2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems

AdOpt: An Adaptive Optimization Framework for Large-scale Power Distribution Systems

Fahad Javed
Department of Computer Science
LUMS School of Science and Engineering
Sector U, DHA
Lahore, Pakistan
fahadjaved (at) lums.edu.pk

Naveed Arshad
Department of Computer Science
LUMS School of Science and Engineering
Sector U, DHA
Lahore, Pakistan
http://na.lums.edu.pk

***Abstract*—Optimizing self-evolving and dynamically changing systems is a grand challenge. In order to apply optimizations almost all conventional optimization techniques require a runtime system model. However, system models and their solution techniques vary in their strengths and limitations. For a rigid system, a single system model is acceptable. But if the system is constantly changing its structure then a rigid model is not able to represent the system properly, resulting in an inefficient use of technique in some cases. Therefore, in this paper we propose a framework for an optimization engine that adapts the optimization technique based on the system state. The adaptation involves selection of techniques based on historical statistics and current data, and dynamic generation of a model at runtime. This runtime model is then used to apply a relevant optimization technique to find a desired optimization plan for the system. We have evaluated the proposed framework on an electricity distribution system. Our results show that the proposed framework is adaptable, fast and able to manage numerous situations.**

I. Introduction

Self-evolving systems, such as peer to peer networks, distributed large-scale computing services, resource management provisioning for server farms, computing in the cloud, etc. are systems that by definition change their dynamics non-deterministically. One of the possible ways to incorporate self-* behavior in these systems is to use an extended control and feedback loop such as SAPE (Sense, Analyze, Plan, Execute) [4]. SAPE in its original form uses an abstracted model of the system in these four phases to incorporate self-* behavior. However, the runtime behavior of many systems change over time and self-* techniques that use one model have applicability only in some selected situations.

This limitation stems from the fact that an abstracted model of the system is developed to apply, for example, a given optimization technique suited for that particular model. Usually this model of a system allows a few observations to be updated at runtime but overall the model is quite rigid.

On the other side each optimization technique has its strengths and weaknesses. But because the runtime model is fixed, the autonomic system has to rely on the results of a single optimization technique available to solve a given scenario. No doubt that the optimization technique caters for the most common cases for optimization. However, in technologies such as cloud computing and peer to peer systems, the “common" scenario is not very common and the "unimaginable" happens more often than accounted for.

Therefore, we believe that to improve the autonomicity of the system we have to incorporate multiple optimization mechanisms. However, in order to incorporate multiple optimization mechanisms we also need more than one runtime model of the system. To this end, in this paper, we propose an “adaptive self-optimization approach” which uses multiple optimization techniques to incorporate a resilient self-optimization in a given system.

In this approach whenever there is a requirement to optimize a given system in a SAPE cycle, our planner does not directly apply a specific optimization technique. Instead it first finds an appropriate optimization technique that is best suited for the given state of the system. Once it selects the optimization technique it generates a runtime model of the system. Thereafter an optimization algorithm is applied to get a plan for the optimal configuration.

II. Motivation

Our main motivation for adaptive self-optimization was based on optimization requirements of an electric power distribution system [11]. A typical power distribution system provides electricity to a locality which consists of tens of thousands of consumers. However, due to power crises in developing countries if the demand of power outstrips supply, the power company cuts off complete power to one or more neighborhoods to keep supply more than demand. To ameliorate this situation in our previous work we proposed a scheme to regulate power distribution based on micro-managing heavy duty electric appliances such as air conditioners and electric water heaters [11]. According to the scheme the control of heavy duty appliances is in the hands of the power company while other low consumption devices are given uninterrupted power. To have fairness, this scheme provided a service-level guarantee to each household. Since such a system has to keep up with the demand and supply pattern of electricity and also has to ensure service-level guarantees for hundreds of thousands of heavy duty electric appliances we used a linear

978-0-7695-3794-8/09 $26.00 © 2009 IEEE
DOI 10.1109/SASO.2009.26

programming model of the system to apply self-optimization on this system. The use of linear programming in self-optimization problems could be complex depending on the dynamics of such a system.

For example, in the most basic form, applying an optimization approach, such as linear programming, requires a fixed set of machines. In this problem a fixed set of linear equations is enough to compute an optimal solution. This set of equations is not required to be updated because the set of entities is fixed and assumed to be present in the system at all times [10].

A linear system of equations is developed based on the entities in the system. But like the power distribution system where consumers are turning on and off their devices, a fixed set of linear equations is not enough. In order to solve such a problem we used a meta-model to generate a set of linear equations at runtime. This set of linear equations is mostly based on the state of the system i.e. the number of electric devices consuming power at the time of optimization. Once the linear system of equations is generated an optimization algorithm such as 'simplex' is used to solve the problem [9]. This approach worked well until the entities which make up the system are small in number.

However, when this number grows beyond say five thousand then generating the linear system of equations and its solution finding using simplex becomes very time consuming. To counter this problem, in our previous work, we used a clustering technique to cluster the entities based on a given variance. The clusters are then used to generate equations and thus a relatively small set of equations was generated.

In solving the power distribution problem, the use of linear programming proved to be considerably fast for even an ultra-large data-set consisting of up to one hundred thousand entities. However, since we were using clusters the solution speed had a penalty of unutilized power that could have been utilized. When a solution is required instantly or is required for a large data-set a small quantity of unutilized power is acceptable. But when the need for distributing the load scaled up or down sharply, we found that we could do a better job with an algorithm that solves the problem and virtually leaves no unutilized power in the system. However, changing algorithms, at runtime is not possible because other algorithms could only be applied if the system is abstracted in a different model. This means that in order to use multiple optimization algorithms the runtime models have to be generated at runtime too.

Therefore, to use multiple optimization algorithms we need a methodology that analyzes the state of the system, recommends an optimization algorithm, generates a runtime model of the system and uses an optimization algorithm to produce a plan for the power distribution.

To achive the aforementioned goal we developed a framework called "AdOpt" short for adaptive optimizations and we describe this framework in this paper.

III. Approach

To realize the goal of using multiple optimizations on a given system, a whole autonomic framework is required. The very basic building blocks of this framework are the optimization algorithms that will eventually optimize the system. The selection of optimization algorithm is partly dependent on the optimization problem in the system and its eventual goals. In this section, we describe the optimization algorithms that we selected to solve our optimization problem.

However, before delving into the details of the algorithms we would like to explain a bit more the optimization scheme of the power distribution problem we are solving. In a nutshell, our optimization scheme turns 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. An hour of usage for each high powered device is divided into six, ten minutes slots. At times when supply is sufficient for demand then all devices are powered for all the six time slots. However, as the demand outstrips supply devices are turned off based on a fair scheme such as round robin. The maximum a device can be turned off is based on a service-level guarantee between the electric company and the consumer. For simplicity purposes, in this paper, we use a two slots service guarantee for all the consumers. This means that a device is to be turned on for at least twenty minutes of an hour. Therefore, the optimization goal is to find a plan for the next hour for each device in the system.

Since a plan is generated for each hour there is no need to recalculate the plan during the course of the hour unless two situations occur: there is a sharp increase in the demand or there is a sharp decrease in the supply. In both cases, the plan has to be recalculated for the rest of the hour. Some relevant information on how the hourly or spike-time plan is discussed in this paper but a detailed account of how it is used and evaluated is discussed elsewhere [11].

A. Overview of Self-optimizing Techniques

We used three optimization methods for calculating a plan for the optimization of electricity. In this section, we describe the scalability and applicability of the three optimization methods .

1) Binary Programming: In our optimization problem we need to find a plan of whether to turn on or turn off an electric device. Binary programming (BP) is an ideal solution to this problem because each device has only an 'on' or an 'off' state during a slot [9]. The advantage of BP is that it gives an exact solution and does not have any rounding off error. This means that there is no unutilized power in the distribution system if BP is used. However, on the downside the running time for BP degrades exponentially as more devices are added into the system. Therefore, BP could only be used if the system has a small number of devices.

BP is a known NP-hard problem. To find the optimal solution takes the problem a bit further and makes it a $\sum^2$ (Sigma2) problem, a class of problems known to be more complex than NP. There are no polynomial time algorithm to solve these problems. The only known way is to enumerate all the possible solutions. However, applying combination of state of the art branch and bound techniques and linear programming can solve small sized problems in a relatively short time.

The formulation of BP encodes each machine-time slot tuple as a single variable. BP decides the state for each tuple subject to the service-level guarantee and the supply and demand constraints. Therefore, the solution provides the state of each machine in the system for each time slot. The formulation of the runtime system model to represent our power distribution system is discussed in next section.

2) Linear Programming: Linear programming (LP) is used to solve the optimization problem of power distribution system. We used two algorithms in linear programming: the simplex method, and the interior point method. As mentioned before, to solve the system using linear programming so as to allow for a large number of machines, we clustered the data based on a distance threshold of the power consumption profile. The clusters are then used to generate the equations that represents the system from a meta-model. Once the equations are generated one of the two aforementioned methods to solve the linear system of equations is used to find a plan for electricity optimization. Because of the inherent modeling of the system both the simplex method and interior point methods produce an error margin i.e. unutilized power.

The main differentiating factor between simplex and interior point is the underlying method of optimization. Simplex traverses along the edges of dimensions and changes its direction when a constraint is encountered. This is slower but in general allows a variable to reach it's maximum point in terms of optimality before optimizing other variables.

Interior point method is the state of the art LP solving method. Interior point method traverses the interior of valid region in search for optimal point. In doing so it changes values for a larger number of variables at the same time. Hence at the end of the optimization, for a degenerate problem, there is a possibility that more variables contribute marginally to the optimal point in comparison to simplex solved solution. A degenerate solution is when more than one combination of values yield the same optimal value.

In short, simplex has less unutilized power but is slower than interior point. On the other hand interior point is fast but could give a larger unutilized power at the end.

B. Runtime Modeler

As mentioned previously, to solve a given optimization problem, we need to represent the system in the form of a system of equations. This system of equations act as an input into a mathematical solver that uses one of the optimization algorithm to solve the given system of equations. Traditionally, representing a system in terms of system of equations is performed at design time and a few variables are updated to represent the runtime state [10]. However, in systems where the system changes its size and structure there is no solution other than to generate the model at runtime. To the best of our knowledge generating mathematical models at runtime is less explored domain than solving them once they are developed at design time.

Since we used three optimization algorithms in our approach, we need to develop three runtime models to find a solution. To generate these models at runtime we developed a meta model in the form of equation templates to generate the system model at runtime. Fortunately, since both simplex method, and interior point method could use the same formulation we have to generate only one model for both of them. However, for binary programming we need to generate a different model. In this section, we describe the meta-model that generates runtime models for binary programming and for the other two optimization methods.

1) Multi-dimensional Multi-Knapsack Model: To use binary programming we modeled our problem as two interwoven multi-knapsack problems. A Knapsack problem is a formulation where hypothetical sacks have a maximum weight and the smaller weights have to be fitted into sacks so that the sack is maximally utilized. In our problem, because of the service-level guarantee, we have to select at least two time slots for each machine. Therefore, the allocation of time slots for each machine becomes a weight. The number of such devices becomes the number of sacks of a knapsack problem as shown in fig. 1 eq. 2.

Concurrently, for each time slot we have to switch-on devices such that the number of devices is maximized while the total power consumed is within the maximum power supply available. This again is a knapsack problem (fig 1 eq. 3). Here our sacks are the time slots.

These two interwoven problems can be expressed independently and then merged together in a single equation matrix for our BP solution method.

The first knapsack problem requires a more dynamic solution than the latter one. As we do not know the number of devices, the number of equations that will be in this subproblem are not known at design time. For each sack, that is for each device, we create at runtime, an equation with six boolean decision variables representing each time period. The sum of these variables should be greater than or equal to the service-level guarantee that the specific device is calibrated to. All of these equations for sacks are aggregated to form two matrices, the variable counting left hand side and the service-level guarantee bound as right hand side as shown in eq. 2.

The second knapsack has time slots as sacks. This means that for our current setup this problem will have six sacks. Though it might be possible that the number of these "sacks" may vary at runtime. For each of these six sacks a sum of product of decision variable and consumption value is calculated as right hand side. The boolean decision variable here is the same as used in the previous knapsack problem. This reuse, or double use of decision variable weaves the two

different knapsacks. The left hand side for each time period is the amount of resource, in our case, power available for the system in that time slot.

Figure 1 is the template for binary programming planning equations. In this figure eq. 2 represents a sack for the first of the intervowen knapsack. As in that problem, each device is considered as a sack, equation 2 is generated for each device in our system. In comparison equation 3 represents a sack of the second knapsack problem. As in this problem, each time slot is considered as a sack equation 3 is generated for each time slot in our system.

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

$$\forall_i \sum_{t} X_{i,t} \geq guarantee \quad (2)$$

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

Fig. 1. Hourly planning BP equations

$$Maximize(Z = \sum_{i,t} X_{i,t}) \quad (4)$$

$$\forall_t \forall_i X_{i,t} \geq MAX_i/guarantee \quad (5)$$

$$\forall_{i,t} X_{i,t} \leq MAX_i \quad (6)$$

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

Fig. 2. Hourly planning LP equations

2) Dimension Reduction and LP Modeling: Linear programming(LP) is a mathematical modeling and solving technique to plan for scarce resources for multiple demands. 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 requirements as well as logical considerations.

LP solutions are in real domain and cannot be restricted to integer or binary values. Doing so makes LP an equivalent of Integer Programming which is an NP-Hard problem. Thus LP on a binary decision problem is not scalable. To derive an answer, within our time constraints, we instead transformed our problem from binary decision to frequency determination domain. This is done by reducing the dimensions of the problem through clustering. A simplified quality threshold algorithm was used to cluster the data [8]. We restrict the radius of clusters and use mean as a representative element. This restriction ensures all our values within a cluster are not far from mean. Through this transformation we are able to change the problem from deciding weather a machine should be kept on or off in a time slice, to, determining the optimal frequency for each cluster of machines. We rounded off the value of cluster to arrive at a sub-optimal, but maximal, plan for our system.

The resulting clustered problem has two logical constraints and one technical constraint. The first logical constraint is that the total energy consumed, as planned by the LP, should not exceed the available power supply. This is represented as equation 7. 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 every machine will be on for atleast 20 minutes (or 33% of the time). Then the constraint will be that atleast 33% of the consumers for each cluster shall be powered "on" in each cycle to ensure the 20 minutes guarantee (equation 5). 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 6.

Figure 2 defines the complete LP meta-model for our problem. The cost function (i.e. equation 4) 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 5 represents the service-level guarantee constraints that in every time period, at-least 1/3rd of systems should be in powered on state. Equation 7 limits the allocation under the maximum available supply and equation 6 puts the technical constraint that the number of allocated consumers in a cluster should not exceed the cluster size.

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.

3) Spike in Supply or Demand: In addition to the hourly planning we also developed models that could be used to replan during an hour. This is necessary because if contrary to the planned optimization there is a sudden upward trend in the demand or a sudden downward trend in the supply then the system should be able to handle it gracefully. These erratic fluctuations in the demand or supply pattern called spikes are handled using a similar model as represented in figures 1 and 2. The only difference is that when an optimization planning during an hour is performed the time window of that planning is small and the system has to keep the service-level guarantees pending for certain consumer devices. Due to the similarity of the spike related meta-model with the hourly planning model we are not going to discuss its formulation. However, we will discuss the spike related results in the evaluation section.

C. CBR Recommendation Engine

In planning an optimization the system has to select a method out of the available optimization methods. Because of competing factors it is not always possible to use a simple rule based system to select it. Therefore, there is a requirement of a recommendation engine that suggests an optimization

method to plan optimizations based on historical data and user preferences. To this end, we have used a recommendation engine that is based on Case-Based Reasoning (CBR) to find the right method when an optimization is required [1].

Case-Based Reasoning (CBR) is a technique to derive a new solution by considering similar past solutions. The CBR engine maintains a case base developed using historical data. Given a new situation the CBR engine compares the new situation with old situations and derives a solution that is closest to it. This engine has the ability to revise and update its case base. We use CBR to select the optimizer for our base problem. The input to CBR are statistics of the system alongwith user policies, constraints, etc. and output is the recommended optimization method i.e. BP, simplex, or interior point.

Another motivation to use CBR as recommendation engine for the system is because we do not have any hard boundaries for a given problem in terms of its size. This means that given a new situation it is not possible for us to select one of the three methods unless we have evaluated its past performance in a similar situation.

The recommendation engine based on CBR requires an initial case-base of historical data and when new cases come it uses the previous experience to recommend an optimization method. The new cases depending on the error rate and speed of their solution are used to improve the set of cases in the CBR engine.

Some constraints such as user policies and service-level guarantees are already fed into the system at design time. At runtime the CBR recommendation engine takes three inputs:

- Total gap between power supply and power demand.
- Total number of machines to be managed.
- Approximate time available for calculating the optimization plan.

Any CBR engine requires a set of initial case bases. We have developed the initial case base using the following methodology: we generated an initial case base of this CBR by running our three solver algorithms on sample data. We varied the size, spread and supply-demand gap of our data. These variations resulted in total of 54 cases initially. The resulting time and unutilized power margin generated by each of these runs along with the inputs are saved in the case base.

Once this initial case base is populated at runtime when an optimization is requested, CBR finds the case closest to the input parameters and selects the solver that can solve the problem within the available time while minimizes the unutilized power. For online learning a feedback system is incorporated in the CBR recommendation engine system.

IV. Architecture

We designed AdOpt so that it is easily integrated into a SAPE cycle. Therefore, we assume that each electric device has a mechanism to send us the information about its status including whether it is powered on by the consumer and its present power consumption. Assuming all this data is received into the system our framework plans for an optimization.

The data of system state, received from the sensors, is fed into CBR Recommendation Engine (CRE). CRE uses this data to generate the selection of optimization algorithm for the given optimization problem. This decision acts as an input to the Runtime Modeler (RM). The model generated by the RM is an input for a mathematical toolbox (MT) that applies the appropriate optimization algorithm to the model. This toolbox also measures the statistics of the solution such as time taken to solve and amount of power unutilized for feedback to the CRE. This process is illustrated in figure 3.

The plan generated by the MT are propagated to the devices that make up the system and executors perform the implementation of this planning. Again sensors and executors are assumed to be implemented and our framework fits in between the two for performing self-optimization.

In this section we describe a particular implementation of AdOpt used to run the simulations of a real power distribution system.

CBR Recommendation Engine: CRE consists of a mathematical summarizer, a CBR engine, result evaluator submodule and a case-base as shown in figure 4. Raw data is fed to the engine for summarization. This summary is used, in conjunction with the case-base to generate the input for subsequent modules. At the other end, CRE has a result evaluator module which receives the statistics of MT at the end of an optimization cycle. These statistics include, number of devices, supply-demand gap, solver selected, solver time and error rate. If the actual time of execution conforms with the case-base then no action is taken. But if the actual time of execution reveals that the solver time was incorrect then the case base is updated. If the number of devices is too far from any of the cases in the case-base then the result is added as a new case.

Runtime Modeler: Details of our Runtime Modeler (RM) are discussed in previous sections. Architecturally, dynamic mathematical modeler consists of three components: a knapsack modeler, an LP modeler and a clustering component for dimension reduction, as discussed previously.

Mathematical Toolbox: Our mathematical toolbox contains our mathematical solvers. These solvers are standard operations research tools. We use the mathematical solvers for the three optimization methods as black boxes. The input to the solvers in MT are the runtime model(s) generated by the RM and the output is an optimization plan. Details of the specific mathematical solvers are given in a later section.

V. Evaluation

The promise of autonomic system in general is to reduce the load of the operator. In order to cater to this need, an autonomic system should be able to cater to as many conditions that it possibly can without operator intervention. In addition, a self-optimizing system should deliver a better result in general cases in comparison to a system where optimization is performed manually.

In general, we claim that AdOpt framework caters to these demands of the self-optimizing system. In fact our system

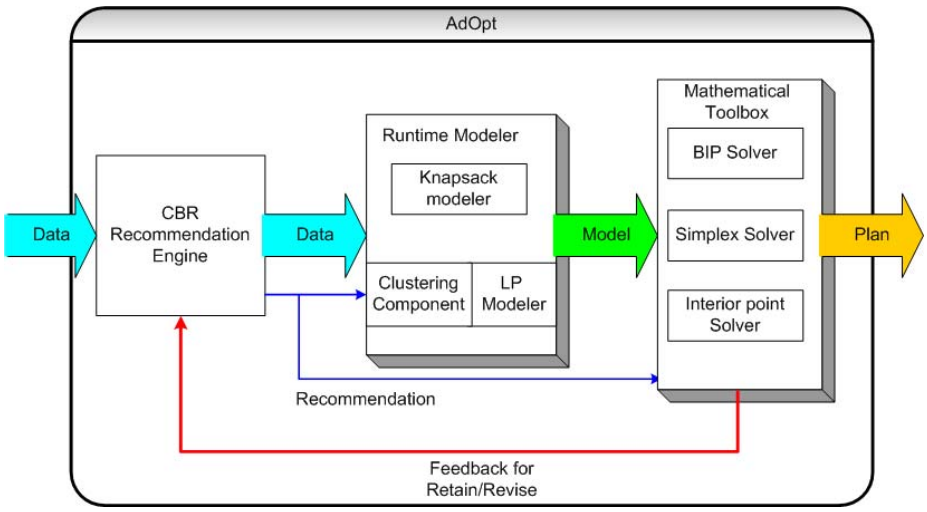

Fig. 3. System Flow

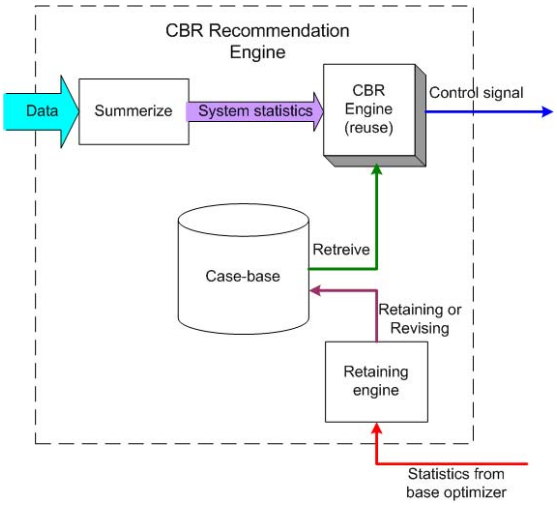

Fig. 4. CRE Architecture

can outperform other self-optimization techniques in a rapidly changing system by leveraging the very changes which cause sub-optimal behavior.

To evaluate our system against these two core requirements we pose the following questions for evaluating our system:

1) Is our system leveraging the change in the dynamics of the system in terms of its size and available time?
2) How much did we optimize or in other words what percentage of power is utilized?

We evaluate our system against two different set of evaluation suites. Our first evaluation suite tested our system for effectiveness, that is, are we able to provide a plan irrespective of variations in the system. We compare our results with three techniques running without adaptation and compare effectiveness and saving.

Our second evaluation suite is built on data from a power distribution network where we compare our framework's results with existing technique. Our savings, or profitability comes from allocating as much power to devices as we can. In this test suite we compare the un-allocated power of AdOpt framework with the un-allocated power of existing techniques.

In this section first we will describe our evaluation environment. Then we will discuss results for the two different sets of evaluation we conducted on our framework.

A. Evaluation Environment

We used a shared 2.4 G.Hz. Pentium Core 2 Duo processor with total of 2.00 GB of RAM to conduct our simulations. The CRE is developed using FreeCBR. The APIs of CBR are integrated with Matlab and uses Matlab's internal JDK to call FreeCBR functions. We wrote our own code for the RT and clustering of raw data. The mathematical solvers in MT uses Tomlab's CPlex [1] solver to solve BP and uses Matlab's optimization toolbox for LP optimizations using simplex or interior point methods.

CBR Initial Case-base

To develop our CBR case-base we varied supply-demand gap and size of the problem. We applied this data to all the three algorithms. Due to practical consideration we limited BP to sizes of 2000 users.

B. Results

We evaluated our claims by running AdOpt on two sets of data. The first set consists of a hypothetical system where we showed the adaptability and efficiency of our system. To validate results further we tested our system on a real data obtained from a consortium of independent power companies in California (CAISO).

[1] http://tomopt.com/tomlab/products/cplex/

1) Adaptability and Efficiency Evaluation: In these experiments we evaluated our system on three key variation of the system. These variation includes the electric devices present in the system, the gap between supply and demand of electricity, and time available to calculate the plan. We made equivalence classes of these three variations to develop the set of experiments to test the system under various set of variations.

We used five equivalence classes of the number of machines present in the system i.e. 500, 1000, 2000, 10000 and 50000. Three variations in demand and supply gaps are considered i.e. 5%, 20% and 50% more demand than available electricity. The available time to calculate the plan is also varied. This is the maximum time available to find a plan. For hourly planning this may not matter because we have a whole hour at maximum to calculate the plan. However, at any time when there is a spike in the system of demand or supply then a plan needs to be recalculated for the hour. Therefore, we considered three upper boundaries for time available to recalculate. These upper boundaries are 5, 20, 60 and 200 seconds respectively.

A cross-product of these three variations results into sixty cases. Showing results of sixty cases is not possible in the paper. Therefore, we used orthogonal arrays to generate twenty combinations of the experimental results that we show here. According to a study by NIST using orthogonal arrays covers 98% for all cases in real situations [20].

These twenty experiments are summarized in table I.

The first three columns of table I (after the experiment numbers) show the three variations for the experimental runs. The # of devices shows the number of electric machine present in the system where a service-level guarantee is required. Shortfall is the different between the demand and supply. A-Time is the maximum time available to calculate the plan.

Using these variations in each test we used the three optimization techniques independently and then used AdOpt to see if AdOpt provides us with a better optimization by selecting one of the optimization technique dynamically.

The C-Time is the time consumed by the technique to find the plan after it receives the raw data in the CRE and produces a plan as output. UP is the unutilized power in the system that an optimization is unable to distribute amongst the electric devices. We have rounded off the percentage numbers to the nearest integer to make a better comparison amongst the techniques.

2) Analysis : As shown in table I during all the experimental runs all optimization methods gave a result in form of a plan except a few cases in binary programming (BP). This is because BP is only applicable at small scales and it runs out of memory in test runs where the number of devices is larger than 2000. Therefore, we did not run BP on problems involving more than 2000 electric devices.

By looking at the results in table I it is clear that AdOpt is able to find a plan in all the cases. Depending on the number of devices, the gap between electricity and demand and available time to calculate it optimizes power by minimizing the unutilized power.

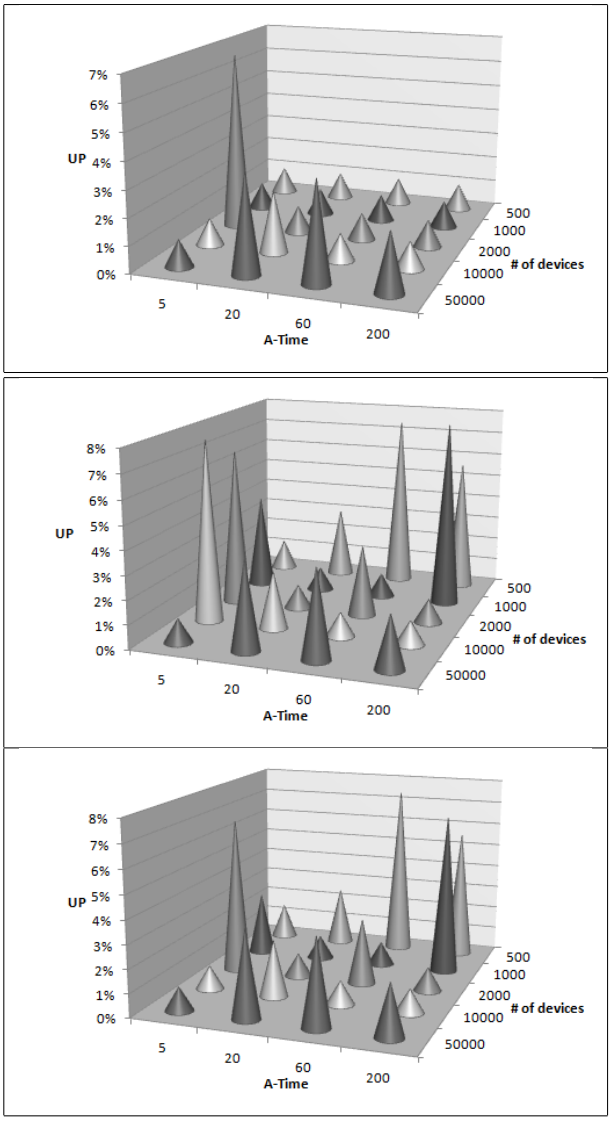

Fig. 5. Summary of Results of Table I. Top most figure shows results of AdOpt, middle one show results from Interior Point and the bottom one shows result of Simplex method.

Fig. 5 shows a summary of the results in terms of power that remain unutilized. We did not show BP evaluation in this figure as BP is only applicable in cases where the number of devices is less than 2000. But in all these cases the unutilized power is less than 1% using BP. figure 5, gives a graphical comparison of UP values given by AdOpt, interior point method and simplex.

The lengths of the peaks in this figure are directly proportional to the amount of unutlized power. Compared to the other two methods i.e. Interior Point, and Simplex used in isolation, AdOpt has a relatively fewer number of large peaks. This means that Adopt is able to minmize the unutlized power by appropriately applying one of the three methods.

It is also clear from the results that if we were to use an optimization technique independently then it can not solve a given problem or solve a given problem efficiently. For

Exp #	Variations			AdOpt		Simplex		Interior point		Binary Programming	
	# of Devices	Shortfall	A-Time	C-Time	UP	C-Time	UP	C-Time	UP	C-Time	UP
1	500	5%	5	0.3120	1%	0.078	1.5%	0.0624	1.27%	0.202	1%
2	500	30%	20	0.3276	1%	0.093	2.5%	0.078	3%	0.1716	1%
3	500	50%	60	2.78	1%	0.062	7.4%	0.062	7.4%	2.57	1%
4	1000	30%	5	4.5396	1%	0.1404	2.7%	0.1248	3.99%	4.148	1%
5	1000	5%	20	0.2964	1%	0.1248	1%	0.1092	1%	0.5928	1%
6	1000	5%	60	0.312	1%	0.1248	1%	0.1404	1%	0.5928	1%
7	1000	50%	200	21.4189	1%	0.1872	6.9%	0.1248	7.9%	1.3104	1%
8	2000	50%	5	0.3276	6.7%	0.202	6.7%	0.2028	6.7%	102.0559	1%
9	2000	5%	20	0.3744	1%	0.2184	1%	0.234	1%	2.028	1%
10	2000	30%	60	46.47	1%	0.234	2.8%	0.182	3%	46.63	1%
11	10000	50%	20	0.9828	2.3%	0.858	2.4%	0.858	2.4%	na	na
12	10000	5%	200	0.9516	1%	0.8892	1%	0.8424	0%	na	na
13	10000	30%	5	1.1076	1%	0.8268	1%	0.8268	7.7%	na	na
14	50000	30%	200	4.735	2.2%	4.3836	2.2%	4.4616	2.2%	na	na
15	50000	5%	5	4.524	1%	4.3524	1%	4.5396	1%	na	na
16	50000	50%	20	4.602	3.7%	4.3056	3.7%	4.3056	3.7%	na	na
17	500	30%	200	0.234	1%	0.046	5.6%	0.078	5.6%	0.2184	1%
18	2000	5%	200	2.076	1%	0.2184	1%	0.2028	1%	1.919	1%
19	10000	30%	60	0.9828	1%	0.8424	1%	0.8112	1%	na	na
20	50000	5%	60	4.504	3.7%	4.3836	3.7%	4.4148	3.7%	na	na

TABLE I
ADAPTABILITY AND EFFICIENCY TEST RESULTS

example, BP is not applicable for 45% of classes of inputs because it can only handle problems where number of entities is small. Moreover, interior point algorithm is applicable to all classes of problems but can put the system in situations where up to 8% of power is unutilized.

It should be noted that result and result times are highly denpendent on input values. As results in this test suite were conducted on variation of input values, this evaluation provides a guarantee that our system can optimally handle any variation of system behavior provided we have certain threshold of supply of electricity.

To summarize, our conclusions are following:

In almost all of the cases UP of AdOpt is minimum. Except in experiment # 8 (i.e 5,2000) (table I). This is because time is very small for this run and our shortfall is 50%. Combination of these two key factors has an adverse effect on time and therefore AdOpt selected the fastest method to calculate the optimization plan i.e. interior point.

Efficiency of AdOpt is inversely proportional to size and shortfall. It is also directly proportional with time. This means that as our size increases, efficiency decreases. Similarly increase in shortfall has adverse effect on efficiency, but as our allowed time increases efficiency increases too.

BP is not able to handle all the situations and in situations where it is applicable it's results are close to 1%. This is why we have not shown time vs devices graph of BP in figure 5.

3) Real Data Evaluation using CAISO Dataset: In our second set of experiments, we applied AdOpt on data collected from State of California in USA. California Independent System Operator (CAISO)[2] provides the energy data including the supply and demand of power in the State of California online. Our motivation to test system with data from CAISO is that California's data is available and the weather pattern in California closely resembles that of Pakistan. With this correlation, we can correlate the behavior of users in our country and simulate the results.

We collected hourly data for seven days. The demand of the system varies between 16000MWh and 25000MWh. During these seven days we used this data from the demand side to see the variations that occur. In the case of California the supply is always greater than demand. Unfortunately, in the case of developing countries this is not true. Therefore, using the demand patterns of California we used a supply pattern of a developing country, where although supply is constant but it is always lower than demands. The top most line in figure 7 shows the demand for a day and the line below it shows the constant supply. A gap of 30-60% between supply and demand is thus added into the system.

Moreover, based on the demand we pro-rated the number of devices that are working in the system . Which means that we assume that if the demand goes up we believe that the number of electric devices also goes up and vice versa.

Using this supply and demand pattern, and the pro-rated number of devices, we used AdOpt to find an optimization plan on the CAISO data.

4) Analysis: We applied AdOpt and simplex algorithm for hourly planning. Since results for simplex and interior point are very close for large data-sets we used only simplex for comparison in this experiment. The bottom line of figure 7 shows the unutilized power using stand alone simplex, and AdOpt. The size of the problem in some cases is too big for BP to solve hence BP is not evaluated as a stand alone technique to solve the problem.

Summary of results for 7 days are shown in fig. 6. AdOpt provided a better resource utilization than a stand-alone optimization method. For the seven day period we generated

[2]http://www.caiso.com/

plans with 27% better utilization of power as compared to a standalone simplex solution.

To better illustrate our system we show results of a day out of the seven days with a lot of fluctuations in demand. Fig. 7 describe the supply and demand variations for this day. As can be seen, the demand for power at the start of the day is low, the demand grows steadily till mid-day and then tapers off towards the night time. If we use an LP based solver to assign power, our unutilized power stays in the range of 6% to 9.5%. In comparison AdOpt's consumption starts off at close to 0% when power demand and supply gap was low. As the power shortfall grew, AdOpt updated it's selection of algorithm to manage the increase in size. At the end of the day, when the demand again dropped sharply, AdOpt also scaled down and applied a more conserving algorithm and increased it's saving.

Our comparison with LP yielded a 27% more cumulative efficient energy allocation for the whole day. This leverage was achieved by applying a more conservative albeit slow technique at low consumption times, mainly at night. During day time, AdOpt used a more robust LP based algorithm to handle the bigger size/gap situation.

An interesting point to note here is that AdOpt was able to scale up and down immediately. This is in contrast with control theoretic approaches which aim for stability and sudden changes are smoothed over and sudden jumps in input space do not translate to a drastic change in solution space.

In a nutshell the summary of our findings is given below:

- Overall AdOpt provides a 27% better service than LP.
- AdOpt can provide solutions to 100% of problems where as BP is not able to solve all the situations in a 24-hour period.
- AdOpt is resilient to sudden changes in the real time power distribution system. If a change in input requires an adaptation then adaptability of AdOpt is instantaneous.

In order to make our adaptation approach usable in a multitude of environments we have used the CBR recommendation engine in AdOpt. Therefore, the whole framework of AdOpt could be applied to a new power distribution system without much effort. In fact we are working on using AdOpt to optimize powerdistribution in large indsutrial settings. The last section of this paper provides some details on this future direction.

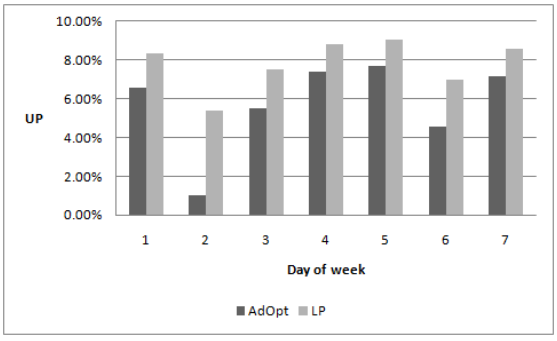

Fig. 6. AdOpt and Simplex comparison on seven day CAISO data

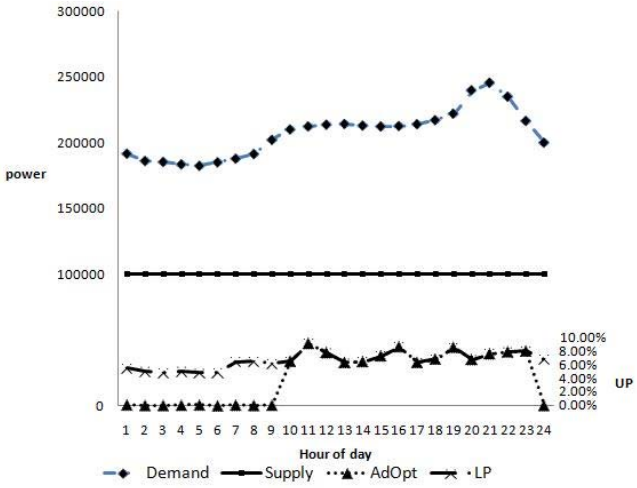

Fig. 7. Supply-demand and comparison of a typical day in CAISO data

VI. Discussion

In this section we discuss some of the issues that a reader may have regarding our framework.

Are the simulations realistic? Data for our simulation comes from a live system from California. Users can download usage data from the website. Hence the projection of demand is actual. We know that in places such as Pakistan, supply-demand gap is as much as 50% so our power provisioning of roughly 55 - 80% of power is a realistic scenario for a developing country.

What is the breaking point of the system? We do not guarantee a plan if the supply is not enough for fulfilling all the guarantees. In such a case, guarantees will need to be scaled down, if possible, and plan calculated afterwards. But it is a policy decision. However, it could be done at runtime without changing anything in AdOpt. .

What about spike handling? Spikes are a reality in a system. We can not ignore them but due to paucity of space, we did not discuss the details of spike handling. Our evaluations, however, did cater for spikes by testing system for various durations where time is critical.

Why use AdOpt when simplex works on all cases? It is true that we were able to allocate upto 92% of electricity through simplex alone, but with AdOpt we have increased this saving to 97%. This minor advantage means that in an area of 10,000 devices, at least 500 more customers will be satisfied than in if we use simplex alone.

Can this system be implemented? In some cities of Germany and USA micro managing large heavy electric devices is possible. However, the power companies in these countries only have to use it in peak demand times which occur very infrequently. In countries such as Pakistan where demand is always more than supply we need a very robust and resilient system that could handle many different variations of load. Legislative changes that allow companies to micro-manage devices is required but discussing these legislative changes are beyond the scope of this paper.

VII. Related Work

Our work has three major themes: scalable self-optimization, best fit adaptation framework, and generating runtime model customized for optimizers. In this section we discuss related work on these themes.

Optimization which is scalable with respect to number of dimensions as well as size is key to adaptive self-optimization. Various techniques have been used for self-optimization. However, for most of the techniques scalability to hyper dimension space has not been proven [21], [15], [14], [16], [2], [13], [21], [18]. These techniques have been mostly applied to domains where number of variables are small or has been reduced through distributed computing. The techniques worked well for a specific domain but are not capable of, or atleast not proven to, handling increase in dimensionality. Almeida and colleagues have introduced a technique which does scale to hyper-dimensions but the technique does not guarantee an optimal solution [3]. The technique is also considerably slow which, reduces it's application. This technique, or similar techniques can become part of our framework and increase efficiency of our framework either through faster response time or better efficiency. Our previous work introduced a way to handle hyper-dimension domains [11]. We have used the techniques discussed in [11] to provide a scalable solution in AdOpt.

Some researchers have described self-optimization as a goal for a framework itself. Böcker and colleagues proposed using "self-optimization" as intelligent controls for industrial engineers [5]. This proposes a framework for controls. Although the stated motivation is to handle changing environment, the authors, however, did not propose an adaptive method to adapt to changing environment.

Research in optimization of power has been worked on by quite a few researchers. However, the research in power domain has been involved in optimizing parameters of power distribution, generation and propagation. However, our work is inherently different from this body of work. Our optimization is applied on coarser level of managing devices. Some relatively recent work by Ogston and Zeman has looked at micro-managing the devices to conserve electricity by clustering the devices based on their power profile [17], [22]. However, the self-managing aspects in this work assumes a fixed set of entities while in our system the number of entities may vary and the framework adapts itself to the varying load. Morever, this work compares reinformcement learning and the homeotaxis methods in managing multigent system that in turn controls the energy optimization which assumes a prebuilt model and improves itself with each cycle. Indeed our work is a step further as it dynamically selects an optimization approach and performs all the generation of models at runtime. Nevertheless, it would be an interesting future work to modify the models proposed by Ogston and Zeman in the AdOpt framework using runtime model generation and variable set of entities.

There has been much work in generating models at runtime. Research has focused on extracting information from a running system and forming an abstract model of the system. For example, Zhang and Figueiredo developed a way to extract key features from a running system [23]. Zhang and colleague developed a regression based modeling technique [24] to model a system. FAME, developed by Kuhn and Verwaest also is used to model a software system [12]. However, these modeling techniques model a system into an abstract representation. Our runtime model in contrast generates a mathematical model which is customized for an optimization technique. The technique mentioned above, and other similar techniques, can be very helpful in analysis phase of SAPE. However, an internal modeler, which takes numeric data and transforms it into optimizer specific mathematical model is necessary for adaptable optimization.

An active community is working on frameworks for adaptive softwares. However, the focus of adaptations, to our knowledge, has been mostly on validation of a new state only and not many have focused on optimality of the new state. In contrast, our technique focuses on optimality of the final state. System such as by Heo and colleagues cater for adaptation [7]. It identifies and avoids conflicts in adaptation paths. They applied this framework on an optimization problem with results strikingly similar to AdOpt. However, the framework is built only to identify and avoid conflicts whereas we are looking for the optimal selection of techniques. In a situation with multiple valid adaptations, AdOpt can easily outperform the said framework.

Trumler and colleagues developed an adaptive self-optimization technique [19]. The work is initial step in adaptive self-optimization. The proposed work, however, is weak in defining architectural details regarding toolbox selection criterion, modeling technique and extensions in the toolbox.

Our work resembles the efforts of Gounaris and colleagues [6]. Whereas we are using CBR and operational research techniques, Gounaris has used switching extremum controls to optimize controller and the system. This is the basis to differ from Gounaris. Our technique has four distinct advantages to Gounaris. First, It has not been proven that control theory solutions can scale to hyper dimensional space. Second, controls were built for electric and electro-mechanical structures hence stability is a major concern for controls developed. Controls tend to smooth the transition even for drastic changes in input domain. This is not ideal for our condition as smoothing out the transition may lead to usage of incorrect technique. Third, control theory has a much steeper learning curve as compared to LP. Fourth, our system learns from our past history whereas controls do not learn and will repeat a mistake over and over again.

These two systems provide an example of adaptable self-optimization and lend credibility to the belief that adaptable self-optimizing systems can be structured to realize the goals of autonomic computing.

VIII. Conclusion and Future work

In this paper we presented a framework that can dynamically optimize itself based on the given situation of the power distribution system. Not only the system optimize itself, but it can also adapt to a number of unforseen situations that are not dealt by other comparative systems. Although, the framework that we have developed is only applicable in the present form to power distribution companies. In this regard our future work is to develop a system that optimizes energy in places like smart homes, large industrial settings and so on. Moreover we are also looking at self-validation system that verifies the optimization plan given by our framework. Furthermore, upcoming commercial applications such as Google PowerMeter, Cisco Smart Energy Networks and IBM SmartGrid could use AdOpt framework to optimize power in a range of settings.

We also believe that with some modifications AdOpt could be used to solve other related problems in computer science. For example, in cloud computing it is not possible to predict the number of nodes that are required to compute the solution of a given problem. With AdOpt this could be made predictable. Other than cloud computing AdOpt could be used in diverse systems such as sensor networks, water management and others.

IX. Acknowledgements

This work is in part supported by grants from the Department of Computer Science at LUMS, ICT Research and Development Fund of Pakistan, and Higher Education Commission of Pakistan.

References

[1] Aamodt, A., and Plaza, E. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun. 7*, 1 (1994), 39–59.

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

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

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

[5] Bocker, J., Schulz, B., Knoke, T., and Frohleke, N. Self-optimization as a framework for advanced control systems. *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on* (Nov. 2006), 4671–4675.

[6] Gounaris, A., Yfoulis, C., Sakellariou, R., and Dikaiakos, M. Robust runtime optimization of data transfer in queries over web services. *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on* (April 2008), 596–605.

[7] Heo, J., Henriksson, D., Liu, X., and Abdelzaher, T. Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study. *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International* (Dec. 2007), 227–238.

[8] Heyer, L. J., Kruglyak, S., and Yooseph, S. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Res. 9*, 11 (November 1999), 1106–1115.

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

[10] Javed, F., and Arshad, N. On the use of linear programming in optimizing energy costs. In *IWSOS* (2008), pp. 305–310.

[11] Javed, F., and Arshad, N. A penny saved is a penny earned: Applying optimization techniques to power management. In *16th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS 2009), 13-16 April 2009, San Francisco, CA, USA* (2009).

[12] Kuhn, A., and Verwaest, T. Fame, a polyglot library for metamodeling at runtime. *Models @ Runtime 2008* (2008), 57–66.

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

[14] Li, Y., Sun, K., Qiu, J., and Chen, Y. 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* (July 2005), 266–273 vol.1.

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

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

[17] Ogston, E., Zeman, A., Prokopenko, M., and James, G. Clustering distributed energy resources for large-scale demand management. *Self-Adaptive and Self-Organizing Systems, International Conference on 0* (2007), 97–108.

[18] Shah, K., and Kumar, M. Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks. *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on* (Oct. 2007), 1–9.

[19] Trumler, W., Pietzowski, A., Satzger, B., and Ungerer, T. Adaptive self-optimization in distributed dynamic environments. In *SASO '07: Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 320–323.

[20] Wallace, D. R., and Kuhn, D. R. Converting system failure histories into future win situations. nist. 2000.

[21] Wang, M., Kandasamy, N., Guez, A., and Kam, M. 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* (13-16 June 2006), 165–174.

[22] Zeman, A., Prokopenko, M., Guo, Y., and Li, R. Adaptive control of distributed energy management: A comparative study. *Self-Adaptive and Self-Organizing Systems, International Conference on 0* (2008), 84–93.

[23] Zhang, J., and Figueiredo, R. Autonomic feature selection for application classification. *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on* (June 2006), 43–52.

[24] Zhang, Q., Cherkasova, L., and Smirni, E. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on* (June 2007), 27–27.