a larger reliance on the day-ahead schedule. Table 1 summarizes the qualitative analysis of consumer preferences based on the choice of lB and uB.

Table 1: Qualitative relation between tuning parameters and Consumer preferences

lB	uB	$uB - lB$	Consumer Preferences
-	low	low	Saving money, does not trust DA Schedule
high	high	low	Willing to pay more, does not trust DA Schedule
low	high	high	Willing to pay more, trusts DA Schedule

The 5 minute price data is downloaded from the API in ComEd's website (*https://hourlypricing.comed.com/hp-api/*). It contains the 5-minute prices along with the time in millis UTC. The master data set is comprised of 5-minute real time price data from December 20, 2015 – July 15, 2017. The data set had values missing for some (roughly 1%) of the slots in between. We ignored such slots and assumed that all data is for consecutive slots. The idea is to verify the results of the algorithm against different training and validation data sets. To test the algorithm rigorously, and to test it against factors/artifacts that can vary depending on the time of year, we create three different sets of validation and training data from the master data set.

1. Set 1: Training data from April 01, '16 to July 15, '17; Validation data from December 20, '15 to March 31, '16.
2. Set 2: Training data from January 01, '16 to August 14, '16 and December 2, '16 to June 15, '17; Validation data from August 15, '16 to November 30, '16.
3. Set 3: Training data from January 01, '16 to March 31, '17; Validation data from April 02, '17 to July 15, '17.

The validation data together is representative of the electricity prices in the whole year. Moreover, to ensure that the performance of the model and algorithm is not due to any correlation that might exist with the training data, the validation data is chosen to be chronologically before, in between and after the training data. For scheduling and comparison, we use the hourly day-ahead and real-time prices from the pricing tables given in ComEd's website (*https://hourlypricing.comed.com/live-prices/*). We consider the problem of using the electricity for 4 hours in between 12:00-8:00PM on each day in the validation data set. For each day, the algorithm determines a schedule, which results in a cost. This cost is expressed as a percentage of the OOS-cost. Since the performance depends heavily on the tuning parameters lB and uB, we plot the difference in cost statistics with respect to OOS for various combinations of lB and uB for all data sets. We consider lB values from 0.6 to 1 in steps of 0.2, and uB values from lB to 2 in steps of 0.1. We do not take absolute value of difference of costs.

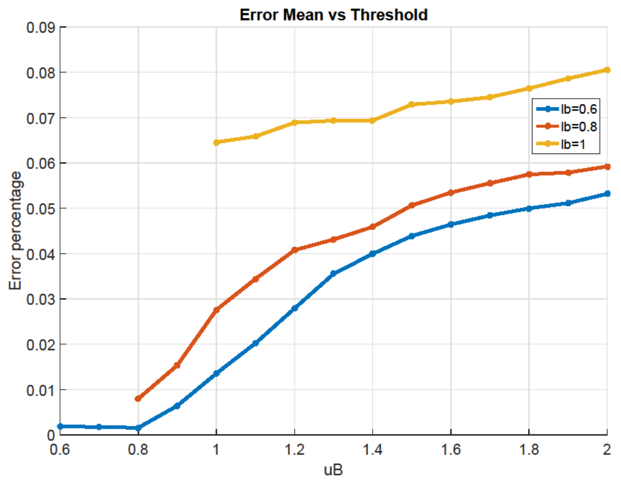

Figure 4: Average difference in costs between the proposed algorithm and the OOS for Dataset 1, plotted for various tuning parameters (lB and uB).

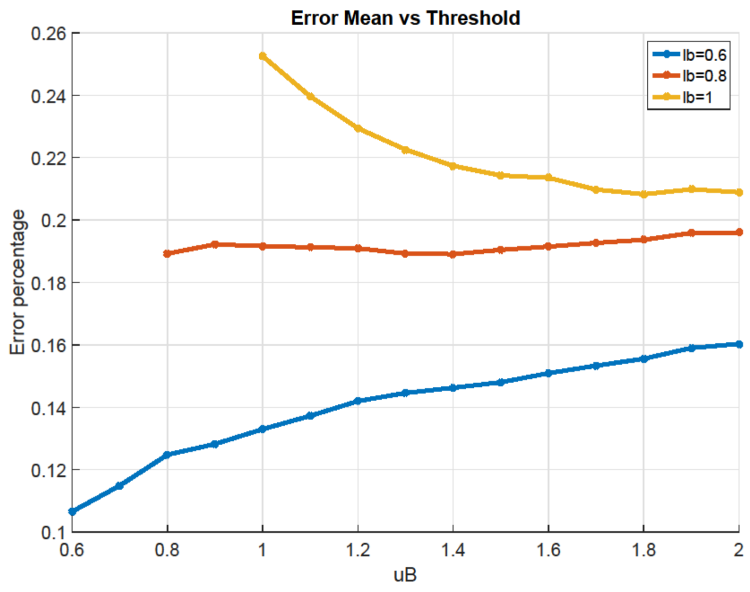

Figure 5: Average difference in costs between the proposed algorithm and the OOS for Dataset 2, plotted for various tuning parameters (lB and uB).

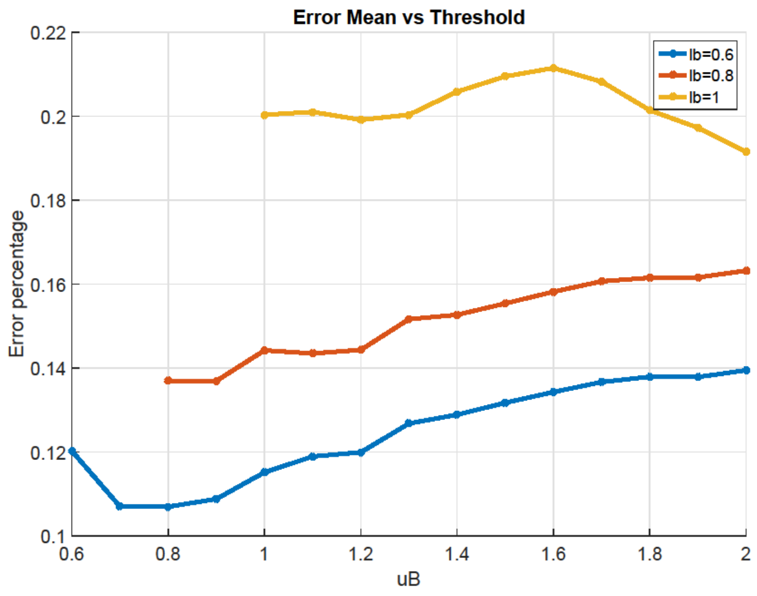

Figure 6: Average difference in costs between the proposed algorithm and the OOS for Dataset 3, plotted for various tuning parameters (lB and uB).

On an average a customer using the proposed algorithm is overcharged by approximately 20%, compared to the OOS. In fact, for Dataset 1 it performs even better. The plot of 90% quantile (Figures 7, 8 and 9) gives us some more insights. It expresses the percentage of cost overcharged with probability 0.9. In other words, it tells us how much is the consumer overpaying 90% of the time. Although the performance against data set 1 is exceptional, generally it can be said that 90% of times, the customer paid no more than 1.5-1.6 times than what she would have paid under the OOS.

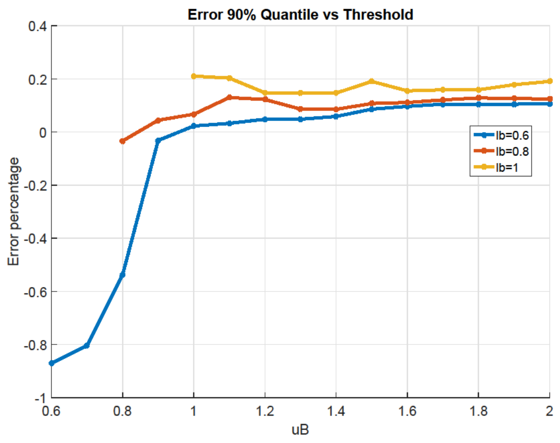

Figure 7: 90% Quantile of difference in costs between the proposed algorithm and the OOS for Dataset 1, plotted for various tuning parameters (lB and uB).

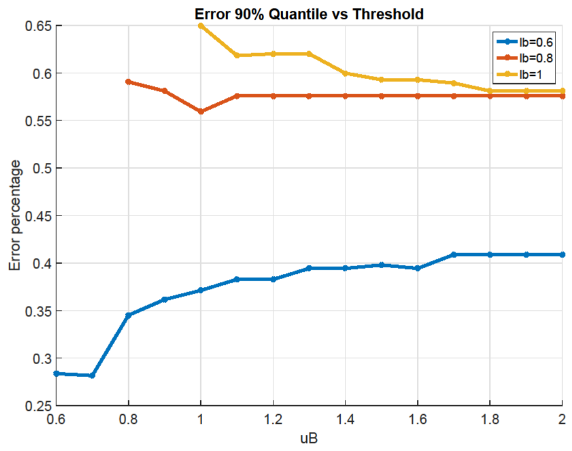

Figure 8: 90% Quantile of difference in costs between the proposed algorithm and the OOS for Dataset 2, plotted for various tuning parameters (lB and uB).

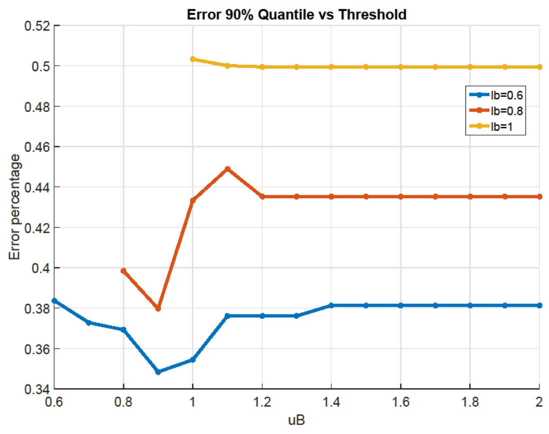

Figure 9: 90% Quantile of difference in costs between the proposed algorithm and the OOS for Dataset 3, plotted for various tuning parameters (lB and uB).

It is to be noted that the optimal schedule calculates the price for 4 hours of charging. If the number of hours of charging is not met, the difference in cost statistic will have lower values (sometimes even negative). Therefore, analysis of percentage of hours missed is complementary to the analysis of difference in cost statistics. The percentage
of hours missed is plotted for various combinations of lB and uB in Figures 10, 11 and 12. As expected, as the value of uB increases, the number of hours missed decreases. The same behaviour is true with respect to lB as well. For each value of lB, the percentage of hours missed is largest when lB is equal to uB. The percentage of hours missed is reasonably lower (<10%) for appropriate tuning parameters.

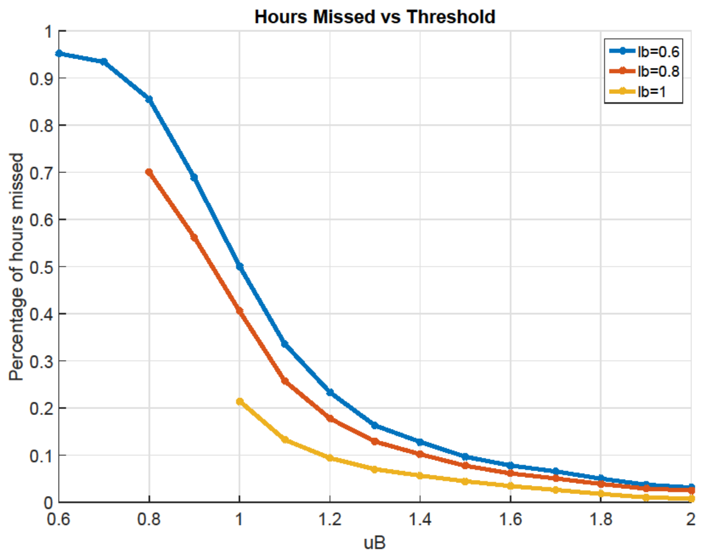

Figure 10: Percentage of hours of charging missed by the proposed algorithm for Dataset 1, plotted for various tuning parameters (lB and uB).

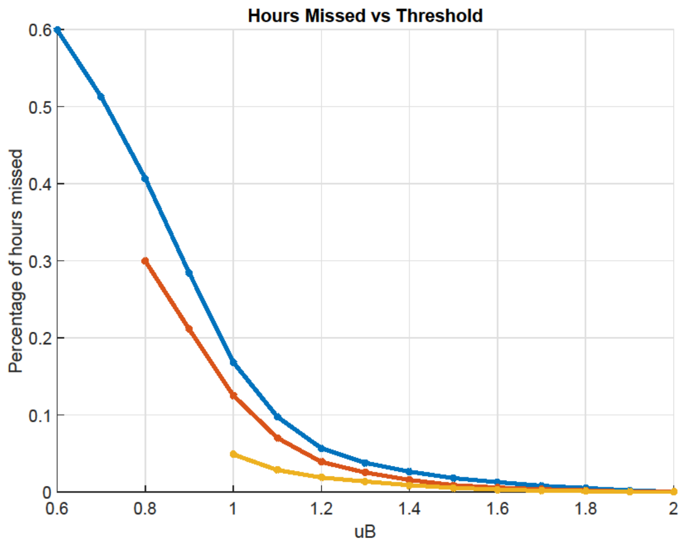

Figure 11: Percentage of hours of charging missed by the proposed algorithm for Dataset 2, plotted for various tuning parameters (lB and uB).

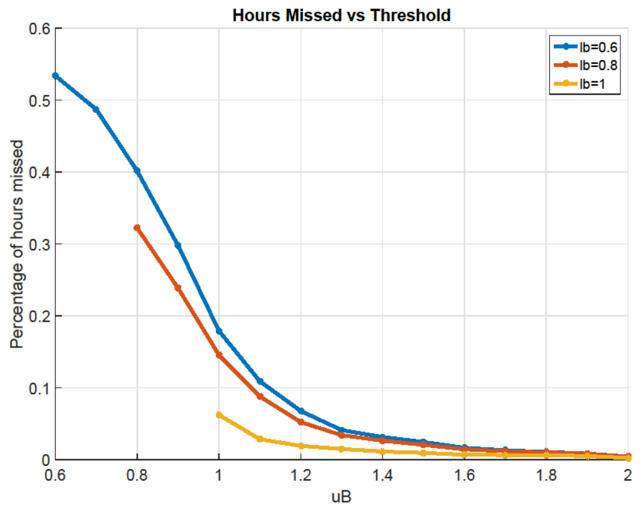

Figure 12: Percentage of hours of charging missed by the proposed algorithm for Dataset 3, plotted for various tuning parameters (lB and uB).

7. Conclusion

We worked under the teleological model that the DAHP has taken into account everything that could possibly be predicted before the day commences (cf. the ``slow dynamics'' discussion in Section 3). The RTHP is essentially the DAHP with additional ``fast dynamics'' that are explicated at the last minute. This helped us to make the prediction-algorithm tractable on the *Edison*. We used a Markov Model to predict the prices. Although we convert the prices into a finite state-space of a reasonable size, such an action inherently introduces inaccuracies in price prediction. For example: assume that the price at 6:00PM is 8 cents per kWh, and that $maxT = 10$. The model will treat the two cases: 1) price at 6:05PM is 25 cents per kWh, and 2) price at 6:05PM is 10 cents per kWh, as being identical. Therefore, it always underestimates high prices. Similarly, it will always overestimate low prices.

We assume that the effect of fast-factors lasts only for 5-minutes. Because of these assumptions we could limit the size of the state space to around 100, and to around 10^4 entries in the transition probability matrix A. For illustration, suppose each entry in A is stored as type *double* (in C++ terminology), which has a size of 8 bytes. Then we need ~80kB of size to store the matrix, which is fairly small. Now, if we relax our assumption and assume that the effect of fast-factors lasts for 10 minutes, then for the Markovian assumption to hold we will have to raise the current state-space to the space where each state is a two-tuple (p_t^i, p_t^{i+1}). This would result in a space of size approximately 10^4, and A will have 10^8 entries which would need around 800MB of space.

We also assume that the probability distribution of prices is stationary. As discussed in Section 1, the stationarity assumption is not necessarily true if we have to predict the prices. We then argued in Section 3 that since we only have to select the cheapest hours, any model would work as long it predicts the *relative ranks* of the hourly prices accurately. This peculiarity gives us room for making the assumption. That said, we can remove this assumption in two ways. The first way is to estimate different transition matrices to be used during different times of the day, which will be a direct extension of this work. The second way is to raise the state space in a way similar to one discussed in the previous paragraph. We would need a 3 or 4-tuple state (the other elements of the state being past hourly prices) to reliably account for stationarity. We end up with a state explosion problem.

As we receive more 5-minute prices for a given hour, the predicted hourly prices become more reliable. But if the consumer waits for a relatively reliable estimate, she loses out on the amount of time she can charge in the given hour. We have not encoded the risk-taking tendencies of consumers in this algorithm.

Since we are looking at the solution from the perspective of the consumer, it is not clear how to pose the scheduling as an optimization problem for minimizing the cost due to the electricity price and hours lost. We have to consider the fact that the "value" of electricity is different for different consumers. In fact, based on the urgency of the task, the "value" of electricity might be different at different times for the same consumer.

Although we have promising empirical results for our algorithm, we do not have any theoretical bounds. Computing theoretical bounds on the *Competitive Ratio* (CR) (i.e. the ratio of the cost obtained by an Online-Algorithm with that of the Omniscient Optimal Algorithm) can be a tall-order even for well-formulated/straight-jacketed academic problems. Attempts at deriving a formal bound for the CR might require making distributional assumptions and other simplifications. A solution towards this end with a practical sanction is an open problem.

We limited the scope of this paper in scheduling devices for which the consumption-periods need not be contiguous. A real-time MILP formulation such as the ones in Qayyum (2015), Antonopoulos (2012) and Sou (2011), but with online prediction of prices remains an open problem. It is a difficult problem considering that it might run into numerical issues arising from the large number of variables in the formulation.

References

Tesla web-link, https://www.tesla.com/charge-at-home, Last accessed: 21-August2017.

B. Berezovskiy, B., Baryshnikov, Y., and Gnedin, A.,The Secretary Problem and Its Extensions: A Review, *International Statistical Review*, vol. 51, no. 2, pp. 189–206, August 1983.

Henke, M., Expectations and Variances of Stopping Variables in Sequential Selection Processes, *Journal of Applied Probability*, vol. 10, no. 4, pp. 786– 806, 1973.

B. Berezovskiy, B., Baryshnikov, Y., and Gnedin, A., On a Class of Best-Choice Problems, *Information Sciences*, vol.
39, pp. 111–127, 1986.

Preater, J. On multiple-choice secretary problems, *Mathematics of Operations Research*, vol. 19, no. 3, August 1994.

Kleinberg, R., A multiple-choice secretary algorithm with applications to online auctions, in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’05. Philadelphia, PA, USA.

Broder, A., Kirsch, A., Kumar, R., Mitzenmacher, M., Upfal, E. and Vassilvitskii, S., The Hiring Problem and Lake Wobegon Strategies,” *SIAM Journal of Computing*, vol. 39, no. 4, 2010.

ComEd 5min prices web-link, https://hourlypricing.comed.com/live-prices/five-minute-prices/, Last Accessed: October 19, 2017.

ComEd live prices web-link, https://hourlypricing.comed.com/live-prices/, Last Accessed: October 19, 2017.

Krieger A. M., and Samuel-Cahn, E., The noisy secretary problem and some results on extreme concomitant variables, *Journal of Applied Probability*, vol. 49, no. 3, pp. 821–837, 2012.

Qayyum, F. A., Naeem, M., Khwaja, A. S., Anpalagan, A., Guan, L. and Venkatesh, B., Appliance scheduling optimization in smart home networks, *IEEE Access*, vol. 3, pp. 2176–2190, 2015.

Antonopoulos, C. P., Kapsalis, V. and Hadellis, L., Optimal scheduling of smart home appliances for the minimization of energy cost under dynamic pricing, in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, September 2012, pp. 1–4.

Sou, K., Weimer, J., Sandberg, H. and Johansson, K., Scheduling smart home appliances using mixed integer linear programming, in *Proceedings of 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, December 2011, pp. 5144–5149.

Ramavarapu, V., Sowers, R. and Sreenivas, R., A smart power outlet for electric devices that can benefit from real-time pricing, in *Proceedings of the 2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, September 2017, pp. 11–17, yogyakarta, Indonesia.

Balani, N., Overview of the Intel Edison module, https://software.intel.com/en-us/articles/what-is-the-intel-edison-module, July 18, 2016, Last accessed: 21-August-2017.

Belkin web-link, http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/, Last accessed 21-August-2017.

USEIA web-link, https://www.eia.gov/energyexplained/index.cfm?page=electricity\factors\affecting\prices, Last Accessed: May 23, 2017.

Yocto web-link, https://www.yoctoproject.org/about, Last Accessed: May 23, 2017.

WeMo web-link, https://www.npmjs.com/package/wemo-client, Last Accessed: May 23, 2017.

Biographies

Arun Raman is currently working towards the Ph.D. degree in Systems Engineering at the University of Illinois at Urbana-Champaign. He was a Research Associate with the Indian Institute of Management, Ahmedabad from 2014-2015. He worked as an Edison Engineer in GE Oil and Gas in Compressor and Gas Turbine simulation from 2012-2014. He has a Master of Science in Systems and Control from the Indian Institute of Technology Bombay, Mumbai in 2012. His research interests lie in the area of classical control, applied mathematics and the control of DEDS systems.

Richard Sowers received the B.S. degree in Electrical Engineering from Drexel University, Philadelphia, in 1986 and then the M.S. in Electrical Engineering and Ph.D. in Applied Mathematics from the University of Maryland at College Park in 1988 and 1991 respectively. He is currently a Professor in the Department of Industrial and Enterprise Systems Engineering and in the Department of Mathematics at the University of Illinois at Urbana-Champaign. He is also the Research Principal, Office of Financial Research since 2012. His current research involves looking at data from a quantitative perspective and with a socially-important goal. The various areas that he is working in include Internet of things, big data approaches to traffic, Virtual Reality and Visual Cliffs and Optimal Transport in Hand-Picked Crops among others.

Ramavarapu Sreenivas received the B.Tech degree in Electrical Engineering from the Indian Institute of Technology, Madras, India in 1985, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA in 1987 and 1990, respectively. He was a Postdoctoral Fellow in Decision and Control at the Division of Applied Sciences, Harvard University, Cambridge, MA, before he joined the University of Illinois at Urbana-Champaign in 1992, where he is an Associate Professor of Industrial and Enterprise Systems Engineering.