Automated Energy Distribution In Smart Grids

John Yu*
Stanford University

Michael Chun†
Stanford University

Abstract

We design and implement a software controller that can distribute energy in a power grid. First, locally-weighted regression is performed on past data to learn and predict the energy demand. Then, reinforcement learning is applied to determine how to distribute power given some state. We demonstrate that our software, in conjunction with smart grids and network communication systems, can automatically and efficiently manage the power grid.

1 Introduction

The goal of a resource distribution system that connects producers and consumers is to ensure that the resource generated by the producer is optimally delivered to the consumer. By optimal, we mean that the resource is delivered only to the consumers that need it in a timely fashion. In this respect, today's electric power distribution systems perform relatively well: blackouts are rare and electricity bills are generally manageable[1]. However, there is plenty of room for improvement. For one, it is good that electricity prices are low, but it would be even better if it was cheaper. One reason the prices are not lower is due to the distribution cost. Specifically, electric power distribution very much remains a manual process, requiring the work of many well-qualified operators and analysists 24 hours a day, 7 days a week. This is a good reason to ponder whether we might be be bettered served if there was a computer program that can decide how the power should be distributed and act accordingly, decresing or eliminating the need for human intervention.

A related area of improvement is energy demand prediction. Over the past couple of years, smart meters and smart power grids implementations have lead to an exponential increase in the amount of information that is available to the power operator. This data allows the operator to develop more efficient, fine-grained power distribution methods. However, there is so much information available that the operator cannot hope to process all the data without computer automation.[1]

As hinted above, at a high level, an energy distribution system needs to two things:

```
1. Predict the energy demand.
2. Based on the prediction, make decisions
   on where to guide energy.
```

Clearly, if an automated system is to replace operators, it must be able to carry out these tasks as well as humans can. Before we go into how the two tasks were tackled, we first present our model of the energy distribution network.

1.1 Energy Model

Figure 1 illustrates the model of the network that our control system will operate under as well as our assumptions. In our model there

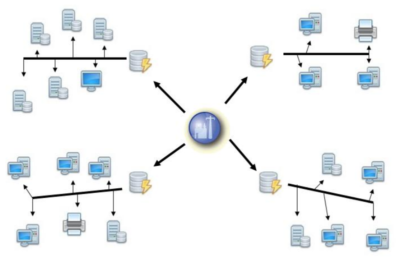

Figure 1: *A simple energy distribution model.*

are some number of rooms that have devices that need to be powered (laptops, printers, servers, etc.). Each room has a single power substation that powers them. The power substations store energy; in fact, we essentially treat them as large batteries that can be charged and discharged. The substations can be charged by drawing power from the power source in the center. By placing a power meter on every device in the room and summing them up, we can determine the amount of energy drawn from the substation that powers the room. Alternatively, we can place a meter on the substation itself, if feasible.

It is easy to see how this model can be cascaded and abstracted to represent much larger networks. For example, the power substations can be treated as leaf nodes (rather than laptops and printers), the central power station can be treated as a substation that needs to be charged by a larger parent station. So this model is simple but scales well. We also assume that power flows one way: if station A provides power to station B, then station B never sends power to station A. Although residential solar panels and other sources of alternative energy are becoming more commonplace, the one-way power model is still by far how things work, so for demonstration purposes we believe this is acceptable.

Now, we place a stronger, more limiting constraint, which is that the power source can only charge one power substation at a time at a predefined rate. This simplifies our controller's behavior into the following loop:

```
while system is running:
    observe state
    decide which station to charge next
    charge the station by some amount
```

Allowing the power source to charge multiple stations simultaneously is probably a more realistic representation, but it was avoided because it greatly increases the number of possible actions the controller can take. This increases the processing time and complicates the learning model, but it does not necessarily offer greater insight into the energy distribution process. In other words, removing this constraint will complicate but not change our learning model, so we can still validate our proposal with the simpler model.

Next we describe the data set that we used to train our controller system.

*e-mail: johnyu@cs.stanford.edu
†e-mail:mschun@cs.stanford.edu
[1]Operators already do make heavy use of computer processing power to do their work, but this same processing is a large component for our machine controller as well.

1.2 Training Data

We used the dataset from the Stanford PowerNet project [2][3]. The PowerNet data was gathered by placing a power meter into 138 devices (laptops, printers, and workstations) that are actively used in the Stanford University Gates computer science building and collecting instantaneous power usage data every second. We restricted our dataset to a 30 day period from September 1 to October 1, 2010. We then divided the data into 6 local rooms, approximately one room per physical floor of the building.

2 Controller Implmentation

As previously stated, our controller needed the ability to learn energy demand and also make decisions based on this learned data and the current state. One way for a reinforcement algorithm to learn the model of the energy demand is to simply let the controller operate for a while and learn what the resultant state is given state and action (rewards can be learned the same way). However, we chose to use locally-weighted linear regression. To understand why, first let us define what the state is for our learning model.

We are interested in keeping the power substations charged at all times. We want to keep the substations from reaching the empty energy level, since this would lead to blackout. Thus, our state is the collective energy levels of all our substations. We also predefine the rewards. When a substation's energy level is more than half, no rewards are given (reward is 0). Otherwise, we give a negative reward. In addition, we gave larger negative rewards as the energy levels got closer to empty. To simulate this reward curve, a square root function was used. i.e:

```
reward = sqrt(energy level)
```

The rewards for the substations were calculated individually and summed to get the total reward of a particular state.

We previously defined the action of the controller as "charging substation X." This is a deterministic action: commanding the controller to charge station A has a 100 percent probability of charging station A by a predefined amount.

So does this mean we know the state transition probabilities? No, since the room is full of electronic devices that are drawing power at various rates from the substations.[2]

Thus we must be able to somehow predict the energy demand. As we noted, we use locally-weighted regression to predict the energy demand rather than allowing the reinforcement learning algorithm to learn by trial-and-record. There are two (related) reasons for this. First, the energy consumption rates are not fixed; they vary from morning to afternoon and from weekday to weekend. Thus, the trial-and-record method, which really only works well in a static environment, is impractical. Second, there are an infinite number of states, both in terms of the energy level of the substations and the power consumption rates of the devices. Trial-and-record is more suited to a finite number of discrete states, which can be learned with a finite number of trials.

Note that locally-weighted regression performs better under both circumstances. Using locally-weighted regression, the controller can predict the power consumption rate at any time during a given day. Thus, varying consumption rates are not an issue. Also, by performing one regression per substation, a continuous curve can relate

[2]Since we do not know the rate at which power is being drawn, even if we know the current state and how much the next action will charge which station, we do not know the next state (the energy levels of the substations).

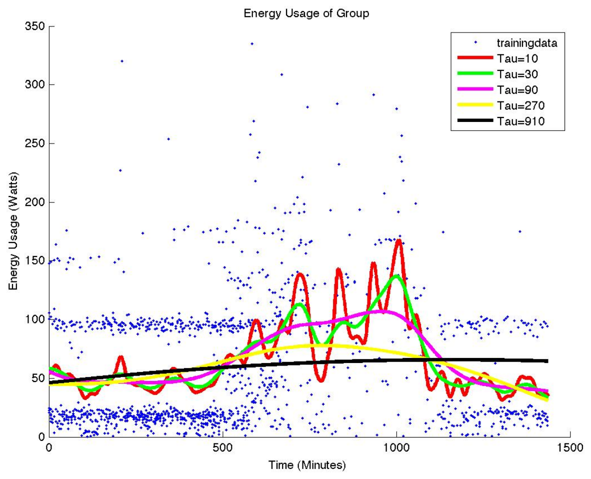

Figure 2: *Training data distribution for group 3 in weekdays and the regression estimates*

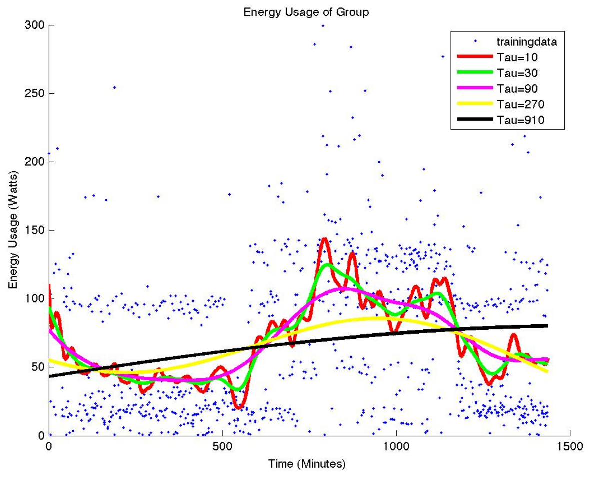

Figure 3: *Training data distribution for group 3 in weekdays and the regression estimates*

the time of day to the aggregate power consumption at the substation. One drawback of regression is that it operates on data from the past. While the past is generally a good indicator of the future, this does mean the prediction is susceptible to unexpected events (such as the school declaring a snow day, for example). However, trial-and-record also operates based on past data, so neither is favored here.

2.1 Energy Demand Prediction

Locally-weighted regression algorithm was used to learn the energy demand of the four rooms in Gates building. This algorithm is parameterized by a bandwidth parameter "τ" which controls the algorithm from being high-bias or high-variance.

To find the optimal τ value, regression was performed over τ values at 10, 30, 90, 270, 910 minutes and compared the training success rate. In our model, success rate is not straightforward because at any given time of a day, energy usage could range from minimal

Table 1: *Statistical measures for regression on group 3 in weekdays*

Taus	Std Dev	Train Success Rate	CV Success Rate
10	55.42	0.8717	0.8794
30	55.28	0.8725	0.8865
90	55.38	0.8717	0.8794
270	57.05	0.8513	0.8582
910	56.78	0.8545	0.8652

Table 2: *Statistical measures for regression on group 3 in weekends*

Taus	Std Dev	Train Success Rate	CV Success Rate
10	69.89	0.9720	0.9681
30	61.27	0.9474	0.9681
90	56.38	0.8808	0.9149
270	57.15	0.7792	0.8085
910	61.05	0.7535	0.7872

(all devices being idle) to maximum (all devices under full load). Since we cannot pinpoint the exact usage, we defined a successfull prediction as the testdata lying within certain range from the prediction. This range is one standard deviation from the predicted regression value. In Figure 2, we can see that the regression tightly follows the dynamics in the dataset when τ is low. Similarly, regression loosely averages values when τ is high. Each data group has a different distribution thus the optimal τ cannot be shared among different groups or different days. In Figure 3, the energy usage distribution for group 3 in weekends is very different from the usage from weekdays. Therefore, the optimal τ is different as shown in Table 1 and Table 2.

Finally, we verified that our τ is neither high-bias nor high-variance by performing 10 percent crossout validation where validation is successful if a data is within one standard deviation away from the regression value. The success rate algorithm turned out to be an effective measure to identify optimal τ because it resulted in the highest crossout validation success rate.

2.2 Making Decisions

Now that we have the ability to predict power demand, our transition function (simulator) is ready, and we can use the Markov Decision Process to learn the behavior of the controller. The simulator is capable of answering the following question:

Given that the current energy levels at substation A, B, and C, are x, y, and z, respectively, what will happen if substation B is charged by some amount w?

Armed with the simulator, the controller can calculate the value at every state using value iteration[4]. Since the state space is continuous, we needed to decide whether to use value function approximation or discretization of the states to calculate the value function. We first tried to use value function approximation but found that it is difficult to approximate parameters θ and some function ϕ of state S. Thus we resorted to discretization.

2.2.1 Discretization

We discretized the energy levels that the power substations contain. For example, if the substation is capable of storing 100 KWH of energy, and we want to discretize this into n = 5 levels, then we treated the energy levels as shown in Table 3.

Now suppose we have 4 substations. Then one possible state might be [10, 30, 90, 50]. There are 5^4 = 625 different states.

Table 3: *Discretization with n = 5*

Actual amount	Discretized amount
0 - 20 KWH	10
20 - 40 KWH	30
40-60 KWH	50
60-80 KWH	70
80-100 KWH	90

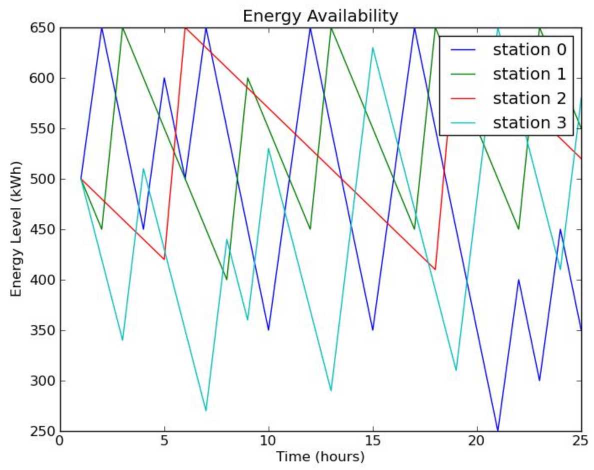

Figure 4: *Result of the simulation. The Y-axis indicates energy levels of a station, and the X-axis indicates time.*

After discretization, we ran value iteration over all 625 different states to calculate the value function for every state.

3 Results

3.0.2 Simulation

Before using the regression data from the Gates building, we ran a simulation to test for correctness and see how our algorithm performs. Our simulated network consisted of a single power source and 4 substations. Each substation consumed power at rates 100 KWH, 50 KWH, 20 KWH, and 80 KWH, respectively. The source is capable of providing 250 KWH of energy to a single station at a time. Note that the input of energy is exactly equal to output, so we are providing just enough energy to meet energy demand. The results are shown in Figure 4.

Figure 4 shows how the energy levels fluctuated over a 24 hour time period. We can note a couple of facts from the graph. One, the energy levels of the stations frequently intersect with each other, which means the decision making process is not biased toward a single station. Two, none of the energy levels reach 0, which means the substations always had energy to provide when needed.

Once we determined that the simulation is running as expected, we proceeded to use the Stanford PowerNet data.

3.0.3 Real Data

We used a slight variation of cross validation to test our algorithm on PowerNet. The power usage data was split into two sets, A and B. Set A contained 90 percent of the data and set B contained the other 10 percent. Set A was used to run locally-weighted regression

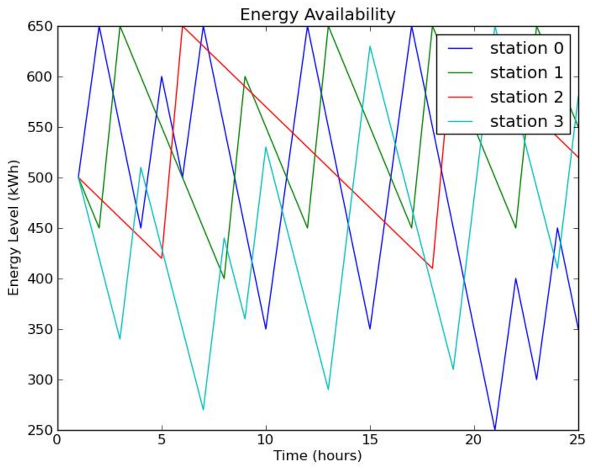

Figure 5: *Testing with PowerNet. Energy levels of the substaions are on the y-axis, and the time passed is on the x-axis.*

Station 0	Station 1	Station 2	Station 3	Next action
1500	1500	1500	1500	0
2116	1415	1441	1141	1
1233	2915	1382	783	0
2116	2830	1323	425	3
1233	2745	1264	1940	0
2022	2660	1205	1582	2
1139	2575	2941	1224	0
2015	2490	2882	866	3
1132	2406	2823	2310	0
2094	2321	2764	1952	-

Table 4: *State transition table. The left 4 columns form a state, and the right column shows the next substation to charge.*

and generate the parameters. Based on this data, the simulator was generated. Essentially, the simulator was a table that listed what the power consumption rates of each of the substations were at a given time. The reinforcement algorithm them used this to learn the value of taking a paticular action given some state, at every state.

Then the controller algorithm was validated on set B. The controller was fed with the current state of the substations and the power consumption rates on a specified day and time (retrieved from set B). Using this data, the controller made a decision that maximizes the value (learned from set A). The controller was allowed to run for 24 hours.

Figure 5 shows the energy availability graph again, this time for PowerNet data. Compared to the simulation data, it appears that there may be slightly higher bias, as rooms 1 and 3 generally tend to stay above the starting amount (1500 KWH). However, the energy levels all stay well above 0. The power consumption rates ranged from 1100 KWH to 0, so we believe providing each substation with a capacity of 3000 KWH is reasonable.

3.1 Performance

Because we discretize the states, it is reasonable to assert that our implementation does not scale to larger environments. For example, if there are 10 substations that need to be managed, and we provide 10 energy levels per substation, there are 10 billion states, each of which we need to calculate values for. But exactly how slow is our algorithm, and when does it become impractical to use?

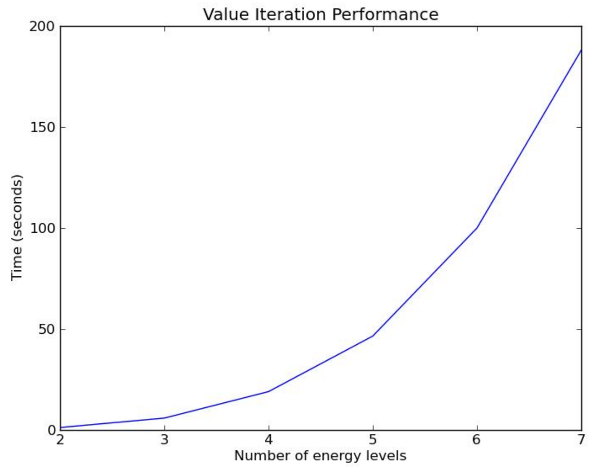

Figure 6: *Value Iteration Performance. As the number of energy levels increase, the running time increases polynomially.*

We measured the time it takes to perform one cycle of value iteration. We also capped the number of iterations to 100. The results are shown in Figure 6.

For our simulation, we used n = 4, so it took about 20 seconds per iteration. This calculation was done once every hour. Once the values have been updated, it takes far less time to make decisions based on incoming state (in the order of milliseconds), so our controller can behave in real-time. However, one can easily see from the graph that performance will quickly grow unacceptable, even with only 4 stations.

4 Conclusion

We were able to develop a functional controller for an energy distribution system. Note the level of granularity that our system can operate under depends purely on the granularity of the power usage data available, and thus smart grid technology is a necessary component to acheive fine-grained power distribution.

Locally-weighted regression was performed mostly in Matlab. However, for the implementation of the simulator and reinforcement learning, we wanted to be able to describe a model of the power distribution network. We chose to use Python with the SciPy library, which provided flexibility and capabilities of a real object oriented language while also providing the large mathematical tool library that Matlib provides.

5 Future work

We primarily concerned ourselves with energy distribution of an electric power network, but as our algorithms deal generally with a resource distribution network linking producers and consumers, this work is widely applicable to other scenarios. For example, our algorithm would apply equally well to the city water distribution system.[3]. The problem of delivering gasoline to the appropriate station can also be solved by our approach.

[3]this might be an even better application than electric grid, since here the resource (water) really does flow in only one direction

One limitation that we noted is that our network model is greatly simplified. In reality, the power source can distribute power to multiple stations at the same time, and in differing amounts, and power flow can be bi-directional. And as analyzed above, we discretize the states.

Also, our system does not account for unexpected events, or forthcomiong events in the future. In particular, our system notes that there are differences in power consumptions between weekdays and weekends, but we do not account for special occasions such as holidays[4].

Our system then has quite a bit of work to do before it can be used in real settings. However, in our view, none of the limitations are insurmountable; it is simply a matter of doing the necessary work.

5.1 Acknowledgments

We thank Maria Kazandjieva for providing us with invaluable power usage data of the Gates building. We also thank Quoc Le and Andrew Ng for giving us advice on choosing our final project topic.

6 References

[1] Dhaliwal, H. & Abraham, S. (2004) Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations, *U.S.-Canada Power System Outage Task Force.* U.S Department of Energy.

[2] Kazandjieva, M., Heller, B., Levis, P. & Kozyrakis, C. (2009) Energy Dumpster Diving. *Workshop on Power Aware Computing and Systems.*

[3] Kazandjieva, M., Gnawali, O., Heller, B., Levis, P. & Kozyrakis, C. (2010) Identifying Energy Waste through Dense Power Sensing and Utilization Monitoring. *Stanford PowerNet technical report.*

[4] Bellman, R. A. (1957) Markovian Decision Process. *Journal of Mathematics and Mechanics 6.*

[4]power consumption in residential area might far exceed the median while business areas may be low