2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems

A Penny Saved is a Penny Earned: Applying Optimization Techniques to Power Management

Fahad Javed
LUMS School of Science and Engineering
DHA, Lahore, Pakistan
fahadjaved at lums dot edu dot pk

Naveed Arshad
LUMS School of Science and Engineering
DHA, Lahore, Pakistan
naveedarshad at lums dot edu dot pk

Abstract

Shortage of electricity is a major problem in many developing countries. Unfortunately, for some of these countries the only solution to this problem is to shut down complete electricity supply to a few neighborhoods to make up for the gap between demand and supply. To this end, we have developed a self-optimization approach to reduce the gap between demand and supply through remotely controlling high powered electric devices such as air conditioners. In this approach we have used mathematical optimization techniques such as linear programming to intelligently manage the electricity distribution. Not only through this approach we have been able to provide service-level guarantees to the consumers but we have also shown that our approach is fast, scalable and has the ability to handle unscheduled spikes in the system.

1 Introduction

The usage of electricity[1] is always on the increase. However, due to the cost of developing new and alternative sources of energy, many developing countries are not able to add these sources fast enough. Due to this limitation, there is an ever increasing gap between the electricity supply and demand. As a result in many countries the only solution to fill this gap is to cut electricity to localities based on a round robin fashion. Commonly, this scheme is known as load shedding. However, not only this kind of power optimization is a great inconvenience to domestic consumers but this scheme is also detrimental to industrial development.

Therefore, in this paper we discuss a power optimization methodology that is motivated by the existing work in the area of self-managing systems. This methodology promises to eliminate or at least reduce the gap between supply and demand. This methodology uses the SAPE (Sense, Analyze, Plan, Execute) cycles and optimizes the use of electricity [3].

Complete implementation details of this methodology are difficult to explain in one paper because of the involvement of many areas of computer science, electrical engineering, law, social sciences, etc. Therefore, we will restrict ourselves to the self-management aspects of the proposed methodology.

In a nutshell, we propose a methodology that turn off high-powered devices for a small duration of time typically determined by a service-level agreement between the electricity company and the consumer. This methodology optimizes the supply of electricity to high-powered devices based on the overall supply/demand situation, a service-level guarantee and other factors. Due to the sheer size of the problem, existing control theoretic solutions are not scalable. In contrast we use clustering to transform the problem in a real domain and then use linear programming to arrive at a solution.

2 Supply and Demand of Electricity

Electricity is a basic utility and almost every device nowadays is dependent on electricity. But with load shedding hitting the level of six hours daily in some countries, consumers resort to setting up UPS (Uninterrupted Power Supply) systems to cater for the basic power consumption needs such as fans and light bulbs. The cost of a UPS prohibits it's use enmasse as it can cost upto $1000 to setup and requires on average another $500 annually for maintenance cost such as battery replacements etc. Moreover, UPS usually only powers low consumption devices and the consumers are not able to use high power appliances like refrigerators and air-conditioners. Another problem is that a UPS is charged from the same supply source where all the power is coming from, therefore, it is really not a power saving but merely a convenience for a few who can afford them. To this end, we need a better power management to

[1]The words 'electricity' and 'power' have been used in this paper interchangeably. However, in this paper for all practical purposes they are synonyms.

978-0-7695-3602-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ECBS.2009.10

	Low usage		High Usage	
Low Power	Vacuum-Cleaner	200 watts	TV	70 watts
			Fan	50 watts
	Shaver	15 watts	Computer	150 watts
High Power	Micro-wave	1000 watts	Air Cond-itioner	2000 watts
	Toaster	1500 watts		

Table 1. Classification of household appliances according to power and usage profile

optimize the distribution of electricity.

Apart from optimized electricity distribution, we also encounter some other issues related to power allocations. One such problem faced by a power supply system are the existence of *spikes*. The power supply to the power grid can increase or decrease any time depending on the availability of electricity generation sources. When the power supply drops the power grid managers are forced to shut down power supply to some areas. This abrupt unscheduled shutdown is damaging and an inconvenience for the customers and their appliances.

To plan a strategy for optimized power allocation and to handle spikes, we look at our consumer usage patterns. In a typical household we could divide electric devices into four broad categories. These four categories are shown in figure 1. The first category includes devices that are low powered and low usage. This means that these devices consume relatively less power i.e. less than 500 watts and are used seldom. In the second category are devices that are low powered but are used more frequently or for a longer duration of time e.g. electric fans, lights etc. The third category are devices that are high powered but are used seldom. Devices in these categories include things like microwave ovens, washing machines etc. Finally, the fourth category are devices that use more power i.e. more than 500 watts and are also used for a longer duration of time typically an hour or more. Devices such as refrigerators and air conditioners fall in this category.

Our hypothesis in this work is that if we somehow can optimize the use of the devices in the fourth category, we could eliminate or at least reduce the gap between supply and demand. In tropical countries, due to very long and hot summers air conditioners of different types makes up for most of the devices in this category. In this work we have managed the usage of air conditioners to optimize the distribution of electricity.

What this optimization entails is that a customer will get full electricity supply for type 1, 2 and 3. However, the air conditioners are regulated by the power company. The power company will be able to remotely switch 'off' the electricity to the air conditioners for short durations. This duration is small enough to retain the cooling effect produced by the air conditioners and long enough to save electricity at a grid station level.

But implementing such electricity optimization has many challenges. First, with the present infrastructure there is no way a power company can turn on or off the air conditioners remotely. But there is one hope and that is to control these devices by using a cell phone. This means that each air conditioner has a cell phone attached to it and the cell phone is able to report the electricity usage of the air conditioners and able to turn it on or off using SMS (Short Messaging Services). The technical details of the implementation of the communication setup shall be discussed in a future paper. There is always a question on the economic aspects of such a solution. We touch on the economic aspects and cost benefit analysis later in the paper.

Second, the number of air conditioners is enormous. In a typical locality thousands of these devices are present. Therefore, we need a technique of self-optimization that can scale to large number of devices without a significant cost overhead.

Third, both the electricity supply and its demand can vary, i.e. spikes can occur in our system. Therefore, the methodology must be dynamic enough to quickly act on supply and demand spikes and take the system back to an acceptable state.

Fourth, the methodology must ensure service-level guarantees i.e. the promise of the power company with the consumer that an air conditioner will not be turned off for more than x minutes in a given hour. This requires a very dynamic self-optimization technique.

In the next few sections we outline our proposed methodology and discuss our solutions to the aforementioned challenges i.e. scalability, spike handling and service-level guarantees.

3 Self-optimization Approach

Our methodology uses a sense-analyze-plan-execute (SAPE) cycle to do the power optimization. Furthermore, the modular architecture of SAPE provides crisp boundaries and input/output parameters. As is the case in our application, each of the four modules could be implemented using a different technique, technology and/or methodology.

The sense and execute phases interact with the systems to collect data and to act on the system, respectively. As sensing the data and executing a plan are application and technology specific, a methodology such as SAPE provide ample opportunity to adapt the same system to a variety of problems by simply changing the sense and/or execute modules. In our methodology the sense module collects data from the ACs and uses SMS as the communication channel.

The data collected from the *sense* is used by the *analyze* phase to see if the state of the system is going out of bounds.

If the system's state is indeed going out of bounds the plan phase is initiated to plan for electricity optimization.

The plan phase implements the algorithms to decide a new configuration of the system. For this purpose, we have used linear programming (LP) to decide new configurations of the system.

Determining which AC to turn off or turn on is a typical binary-programming problem. However, binary programming is NP-hard and no known solution is available to solve the problem efficiently. We also thought about integer programming but integer programming is also NP-hard. To solve this problem, we have devised a transformation. We cluster the data using an incremental clustering technique. We then use the frequency and mean of the clusters for LP. Our LP runs in real domain and our final solution is an optimal distribution with a bounded rounding error.

To meet with the challenges of our problem, we have devised three SAPE cycles in our methodology. The first SAPE cycle is a recurring SAPE cycle that works on an hourly basis and decide a configuration for the next one hour. This SAPE cycle uses historical usage data and builds up a plan for the hour that follows. However, during an hour two situations could occur in the system that requires a plan to be modified.

These two situations are an increase in the demand of electricity when supply is constant, and a decrease in supply when the demand is constant. For simplicity purposes we have not considered other situations such as where supply is going down and demand is going up simultaneously or situations where the supply is increasing or the demand is decreasing..

The other two SAPE cycles are designed specifically to handle these situations. The three SAPE cycles have many commonalities. Therefore, first we discuss the planned hourly optimizations SAPE cycle followed by discussion on spikes in demand and supply sides while identifying their similarities and differences.

4 Planned Optimization

Planned optimization is the SAPE cycle that runs on an hourly basis. A supply and demand graph is shown in figure 3. This is a typical supply and demand situation on a summer day for a locality. Assuming that we have such historical data available we plan for the electricity optimizations for the next hour.

The usage of electricity is a very dynamic. Therefore, in order to cater for any short and temporary spike we define a **reserve margin**[2] between the peak demand and supply. Reserve margin is a buffer between the maximum anticipate demand and the supply that is available to the system.

[2]Power companies usually have this reserve margin already but the way we have calculated the reserve margin is different than that of power companies.

[clusters] = clusterize(n)

```
sort (n)
k = 1
for i = 1..n
clusters[k].insert( n[i] )
if(variance(cluster[k] > errorThreshold))
   clusters[k].remove(i)
   k = k + 1
   cluster[k].insert (n[i])
```

(Where: n = Data points to be clustered; clusters = two dimensional array. each 1D array is a cluster; k = number of clusters)

Figure 1. Clustering Algorithm

This reserve margin is maintained to cater for any growth in demand beyond the supply. The motivation and way to calculate this reserve margin is described in section 5.2.

In our scheme, if we do need to replan the optimization of electricity this reserve margin provides the time necessary to replan the distribution of electricity.

For our methodology this margin is $margin \leq calcTime \times \Delta$. Where Δ is the slope when the global demand function approaches maximum supply and $calcTime$ is the maximum time taken to analyze, plan and execute the plan. The derivation of this equation is provided in the evaluation section. We subtract the margin from the electricity supply value and used the new number to plan for electricity optimization. The supply value minus the reserve margin gives what we call an **adjusted supply**.

4.1 Sense

'Sense' collects the data from the system. Three types of the data is required in our methodology. First, is the details of electricity supply from the electric producing companies. At a power grid station level this data is easily available. Alongwith the supply data our system also requires historical data of electricity usage of the last few days for the entire system.

The second type of data is the demand of electricity at any given time. Again this data is easily available at the grid station level alongwith its historical figures.

The third type of data and also the most critical data from our methodology's point of view is the data from the many air conditioners installed at customer locations. This data includes the type of the air conditioner, its electricity consumption rate for a given hour, whether it is in an 'on' state or an 'off' state. As we mentioned previously in this paper we are not going into the details of how we collect this data from individual air conditioners. However, our assumption is that we will have access to this data.

4.2 Analyze

We use the latest power usage profiles of individual air conditioners for our analysis. In this analysis we cluster the ACs on their power consumption profiles by using an incremental clustering technique[6].

To cluster we first sort the power profiles in incremental order. For each data point, we include it in the current cluster and then calculate the variance (σ^2) of the current cluster. If $\sigma^2 < errorThreshold$ then we proceed to the next data point. Otherwise we pull out the data point and create a new cluster for this data point. here threshold is a user defined parameter limiting the variance of a cluster. As our values are sorted prior to the clustering, we are always sure that our variance for each cluster will be less than threshold.

In order to make the best out of clustering we had to reduce the inter cluster variance. Therefore, to counter this problem we set our cutoff criterion for the clustering to restrict the variance (σ^2) of a cluster within a threshold. This means that each value in the cluster is in the range of $\mu \pm errorThreshold$. Here μ is the mean of a specific cluster.

To cluster we could also have used another clustering algorithm such as k-means. But k-means limits of number of clusters (k) where our requirements was to stabilize the system with respect to the inter-cluster variance and the number of clusters is not important.

The output of the analysis phase hence will be the air-conditioners usage information divided into clusters. This information is then used to plan the actual optimizations.

4.3 Planning

The input to the planning module is the cluster mean and frequency data and the electricity power supply available. In this phase we use LP to plan shutdowns in a pre-defined scheme.

Linear Programming (LP) is a well known technique to plan resource allocation for competing demands where the supply is scarce [6]. The system is modeled as a series of linear equations. These equations define the whole system including the constraints, cost-functions and decision variables. These equations are usually derived from the technical considerations as well as logical requirements.

We have two logical constraints and one technical constraints in our system. First logical constraint is that the total energy consumed, as planned by the LP, should not exceed the supply (precisely the total supply minus the reserve margin). This is represented as equation 3. The second constraint is that the number of machines to be powered "on" in each cluster for each time period should at least be $x\%$ of the total number of machines in that cluster. Where x is the minimum service level guarantee for the specific cluster . For example if the guarantee for the system is that no machine will be off for more than 20 minutes (or 33% of the time). Then the constraint will be that 33% of the consumers for each cluster shall be powered "on" in each cycle to ensure the 20 minutes guarantee(equation 2). The technical constraint is that the number of machines powered on in each cycle should not exceed the number of machines in that cluster as given in equation 4.

Figure 2 defines the set of equations to define LP. The cost function (Eq. 1) maximizes the total number of machines in each cluster for all time periods. Here $X_{i,t}$ represents the i^{th} cluster in t^{th} time period. As we do not have any priority, for clusters all machines have equal chance of getting selected. Our z will give us the total number of 'on' machines and value for X_{it}will give us the number of machines to switch in i^{th}cluster at the t^{th} time period. Equation 2 represents the service level guarantee constraints that in every time period, at-least 1/3rd of systems should be in powered on state. equation 3 limits the allocation under the maximum available supply and equation 4 puts the technical constraint that the number of allocated consumers in a cluster should not exceed the cluster size.

For air conditioners, it is recommended that a delay of 10 minutes should be given between power cycle of an AC. To cater this requirement, we divide our total time (1 hour) into 6 chunk of 10 minutes each. A cluster for each 10 minute time period is represented as a decision variable. So if we have 100 clusters and t is 6 then we will have 600 decision variables, each variable defining the number of consumer that should get power in that 10 minute period. The optimization function is to maximize the number of consumer getting the supply for the entire duration. In should be noted here that for other problems, the t can be increased or decreased accordingly.

Linear programming assumes real values for all variables. This means that LP outputs non-integer values for state frequencies for each cluster. For example, LP can output 5.2 ACs to be in "on" state in time t. We will take the *floor* of the cluster values. For our problem this adds an element of error. However this error will be order of (kt) where k are the number of clusters and t is the number of time periods. A perfect solution, maybe by integer programming, will allocate power to some of these kt machines. By clustering and rounding off all of these kt values we might be under allocating the resource. However in our experiments we observed that our maximum value for k is approximately 1% of total number of machines (n) and as our $t = 6$, our total missed allocation will not exceed 6%.

The plan given by LP thus consists of the machines to be turned off in each ten minute period of an hour in each cluster. Therefore, this is an hourly optimization plan to be implemented every hour. This plan is provided to the execution module as input.

4.4 Execution

Execution phase implements the plan developed in the planning phase. The execution phase implements a plan in two steps. The plan only states the number of machines to be turned off in a given ten minute period of time without pointing to the specific machines. Therefore, the first step

$$Maximize(Z = \sum_{i,t} X_{i,t}) \quad (1)$$

$$\forall_t \forall_i X_{i,t} \geq MAX_i/3 \quad (2)$$

$$\sum_{i,t} \mu_i X_{i,t} \leq supply \quad (3)$$

$$\forall_{i,t} X_{i,t} \leq MAX_i \quad (4)$$

Figure 2. Hourly planning LP equations

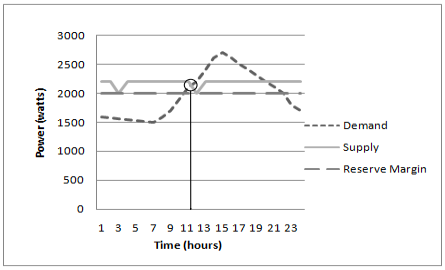

Figure 3. Typical Supply spike in system

is to apply a fair policy-based mechanism to transform the cluster frequency output of the plan to discrete on/off values for each time period. The second step is to propagate this information to individual air conditioners. Again this information in our proposed methodology is communicated through SMS. The cell phone attached with individual ACs performs the switching on or switching off of ACs.

5 Spike Handling

A spike is a sudden upward or downward surge in supply or demand. Figure 3 defines a typical power usage pattern at a grid level. The power provider guarantees a 2200 KW power supply. But this supply could drop arbitrarily. The fall in power supply when demand is low does not effect the system much. But when demand matches or exceeds supply, like in the 11th time period create problems.

A demand spike occurs when a predicted maximum load is crossed. Figure 6 shows the predicted and actual load of a system in real time. The maximum predicted load was 2600 KW. Four hundred kilowatts was a reserve margin. However, still the demand outstrips supply.

A downward spike in demand or an upward spike in supply does not effect our system. For the sake of simplicity we do not handle these two types of spikes.

To deal with a upward spike in demand and a downward spike in supply we use two SAPE cycles, respectively. Some of the modules like the sense and analyze are used almost in a similar fashion but the analyze and plan are designed differently.

5.1 Supply-side Spike

Supply side spike occurs when the power supply company faces a sudden loss of a power generation source such as a unit in a thermal power plant. This kind of spike occurs almost instantly. However, sometimes there is a margin of

$$Minimize(Z = \sum_{i,t} \mu_i \times X_{i,t}) \quad (5)$$

$$\forall_t \forall_i X_{i,t} \geq MAX_i/3 \quad (6)$$

$$\sum_{i,t} \mu_i X_{i,t} \leq supplyNew \quad (7)$$

$$\forall_{i,t} X_{i,t} \leq MAX_i \quad (8)$$

$$\forall_{t':t'<currenttime} \forall_i X_{i,t'} = allocX_{i,t} \quad (9)$$

Figure 4. Spike handling LP equations

a few seconds. We assume that we use these few seconds to handle the spike.

As soon as the system sense a supply side spike we initiate a replanning process.

In the *analyze* phase, we assume that our initial prediction and clustering was correct.

However, in the *plan* phase we use a proactive approach. We calculate the minimum threshold power that is needed in every 10 minute time period without violating the service guarantee. We calculate this immediately after calculating the hourly plan. That is, at the start of each hour, in addition to the main plan, we have 5 additional plans for failure at the 10^{th}, 20^{th}, 30^{th}, 40^{th}, and 50^{th} minutes.

The planning for each 10 minute period uses the LP in figure 4. Here we find the plan which minimizes the possible power supply without violating the service level agreements (eq. 5).

Assume that the spike occurs at time t'. As at time t' all the power allocations for time slices before t' must have already been implemented, we consider the values for decision variables for timings before t' as constant. This is represented as equation 9. Our demand constraint and guarantee constraint remain the same (equations 6 and 8) . Our optimization function is to choose a set of decision variables for which the total power needed is minimum (equation 5).

If at time t' the power does go below the amount promised at the start of the hour, we have a plan ready which can be propagated to the system instantaneously. We assume here that our communication network is fast and stable enough to propagate the new plan. As mentioned previously *execution* phase is implemented using the same communication network i.e. SMS.

Since our planning is at discrete time intervals, we revert the system to a plan older than current time. For example, if a drop in supply occurs in the $24th$ minute then we will revert to the plan 2 made for $t = 20$ as shown in figure 5.

The decision to revert or to move forward is a trade off between fairness and guarantee constraints. Reversion makes sure that the guarantees are met. However reversion can be unfair in distribution of power within a cluster. For example, if in the same scenario, we use the future plan then an AC with power supply "on" at the $24th$ minute will be considered as having electricity for the entire 20-30 minutes period for our calculation. This means that although the

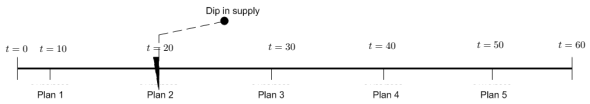

Figure 5. System response for spike at $20 < t < 30$

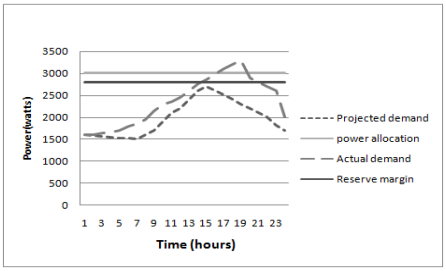

Figure 6. Typical demand spike in system

AC got only 4 minutes of AC, the system counts it as 10 minutes. If this machine had already gotten power before then LP will consider the guarantee for this machine met. However the AC should get atleast 6 more minutes of power for guarantee. A reversion on the other hand will not count the 4 minutes provided to the AC which is in "on" state. This means that a LP will not count this user's consumption and will allocate power again if the guarantee is not met. What this means is that this user gets 24 minutes of power where as those who had their machines in "off" state at the $24th$ minute will get only 20 minutes of power. However for us, satisfying service level guarantee is more important than fairness hence when a drop in supply occurs we revert to the passed plan as shown in figure 5.

5.2 Demand-side Spike

Demand side spike occurs when the demand from the consumers outstrip the projected demand calculated at the start of an hour. Figure 6 shows a typical hot summer day when the real demand outstrips than the projected demand. We are calculating demand on per home basis. The sum of our demands give us a global picture of how much energy is needed.

One thing to note here is that unlike supply-side spike demand-side spike almost always grow smoothly. This give us breathing space to deal with this kind of spike in a much robust way. Therefore, we use a reactive method in this approach.

In the *sense* phase we have the data of the aggregate demand as well as the usage profile data from individual air-conditioners. Assuming that we use a pull model to get the data from the air conditioners we have this data whenever necessary.

Analyze for demand-side spike is a two step process. First, we evaluate that how fast is our demand approaching our supply since it is non-deterministic. This means that an increase in demand may not be a continuous increase in

$$reserveMargin \leq \Delta \times CalcTime \quad (10)$$

Figure 7. Reserve margin lower limit

demand in which case we would like wait and see if a re-planning is really required. To do this we take the demand data at every minute and use a linear regression to find a fit for our global demand. We then use relation defined in equation 10. In this relation Δ is the slope of our linear regression fit and $calcTime$ is the time to cluster, plan and propagate the updated plan. If the relationship does not hold then we proceed to step two of analysis.

This relationship is the trigger to a replanning. As we measure the rate of growth and the current state and the relate it with the time it takes for us to react, we ensure that we always have a plan ready for an eventuality.

We start our step two with pulling the data from all the AC units. A demand different from the projected demand requires a re-clustering of the air conditioners usage data. To do a better job at clustering we use the historical data of the air conditioners and take a max of the two pieces of data for each AC. Here we use the same incremental clustering algorithm with similar threshold.

In the *plan* phase we use an LP which borrows from the two LPs discussed so far. As our goal here is the same as that of program in figure 2, that is to maximize the number of users, we use the optimization function from that LP (equation 1). The constraints equations will be mathematically same as those in figure 4. However, we will change the constants differently now. Since in this scenario our mean(μ) for clusters has changed and the supply is same as before, our LP will have a changed left hand side (μ) instead of right hand side($supply$). As these values are input parameters, we will not need to change the actual LP equations and only changing the constants will be sufficient. We can also interpolate that the running time and complexity will be the same for both the LP.

The *execute* phase is similar to the execute phase of other SAPE cycle.

6 Evaluation

We have evaluated our methodology on the challenges mentioned in section 2. These challenges are scalability, spike handling and ensuring service-level guarantees. The equations of the linear programming has already proven the effectiveness of the technique in ensuring supply-side guarantees. Therefore, in this section we discuss the simulation results that proves the scalability of our approach and a mathematical derivation that proves that our approach is effective in handling spikes.

Moreover, in the SAPE cycle since plan and execute are tightly tied with the way we receive and execute the commands using cell phones. Therefore, we have not evaluated the sense and analyze phases of our proposed SAPE

methodology in this evaluation. Instead we have concentrated on analyze and plan phase.

We used two different methodologies to evaluate our systems and support our hypothesis. We first conducted simulations with varied data sets and sizes to test the scalability and correctness of our system. As our spike handling uses the same clustering and a similar LP, we used mathematical derivation to evaluate and prove the correctness and resilience of our spike handler since the scalability and correctness of clustering and LP has already been shown.

We start with discussion of our complexity of clustering and LP with respect to scalability. In section 6.1 we evaluate our test results for scalability of the system. We prove that our running time for an increase in number of controlled ACs can be managed thus answering the scalability question. In section 6.2 we prove the correctness of spikes mitigation system. This will give us the answer for the effectively handling of un-scheduled updates question.

6.1 Is the Solution Scalable?

Since we are controlling individual ACs in this methodology the shear number of the ACs require a very scalable solution. This also means that in no way we can afford a non-polynomial solution.

The scalability of our methodology is dependent in most part on the analyze and plan phases. In analyze we have used an incremental clustering technique. Our algorithm tries to insert each element in one cluster only, and then calculates the variance of that cluster. that is our complexity is dependent on the length of the largest cluster l, the number of clusters k and the number of elements n. The order of this incremental clustering algorithm is $O(nkl^2+n(log(n))$ where $n(log(n))$ is the complexity of sorting. A point to note here is that the number of cluster and size of clusters are very closely related with the error threshold and the variance of the distribution. We discuss this relationship and its effects shortly.

In the plan phase we use a linear programming (LP) algorithm. The complexity of LP is dependent on the input number of decision variables. Our LP will decide the number of machines that should be on for each cluster. We used Matlab's LIPSOL algorithm [17]. This algorithm is based on Mehrotra's Predictor-corrector interior point algorithm [11]. Complexity of a modification of this algorithms in big-O terms was calculated by Salahi and colleagues and was found to be $O(k^2L)$ [14]. Here k is the number of variables and L is the length of string needed to encode the input. Recall k is the number of clusters that we created in previous step.

A grid station typically supports a few thousand consumers. As the consumption of individual ACs could vary we used random data generated using an upper and lower bound to mimic actual AC power consumption. Since consumption of electricity can be considered a natural phenomenon and our sample size is large enough, use of normally distributed data sufficiently models our system.

Variance	Size	Cluster count	Clustering Time	LP Time	Total Time
1	10,000	7	1.96	0.06	2.02
	50,000	7	41.76	0.05	41.71
	100,000	6	151.8	0.09	151.89
10	10,000	64	.76	0.16	.92
	50,000	76	6.48	0.21	6.69
	100,000	80	21.49	0.23	21.72
50	10,000	304	0.5	0.37	0.87
	50,000	348	3.8	0.56	4.36
	100,000	372	9.12	0.88	10

Table 2. Total time for analyze/plan (Error threshold = .1)

The scalability is affected by three variables: 1) the number of AC usage profiles i.e. n, 2) the variance σ^2 of the usage profiles, and 3) the threshold e within a cluster the threshold we will tolerate within a cluster.

To evaluate the system we considered AC usage profiles between 10,000 and 100,000. We used data with a variance between 1 kilowatt and 50 kilowatt.

Finally, we varied the threshold within a given cluster between 0 .01 and 0.1.

For the aforementioned variables the clustering results with an error threshold of 0.1 are given in table 2 and with an error threshold of 0.01 are given in table 3.

It can be observed that for a typical 50,000 values and error threshold of 0.01, our clustering time is less than a minute. In fact our worst time for clustering (151.8 seconds) was with the lowest variance (1) of data and with the highest error threshold (0.1).

An interesting observation to note here is that a higher error threshold or a higher variance, or both reduces the running time of the algorithm. This is almost counter-intuitive. We see that as the variance of the system increases from 1 to 50, the time actually reduces. Similarly the time for $e = .1$ for $n = 500,000$ is 151 seconds. But for $e = .01$ and same n our clustering time is 58 seconds, a three fold reduction. However, the answer to this could be explained by the clustering algorithm. Every time we insert an element in a cluster, we run a l^2algorithm where l is the size of the cluster. If all our elements are in a single cluster then $l \rightarrow n$. As a result our order could be $O(n \times n^2) = O(n^3)$. However, if our k is large, then for a normal distribution each cluster will have fewer elements. That is l in general will be small compared to n. A larger $\sigma^2$and a smaller e increases k yet reduces our time.

Our observation is that clustering infact takes the major chunk of time as time to calculate is extremely small. For example with a value of $k = 1045$ the plan is calculated in

Variance	Size	Cluster Count	Clustering Time	LP Time	Total Time
1	10,000	23	1.12	0.12	1.24
	50,000	24	14.5	0.09	14.59
	100,000	26	58	0.11	0.69
10	10,000	201	0.56	0.43	0.99
	50,000	220	4.2	0.34	4.54
	100,000	234	11.2	0.43	11.63
50	10,000	857	0.5	1.40	1.9
	50,000	1004	3.86	1.45	5.31
	100,000	1045	8	2.04	10.04

Table 3. Total time for analyze/plan(Error Threshold=.01)

just 2.1 seconds. The total time required for both clustering and LP is given in tables 2 and 3. It can be seen that LP adds a fraction of time to the total calculations. In contrast an integer programming solution takes close to 30 minutes for a dataset of size 30!

6.2 Is the Solution Effective in Handling Spikes?

As discussed previously, our system will encounter two types of spikes. We discuss the effectiveness and response time for each source individually. Since our demand spike only uses an LP which is similar to the one discussed in previous section. And our demand side spike uses the same clustering algorithm as discussed previously, we do not re-evaluate our clustering or LP scalability here. Instead our focus in this section will be to prove that the spike handling mechanism is robust and able to maintain the guarantees, if possible, in real time.

6.2.1 Supply side spike

A sudden dip in supply is very much a possibility and is inevitable. Since our system requirements stipulate that a machine shut down should not be restarted in the next 10 minutes hence any change in supply is immediately transferred to the customers. For the next cycle, however, due to our proactive approach we already have a plan ready and this plan will simply replace the current plan. Hence we will seamlessly integrate the change. It might be possible that the dip in power in later stages of an hour might give us a situation where reaching a guarantee is not possible. This case is a policy decision and is beyond the scope of our work. In case the system managers would like to lower their guarantee with some penalty then such a plan can also be calculated apriori and enforced in such a situation.

In a nutshell, at the time of supply side spike the system goes down to guaranteeing minimum service level. That is the system will try to conserve energy and only ration 20 minutes of usage per AC. However if the power dip does not reach our minimum level, we can always recalculate the plan using LP in figure 2 for next time period and update the plan like wise. Since maximum time to calculate a plan is less than 4 minutes, and our time period is 10 minutes, we have the ability to update a plan if needed.

6.2.2 Demand side spike

As discussed previously demand side spike is a result of growth of consumption beyond predictions. A point to note is that the demand growth over time is not as drastic as the supply side spike. Especially when considering that we have tens of thousands of consumers. In order to predict demand side spikes we need to find out when the SAPE process should start when demand changes. Another related aspect is the reserve margin that we need to subtract from the supply to get the adjusted supply.

For the trigger mechanism we must ensure that we have enough time to run the SAPE process before the demand is at an unacceptable level. Planning is not possible as the growth in demand is quite non-deterministic. Our limitation is that we need to start our process $calcTime$ seconds before we reach our max supply point. We derive a formula for the trigger as follows:

Let $calcTime$ be the time to analyze plan and execute our SAPE cycle. Therefore, if we want to have enough margin so that we trigger our calculations before we overrun our supply then our trigger will be:

$$t' - t \leq calcTime \tag{11}$$

Where t is the current time and $t'$is that time when, according to current estimates, our demand will reach our supply.

We do a regression analysis for values of demand in previous five minutes to approximate the movement of our demand. The regression analysis gives us the equation:

$$currentdemand = \Delta t + c \tag{12}$$

we can say that t'can be given as:

$$supply = \Delta t' + c \tag{13}$$

replacing values in 11 with equations 12 and 13, we get

$$\frac{supply - currentDemand}{\Delta} \leq calcTime \tag{14}$$

Simplifying we get our trigger as:

$$currentDemand \geq supply - \Delta \times calcTime \tag{15}$$

That is, when our $currentDemand$ goes below $supply - \Delta \times calcTime$ we will initiate our demand spike planning module.

The second value that we need to know is an acceptable reserve margin. Recall from an earlier section that we introduced lower limit of margin as $margin \leq calcTime \times \Delta$

. If our predictions were correct then at the time of peak usage we will not trigger a spike mitigation if our $margin$is greater than or equal to $calcTime \times \Delta$.

To ensure that at peak time, our demand spike mechanism is not started when we are following our demand pattern, we put a lower limit to our margin.

$$reserveMargin \geq f(t') - f(t) \tag{16}$$

where$f(t)$is the maximum demand and $f(t')$is the hypothetical demand at time t'. If the demand would have continued based on regression analysis for a window from five minutes before maximum demand till the maximum demand. We use regression analysis on the predicted demand function to find $f(x)$. The equation given by regression analysis is:

$$f(x) = demand(x) = \Delta x + c \tag{17}$$

we can hence write equation 16 as

$$reserveMargin \geq \Delta(t' - t) \tag{18}$$

We know that $(t' - t)$ is bounded by $calcTime$. For calculating margin limit, we use the lower bound for $calcTime$hence we can say that.

$$reserveMargin \geq \Delta \times calcTime \tag{19}$$

This gives us the lower bound for the $reserveMargin$.

Using these mathematical proofs we can say with confidence that as long as our $reserveMargin$ is more than our $\Delta \times calcTime$ and our trigger is calculated regularly and we will always have enough time to calculate a new solution for an increasing demand.

7 Discussion

In this section we discuss some of the observations and frequently asked question about our methodology.

Is the solution cost effective? We have calculated that to control each air-conditioner remotely we need equipment worth about $30 including relays, GSM cell phone, circuits, wiring, etc. The cost of running the system is also quite low because many GSM companies provides packages of unlimited SMS in many countries for just $2 a month.

Who will pay the cost/ who will be responsible for setup? We envision that our system will be adopted by townships and/or power distribution companies. A city council or a power providing company will decide on setting up a system. In such a case, either the power company or the city council will enforce such a system and will devise a cost breakdown mechanism. Such adaptation will have legal and social implications however this discussion is beyond the scope of this paper. Legal changes will be required to enforce such a system such as, made by Colorado and California states in USA.

What is the learning curve of applying this technique? Our technique uses basic clustering and mathematics. The modeling framework required to adopt a new problem is much less than many other proposed techniques [4, 2, 1, 8, 16]. There are open source tools available (GNU linear Programming Kit) available for solving LP. Indeed the methodology we have proposed is not only simple but is so commonly used by economists, managers, engineers and planners.

8 Related work

Although we started our work before the publication of work of Keeton and colleagues, their building of argument for using operations research techniques strengthens our hypothesis [7].

Mathematical optimization techniques have been used previously. Diao and colleagues used Nelder-Mead Simplex method [4]. Aleida and colleagues used a heuristics based technique [2]. However, both these methods are heavy to calculate, have a sharp learning curve, and do not guarantee a global optimum value. In contrast our technique is much faster, requires basic mathematical modeling training and further more guarantees a global maximal solution.

Artificial Intelligence (AI) techniques have also been applied for self-optimization. These techniques use heuristic and stochastic processes to "guess" a starting point and start a search. However, AI techniques are known to get trapped in local extremum [13]. Some issues with heuristic-based techniques were also pointed out by Sirer [15]. For instance, SORA is proposed by Mainland et al. [10]. SORA uses re-enforced learning. The method however follows greedy paradigm and decision making is done on node level without a view of the global demand and supply. This can result in maximal solution for individual nodes. But for problems not exhibiting sub-problem optimality property, the real optimum value cannot be guaranteed.

To our knowledge, our work is the first attempt at using LP for time variate decision problem. Although Femal and Freeh used LP to derive an optimal solution for boosting performance [5], but it can be shown that a greedy technique would have been more suitable to find the solution.

There are many greedy or dynamic programming techniques used for self-optimizations. However, these techniques fall short when the dimensions of input and output domain increases [9, 12].

Apart from operations research and mathematical optimizations, researchers have also used control theory to provide predict and plan for optimality. Contribution from Abdelwahed and colleagues used control theory to predict the workload on a distributed web-server [1]. Wang et al. and Lefurgy et al. used control theory to optimize such a system. Lefurgy optimizes by using low-level power management [8]. Wang attempts to solve a similar problem but

through tampering with the operational frequency of the processor [16]. These techniques, though workable for the sample problem, are not proven to be scalable to hyper-dimensional input and output space.

9 Conclusion and Future Work

We have presented an approach to manage the electricity management using techniques from operation research. We believe that this technique could be applied to the domains as well. In addition we have shown a near optimal solution to otherwise an NP-hard problem.

This approach could be improved in a number of ways. First, we will look at situations where the demand and supply are changing at the same time. Moreover, we will also look at techniques where in case of an increase in supply or a decrease in demand we can transfer the benefits to the consumers.

Second, we have looked at only one type of machine i.e. airconditioner in this approach. Our future work includes identifying more such devices and/or opportunities that we could use to manage electricity further.

Third, we have looked at only a single level of service guarantee in this system. We will be looking at the ways in which we could accommodate multiple levels of service guarantees.

In a way this paper focuses on demonstrating the effectiveness of the analyze and plan phases of our approach. Therefore, a very important aspect of the success of this approach is to ensure the integration of the sense and analyze phases.

Another interesting dimension is to apply this technique to software configuration such as Moodle and Apache. However, a big challenge is to represent large configuration spaces in a set of linear equations.

References

[1] S. Abdelwahed, N. Kandasamy, and S. Neema. A control-based framework for self-managing distributed computing systems. In *WOSS '04: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, pages 3–7, New York, NY, USA, 2004. ACM Press.

[2] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian. Resource management in the autonomic service-oriented architecture. *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pages 84–92, 13-16 June 2006.

[3] N. Arshad, D. Heimbigner, and A. L. Wolf. A planning based approach to failure recovery in distributed systems. In *Proceedings of the ACM SIGSOFT International Workshop on Self-Managed Systems (WOSS'04)*. ACM Press, Oct./Nov. 2004.

[4] Y. Diao, F. Eskesen, S. Froehlich, J. L. Hellerstein, L. F. Spainhower, and M. Surendra. Generic online optimization of multiple configuration parameters with application to a database server. In *Distributed Systems, Operations and Management (DSOM)*, volume 2867/2004 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003.

[5] M. Femal and V. Freeh. Boosting data center performance through non-uniform power allocation. *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pages 250–261, 13-16 June 2005.

[6] F. Hillier and G. Lieberman. *Introduction to operations research*. McGraw-Hill, 2001.

[7] K. Keeton, T. Kelly, A. Merchant, C. Santos, J. Wiener, X. Zhu, and D. Beyer. Don't settle for less than the best: use optimization to make decisions. In *HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[8] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, pages 4–4, 11-15 June 2007.

[9] Y. Li, K. Sun, J. Qiu, and Y. Chen. Self-reconfiguration of service-based systems: a case study for service level agreements and resource optimization. *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 266–273 vol.1, July 2005.

[10] G. Mainland, D. C. Parkes, and M. Welsh. Decentralized, adaptive resource allocation for sensor networks. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 315–328, Berkeley, CA, USA, 2005. USENIX Association.

[11] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[12] R. Nathuji, C. Isci, and E. Gorbatov. Exploiting platform heterogeneity for power efficient data centers. *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, pages 5–5, 11-15 June 2007.

[13] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[14] M. Salahi, J. Peng, and T. Terlaky. On mehrotra-type predictor-corrector algorithms. *SIAM J. on Optimization*, 18(4):1377–1397, 2007.

[15] E. G. Sirer. Heuristics considered harmful: Using mathematical optimization for resource management in distributed systems. *IEEE Intelligent Systems, Special Issue on Self-Management through Self-Organization in Information Systems*, 21(2):55–57, April 2006.

[16] M. Wang, N. Kandasamy, A. Guez, and M. Kam. Adaptive performance control of computing systems via distributed cooperative control: Application to power management in computing clusters. *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pages 165–174, 13-16 June 2006.

[17] Y. Zhang. Solving large-scale linear programs by interior-point methods under the matlab environment, 1997.