The eighth annual GreenMetrics Workshop was held on June 5, 2017 in Urbana-Champaign, IL, USA, in conjunction with the ACM SIGMETRICS 2017 conference. For the past years the workshop has been expanded from topics on the energy and ecological impact of Information and Communication Technology (ICT) systems, to include emerging work on the Smart Grid. Topics of interest fall broadly into three main areas: designing sustainable ICT, ICT for sustainability, and building a smarter, more sustainable electricity grid. The workshop brought together researchers from the traditional SIGMETRICS and Performance communities with researchers and practitioners in the three areas above, to exchange technical ideas and experiences on issues related to sustainability and ICT.

The workshop program included three 45-min keynote talks, and eleven 20-min presentations of technical papers.

In the first keynote “Data and metrics for power grids and energy supply sustainability”, Peter W. Sauer, locally from UIUC, presented fundamental issues and opportunities for Direct Data Driven Applications as well as Data Analytics for energy delivery systems, which included the challenges of full utilization of renewable resources, customer interaction, energy markets, and optimal strategies. One particular focus was exploiting the vast supply of Phasor Measurement Units and the potential for model-less analysis and control.

David Irwin from University of Massachusetts Amherst presented the second keynote talk, titled “Staring at the Sun: Solar Energy Analytics and their Privacy Implications”. He first presented SunDance, a technique for disaggregating solar power from a building’s net energy usage. Since the vast majority of solar deployments are “behind the meter”, accurate solar disaggregation can significantly improve utilities’ visibility into distributed solar generation. Unfortunately, solar energy data is not anonymous: since every location on Earth has a unique solar signature, it embeds detailed location information. To explore the severity and extent of this privacy threat, David continued to present SunSpot, a technique for localizing “anonymous” solar-powered buildings from their solar energy data.

Our third keynote talk was by Thomas F. Wenisch from University of Michigan, titled “Report from the Arch2030 Visioning Workshop: Where are Computer Architects headed over the next 15 years?”. Thomas summarized the outcome of the workshop report delivered to the National Science Foundation and discussed possible implications for the Sigmetrics/Greenmetrics community.

After considering many good candidates, the best student paper award was given to “GPGPU Power Estimation with Core and Memory Frequency Scaling”, by Qiang Wang and Xianwen Chu from Hong Kong Baptist University. The authors proposed a fast prediction model based on Support Vector Regression, which can estimate the average runtime power of a given GPU kernel using a set of profiling parameters under different GPU core and memory frequencies.

Demand response is discussed in the following papers. In “Distributed Algorithm Design for Probabilistic Demand Response”, Joshua Comden et al. proposed a simple contract between customers and the LSE to allow flexibility in demand response commitment, and design a distributed algorithm to find the optimal contract parameters. In “Summary of Recent Results: Crowd-Sourced Storage-Assisted Demand Response in Microgrids”, Mohammad Hajesmaili et al. devised an online algorithm with logarithmic bi-criteria competitive ratio for the problem of utilizing energy storage systems to perform demand-response in microgrid.

The following two papers deal with data center energy management. In “Optimal Energy Procurement for Geo-distributed Data Centers in Multi-timescale Electricity Markets”, Tan N. Le et al. developed energy procurement algorithms for geo-distributed data centers that utilize multi-timescale markets to minimize the electricity procurement cost. In “Phase Balancing in Power Distribution Network with Data Center”, Wei Wang and Nanpeng Yu proposed a new paradigm which coordinates the operation of data center and distributed energy resources to reduce phase imbalance and improve the reliability and efficiency of electric distribution networks.

Hitoshi Oi presented “A Case Study of Energy Efficiency on a Heterogeneous Multi-Processor”, in particular, results on Exynos 5422 based on the ARM big.LITTLE architecture. In “Black-box Solar Performance Modeling: Comparing Physical, Machine Learning, and Hybrid Approaches”, Dong Chen and David Irwin presented a configurable hybrid approach enabling users to select the parameters they physically model versus learn via ML. Pengcheng You et al. presented “Battery Swapping Assignment for Electric Vehicles: A Bipartite Matching Approach” and in particular, an efficient solution based on the Hungarian algorithm. In “Distributed Lagrangian Method for tie-Line Scheduling in Power Grids under Uncertainty”, Thinh Doan et al. proposed a distributed Lagrangian algorithm and established the convergence rate. Mohammad A. Islam et al. presented “A First Look at Power Attacks in Multi-Tenant Data Centers”, in which they developed a state-augmented Kalman filter that guides an attacker to precisely time its power attacks. Finally, in “Load-side Frequency Regulation with Limited Control Coverage”, John Pang et al. obtained a control law that rebalances power and asymptotically stabilizes frequency after a disturbance.

The papers presented at the workshop reflected a current concern of green computing, demand response, and other fundamental issues in smart grid. The workshop incited interesting discussions and exchange among participants from North America, Europe, and Asia.

Niklas Carlsson, Linköping University
Zhenhuan Liu, Stony Brook University
Thu Nguyen, Rutgers University
Catherine Rosenberg, University of Waterloo
Adam Wierman, Caltech
1. INTRODUCTION

One of the major issues with the integration of renewable energy sources into the power grid is the increased uncertainty and variability that they bring [1]. The limited capability to accurately predict this variability makes it challenging for the load serving entities (LSEs) to respond to it [3]. If this variability is not sufficiently addressed, it will limit the further penetration of renewables into the grid and even result in blackouts [4]. Various approaches have been implemented or proposed to address this issue. These include improving renewable generation forecast, aggregating diverse renewable sources, fast-responding reserve generators, energy storage, and demand response (DR), among others. Compared to energy storage, DR has advantages to provide reserves to the LSEs in a cost-effective and environmentally friendly way [11, 16].

DR programs work by changing customers’ loads when the power grid experiences a contingency such as a mismatch between supply and demand. The decision that must be made by DR programs is how much load demand each customer should change. Uncertainties from both the customer-side and LSE-side make designing algorithms for DR a major challenge. LSEs make predictions about the net load demand and purchase capacity to dispatch controllable supply accordingly, but they do not know the true mismatch between supply and demand until the time arrives. On the other hand, customers who are accustomed to having electricity supply on demand are not be able to accurately estimate how much disutility a future change in load demand would bring them. These uncertainties mean that the LSE does not know how exactly how much aggregate DR it will need and the customers do not know how much change they will be willing to provide.

Also for large-scale algorithms with many decisions such as DR programs, distributed algorithm design is important. As networks grow, the increased communication overhead required for a centralized decision maker becomes infeasible. More importantly, privacy requirements may not allow a central entity to know the objectives and constraints of all the users. This is especially true in the case of deregulated power markets [8]. Distributing algorithms is a major challenge for DR because the LSE needs information about the customers’ decisions.

This paper makes the following main contributions: First, we model the social cost minimization problem using stochastic optimization that jointly optimizes DR participation and capacity planning (Section 2). Second, we propose a simple contract between customers and the LSE, and design a distributed algorithm to find the optimal contract parameters (Section 3). Third, we show that the distributed algorithm converges quickly with a real world trace (Section 4). The major difference in our work as compared to existing work is that we design a contract between each customer and the LSE that adjusts the customers’ loads according to real-time system parameters in contrast to robust real-time pricing strategies, e.g [10, 12, 15].

2. MODEL

We consider a two-stage decision problem in which an LSE first purchases capacity to handle an unknown amount of supply-demand mismatch, and second procures a total amount of load reduction from a set of customers \( \mathcal{V} \) after the supply-demand mismatch is revealed. Note that we only frame DR as a load reduction for ease of explanation. Our approach is general enough for DR to be framed as any change of load demand. We ignore the power network constraints in this paper. However, our model and algorithms can be extended with extra effort to incorporate those power network constraints.

Customers

Let \( d_i \) be the actual power demand and \( \hat{d}_i \) be the predicted power demand of customer \( i \in \mathcal{V} \) for a particular timeslot. Therefore, we denote \( \delta_i := d_i - \hat{d}_i \) as the customer demand’s mismatch and model it as a random variable. In real-time, the customer observes its own real power demand \( d_i = \hat{d}_i + \delta_i \) in the absence of a DR program and decides \( x_i \) under a particular DR program as the amount of demand reduction from \( d_i \). So the actual power consumption under a DR program becomes \( d_i - x_i \).

To model the loss of utility caused by the reduction in power consumption \( x_i \) from the original demand \( d_i \), we assume there is a cost function \( C_i(x_i) \). The function is inherently different for different timeslots and may not be known until at (or just before) the time of consumption. The uncertainty in this function can be modeled by using a random variable that parameterizes \( C_i(x_i) \), (e.g. \( a_i \) in \( C_i(x_i) = a_i x_i^2 \)). We use the function \( C_i(x_i) \) to represent its estimated cost function.

Load Serving Entity

We consider the general case where the LSE has volatile renewable energy generation and is responsible for handling any supply-demand mismatch with dispatchable resources. The uncertainty from the renewables and the customers’ demands combine to form the aggregate mismatch \( D \) which is modeled as a random variable. Therefore, after the customers apply their load reductions, the LSE has \( D - \sum_i x_i \) remaining mismatch to manage.

For any remaining mismatch, the LSE has to bear the cost denoted by the penalty function \( C_p(D - \sum_i x_i) \). Specifically, this is the cost imposed on the LSE to close the gap through actions such as employing fast responding reserves or grid energy storage.

In order for the LSE to tolerate the mismatch and prevent
blackouts, the LSE must purchase long-term energy storage or reserves in some forward market denoted by \( \kappa \). Denote \( C_{\text{cap}}(\kappa) \) as the cost of the energy storage/reserve capacity \( \kappa \) amortized to a single timeslot. This gives us the mismatch constraint
\[
-\kappa \leq D - \sum_{i \in V} x_i \leq \kappa
\]
so that the LSE has the ability to handle any remaining mismatch.

### Optimization Problem

The goal is to simultaneously decide the capacity planning \( \kappa \) and a practical DR policy \( x(D, \delta) \) to minimize the expected social cost caused by a random aggregate supply-demand mismatch \( D \) (which captures mismatches from both the generation side and the load side).

\[
\begin{align*}
\min_{\kappa, x(D, \delta)} & \quad C_{\text{cap}}(\kappa) + \mathbb{E}_{D, \delta, C_i(\cdot)} \left[ \sum_{i} C_i(x_i(D, \delta_i)) + C_g \left( D - \sum_{i} x_i(D, \delta_i) \right) \right] \\
\text{s.t.} & \quad \max_{D, \delta} \left\{ D - \sum_{i} x_i(D, \delta_i) \right\} \leq \kappa \quad (2a) \\
& \quad \min_{D, \delta} \left\{ D - \sum_{i} x_i(D, \delta_i) \right\} \geq -\kappa. \quad (2b)
\end{align*}
\]

The expectation is taken with respect to the aggregate mismatch \( D \), individual customer mismatches \( \delta_i \), and the customer cost functions \( C_i(\cdot) \) since they are not known before DR operation. We note that (2a) and (2b) are worst-case constraints so that the remaining mismatch does not go beyond the purchased capacity.

We make the following mild assumptions. We assume that the cost functions are convex. The convexity assumption on the customer-side is consistent with the concavity assumption of customer utility functions as was done in [14]. A simple but widely used example is the quadratic function, i.e., \( C_i(x_i) = a_i x_i^2 \) [17]. On the LSE-side, \( C_g(\cdot) \) can also be a quadratic function [13], and \( C_k(\cdot) \) can be linear. Also, the randomness in a customer’s cost function \( C_i(\cdot) \) and the mismatch \( D \) are assumed to be both stationary. This assumption is reasonable since the randomness in \( D \) is due to the prediction error of the customers’ load demands and renewable energy supply.

The two main challenges of Problem (2) are (i) deciding the optimal capacity \( \kappa \) before operating the DR policy, and (ii) optimizing an online DR policy.

### 3. POLICY DESIGN

#### Linear contract

Motivated by the desire to find a simple DR policy \( x(D, \delta) \) that preserves convexity and can be decided jointly with capacity, we focus on a simple but powerful linear contract that is a function of the aggregate and individual mismatches:
\[
x_i(D, \delta_i) = a_i D + \beta_i \delta_i + \gamma_i \quad (3)
\]
which is the optimal form of a DR policy when the cost functions are quadratic as shown in our extended version [9].

This contract combines the global aggregate mismatch \( D \) with each customer’s local mismatch \( \delta_i \) to decide what that customer’s change in demand should be. Intuitively, there are three components: \( a_i D \) implies each customer shares some predefined fraction of the global mismatch \( D \); \( \beta_i \delta_i \) means customer \( i \) may need to take additional responsibility for the mismatch due to his own demand fluctuation and estimation error; finally, \( \gamma_i \), the constant part, can help when the random variables \( \mathbb{E}[D] \) and/or \( \mathbb{E} [\delta_i] \) is nonzero. Then the LSE needs to solve (2) with (3) to obtain the optimal parameters for the linear contract, i.e., \( a_i, \beta_i, \gamma_i \) as well as the optimal capacity \( \kappa \). We have the following theorem proved in [9]:

**Theorem 1.** Problem (2) with the linear contract (3) is a convex optimization problem.

### Distributed algorithm

In most cases, the LSE’s information on the customers’ cost functions is much less accurate than the customers themselves. This can also be due to privacy concerns. To handle this, we design a distributed algorithm so that the LSE does not need the information of the customer cost functions.

First, we introduce and substitute \( (u_i, v_i, w_i) \) as the customer’s copy of \( (\alpha, \beta, \gamma) \) in each of their estimated cost functions \( C_i(\cdot) \) to get
\[
\begin{align*}
\min_{\alpha, \beta, \gamma, u, v, w, \kappa} & \quad C_{\text{cap}}(\kappa) + \sum_{i \in V} \mathbb{E}_{D, \delta, \hat{C}_i(\cdot)} \left[ \hat{C}_i(u_i D + v_i \delta_i + w_i) \right] \\
& + \mathbb{E}_{D, \delta} \left[ C_g \left( D - \sum_{i \in V} (\alpha_i D + \beta_i \delta_i + \gamma_i) \right) \right] \\
\text{s.t.} & \quad (2a), (2b) \\
& \quad u_i = \alpha_i, \quad v_i = \beta_i, \quad w_i = \gamma_i, \quad i \in V \quad (4a)
\end{align*}
\]

Problem (4) can be split where each customer controls its own \( (u_i, v_i, w_i) \) and the LSE controls \( (\alpha, \beta, \gamma) \) by using dual decomposition of constraint (4b). Let \( (\pi_i, \lambda_i, \mu_i) \) be the dual prices for each customer corresponding to constraint (4b). Therefore \( \pi_i u_i + \lambda_i v_i + \mu_i w_i \) is the total payment to customer \( i \) for following the linear demand response contract. Accordingly, (4) is decomposed into the individual customer optimization problem
\[
\begin{align*}
\min_{u_i, v_i, w_i} & \quad \mathbb{E}_{D, \delta, \hat{C}_i(\cdot)} \left[ \hat{C}_i(u_i D + v_i \delta_i + w_i) \right] - \pi_i u_i - \lambda_i v_i - \mu_i w_i, \quad (5)
\end{align*}
\]
and the LSE’s optimization problem among all the customers
\[
\begin{align*}
\min_{\alpha, \beta, \gamma, \kappa} & \quad C_{\text{cap}}(\kappa) + \sum_{i \in V} (\pi_i \alpha_i + \lambda_i \beta_i + \mu_i \gamma_i) \\
& + \mathbb{E}_{D, \delta} \left[ C_g \left( D - \sum_{i \in V} (\alpha_i D + \beta_i \delta_i + \gamma_i) \right) \right] \\
\text{s.t.} & \quad (2a), (2b). \quad (6)
\end{align*}
\]

Problems (5) and (6) can be solved with standard stochastic optimization techniques such as the Stochastic Subgradient Method with Monte Carlo sampling [7]. To solve the decomposed problems, we must ensure the customers’ and LSE’s decisions satisfy (4b). We achieve this by applying...
the Subgradient Method (see [6] Chapter 6) to obtain the optimal dual prices in the following

**Distributed Algorithm:**

0. **Initialization:** $\{\alpha, \beta, \gamma, u, v, w, \pi, \lambda, \mu\} = 0$.

1. **LSE:** receives $(u, v, w)$ from each customer $i \in V$.
   - Solves Problem (6) and updates $(\alpha, \beta, \gamma)$ with the optimal solution.
   - Updates the stepsize:
     \[
     \eta = \frac{\zeta}{k}
     \]
     (7)
     where $\zeta$ is a small constant and $k$ is the iteration number.
   - Updates the dual prices, $\forall i \in V$:
     \[
     (\pi, \lambda, \mu_i) := (\pi, \lambda, \mu_i) + \eta ((\alpha, \beta, \gamma) - (u, v, w))
     \] (8)
     - Sends $(\pi, \lambda, \mu_i)$ to the customer respectively.
   2. **Customer** $i \in V$: receives $(\pi, \lambda, \mu_i)$ from LSE.
   - Solves Problem (5) and updates $(u, v, w)$ with optimal solution.
   - Sends $(u, v, w)$ to the LSE.
   3. Repeat Steps 1-2 until $\|((\alpha, \beta, \gamma) - (u, v, w))\|_2 \leq \epsilon$ where $\epsilon$ is the tolerance on magnitude of the subgradient.

When the LSE signals the customers for DR and they respond accordingly, each customer is paid $\pi_i u_i + \lambda_i v_i + \mu_i w_i$ by the LSE. We now establish the convergence and optimality for the proposed distributed algorithm proved in [9]:

**Theorem 2.** The distributed algorithm’s best dual prices converge to the optimal dual prices of Problem (4).

### 4. PERFORMANCE EVALUATION

We aim to use realistic parameters in the experimental setup to validate the convergence of our distributed algorithm. We model an LSE supplying power to 300 customers for a demand response timeslot that is five minutes long and observe its social cost over a year long operation. The LSE must first purchase capacity for which we model the cost as a linear function $c_k$ with a cost parameter $c = \$1/kW$-mo. The generation cost function for the LSE is modeled as a quadratic function $A (D - \sum x_i)^2$ with the parameter $A = \$/kW^2$. For this cost function setting, a deviation of 60kW for five minutes is equivalent to an energy cost of $0.05/kWh$ and matches the intuition that larger mismatches are increasingly more expensive to manage. Each customer has a particular demand of load. To model this we utilize the traces obtained from the UMass Trace Repository which give very granular load measurements from three homes [5]. We model the cost incurred by each customer to change its consumption as a quadratic cost $a_i x_i^2$ with the parameter $a_i \in [1, 10]/kW^2$. Under these settings, a consumption decrease of 0.3kW for five minutes would cost the customer an energy price equivalent to $0.025-0.25/kWh$. To generate customer cost uncertainties we randomly choose $a_i$ from a bounded normal distribution for each customer’s estimated cost function. Renewable generation is incorporated into our simulations by using the ISO-NE’s data on hourly wind power production for the same dates as the UMass data [2]. The amount of wind capacity is scaled to 100kW. The historical data sets for each customer were generated from the available trace data (Homes A,B,C and ISO-NE wind production). They were made by bootstrapping 100 customers from each of the UMass Homes A,B,C. We also do this for the ISO-NE wind data which is first normalized by the maximum power output so that we can scale wind power accordingly. Sampling from these historical sets is how the expectation is evaluated in Equations (5) and (6).

**Convergence of the distributed algorithm.**

We consider the convergence of our distributed algorithm. Figure 1(a) illustrates that the social cost of the distributed algorithm converges quickly to that of the centralized algorithm and Figure 1(b) gives the trajectory of the total fraction of aggregate mismatch absorbed by all of the customers. It validates the convergence analysis for the distributed algorithm. For the parameters, even if we start with $\alpha_i = 0: \forall i \in V$, it quickly converges to the optimal $\alpha$ and stays there.

### 5. ACKNOWLEDGMENTS

We thank all of the reviewers for their valuable feedback. This research is supported by NSF grants CNS-1464388 and CNS-1617098.

### 6. REFERENCES


Summary of Recent Results: Crowd-Sourced Storage-Assisted Demand Response in Microgrids

Mohammad H. Hajiesmaili†, Minghua Chen†, Enrique Mallada†, and Chi-Kin Chau§

†Department of Electrical and Computer Engineering, The Johns Hopkins University
‡Department of Information Engineering, The Chinese University of Hong Kong
§Masdar Institute of Science and Technology, UAE
{hajiesmaili, mallada}@jhu.edu, minghua@ie.cuhk.edu.hk, ckchau@masdar.ac.ae

ABSTRACT

This paper studies the problem of utilizing energy storage systems to perform demand-response in microgrids. The objective is to minimize the operational cost while balancing the supply-and-demand mismatch. The design space is to select and schedule a subset of heterogeneous storage devices that arrive online with different availabilities. Designing a performance-optimized solution is challenging due to the existence of mixed packing and covering constraints in a combinatorial problem, and the essential need for online design. We devise an online algorithm and show that it achieves logarithmic bi-criteria competitive ratio. Experimental results demonstrate the effectiveness of our algorithm.

1. INTRODUCTION

We study crowd-sourced storage-assisted demand response in microgrid. In this approach, hundreds of electric vehicles and residential storages, residing in a microgrid, with huge aggregate capacity can participate in microgrid demand response through reducing their charging demand or discharging and selling back the electricity to the microgrid, e.g., through vehicle-to-grid scheme for EVs. In this way, not only the microgrid can reduce its electricity usage from main grid, but also, the customers can benefit by participating in this scheme.

We consider a scenario in which the microgrid operator solicits contribution of heterogeneous energy storage systems (or sources), such as EVs and residential batteries, in demand response through storage crowd-sourcing paradigm. After receiving all the information of the available sources, the operator selects a subset of sources and schedules their participation volume, by either reducing their charging rate or discharging, to (i) fulfill the supply shortage of microgrid, for reliable operation, and (ii) minimize total cost of involving the chosen sources, for economic operation.

Challenges. It turns out that achieving the above objectives is a formidable task since it requires solving a joint Source Selection and Scheduling Problem (S3P), which is uniquely challenging to solve because of two critical challenges:

Heterogeneity of the sources, in terms of cost, capacity, and availability in time leads to a combinatorial problem with both packing constraints, i.e., capacity constraint of the sources, and covering constraints, i.e., supply shortage of the microgrid. It turns out that a simplified version of the S3P can be re-expressed as the Capacitated Facility Location Problem (CFLP) [Levi et al. 2004], which is known to be as a fundamental theoretical CS problem [Williamson and Shmoys 2011]. The S3P is more complicated than the CFLP as it involves a topological constraint caused by the availability of sources.

The second challenge is the essential need for online solution design. In practice, the supply shortage as well as the availability of sources reveal online. The underlying problem, however, is coupled over the time, i.e., the current decision depends on input of future slots, thereby it is challenging to make online scheduling decisions without knowing future input. It turns out that in online setting, achieving a bounded performance against offline optimum without violating either packing or covering constraints is infeasible [Azar et al. 2013]. Therefore, we follow bi-criteria competitive online algorithm design paradigm [Williamson and Shmoys 2011] which jointly minimizes both cost and the amount of packing constraint (capacity) violation.

Contributions. We focus on designing an online fractional algorithm for the linear-relaxed version of the S3P. Note that even linear version of the problem is still difficult in online scenario, because the input to the time-coupled linear problem is not known in advance. By adapting the recently proposed framework for online mixed packing and covering problems [Azar et al. 2013], we propose an online fractional algorithm called OnFrc. In the OnFrc at each slot, we obtain a fractional solution for the S3P by constructing a potential function that is linear in cost and exponential in violating the capacity constraint of the storage sources.

We analyze the performance of the OnFrc using bi-criteria competitive ratio analysis\(^1\) and demonstrate that the OnFrc is a bi-criteria \(O(\log n, \log n)\)-competitive online algorithm, where \(n\) is the number of sources. By experiments using real-world data traces, we investigate the performance of the algorithm. Note that although the proposed algorithm is logarithmic competitive, this is a worst-case bound and our results demonstrate much better performance in practical settings.

2. PROBLEM FORMULATION

\(^1\)In context of our problem, a bi-criteria \((\alpha, \beta)\)-competitive online algorithm produces a solution at cost of at most \(\alpha\) times of the offline optimum, while violating the capacity constraints by no more than a \(\beta\) factor.
We assume that the system is time-slotted, where each time slot $t \in T$, $(T \triangleq |T|)$ has a fixed length (e.g., 1 hour). At slot $t$, the microgrid has a shortage $d_t \geq 0$ in supply whose value is revealed at the beginning of the slot. For the future slots, we have no assumptions on the exact or stochastic modeling of $d_t$. Let $I$, $(n \triangleq |I|)$, be the set of energy storage systems (sources, used interchangeably) in the microgrid that are available to contribute in demand response scheme. By storage system, we mean any devices like EVs and residential batteries that can be connected to the microgrid and participate in demand response by either reducing their charging rate or discharging back to the microgrid. The sources are heterogeneous in terms of availability over time horizon, capacity, and operating cost. Source $i$ is available in interval $T_i \subseteq T$, where $T_i = [a_i, b_i]$ and $a_i$ and $b_i$ are the arrival and departure slots. This captures the availability of sources, e.g., EVs are available in different intervals in parking lots. In our model, we also assume sources arrive online. Source $i$ announces total capacity $c_i$ which can be used arbitrarily in its availability window for demand response scheme. Cost model of source $i$ is: (i) a fixed cost $f_i$, as the participation cost, which is fixed value regardless of the amount of energy that is solicited, and (ii) a unit cost $u_i$ which must be multiplied by the volume of energy that is contributed by source $i$.

Given the set of heterogeneous sources, the objective is to use the potentials of the available sources by selecting a subset of them such that by a proper scheduling, the supply shortage during time horizon is covered, and at the same time the aggregate fixed and unit costs are minimized. The optimization problem is a joint source selection and scheduling problem (S3P) that is formulated as

$$\text{S3P: } \min \sum \limits_{i \in I} \left( f_i x_i + u_i \sum \limits_{t \in T_i} y_i(t) \right)$$

s.t. \hspace{1cm} \sum \limits_{t \in T_i} y_i(t) \leq c_i x_i, \quad \forall i \in I, \quad (1a)$$

$$\sum \limits_{i \in I, t \in T_i} y_i(t) \geq d_t, \quad \forall t \in T, \quad (1b)$$

vars. $x_i \in \{0, 1\}, \quad \forall i \in I,$

$y_i(t) \geq 0, \quad \forall i \in I, t \in T,$

where optimization variables are $x_i$ and $y_i(t)$. $x_i = 1$, if source $i$ is selected; $x_i = 0$, otherwise. In addition, $y_i(t)$ denotes the amount of energy that is covered by source $i$ at time $t$. Constraint (1a) is the capacity (packing) constraint of the sources. Constraint (1b) is the covering constraint that guarantees that total acquired energy by the chosen sources covers the shortage at each slot. Finally, note that the S3P is an NP-hard mixed integer linear program which is difficult to solve, in general, even in offline setting. Recall that the S3P without availability limit of the sources is equivalent to the CFLP, which is generally difficult to tackle even in offline setting.

Our design is even more challenging since the S3P requires online solution design. The online inputs in our problem are two-fold. First, supply shortage $d_t$ is revealed in slot-by-slot fashion. Second, the sources arrive online. In this way, all the characteristics (cost, capacity, and departure time) of available sources reveal at the beginning of each time slot. In terms of underlying optimization problem, both packing and covering constraints arrive online.

### 3. Online Solution Design

Since the S3P encounters mixed packing and covering constraints, to achieve a competitive ratio better than $O(n)$ in online scenario, it is inevitable that either packing or covering constraint is violated [Hajiesmaili et al. 2017]. In demand response, however, it is critical that the shortage is fulfilled by the chosen sources. Hence, in our online algorithm design, we force the covering constraint to be respected, and as a result, violation of capacity constraints of the sources is permitted. As such, our goal is to minimize the capacity violation of selected sources, in addition to total cost minimization.

In this work, we briefly explain our online algorithm design for the linear version of the S3P. In [Hajiesmaili et al. 2017], we propose a randomized rounding approach to find an integral solution based on the fractional solution obtained from linear S3P. We skip the details of randomized integral solution due to space limit and refer to [Hajiesmaili et al. 2017].

We assume that the number of time slots ($T$) is known in advance and the optimal offline cost $\text{Opt}$ is given. In general, $f_i \leq \text{Opt}, \forall i \in I$, otherwise, we exclude the sources with fixed cost greater than $\text{Opt}$. Now, we introduce $\hat{f}_i$ as the scaled fixed-cost of source $i$ as $\hat{f}_i = \max\{(f_i/\text{Opt}) \cdot 1, 1\}$, and $\hat{u}_i(t)$ as the normalized unit cost of the source $i$ at time $t$ as $\hat{u}_i(t) = (u_i d_i n) / \text{Opt}$. Finally, we define $d_i(t) = d_i / c_i$. Note that by multiplying fixed and unit cost parameters by $n/\text{Opt}$, the optimal value of the problem changes to $3$These assumptions are reasonable since $T$ is fixed usually.

The optimal offline cost also can be estimated based on historical data. Nevertheless, the algorithm can be extended to the case that the optimal offline cost is not known at the expense of adding a multiplicative logarithmic order in competitive ratio [Azar et al. 2013].

---

**Algorithm 1: OnFrc- Online Fractional Algorithm, at time $t$**

1. **Initialization**
2. $I_t \leftarrow$ an ordering of sources that are available in $t$ in ascending order of $v_i(t)$ in Eq. (2)
3. $P_t \leftarrow$ the maximal subset of $I_t$ such that $\sum_{i \in P_t} x_i < 1$
4. $I_t \leftarrow$ the first source in $I_t$ that is not in $P_t$
5. while $\sum_{i \in P_t \cup \{l_t\}} z_i(t) < 1$
6. \hspace{1cm} **foreach** $i \in P_t \cup \{l_t\}$
7. \hspace{2cm} if $i \in P_t$ or ($i = l_t$ and $x_i < 1$) then
8. \hspace{3cm} $x_i \leftarrow \min\{x_i + z_i(t)/T, 1\}$
9. \hspace{3cm} $z_i(t) \leftarrow \min\{2x_i, \frac{c_i - 1}{d_i(t) + u_i(t)/T}\}$
10. \hspace{2cm} end
11. \hspace{1cm} if $i = l_t$ and $x_i = 1$ then
12. \hspace{2cm} $z_i(t) \leftarrow z_i(t) + 1/v_i(t)T$
13. \hspace{1cm} end
14. **end**
15. **end**
O(n), facilitating the competitive analysis. Let \( x_i \geq 0 \) be the relaxed integer source selection variable. Moreover, \( z_i(t) \in [0, 1] \) is the portion of supply shortage \( d_i \) that is fulfilled by source \( i \), provided that \( t \in T_i \). Indeed, \( y_i(t) = d_i z_i(t) \).

The proposed online competitive algorithm, summarized as Algorithm 1, accomplishes source selection and scheduling by providing an ascending ordering among the available sources at each slot. In this way, our main endeavor is to construct a metric to be used to sort the sources. The sorting metric \( v_i(t) \) which we refer to it as virtual cost of source \( i \) at time \( t \) is defined as follows.

\[
v_i(t) = \begin{cases} \hat{f}_i \theta \gamma_i(t) - d_i(t) + \hat{u}_i(t), & \text{if } x_i = 1, \\ \hat{f}_i d_i(t) + \hat{u}_i(t), & \text{otherwise}, \end{cases}
\]

where \( \theta > 1 \) is a constant factor and \( \gamma_i(t) = \sum_{\tau \in T_i, \tau \leq t} d_i(\tau)z_i(\tau) \) is the current congestion level of source \( i \). Hence, we call \( \theta \) the congestion parameter. By this definition, when source \( i \) is not fully chosen \( (x_i < 1) \), the cost is linear in both normalized fixed and unit cost. On the other hand, if source \( i \) is already fully chosen \( (x_i = 1) \) in the fractional solution due to the scheduling in the previous time slots, the virtual cost is linear in fixed cost, however exponential in the congestion level \( \gamma_i(t) \). The following theorem characterizes the competitive ratio of the OnFrc.

**Theorem 1.** [Hajiesmaili et al. 2017] Given \( 1 < \theta < 1.5 \), OnFrc generates a fractional schedule that is \( O(\log n, \log n) \)-competitive.

### 4. PERFORMANCE EVALUATIONS

In this section, we report a selected set of results of the online fractional algorithm (in Sec. 3) and the randomized rounding algorithm in [Hajiesmaili et al. 2017] that is built on top of the fractional one and leads to an integral solution. The electricity data traces are from [Commission 2006] and we assume that on average 10% of the demand is regarded as supply shortage in each slot. The unit cost for each source is randomly generated in \([10, 70]\)kWh. We set \( T = 12 \) and the length of each slot to 1 hour. We compare the result of online algorithms to the offline optimum.

In Fig. 1, we report the total costs of offline optimal solution and our fractional and integral algorithms as a function of different input parameters. The result in Fig. 1(a) shows that as the number of sources increases, total cost of all algorithms decreases, which is reasonable since with the increase in sources, there is more freedom to pick more cost-effective sources. The average cost ratios, i.e., the cost of the algorithms over the offline optimums cost, for the fractional and integral algorithms are 1.56 and 1.71, respectively which demonstrate sound performance of our algorithms. The obtained empirical cost ratios demonstrate that our algorithms can achieve much better results than those obtained in theoretical analysis. The difference between the cost of fractional and integral algorithms is due to the integrality gap made by randomized rounding [Hajiesmaili et al. 2017]. Fig. 1(b) shows that as the number of slots increases, total cost increases for all solutions. This is reasonable since with fixed number of sources as the number of slots increases, more demand must be covered and hence total cost increases. The results in Fig. 1(c) show that as the capacity of sources scales, total cost decreases since each source can cover more supply shortage.

### 5. CONCLUSIONS

This paper advocates the idea of using the potentials of existing sources in a microgrid to perform crowd-sourced storage-assisted demand-response. It formulates a joint problem of source selection and scheduling with the goal of minimizing the cost, while respecting mixed packing and covering constraints. An efficient online competitive algorithm for the problem is devised and experiments show that the performance of the algorithm is near optimum. The underlying problem could be imagined as a natural expansion of the minimum knapsack problem over time, and one can find several applications rather that demand-response in microgrids. In fact, our formulation makes sense in any application in which different sources can contribute in fulfilling a service that arrives in time.

### 6. REFERENCES


Optimal Energy Procurement for Geo-distributed Data Centers in Multi-timescale Electricity Markets

Tan N. Le\textsuperscript{1,2}, Jie Liang\textsuperscript{1}, Zhenhua Liu\textsuperscript{1}, Ramesh K. Sitaraman\textsuperscript{3,4}, Jayakrishnan Nair\textsuperscript{5}, Bong Jun Choi\textsuperscript{1,2}, Stony Brook Univ.\textsuperscript{1}, SUNY Korea\textsuperscript{2}, Univ. of Massachusetts at Amherst\textsuperscript{3,4}, Akamai Tech.\textsuperscript{4}, IIT Bombay\textsuperscript{1}

ABSTRACT

Heavy power consumers, such as cloud providers and data center operators, can significantly benefit from multi-timescale electricity markets by purchasing some of the needed electricity ahead of time at cheaper rates. However, the energy procurement strategy for data centers in multi-timescale markets becomes a challenging problem when real world dynamics, such as the spatial diversity of data centers and the uncertainty of renewable energy, IT workload, and electricity price, are taken into account. In this paper, we develop energy procurement algorithms for geo-distributed data centers that utilize multi-timescale markets to minimize the electricity procurement cost. We propose two algorithms. The first algorithm provides provably optimal cost minimization while the other achieves near-optimal cost at a much lower computational cost. We empirically evaluate our energy procurement algorithms using real-world traces of renewable energy, electricity prices, and the workload demand. Our empirical evaluations show that our proposed energy procurement algorithms save up to 44% of the total cost compared to traditional algorithms that do not use multi-timescale electricity markets or geographical load balancing.

1. INTRODUCTION

Data centers are becoming the largest and the fastest growing consumers of electricity in the United States. It is reported that US data centers consumed 91 billion kilowatt-hours (kWh) in 2013, which is more than twice of the electricity consumed by households in New York City (see [15]). In the same report, the electricity consumption of the data centers is estimated to reach 140 billion kWh in 2020 due to the explosion of demand for cloud computing and other Internet-scale services.

Multi-timescale electricity markets have been proposed to improve the efficiency of electricity markets [4]. Multi-timescale electricity markets encompass both forward (futures) and spot (real-time) markets. While energy is procured at the time of consumption in a spot market, forward markets allow customers to buy electricity a day ahead or even several months ahead of when it is consumed. Forward electricity markets reduce the risk for both the supplier and consumer by reducing the quantity of energy being traded in the real-time spot markets [2]. Furthermore, purchasing electricity ahead of time can facilitate the expansion of renewable energy sources [6].

Utilizing multi-timescale markets has great potential for electricity cost savings of cloud providers who operate one or more data centers. There has been much recent work that exploits the variation of real-time electricity prices in the temporal and spatial dimensions to reduce the total electricity cost [12, 13]. Other papers exploit temporal variation in the real-time energy price and use energy storage to reduce the electricity costs [7, 17], i.e., the storage device is charged during the times when the electricity price is low and discharged when the price is high. However, while these works focus on traditional real-time markets, our paper studies the potential of using multi-timescale markets in the context of a cloud provider.

In particular, using forward markets to lower the electricity cost for a cloud provider is challenging for multiple reasons. The optimal amount of electricity that a cloud provider should purchase in advance for a particular location depends on the workload, the availability of onsite renewables, and the real-time electricity price at that location. The main challenge is that the future workloads, renewables, and real-time electricity prices are not perfectly predictable and are subject to significant forecasting errors. Note that if the cloud provider is too conservative and buys too little from the forward market, any shortfall in electricity would need to be covered by purchasing it from the more expensive\textsuperscript{1} real-time market. Likewise, if the cloud provider is too aggressive and buys too much from the forward market, any excess in electricity will go to waste. In addition, the ability of a cloud provider to move the load from one data center to another, possibly incurring a performance penalty that we characterize as the “delay cost”, adds an additional level of complexity that needs to be optimized. In this work, we provide an optimization framework for tackling the aforementioned challenges. There are a few recent papers that consider forward markets; these papers deal with the financial risk of a single data center arising from the uncertainties in electricity prices and workload [14, 16]. Geographical load balancing systems with both day-head market and real-time market have been studied in a recent publication [5]. However, the proposed solution is somewhat restrictive to particular distributions to facilitate stochastic optimization.

\textsuperscript{1}In some cases, the prices in the forward markets might be (on average) higher than real-time prices. If so, instead of saving electricity expenditure, the cloud provider can participate in forward markets to reduce cost variations. Our model can be extended to handle either case.
and does not provide any optimality guarantee. Our contributions are two-fold.

(1) **Optimal algorithm development.** We develop two algorithms for a cloud provider with geo-distributed data centers to buy electricity in multi-timescale markets: one algorithm provides optimality guarantees, while the other is simpler in that it uses limited predictions but achieves near-optimal performance. To develop the energy procurement system, we first model the problem of procuring electricity for geo-distributed data centers in multi-timescale markets in Section 2. The system model is general and applicable to any global cloud provider with access to multi-timescale electricity markets. We focus on two-timescale markets that consist of one long-term market and one real-time market. However, our model and algorithms can be extended to handle multiple markets at various timescales. We present the characteristics of the objective functions and the optimal solution in Section 3, which forms the theoretical basis for our algorithm design. The two algorithms that we design, prediction based algorithm (PA) and stochastic gradient based algorithm (SGA), are described in Section 4.

(2) **Empirical evaluation.** We carry out a detailed empirical evaluation of our proposed energy procurement systems using real world traces. In Section 5, we demonstrate that SGA can converge to the optimal solution in a small number of iterations. Moreover, we show that PA, our heuristic algorithm, surprisingly achieves a near-optimal solution. This is partially because the real-time optimization takes into consideration the trade-off between energy cost and delay cost, and is able to compensate for some prediction errors by redirecting workloads. The proposed energy procurement systems are compared with other comparable energy procurement strategies to highlight their benefits.

## 2. MODEL

### 2.1 System model

**Two-timescale markets.** A service provider operating geo-distributed data centers can purchase electricity in two markets—a long-term market and a real-time market. The electricity consumed at time \( t = 0 \) must be procured from the real-time market at \( t=0 \) and/or from the long-term market \( t \leq -T_l \).

**Geo-distributed data centers.** We consider a set \( N \) of geo-distributed data centers serving workload demands from a set \( J \) of sources as illustrated in Figure 1. The workload demand from each source is split between the \( |N| \) data centers. Here, a source can represent the aggregate demand from a group of local users, such as users of a particular city, ISP, or geographical region. Each data center has access to renewable energy sources. Further, each data center participates in a (local) long-term electricity market and a (local) real-time electricity market. In other words, each data center \( i \) can buy electricity ahead of time in its long-term market, and can also buy additional electricity in its real-time market if necessary.

**Energy procurement system (EPS).** Our proposed energy procurement system for geo-distributed data centers is depicted in Figure 2. There are three main components, namely, the long-term forecaster, the energy procurement (EP) in long-term markets and the geographical load balancing (GLB). The long-term forecaster provides the forecasted information for the energy procurement. The forecasted information includes the predicted values and the prediction error distributions of IT workload, renewable energy generations, and electricity prices. The EP component procures electricity for each data center in the corresponding long-term markets (at time \( t = -T_l \)) based on the electricity prices in the long-term markets and forecasts of real-time prices, workload, and renewable generation. The GLB component (at time \( t = 0 \)) distributes (routes) the realized workload from sources to data centers, provisions the required computing capacity at each data center, and procures additional electricity as needed in the real-time markets.

**Data center.** Let \( M_i \) denote the number of servers in data center \( i \). The number of active servers at real-time (time \( t = 0 \)) is denoted by \( m_i \), which is a control parameter. In practice, there can be more than a hundred thousand servers in a single data center. Thus, we treat \( m_i \) as a continuous number satisfying \( 0 \leq m_i \leq M_i \).

At time \( t = 0 \), the power consumption of data center \( i \) is denoted by \( d_i^r \). In general, the power consumption of data center \( i \) is dependent on the number of active servers \( m_i \) and the workload arrival \( \lambda_i \). For simplicity, we assume that \( d_i^r = m_i \), which implies that the power consumption is proportional to the number of active servers, and is independent of the workload \( \lambda_i \).

**Workload.** Workload demand from source \( j \) in real-time ( \( t = 0 \) ) is denoted as \( L_j^r \). We assume that the exact realization of the random vector \( L_j = (L_j^r, j \in J) \) is known to the cloud provider at time \( t = 0 \), and is an input to GLB. Let \( \lambda_{ij} \) denote the distributed workload arrival from

---

2The proportionality constant relating the number of active servers and the power consumption is taken to be 1 without loss of generality. Also, our analysis can be easily generalised to the case \( d_i^r = O(\lambda_i, \lambda_j) \), where the function \( O_i \) is continuously differentiable, convex, and non-decreasing in each coordinate.
source $j$ to data center $i$ at time $t = 0$ (set by GLB). Thus, $L_j^0 = \sum_{i \in N} \lambda_{ij}$ ($j \in J$), and $\lambda_i = \sum_{j \in J} \lambda_{ij}$ ($i \in N$).

**Renewable energy.** Data centers can utilize their integrated renewable energy sources. Let $w_i^r$ denote the renewable energy generation at data center $i$ in real-time ($t = 0$).

We assume that the exact realization of the random vector $w^r = (w_i^r, i \in N)$ is known at time $t = 0$, and is an input to GLB.

**Electricity price.** For each data center, the cloud provider can purchase electricity at time $t = -T_l$ in the local long-term market and then purchase any additional electricity needed in the local real-time market at time $t = 0$. For data center $i$, let $p_i^r$ denote the long-time price for 1 unit of electricity, and $p_i^l$ denote the real-time price for 1 unit of electricity. We assume that $p_i^r = (p_i^r, i \in N)$ is fixed (or equivalently, is known at the time of the long-term procurement), and $\mathbf{p}^l = (p_i^l, i \in N)$ is a random vector whose exact value is known at time $t = 0$ and is an input to GLB.

### 2.2 Cost model

The total cost of operating geo-distributed data centers is composed of a delay cost and an energy cost.

**Delay cost.** The delay cost represents the monetary cost incurred due to the delay experienced by the sources. We model the delay cost $h_{ij}(m_i, \lambda_i)$ of routing and processing each unit of workload from source $j$ to data center $i$ as follows:

$$h_{ij}(m_i, \lambda_i) = \beta \left( \frac{1}{\mu_i - \lambda_i/m_i} + \pi_m \right) \quad (\lambda_i < m_i \mu_i) \quad (1)$$

Here, the parameter $\beta$ weighs the delay relative to the energy cost. The first term above captures queuing delay at delay center $i$, which is based on the well-known mean delay formula for the M/GI/1 processor sharing queue; $\mu_i$ is the service rate of servers in data center $i$. The second term captures the network delay from source $j$ to data center $i$.

Note that for stability, we need that $\lambda_i < m_i \mu_i$. Our delay cost model assumes a linear relationship between delay and its associated monetary cost, as is suggested in [3].

**Energy cost.** Let $q_i^r$ and $q_i^l$ respectively denote the amount of electricity purchased in the long-term market and the real-time market by data center $i$. Here, we require that sufficient electricity is procured to process the workload routed to each data center as

$$q_i^r + w_i^r + q_i^l \geq d_i^r = m_i \quad (i \in N).$$

The electricity bills of data center $i$ in the long-term and real-time market are respectively computed as

$$R_i^r(q_i^r) = p_i^r q_i^r \quad i \in N, \quad R_i^l(q_i^l) = p_i^l q_i^l \quad i \in N.$$

### 2.3 Formulation of optimal energy procurement in multi-timescale markets

Recall that the total cost of operating geo-distributed data centers in our two-timescale market setting is the sum of the energy cost and the delay cost, given by

$$F = \sum_{i \in N} R_i^r(q_i^r) + \sum_{i \in N} R_i^l(q_i^l) + \sum_{i \in N, j \in J} \lambda_{ij} h_{ij}(m_i, \lambda_i).$$

We seek to minimize $\mathbb{E}[F]$ subject to the aforementioned constraints. Note that this optimization is performed on two timescales, with different sets of information available at each. The EP component optimizes the long-term procurements $q_i^r = (q_i^r, i \in N)$ given only distributional information of the real-time workload $L'$, the real-time renewable generation $w^r$, and the real-time electricity prices $p^l$. The GLB component optimizes the workload routing $\lambda = (\lambda_{ij}, i \in N, j \in J)$, the number of active servers $m = (m_i, i \in N)$ at the data centers, and the real-time procurements $q_i^l = (q_i^l, i \in N)$ given the prior long-term procurements $q_i^r$, and the exact realization of $(p^l, L', w^r)$. Below, we first formalize the real-time optimization, followed by the long-term optimization.

**Geographical load balancing in real-time markets.** Note that in real-time, GLB optimizes the real-time procurements $q_i^l$, the numbers of active servers $m$, and the workload routing $\lambda$, given the long-term procurements $q_i^r$ and the realization of the random vector $(p^l, L', w^r)$. The total cost as seen by GLB is

$$F^r(q_i^r, m, \lambda, p^l) := \sum_{i \in N} R_i^r(q_i^r) + \sum_{i \in N, j \in J} \lambda_{ij} h_{ij}(m_i, \lambda_i).$$

Thus, the real-time optimization is defined as follows.

**GLB-RT:**

$$\min_{m, \lambda, p^l} \quad F^r(q_i^r, m, \lambda, p^l)$$

s.t. \begin{align*}
\lambda_{ij} & \geq 0 \quad \forall i \in N, j \in J \\
\sum_{i \in N} \lambda_{ij} &= L_j^r \quad \forall j \in J \quad (2a) \\
\lambda_i & \leq m_i \mu_i, \quad \forall i \in N \quad (2b) \\
0 & \leq m_i \leq M_i, \quad \forall i \in N \quad (2d) \\
q_i^l & \geq 0, \quad \forall i \in N \quad (2e) \\
m_i - q_i^r - w_i^r & \leq q_i^l \quad \forall i \in N. \quad (2f)
\end{align*}$$

Since $p_i^l \geq 0$, it easily follows that any solution of the above optimization problem satisfies $q_i^l = [m_i - w_i^r - q_i^l]_+$, where $[x]_+ := \min\{0, x\}$. Thus, the real-time objective can be re-written as

$$F^r(q_i^r, m, \lambda, p^l, w^r) = \sum_{i \in N} p_i^l \left[ m_i - w_i^r - q_i^l \right]_+ + \sum_{i \in N, j \in J} \lambda_{ij} h_{ij}(m_i, \lambda_i). \quad (3)$$

**GLB-RT problem** is a convex optimization problem and hence can be solved efficiently using standard methods [11].

**Energy procurement in long-term markets.** At time $t = -T_l$, the cloud provider purchases electricity $q_i^r$ in long-term markets that will be used at real-time. Note that optimization of the long-term procurements has to be performed based only on distributional information for the random vector $(p^l, L', w^r)$, and subject to the real-time optimization that will be subsequently performed based on the realization of the random vector $(p^l, L', w^r)$.

Let us denote the optimal value of the optimization GLB-RT by $F^{\ast r}(q_i^r, p^l, L', w^r)$. The long-term objective is thus defined as

$$F^l(q_i^l) := \sum_{i \in N} R_i^l(q_i^l) + \mathbb{E} \left[ F^{\ast r}(q_i^r, p^l, L', w^r) \right].$$

Note that the above expectation is with respect to the random vector $(p^l, L', w^r)$. The long-term optimization problem is then given by:

**EP-LT:**

$$\min_{q_i^l} F^l(q_i^l)$$

s.t. $q_i^l \in \mathbb{R}^N_+$. 

The above optimization is more challenging than GLB-RT. In Section 3, we prove that EP-LT is a convex optimization and characterize the gradient of the objective function. These results are then used to arrive at a provably optimal stochastic gradient algorithm in Section 4.

3. CHARACTERIZING THE OPTIMA

In this section, we collect useful properties of the optimizations EP-LT and GLB-RT. These are important for understanding the behavior of the energy procurement system, and also for proving convergence of the stochastic gradient algorithm for EP-LT in Section 4.

Our first result is that EP-LT is indeed a convex optimization, which suggests that EP-LT is a tractable optimization.

**Theorem 1.** $F^i(q^i)$ is convex over $q^i \in \mathbb{R}_+^N$.

We provide the proof of Theorem 1 in Appendix A.1 of the full version [9]. Next, we characterize the gradient of the EP-LT objective function as follows.

**Theorem 2.** The gradient of $F^i(\cdot)$ is characterised as follows.

$$
\frac{\partial F^i(q^i)}{\partial q^i_l} = p^i_l + E \left[ \frac{\partial F^{\infty}(q^i, p^i, L^i, w^i)}{\partial q^i_l} \right]
$$

$$
= p^i_l - E \left[ \varrho_i(q^i, p^i, L^i, w^i) \right],
$$

where $\varrho_i(q^i, p^i, L^i, w^i)$ is the unique Lagrange multiplier of GLB-RT corresponding to the constraint (2f).

Note that the first equality in the theorem statement asserts that the order of an expectation and a partial derivative can be interchanged. The second equality relates the partial derivative of $F^\infty$ with respect to $q^i_l$ to a certain Lagrange multiplier of GLB-RT. We provide the proof of Theorem 2 in Appendix A.2 of the full version [9].

We note that Theorem 2 does not enable us to compute the gradient of the $F^i(\cdot)$ exactly. Indeed, the expectation the Lagrange multiplier $\varrho_i$ with respect to $(p^i, L^i, w^i)$ would in general be analytically intractable. However, Theorem 2 does enable a noisy estimation of the gradient of the $F^i(\cdot)$ via Monte Carlo simulation as follows. Suppose we simulate a finite number, say $M$, of samples from the distribution of $(p^i, L^i, w^i)$. In practice, we can obtain these samples by using real-world traces. For each sample, the Lagrange multipliers $(\varrho_i, i \in N)$ can be computed efficiently by solving GLB-RT. By averaging the $M$ instances of $(\varrho_i, i \in N)$ thus obtained, we get an unbiased estimate of the gradient of $F^i(\cdot)$. This, in turn, enables us to solve EP-LT using a stochastic gradient descent method; details follow in Section 4.

As there are two timescales in optimization, it is critical to investigate how EP-LT affects the operation of geographical load balancing in real-time. We start by answering the following question: how does the long-term procurement $q^i$ impact the number of active servers $m_i$ in data center $i$? Formally, we have the following intuitive result:

**Lemma 3.** At any data center $i$, an optimal solution always utilizes the long term energy procurement $q^i$, and renewable generation $w^i$ as much as possible. It is simply represented by

$$
m_i \geq w^i + q^i \quad \text{if } w^i + q^i < M_i,
$$

$$
m_i = M_i \quad \text{if } w^i + q^i \geq M_i.
$$

**Proof.** Appendix A.3 of the full version [9].

The above lemma states that a data center $i$ uses up the reserved electricity, including free renewable energy and pre-purchased electricity, because doing so reduces the queueing delay.

4. ALGORITHM DESIGN

The energy procurement system needs algorithms for both energy procurement in long-term (EP-LT) and geographical load balancing in real-time (GLB-RT). GLB-RT is a convex optimization problem that can be solved efficiently in real-time by standard techniques [11]. Thus, we focus on designing algorithms for energy procurement in the long-term markets. Note that even though EP-LT is a convex optimization (see Theorem 1), neither the objective function nor its gradient admit a closed-form representation, which presents significant challenges.

4.1 Prediction based Algorithm (PA)

Prediction based algorithm (PA) relies on the mean values of renewable generation, workload, and electricity price. The predicted values $L^i_j, w^i, \hat{p}^i$ are the estimates of the mean values of renewable generation, workload, and electricity price.

PA computes the long-term procurement $q^i$ by solving EP-LT and GLB-RT at the same time, with the random variables $w^i, L^i_j, \hat{p}^i$ replaced by their predicted values. Formally, this is done by solving the following deterministic convex optimization problem.

$$
\text{LT-PA: } \min_{m, \lambda, q^i} \sum_{i=1}^N p^i_l q^i_l + \sum_{i=1}^N \hat{p}^i [m_i - w^i - q^i_l]_+ + \beta \sum_{i=1}^N \sum_{j \in J} h_{ij}(m_i, \lambda_{ij})
$$

$$
s.t. \ (2a), \ (2c)-(2e) \sum_{i \in N} \lambda_{ij} = L^i_j \quad \forall j \in J
$$

$$
\lambda_{ij} \geq 0 \quad \forall i \in N
$$

The objective function of LT-PA is similar to that of the EP-LT without the expectation operation. The constraints over $m$, $\lambda$, and $q^i$ of LT-PA are identical to those of GLB-RT and EP-LT. LT-PA is a convex optimization problem and can be solved efficiently by standard techniques.

4.2 Stochastic Gradient-based Algorithm (SGA)

Although PA can offer a quick heuristic decision, it is desirable to have an algorithm that optimally procures electricity in long-term markets. To this end, we exploit the gradient characterization of the long-term objective (see Theorem 2) to design a stochastic gradient descent algorithm. The algorithm, namely, SGA, is summarized in Algorithm 1. The main idea of the algorithm is to compute a noisy estimate of the gradient of the long-term objective by averaging the gradient of the (random) total cost over a finite number of sample paths. This noisy gradient is used to perform a stochastic gradient descent. Stochastic approximation theory can then be used to prove convergence to the set of optimal solutions, as long as the step-size sequence is appropriately diminishing [8].
The electricity prices in real-time markets are the industrial electricity prices of each state in May 2010 [1]. We set the long-term prices such that the ratio $\frac{\bar{p}_t}{\bar{p}_t}$ = 2.5. To simulate the uncertainties, the error distributions from the full version [9] are used to generate the samples of renewable energy generation (wind generation), workload, and electricity price. The mean absolute errors (MAE) of prediction errors for wind generation, electricity price, and workload demand are 65%, 40%, and 35%, respectively. The penetration of renewable penetration is 50%.

![Figure 3: The proposed algorithms PA and SGA are very close to the lower bound, OA and outperform the traditional methods up to 44%.](image)

Cost savings. We highlight the benefit of our proposed system by comparing with the baselines in Table 1. Nearest is the naive load balancing scheme that forwards the workload to the closest data centers. Fixed LT just pre-purchases electricity for a certain percentage of workload in long-term markets. In addition to the baseline algorithms, we compare our algorithms to Oracle Algorithm (OA). OA is an unrealizable algorithm that is given to the absolute performance limit by assuming assumes all realizations of renewable energy, workload, and electricity prices are fully known apriori. Similarly to PA, the problem of long-term procurement can then be solved efficiently. The cost of OA is measured by averaging its output over many realizations.

We give the proof of Theorem 4 in Appendix B of [9].

The convergence of SGA is asserted by the following theorem.

THEOREM 4. Under Assumption 1, almost surely, the iterates $\mathbf{q}^i$ generated by SGA converge to the set of optimal solutions of EP-LT as $\tau \to \infty$.

We prove that SGA converges to the set of optimal solutions of EP-LT under the following standard assumption on the step-size sequence.

ASSUMPTION 1. $\sum_{\tau=1}^{\infty} (\eta_\tau) = \infty$ and $\sum_{\tau=1}^{\infty} (\eta_\tau)^2 < \infty$.

The bottleneck of SGA is the computation of the noisy gradient estimate, which involves solving $S$ instances of GLB-RT. Moreover, the diminishing step-size sequence implies that SGA requires a large number of iterations to compute a near-optimal solution. However, it is important to note that since this algorithm is only used for long-term procurement, its computation time would not be a bottleneck in practice.

5. NUMERICAL RESULTS

Experimental Setup. There are 10 logical data centers which are located in 10 different states. We assume that there are one million servers distributed across the ten logical data centers. The peak power consumption for each server is 300W. We consider 40 sources, corresponding to 40 states of the US; the corresponding workload data is obtained from Akamai Technologies. The average workload is 30% of the total capacity of the data centers. The network delays are estimated to be proportional to the distance between sources and data centers. The importance of delay is estimated according to the fact that 100 ms latency costs 1% of Amazon in sales [10].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EP-LT</th>
<th>fixed LT</th>
<th>GLB-RT</th>
<th>Nearest</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLTnGLB</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>fLTnGLB</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>nLT</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>fLT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Mean Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>long-term energy cost</td>
<td>1.5</td>
</tr>
<tr>
<td>real-time energy cost</td>
<td>2.0</td>
</tr>
<tr>
<td>queueing delay cost</td>
<td>1.0</td>
</tr>
<tr>
<td>network delay cost</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1: Baseline algorithms.
and the compensation of GLB-RT at real-time markets. The details can be found in the full version [9].

**Why do our proposed algorithms perform so well?**

The intuition behind the small performance gaps between PA, SGA and OA is the compensation of GLB-RT at real-time markets. In particular, GLB-RT can utilize the available renewable energy and cheap electricity to partially compensate for performance gap caused by the prediction errors in long-term. More interestingly, PA and SGA are noticeably aggressive in long-term markets as in Figure 3. In addition, PA and SGA are even more aggressive than OA. In fact, Lemma 3 allows PA, SGA, and OA to purchase a lot of electricity in long-term markets, because the over-provisioned energy can be used up to reduce queuing delay in real-time. Thus, there is the trade-off between the energy costs and delay costs that helps our proposed methods become close to OA.

**6. CONCLUDING REMARKS**

In this paper, we present a systematic study of optimal energy procurement systems for geo-distributed data centers that utilize multi-timescale electricity markets. The contributions of this paper are three-fold: (i) designing algorithms for long-term electricity procurement in multi-timescale markets; (ii) analyzing long-term prediction errors using real-world traces; and (iii) empirically evaluating the benefits of our proposed procurement systems. In particular, we proposed two algorithms, PA and SGA, both of which save up to 44% of the energy procurement cost compared to traditional algorithms that do not use long-term markets or geographical load balancing. While SGA provably converges to an optimal solution, PA surprisingly achieves a cost that is nearly optimal with much less computing effort.

**7. ACKNOWLEDGEMENTS**

This research is supported by NSF grants CNS-1464388, CNS-1617698, and CNS-1413998. This research was partially funded by the MSIP, Korea, under the “ICT Consilience Creative Program” (IITP-2015-R0346-15-1007) supervised by the IITP and by the NRF, Korea, under the “Basic Science Research Program” (NRF-2015R1C1A1A01053788).

**8. REFERENCES**

Phase Balancing in Power Distribution Network with Data Center

Wei Wang  
Dept. of Electrical and Computer Engineering  
University of California, Riverside  
wwang031@ucr.edu

Nanpeng Yu  
Dept. of Electrical and Computer Engineering  
University of California, Riverside  
nyu@ece.ucr.edu

ABSTRACT
High degree of unbalance in electric distribution feeders can significantly affect power quality, damage electrical equipment, and result in tripping of protective devices. If not properly managed, integration of new data center and distributed energy resources into the power distribution network will exacerbate the problem. This paper proposes a new paradigm which coordinates the operation of data center and distributed energy resources to reduce phase unbalance and improve the reliability and efficiency of electric distribution networks. The coordination scheme is implemented within the framework of a distribution system operator managed electricity market. The proposed phase balancing algorithm with data center is validated using a modified IEEE distribution test feeder. The simulation results show the proposed data center and distributed energy resources coordination scheme not only significantly reduces the degree of unbalance of distribution feeders but also results in sizable reduction in data center electricity costs.

Keywords
Demand response, phase balancing, data center, power distribution network

Nomenclature

- $P_{D_q}$: Real power of total demand at node $q$ with phase $g$.
- $P_{dc}$: Real power demand of data center on phase $m$.
- $P_{dyn}$: Dynamic power demand of the $i$-th server on phase $p$.
- $P_i$: Flexible loads at node $i$ with phase $m$.
- $PG$: Real power of generation at the reference bus on phase $m$.
- $P_{idle}$: Idle power demand of the $i$-th server on phase $p$.
- $P_{loss}$: Total real power losses on phase $p$.
- $PLimit$: Real power flow limit between node $i$ and $k$ with phase $p$.
- $P_k$: Real power demand of the $i$-th server on phase $p$.
- $R^p$: Vector of equivalent requests of moving VMs from the servers on phase $p$ to phase $r$.
- $T_p$: The response time of the servers on phase $p$.
- $TSLA$: The service level agreement limit.
- $\gamma$: Power imbalance limit between phases.

1. INTRODUCTION
With the cloud-computing industry experiencing exponential growth, data center is becoming a significant electricity consumption source. According to the U.S. data center energy usage report [12], an estimated 70 billion kWh of electricity is consumed by data center in 2014 representing about 1.8% of total U.S. electricity consumption. The electricity usage by data center is also expected to increase 4% annually in the next five to ten years. Electricity is also the fastest growing operational costs of a medium-scale or large-scale data center which pays millions of dollar in annual electricity bill. Therefore, it is imperative for data centers to improve operational efficiency and reduce electricity costs.

Data center typically receives electric power through a three-phase transformer connected to an unbalanced three-phase distribution feeder. In power distribution systems, unbalanced feeders with unbalanced electric loadings are very common. High degree of unbalance in distribution feeders can significantly affect power quality, damage electrical equipment and appliances [16], and result in highly unbalanced three-phase voltages. In addition, unbalanced systems are more likely to experience overloading on a phase wire or a neutral wire. The overloading will not only cause overheating but also lead to tripping of a protective device if there

This work was supported by California Energy Commission (CEC) under award EPC-16-030, National Science Foundation (NSF) under award 1637258, and Department of Energy (DOE) under awards DE-OE0000840 and DE-EE0007328.
is large neutral current [3]. If not properly managed, the integration of a new data center into an existing distribution network can exacerbate the degree of unbalance in the distribution circuit and may require expensive distribution system upgrades.

This paper proposes a new paradigm which coordinates the operation of data center and distributed energy resources (DERs) to reduce phase unbalance and improve the reliability and operational efficiency of electric distribution networks. The coordination scheme is implemented within the framework of a distribution system operator (DSO) managed electricity market. The data center and DERs proactively participate in the resource dispatch and market price formation processes [19]. The electricity sold and purchased on three different phases are settled using the three-phase locational marginal prices (LMPs) [17]. The LMPs on the three phases serve as the coordination signals with high prices discouraging electricity consumption and low prices encouraging consumption on a phase wire.

The existing research in the field of green computing tries to improve data center energy efficiency at five different levels. At the processor level, dynamic voltage and frequency scaling (DVFS) techniques have been shown to be highly effective in improving the energy-efficiency [5, 13, 8]. At the server level, various scheduling policies have been designed to create opportunities for deep sleep [10, 11]. At the data center level, virtual machine migration and autoscaling techniques have been proposed to optimize energy consumption [15, 9, 20]. The trade-off between minimizing energy cost and maximizing cloud computing services for a data center was analyzed in [4]. At the transmission grid level, receding horizon control approach [7], game theoretic approach [18, 14], and distributed control approach [21] have been developed to coordinate the operations of data centers and distributed energy resources such as renewable generation and electric vehicles.

Our work differs from the existing research by exploring the ways to coordinate the operations of data centers and distributed energy resources at the electric power distribution system level. The existing work ignored the three-phase electrical wiring within a data center and modeled only balanced three-phase power systems. In this paper, we filled the knowledge gap by carefully modeling the realistic three-phase unbalanced electric power distribution network and the data center. We propose solving the distribution network phase balancing problem by shifting computational loads among the servers connected to three different phase wires in a data center. The unique contributions of this paper are listed as follows.

- To the best of our knowledge, this is the first paper that solves the distribution network phase balancing problem by shifting computational loads among the servers connected to three different phase wires in a data center.
- This paper proposes an iterative scheme to coordinate the operations of data center and distributed energy resources within a DSO managed electricity market (Section 2).
- We also derived the three-phase Locational Marginal Prices (LMPs) sensitivities in a distribution market and embedded the price sensitivities into the data center’s electricity cost minimization problem (Section 3).
- The operational coordination strategy for data center and DERs is very effective in reducing phase unbalance, improving distribution network operational efficiency and reliability. The simulation results show that the degree of unbalance of a distribution feeder can be reduced by up to 100% and the electricity cost of the data center decreases by more than 4.0% (Section 4).

The remainder of this paper is organized as follows. Section 2 provides an overview of the coordination scheme for the data center and DERs within the framework of a DSO managed electricity market. Section 3 formulates the DSO market clearing problem, derives the three-phase LMPs sensitivities, and presents the electricity cost minimization problem formulation for a data center. The numerical study results are shown in Section 4. The conclusions are stated in Section 5.

2. OVERALL FRAMEWORK

The overall framework of coordinating the operation of data center and DERs to reduce phase unbalance and improve operational efficiency of electric distribution networks is depicted in Fig. 1. The coordination framework involves interactions among three decision making entities in the DSO managed electricity market. They are the DSO, the DERs, and the data center.

![Figure 1: Overall coordination framework](image)

2.1 Distribution System Operator

The DSO manages the distribution electricity market and adopts a transactive and iterative approach to coordinate the operations of DERs and data center. In each iteration of the market clearing process, the DSO tries to maximize the social welfare in the distribution circuit with the three-phase DC optimal power flow (DCOPF) algorithm. The inputs to the three-phase DCOPF algorithm include the price-sensitive energy bid curves from the DERs, the electricity consumption target from the data center, and forecast for fixed loads in the distribution feeder. The outputs of the three-phase DCOPF algorithm include the three-phase LMPs, the dispatch levels for the DERs, and the LMPs sensitivities. After the distribution electricity market is cleared, the DSO will send the LMPs and the dispatch operating points to the DERs, the LMPs and the prices sensitivities to the data center. The three-phase DCOPF algorithm and the derivation for three-phase LMPs sensitivities are described in Sections 3.1 and 3.2.

2.2 Distributed Energy Resources

The DERs proactively participate in the DSO managed distribution electricity market by submitting their single-phase price-sensitive energy bid curves on an hourly basis. The single-phase bid curves can be constructed based on the resource control model and the customers’ preferences as described in [19]. If the DER is a load resource, then a price-sensitive demand bid curve will be submitted to the DSO. The demand bid curve is a graphical representation of the relationship between quantity of electricity demand and...
customer’s willingness-to-pay. The demand bid curve must be monotonically decreasing in the price-quantity space.

2.3 Data Center

It is not straightforward to construct energy bid curves for data centers due to the electrical and computational coupling among the servers on the three individual phases. To illustrate this coupling effect, a simplified electrical wiring diagram of a typical data center is shown in Fig. 2. As shown in the figure, the computational load and electrical consumption can be partially shifted among servers connected to different phase wires by migrating computational services. However, the total electrical consumption and service level agreement constraints which link all three phases depend on the level of load shifting. Therefore, it is difficult for the data center operator to decompose its six dimensional bid curves with prices and loads of the three phases directly into three independent two-dimensional energy bid curves.

Figure 2: Electrical wiring diagram of a data center.

In order to enable the proactive participation of data center in distribution electricity market, an iterative approach is proposed to facilitate the negotiation between the DSO and the data center. In each iteration of the negotiation, the data center operator first determines the optimal load shifting plan with the latest three-phase LMPs and price sensitivities information. The data center operator then submits its electricity consumption targets for the three individual phases to the DSO. The DSO will clear the distribution electricity market and sends the updated three-phase LMPs and price sensitivities to the data center. The data center electrical cost minimization algorithm is described in Section 3.3.

3. PROBLEM FORMULATION

3.1 Distribution Network Optimization with Three-phase DCOPF

The DSO adopts a transactional approach to coordinate the operations of DERs and data center in a distribution electricity market. To clear the distribution electricity market, the DSO runs the three-phase DCOPF algorithm [17]. The objective of the three-phase DCOPF problem is to maximize the social welfare, which is the summation of the surplus of electricity customers and energy suppliers in a distribution system as shown in equation (1). The energy supplier from the transmission system is assumed to be submitting a supply offer from the reference bus to the DSO. The flexible loads are located at all other buses in the distribution network. The bid curves of DERs or flexible loads are assumed to be linear for simplicity. Hence, the customer’s willingness-to-pay function at node i with phase m for electricity with the amount of $P_i^m$ is in a quadratic form. The operating constraints in the distribution system include the real power balance constraints (2), the distribution line thermal limit constraints (3), and the phase imbalance constraints (4).

$$\max_{P_j} \sum_{i=2}^{N} \sum_{m=1}^{3} \left( a_{im} P_j^m + b_{im} P_j^m - C g \right) \sum_{m=1}^{3} P_G^m$$

subject to

$$P_G^p - \sum_{i=2}^{N} \sum_{m=1}^{3} (DF_{im})^p \cdot P_i^m$$

$$+ P^p_{Loss}(FP) - P^p_{Loss}(FQ) = 0, p = 1, 2, 3$$

$$| \sum_{q=2}^{N} \sum_{g=1}^{3} GSF_{p}^{r,s} \cdot (-PD_q^g - E_q^g) | \leq \text{PLimit}_g, \forall i, k \text{ and } i \neq k$$

$$| \sum_{n=2}^{N} P_n^i - \sum_{n=2}^{N} P_n^j | \leq \gamma, i, j = 1, 2, 3 \text{ and } i \neq j$$

The three-phase generation shift factors $GSF_{p}^{r,s}$ for the fictitious nodal demands (ENDs) $E_q^g$, and delivery factors (DFs) $(DF_{im})^p$ are derived from three-phase power flow equations and three-phase admittance matrix. The three-phase GSFs, FNDs, and DFs are very different from that of the single-phase systems. These differences arise from the mutual coupling among three phases of distribution system lines. The derivation details can be found in [17]. The iterative algorithm used to solve the FN-based three-phase DCOPF problem is described in Algorithm 1. The outputs of the iterative three-phase DCOPF algorithm include the dispatch for flexible loads, generation, the LMPs at each bus with all three phases.

The three-phase LMPs can be decomposed using the Lagrangian function of three-phase DCOPF problem. Define $\lambda^g$ as the Lagrange multiplier of the constraints (2), $\mu_g^{p+}$ and $\mu_g^{p-}$ as the Lagrange multipliers of the constraints (3), and $\mu_g^{pm+}$ and $\mu_g^{pm-}$ as the Lagrange multipliers of the constraints (4). As shown in [17], the LMP of node i with phase g can be decomposed as

$$LMP_i^g = \sum_{p=1}^{3} \lambda^g (DF_{im})^p + \sum_{b=1}^{B} \sum_{m=1}^{3} \mu^{p+}_{b} GSF_{b}^{r,s} + \mu^{p-}_{b} GSF_{b}$$

Where $GSF_{b}^{r,s}$ is generation shift factor for real power flow of the branch b with phase p when power injection is at node i with phase g. $B$ is the set of total branches. The Lagrange multipliers $\mu^{p+}_b$ and $\mu^{p-}_b$ are defined as

$$\mu^{p+}_b = \mu^{p+}_b - \mu^{p-}_b$$

$$\mu^{p-}_b = \begin{cases} \mu_{12}^+ + \mu_{13}^+ - \mu_{12}^- - \mu_{13}^- & \text{if } g = 1; \\ -\mu_{12}^+ + \mu_{23}^+ + \mu_{12}^- - \mu_{23}^- & \text{if } g = 2; \\ -\mu_{13}^+ + \mu_{23}^- + \mu_{13}^- - \mu_{23}^- & \text{if } g = 3. \end{cases}$$

$\mu^{p+}_b$ is the equivalent Lagrange multiplier of the thermal limit constraints (3) for branch b with phase p. $\mu^{p-}_b$ is the equivalent Lagrange multiplier of the phase imbalance constraints (4) related to loading limit on phase g.
Algorithm 1: Iterative algorithm for three-phase DCOPF

Initialize DFs, FNDs, and power losses.
Solve linear optimization problem using (1)-(4).

while 1 do
Update the values of FNDs, power losses, and DFs;
Solve linear optimization problem using (1)-(4);
if the difference of the dispatch of loads and generation
between the current iteration and previous iteration’s
result is larger than the pre-defined tolerance then
break;
end if
end while

3.2 Three-phase LMPs Sensitivities

The sensitivities of single-phase LMPs in transmission
electricity market can be derived using a perturbation
approach [1]. In this section, we extend the derivation for
sensitivities to three-phase LMPs in the distribution electricity
market. In particular the three-phase LMPs sensitivities
with respect to changes in bus demands are derived here.

Denote \( h(x, a) \) and \( g(x, a) \) as the set of equality and
inequality constraints of three-phase OPF problem respectively.
\( x \) represents the load and generation dispatch variables.
\( a \) stands for the vector of electricity demands at all
nodes.

Define \( z \) as

\[
z = f(x, a) = C g \sum_{m=1}^{N} PG^m - \sum_{n=2}^{N} \sum_{m=1}^{3} \left(a_i^m (P_i^d)^2 + b_i^m P_i^d\right)
\]

The three-phase DCOPF can be written in compact form as

\[
\min_z z = f(x, a)
\]

subject to

\[
h(x, a) = 0 \quad (8)
\]
\[
g(x, a) \leq 0 \quad (9)
\]

By applying the perturbation technique on top of the Karush-Kuhn-Tucker
first-order optimality conditions, we can obtain the sensitivities with respect to the electricity demands
[2],

\[
\left[dx^T, d\lambda^T, d\mu^T, dz\right]^T / da = U^{-1} S
\]

where \( \lambda \) and \( \mu \) are the Lagrange multipliers vector for the
equality and inequality constraints respectively. Matrix \( U \)
and vector \( S \) can be derived as

\[
U = \begin{bmatrix}
F_x & 0 & 0 & -1 \\
F_{xx} & H_x^T & G_x^T & 0 \\
H_x & 0 & 0 & 0 \\
G_x & 0 & 0 & 0
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
F_x^T, F_{xx}^T, H_x^T, G_x^T
\end{bmatrix}^T
\]

with respect to \( x \) and then \( a \). \( H_x, H_a, G_x, \) and \( G_a \) are the
first order derivatives of the equality and binding inequality
constraints with respect to \( x \) and \( a \). The detailed derivation
can be found in [2]. Taking derivatives on both sides of
equation (5) with respect to fixed demand of node \( u \) phase \( v \),
we get the LMPs sensitivities of node \( i \) phase \( g \) with respect to \( P^u_g \).

\[
\frac{\partial LMP^p_g}{\partial P^u_g} = \sum_{p=1}^{3} \frac{\partial \lambda^p}{\partial P^u_g} (DF^p_g)^2 + \sum_{b=1}^{B} \sum_{p=1}^{3} \frac{\partial \mu^{p,b}}{\partial P^u_g} GSFPP^{p,b} + \frac{\partial \mu^{p,b}}{\partial P^u_g}
\]

Where the derivatives of Lagrange multipliers of non-binding
inequality constraints with respect to \( P^u_g \) are zeros. Now,
the derivatives of Lagrange multipliers in equation (10) can be
substituted into (13) to calculate the three-phase LMPs sensitivities.

3.3 Data Center Electricity Cost Minimization

Since majority of the electrical appliances used in the data
center cooling systems consume three-phase electrical power,
they can not be leveraged to address phase balancing
problem. Hence, they are not modeled in this paper. In this section,
the data center electricity cost minimization problem only
considers the electricity cost from the servers. Assume
there are \( N^S \) servers on phase \( p \). Define \( M^p \) as a \( N^S \times 1 \) binary variable vector. If the \( i \)-th element of \( M^p \) is 1, it
means that the \( i \)-th VM is moving from a server on phase \( r \)
to phase \( p \). \( N^S \) denotes the number of VMs running on the
servers of phase \( r \) initially.

The objective function of the data center is to minimize
its electricity cost as shown in equation (14). The electricity
costs of all servers equal to the dot product of updated
LMPs vector after the VMs live migration and the vector of
per phase electricity consumption of servers \( P^{dc} \). In the
objective function, \( \text{LMP} = [\text{LMP}^1, \text{LMP}^2, \text{LMP}^3]^T \) denotes
the LMPs for the three different phases at the data
center bus and \( P^{dc} = [P^{dc}_1, P^{dc}_2, P^{dc}_3]^T \) denotes
the electricity consumption from the servers on the three-phases.
\( P^{dc} \) stands for the initial value of \( P^{dc} \). The updated LMPs vector
after the VMs live migration is estimated by using the
LMPs sensitivities \( \partial \text{LMP} / \partial P^{dc} \), which is a \( 3 \times 3 \) matrix.
The LMPs sensitivities are introduced into the data center
electricity cost minimization process to serve as a damping
factor which prevents oscillating of computing load shifts in
the data center and the DSO’s negotiation process. Without
the LMP sensitivities, the data center will aggressively move
its load from the phase with higher price to the phase with
lower price without considering the impacts of the move on
LMPs. This could prevent the iterative negotiation process
between the DSO and the data center from reaching an equi-
librium point.

The electricity consumption of a particular phase equals to
the sum of electricity consumptions from each individual
server connected to the phase wire in equation (15). The
electricity consumption of each server includes a dynamic
component and an idle component as shown in equation
(16) [4]. The dynamic component of the server electricity
consumption is closely related to the server utilization
rate which is modeled in equation (17). The utilization
rate of server \( i \) on phase \( p, U^i_p \), can be estimated based
on the requests arrival rate for servers on phase $p$ and the live migration of VMs [15]. During the migration period, computational loads increase on the servers which the VMs migrated to and from. After the migration period, computational loads increase/decrease on the servers which the VMs migrated to and from. For simplification purpose, a uniform utilization of servers is assumed for each phase.

$$\min_{M^{pr}, p=1,2,3, r \neq r'} [LMP + \frac{\partial LMP}{\partial P_{dc}} (P_{dc} - P_{dc}^0)]^T P_{dc}$$

subject to

$$P_{dc}^p = \sum_{i=1}^{N^{ip}} P^p_i, \quad p = 1, 2, 3$$

$$P^p_i = P^p_{dyn}\times U^p_i + P^p_{idle}, \quad \forall i, p = 1, 2, 3$$

$$U^p_i = \{(A^p)^T \cdot 1 - \sum_{r=1, r \neq p}^{3} [(1 - \frac{T^{pr}}{T_{int}})(A^p)^T \cdot M^{pr}]$$

$$\sum_{r=1, r \neq p}^{3} [(1 - \frac{T^{pr}}{T_{int}})(A^p)^T \cdot M^{pr}] + \sum_{r=1, r \neq p}^{3} R^p M^{pr}$$

$$+ \sum_{r=1, r \neq p}^{3} R^p M^{pr})/ \sum_{i=1}^{N^p} R^p_{cap_i}, \quad \forall i, p = 1, 2, 3$$

In equation (17), $T^{pr}$ denotes the VM migration time from phase $p$ to $r$ and $T_{int}$ is the market clearing time step. $A^p$ denotes the vector of requests arrival rate of VMs on phase $p$ before the migration. It has a dimension of $N^p \times 1$. $R^p_{cap_i}$ denotes the request processing capability of server $i$ on phase $p$. 1 is a vector of ones with the same dimension as $A^p$.

The data center electricity cost minimization formulation also includes two sets of constraints related to the service level agreement (18) and (19).

During the live migration of VMs, the response time constraints in the service level agreement is modeled as

$$T^p = a_0^p + a_1^p \cdot (A^p)^T \cdot 1 + \sum_{r=1, r \neq p}^{3} R^p M^{pr}$$

$$+ \sum_{r=1, r \neq p}^{3} R^p M^{pr})/ N^p_S \leq T_{SLA}, \quad p = 1, 2, 3$$

After the migration of VMs, the response time constraints in the service level agreement is modeled as

$$T^p = a_0^p + a_1^p ((A^p)^T (1 - \sum_{r=1, r \neq p}^{3} M^{pr}) + \sum_{r=1, r \neq p}^{3} (A^p)^T$$

$$M^{pr})/ N^p_S \leq T_{SLA}, \quad p = 1, 2, 3$$

### 4. NUMERICAL STUDY

#### 4.1 Simulation Setup

The standard IEEE 4-bus distribution test feeder [6] is modified to validate the effectiveness of the proposed phase balancing algorithm with data center. The data center is located at node 2. The fixed loads and flexible loads are located at node 4. The transmission system is assumed to supply electric power to the distribution network through the distribution substation at a price of $0.6/kWh. The total amount of fixed demands and price-sensitive demands are summarized in Table 1. Two simulation cases with different degree of unbalance are created. The distribution feeder is slightly unbalanced in case 1 and heavily unbalanced in case 2. The power imbalance limit between any two phases in the distribution feeder is set to be 60kW.

The price-sensitive demand bid curves of flexible loads on the three phases are assumed to be linear functions as $Price^1 = 1 - P^1_{dc}/200$, $Price^2 = 1 - P^2_{dc}/250$, and $Price^3 = 1 - P^3_{dc}/300$. $Price^1$, $Price^2$, and $Price^3$ are the bidding prices for the three phases. The price ranges of the three demand bid curves are from $0.1/kWh$ to $0.45/kWh$.

#### 4.2 Simulation Results

The LMPs and electricity consumptions on all three phases of the data center are calculated with and without phase balancing for the two different unbalance cases. As shown in Table 1, in both cases electricity load shifts from servers on phase $c$ which has higher price to phase $a$ which has lower price. After phase balancing, the degree of unbalance of the distribution feeder is reduced which leads to smaller price difference between phase $a$ and phase $c$. The amount of load shift and reduction in price difference is higher in the heavily unbalanced case than the slightly unbalanced.

<table>
<thead>
<tr>
<th>Case</th>
<th>Phase balancing</th>
<th>Price &amp; Power</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Without</td>
<td>$($/kWh) 0.5048</td>
<td>0.6015</td>
<td>0.6988</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 120.0</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Without</td>
<td>$($/kWh) 0.3241</td>
<td>0.6015</td>
<td>0.8795</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 120.0</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>With</td>
<td>$($/kWh) 0.4543</td>
<td>0.6015</td>
<td>0.7494</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 130.5</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
</tbody>
</table>

### Table 1: Fixed and Flexible Load Profile

<table>
<thead>
<tr>
<th>Node 4</th>
<th>Fixed Load</th>
<th>Capacity (KW)</th>
<th>Flexible Load Capacity (KW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>460</td>
<td>500</td>
<td>300</td>
</tr>
<tr>
<td>Case 2</td>
<td>420</td>
<td>500</td>
<td>300</td>
</tr>
<tr>
<td>Case C</td>
<td>530</td>
<td>580</td>
<td>270</td>
</tr>
</tbody>
</table>

### Table 2: LMPs and Data Center Electricity Consumption

<table>
<thead>
<tr>
<th>Case</th>
<th>Phase balancing</th>
<th>Price &amp; Power</th>
<th>Phase A</th>
<th>Phase B</th>
<th>Phase C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Without</td>
<td>$($/kWh) 0.5048</td>
<td>0.6015</td>
<td>0.6988</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 120.0</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Without</td>
<td>$($/kWh) 0.3241</td>
<td>0.6015</td>
<td>0.8795</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 120.0</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>With</td>
<td>$($/kWh) 0.4543</td>
<td>0.6015</td>
<td>0.7494</td>
<td></td>
</tr>
<tr>
<td></td>
<td>($/kWh) 130.5</td>
<td>$(/kWh) 6015$</td>
<td>120.0</td>
<td>120.0</td>
<td></td>
</tr>
</tbody>
</table>
The electricity cost of data center and total surpluses are reported in Table 3. As shown in the table, the phase balancing algorithm not only reduces the electricity bill of the data center but also increases the total surpluses of the flexible loads and the supplier. The savings in the heavily unbalanced case is much more significant than the slightly unbalanced case. In case 2, the phase balancing algorithm reduces the electricity bill of the data center by more than 4.0% and increases the total surpluses by 35%. Note that the savings also depend on the price elasticity of demand.

<table>
<thead>
<tr>
<th>Case</th>
<th>Without Phase Balancing</th>
<th>With Phase Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Center ($)</td>
<td>216.6</td>
<td>215.8</td>
</tr>
<tr>
<td>Total Surpluses ($)</td>
<td>57.3</td>
<td>58.9</td>
</tr>
</tbody>
</table>

Instead of keeping the phase imbalance limit $\gamma$ in equation (4) constant at 60 KW, we try to reduce the phase imbalance as much as possible without making the savings of the data center and the flexible loads worse. The simulation results show that the percentage reduction in phase imbalance varies with the electric power rating of the data center. In the simulation setup of case 2, the power rating of the data center is about 25% of the total feeder demand. In this case, our proposed algorithm can reduce the phase imbalance by 100%. In other words, by shifting computational loads in the data center, the electric loads on the distribution feeder can be completely balanced. By gradually reducing the size of the data center by 40%, 45%, and 50%, the reduction in phase imbalance also decreases to about 93%, 83%, and 73%.

5. CONCLUSIONS

This paper develops an iterative scheme to coordinate the operations of data center and DERs to tackle the electricity cost minimization algorithm. We also derived the three-phase LMPs sensitivities in a distribution network phase balancing problem. We also derived the three-phase LMPs sensitivities in a distribution network phase balancing problem. We also derived the three-phase LMPs sensitivities in a distribution network phase balancing problem.

6. REFERENCES


A Case Study of Energy Efficiency on a Heterogeneous Multi-Processor

Hitoshi Oi
The University of Aizu, Aizu-Wakamatsu, JAPAN
oi@oslab.biz

ABSTRACT
In this extended abstract, we present a case study of power-efficiency on a heterogeneous multi-core processor, Exynos 5422 based on the ARM big.LITTLE architecture. We show the effect of thermal management on the big (faster) cores and the comparisons between big and LITTLE (slower) cores using the EEMBC CoreMark-Pro benchmarks. As expected, the LITTLE cores are more energy efficient than the big cores at the maximum performances of both cores for all workloads. However, the big cores are similarly or more power efficient as LITTLE cores for 5 out of 9 workloads when the performance of both cores are matched by lowering the clock frequency of big cores. Delay-insertion for matching the performance is only effective for one workload, but it may be useful in a multi-programmed environment when the frequency of each core cannot be set independently (which is the case for Exynos).

Keywords
Workload Analysis, Performance Evaluation, Energy Efficiency, Heterogeneous Multi-Core Processor

1. INTRODUCTION
Heterogeneous multi-core processing (HMP) is very popular these days, but in commercial products, they are mostly combinations of general purpose CPUs and compute-intensive cores (e.g. GPGPUs). Another type of heterogeneous multi-core architecture is that the cores are implemented in different microarchitectures but still have an identical instruction set. The big.LITTLE architecture from ARM is one of commercial products in this type. Samsung, among others, has implemented it in their Exynos series [1]. Task scheduling among different types of cores has been one of the most active topics in this type of HMPs [2]. However, it can also be a suitable platform to study the effect of microarchitecture on the performance and energy efficiency because all components other than cores are identical on a system with such an HMP processor. In this extended abstract, we present an energy efficiency study on an Odroid-XU3, a single-board computer with an Exynos 5422 processor [3]. In the next section, the measurement platform and methodologies are described. The measurement results and analysis are reported in Sections 3 to 6. Section 7 summarizes the results obtained so far and also describes the ongoing work.

Copyright is held by author/owner(s).

2. PLATFORM AND METHODOLOGIES
Table 1 shows the specifications of the measurement platform Odroid-XU3 [3]. It is a single board computer from Hardkernel with an Exynos 5422 processor (Figure 1). Both A15 (big) and A7 (LITTLE) cores have core-private L1 instruction and data caches of 32KB each. The sizes of the shared L2 caches are 2MB (A15) and 512KB (A7). Both are DVFS capable in 0.1GHz steps with frequency ranges of 0.2 to 2GHz (big) and 0.2 to 1.4GHz (LITTLE). A15 is 3-way out-of-order superscalar and A7 is 2-way in-order.

Table 1: Odroid-XU3 Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Samsung Exynos 5422 (ARM big.LITTLE: 4\times A7 + 4\times A15)</td>
</tr>
<tr>
<td>Memory</td>
<td>2GB (LPDDR3)</td>
</tr>
<tr>
<td>OS</td>
<td>Arch Linux ARM (Kernel 3.10.104-8)</td>
</tr>
<tr>
<td>Sensors</td>
<td>Power (big and LITTLE cores, GPU &amp; Memory), Temperature</td>
</tr>
</tbody>
</table>

Odroid-XU3 has four built-in power sensors, measuring big and LITTLE cores, memory and GPU. It also has a temperature sensor measuring the die temperature. To measure the total power consumption, a power meter, Odroid Smart Power [4] is attached to the DC-input of the Odroid-XU3. The sensors and power meters are sampled every second. The idle power consumption is 0.1969 Watt by the sum of sensors and 3.2091 Watt by the power meter.

Figure 1: Exynos 5422 Diagram

We first obtain the normalized per-task energy $E_n = \sum_{ET} (P_d - P_i) / NT$, where $P_d$ is the power consumption (either by sensors or power meter), $P_i$ is the idle power, $NT$ is the number of tasks executed during execution and $ET$ is the execution time. As the energy efficiency metrics, we use the relative energy consumption which is the ratio of $E_n$ of big cores with 2GHz against with the clock frequency in MPC in Table 3 (Section 3) and LITTLE cores against big cores (Sections 4 to 6).
While further investigation is required, the total includes the power consumption of the cooling fan, which consumes 0.84 Watt of power, and this could be the reason of total’s being higher than the CPU+Memory. CJPEG benchmark had the highest energy overhead. If it were run at 2GHz, it would consume 22 and 38 % more energy than 1.7GHz in CPU+Memory and total, respectively.

Figure 2: Effect of Thermal Management on Performance (left y-axis) and Normalized Energy (right y-axis).

4. BIG.LITTLE COMPARISON

Figure 3 compares the performance and energy consumption of the LITTLE cores against the big cores. The clock frequency of LITTLE cores is fixed at 1.4GHz (the highest frequency). The big cores were executed at the clock frequency listed in the MPC columns in Table 3 because the highest frequency (2GHz) resulted in sub-optimal performance and energy efficiency.

The relative performance of the LITTLE cores (left bars) varies widely among benchmarks, from 26% of Radix2 to 51% of Core. The relative energy consumption seems to be in a reciprocal relation to the performance: the higher the performance, the lower the energy consumption (vice versa). For example, Radix2 and Core consumed the most and the least energy, which is the opposite to the relation of these two programs in terms of performance. It should also be noticed that the total energy was smaller than the CPU+Memory energy. This may be because the cooling fan might be active only in the big core executions and it became an energy overhead.

5. LOWERING CLOCK FREQUENCY

We lowered the clock frequency of the big cores and sought the frequency at which the big cores achieved the nearest throughput as the LITTLE cores (Figure 4). The right bars represent the clock frequency (GHz) at which the big cores performed the nearest throughput as the LITTLE cores.

In terms of energy efficiency, the LITTLE cores may not always be advantageous against the big cores as in Figure 3. In Radix2 and Parser (Linear, Loop, Nnet and Zip), the LITTLE cores consumed more (similar) energy per task than (as) the big cores. In other words, it might be better to run these workloads on the big cores with a lower clock frequency than on the LITTLE cores if the energy efficiency is the primary concern. Another characteristic we noticed

---

### Table 2: CoreMark-Pro Benchmark Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJPEG</td>
<td>JPEG Image Compression</td>
</tr>
<tr>
<td>Core</td>
<td>Derivation of CoreMark</td>
</tr>
<tr>
<td>Linear</td>
<td>Linear algebra solver (from Linpack)</td>
</tr>
<tr>
<td>Loops</td>
<td>Modified Livermore loop</td>
</tr>
<tr>
<td>Neural Network</td>
<td>Pattern evaluation using neural net</td>
</tr>
<tr>
<td>XML Parser</td>
<td>XML parsing and ezxml creation</td>
</tr>
<tr>
<td>SHA</td>
<td>256-bit Secure Hash Algorithm</td>
</tr>
<tr>
<td>Radix2</td>
<td>FFT Radix 2</td>
</tr>
<tr>
<td>ZIP</td>
<td>zlib compression benchmark</td>
</tr>
</tbody>
</table>

For the workloads of this study, we use the CoreMark-Pro benchmark programs from EEMBC [5]. It is a collection of nine benchmark programs (five integer and four floating-point) as shown in Table 2. Each benchmark is run for minimum 1000 iterations or 10 seconds (whichever more/longer), and the averages of ten runs are reported.

### Table 3: Thermal Management Effect. TM: % of time thermal management was activated. MPC: Clock frequency for the maximum performance.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>TM (%)</th>
<th>CJPEG</th>
<th>Core</th>
<th>Linear</th>
<th>Loops</th>
<th>Nnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM (%)</td>
<td>39.3</td>
<td>45.7</td>
<td>22.5</td>
<td>48.6</td>
<td>33.6</td>
<td></td>
</tr>
<tr>
<td>MPC (GHz)</td>
<td>1.7</td>
<td>1.8</td>
<td>1.7</td>
<td>1.7</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Parser</th>
<th>Radix2</th>
<th>SHA</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM (%)</td>
<td>22.2</td>
<td>14.1</td>
<td>44.1</td>
<td>24.1</td>
</tr>
<tr>
<td>MPC (GHz)</td>
<td>1.9</td>
<td>1.9</td>
<td>1.8</td>
<td>1.9</td>
</tr>
</tbody>
</table>

Table 3 shows the the percentage of the execution time during which the thermal management was activated (TM) and the clock frequency at which the maximum performance was achieved (MPC). Figure 2 shows the relative performance (left Y-axis) and the energy consumption (right Y-axis) of the the 2GHz execution against the execution with the MPC clock frequency in Table 3. Linear (linear algebra solver) was suffered from the thermal management most and running it with the 2GHz clock actually slowed it down to 64% of the 1.7GHz execution. Please note that the higher percentage of the thermal management activation did not necessarily result in the lower performance. We can see this example in Linear vs Loops: the latter has a higher thermal management percentage (22.5 vs 48.6) but the former has a lower relative performance (64.4 vs 82.4).

The center and right bars in Figure 2 show the relative energy consumption of the 2GHz cases versus the maximum performance cases. In general, the total (DC) energy shows higher values than the CPU+Memory (P+M).
In this work, we presented a case study of energy-efficiency on a platform with a heterogeneous multi-core processor. We compared the big and LITTLE cores, at their maximum performance and also at the matched performance by means of lowering clock frequency and delay-insertion. The lowering clock frequency made the big cores similarly or more power-efficient than LITTLE cores in 5 out of 9 workloads.

Until recently, it was considered that only LITTLE cores could be profiled with the performance counters [7]. Actually, it was not true and a tool to profile both types of cores became available [8]. We are currently investigating the results presented in this work and further insight of the behavior of the HMP using such tools.

8. REFERENCES


GPGPU Power Estimation with Core and Memory Frequency Scaling

Qiang Wang
Department of Computer Science
Hong Kong Baptist University
qiangwang@comp.hkbu.edu.hk

Xiaowen Chu
Department of Computer Science
Hong Kong Baptist University
chxw@comp.hkbu.edu.hk

ABSTRACT
With the increasing installation of Graphics Processing Units (GPUs) in supercomputers and data centers, their huge electricity cost brings new environmental and economic concerns. Although Dynamic Voltage and Frequency Scaling (DVFS) techniques have been successfully applied on traditional CPUs to reserve energy, the impact of GPU DVFS on application performance and power consumption is not yet fully understood, mainly due to the complicated GPU memory system. This paper proposes a fast prediction model based on Support Vector Regression (SVR), which can estimate the average runtime power of a given GPU kernel using a set of profiling parameters under different GPU core and memory frequencies. Our experimental data set includes 931 samples obtained from 19 GPU kernels running on a real GPU platform with the core and memory frequencies ranging between 400MHz and 1000MHz. We evaluate the accuracy of the SVR-based prediction model by ten-fold cross validation. We achieve greater accuracy than prior models, being Mean Square Error (MSE) of 0.797 Watt and Mean Absolute Percentage Error (MAPE) of 3.08% on average. Combined with an existing performance prediction model, we can find the optimal GPU frequency settings that can save an average of 13.2% energy across those GPU kernels with no more than 10% performance penalty compared to applying the default setting.

1 INTRODUCTION
Over the past few decades the Graphics Processing Units (GPUs) have been increasingly adopted to massive parallel computing across a wide range of industrial and academic areas, including big data analysis [7], bio-informatics [27], image recognition based on deep neural network [10], etc. Apart from over 10 TFLOPS of the peak single-precision performance, modern GPUs also bring brilliant advantages in performance-per-watt. In the GREEN500 supercomputer list [8] of June 2017, 9 of the Top 10 are equipped with GPUs. Especially the top 1 machine can provide up to 14.11 GFLOPS per watt with NVIDIA Tesla P100. However, to handle exponentially increasing data generated every day, GPU-based technology could suffer from high electricity cost. These facts reveal great emergency and potential of energy conservation of GPUs with hunger of efficient power management techniques.

Dynamic Voltage and Frequency Scaling (DVFS) [13] is a traditional and useful technique to save energy of modern processors. It allows the processors to achieve better energy efficiency with proper voltage and frequency settings. Compared with relatively mature CPU DVFS technology, GPU DVFS is still at an early stage. Unfortunately, according to existing studies [1, 17, 25], CPU DVFS technology can not be directly adopted to GPUs. That might be caused by the fact that modern GPUs have two main frequency domains (core and memory frequencies), which makes it more complicated. Besides, scaling up the frequency is proved be energy efficient for CPUs but not always for GPUs [12]. To achieve the maximum energy conservation of one given GPU application with DVFS, it is necessary to predict its energy consumption, which requires precise power estimation under different frequency settings. Besides, such power estimation could also help solve energy efficient tasking scheduling problem on CPU-GPU grid which has been discussed in [4, 16].

Several previous literatures [1, 17] have explored GPU power characteristics with DVFS and reveal the fact that frequency settings should be determined carefully to achieve the best energy efficiency. Usually neither the lowest nor the highest is optimal. To model those complex correlations, recent papers [1, 6, 26] attempt to apply machine learning methods. From linear regression to K-means clustering, they reveal the considerable potential to precisely catch the statistical correlations between performance metrics of the GPU applications and the final power consumptions. However, there might exist one or more drawbacks of each work. For example, some of them applied too few performance features of the GPU kernels which may result in large bias from the ground truth. Furthermore, some works adopted naive pre-processing, especially like normalization, to the performance features before feeding them to the machine learning algorithm. Even the effects of core and memory frequency scaling are rarely considered in their models.

To address those problems, this paper proposes a power estimation model based on support vector regression machine (SVR) which is applicable to real GPU hardware. With special treatments to the features extracted by the profiling tool during kernel runtime, we can utilize them as input of the SVR model to let it learn the effects on the power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by the fact that modern GPUs have two main frequency domains (core and memory frequencies), which makes it more complicated. Besides, scaling up the frequency is proved be energy efficient for CPUs but not always for GPUs [12]. To achieve the maximum energy conservation of one given GPU application with DVFS, it is necessary to predict its energy consumption, which requires precise power estimation under different frequency settings. Besides, such power estimation could also help solve energy efficient tasking scheduling problem on CPU-GPU grid which has been discussed in [4, 16].

Several previous literatures [1, 17] have explored GPU power characteristics with DVFS and reveal the fact that frequency settings should be determined carefully to achieve the best energy efficiency. Usually neither the lowest nor the highest is optimal. To model those complex correlations, recent papers [1, 6, 26] attempt to apply machine learning methods. From linear regression to K-means clustering, they reveal the considerable potential to precisely catch the statistical correlations between performance metrics of the GPU applications and the final power consumptions. However, there might exist one or more drawbacks of each work. For example, some of them applied too few performance features of the GPU kernels which may result in large bias from the ground truth. Furthermore, some works adopted naive pre-processing, especially like normalization, to the performance features before feeding them to the machine learning algorithm. Even the effects of core and memory frequency scaling are rarely considered in their models.

To address those problems, this paper proposes a power estimation model based on support vector regression machine (SVR) which is applicable to real GPU hardware. With special treatments to the features extracted by the profiling tool during kernel runtime, we can utilize them as input of the SVR model to let it learn the effects on the power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumption caused by both core and memory frequency scaling.

Section 4 proposes our GPGPU power estimation model with SVR. Model evaluation and experimental results are revealed in Section 5. Finally we conclude our work in Section 6.
2 RELATED WORK

In this section we review some previous work about GPU DVFS characteristics and power modeling. As mentioned before, GPU DVFS can be very different from traditional CPU DVFS. Mei et al. [17] conducted real hardware experiments on Fermi GTX560Ti and Maxwell GTX980 and observed that the best energy efficient setting of core voltage and two frequency domains varies from kernel to kernel. Even the default setting is usually not the best choice, which may bring up to 34% potential energy wastes. Jared et al. [9] explored energy, power and performance characteristics of 34 selected GPGPU benchmark programs by varying bth core and memory frequency, input data size and ECC switch and revealed some relationships between average runtime power and those three factors, which suggests the potential of energy conservation with proper settings.

As for power modeling work, Hong and Kim [11] estimated the access rates of different components on the GPU based on the dynamic number of instructions and the cycles of each instruction. They then designed a suite of micro-benchmarks to search for separated power of those components. After that they could estimate the runtime GPU power consumption of a new kernel. However, today’s GPU architecture can be more complicated with new hardware features and instructions, which gives much more challenges of power modeling. Leng et al. packed Hong and Kim’s power modeling to GPGPU-Sim, to form GPUWatch, which could estimate the runtime GPU power with different voltage/frequency settings at cycle-level [14]. The authors refined Hong and Kim’s model with supplemental micro-benchmarks, to overcome the power uncertainties brought by the new Fermi hardware. The prediction error was 15% for the micro-benchmarks, and 9.9% for the general GPU kernels, on the Fermi GTX480 GPU. Although those papers achieve good accuracy, they can not avoid trivial micro-benchmark procedures for new hardware and long time simulations.

Recent years witness the popularity of statistical methods applying to GPU power modeling, which treats the GPU hardware as a black box and predicts the power with some monitored runtime signals. Abe et al. built regression models to estimate the average power consumption of the NVIDIA Tesla, Fermi and Kepler GPUs under different levels of core and memory frequency [1]. Particularly, they defined three core/memory frequency settings as part of the model inputs. They also chose 10 most relative performance counters who provided the best-fitting results. The average prediction errors range from 15% to 23.5% depending on the kernel characteristics and the generations of GPU architecture. Newer hardware had larger prediction error, which indicates that simple linear regression is not sufficient to catch those correlations between power and performance counters.

Song et al. [23] attempted to use artificial neural network (ANN) to model the non-linear relationship between runtime events and runtime average power of a given GPU kernel. They used two independent threads to collect performance events with CUPTI API and power metrics with NVML API respectively. Then they selected a collection of most relevant events as input of ANN. Their model achieved average prediction error of 2.1% under default frequency settings. Wu et al. extensively studied the GPU power and performance, with different settings of GPU frequency, memory bandwidth and core number [26]. They adopted K-means clustering and ANN simultaneously. In the ANN modeling process, they first used K-means to cluster the kernels according to the similarity of scaling behaviors. Then for each cluster, they trained an ANN with two hidden layers. The reported average power prediction error over all frequency/unit configurations was 10%.

3 BACKGROUND AND MOTIVATION

3.1 Dynamic Voltage and Frequency Scaling

As mentioned before, DVFS is one of the most important techniques for energy conservations of not only traditional CPUs but also GPUs. The dynamic power is usually modeled by Equation (1), where \( a \) denotes a utilization ratio, \( C \) denotes the capacity, \( V \) denotes the chip supply voltage and \( f \) denotes the frequency [13]. Since the total energy consumption of one application is the product of average runtime power and total execution time, power modeling plays an important role in energy conservation with different DVFS settings.

\[
    P_{\text{dynamic}} = aCV^2f
\]  

Notice that the power has a linear correlation with the frequency. However, some previous GPU DVFS work indicates that GPUs have more complex power scaling behaviors when adopting DVFS [17]. One reason is that modern GPUs have two main frequency domains. One is core frequency, which mainly controls the speed of stream multiprocessors (SMs), while the other is memory frequency, which affects the bandwidth of DRAM. Table 1 summarizes the dominating frequency for different types of memory. Note that only DRAM works under memory frequency and L2 cache works under core frequency though they both serve the global memory requests.

<table>
<thead>
<tr>
<th>Components</th>
<th>Dominating Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>memory frequency</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>core frequency</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>core frequency</td>
</tr>
<tr>
<td>Texture Cache</td>
<td>core frequency</td>
</tr>
<tr>
<td>Register</td>
<td>core frequency</td>
</tr>
<tr>
<td>CUDA cores</td>
<td>core frequency</td>
</tr>
<tr>
<td>Special Function Units</td>
<td>core frequency</td>
</tr>
</tbody>
</table>

3.2 Power Characteristics of Frequency Scaling

Because different applications can have diverse workloads on different units on the GPU, scaling both core and memory frequency may result in completely disparate power scaling behaviors on them. In this section, we demonstrate an example of Nvidia GTX 980 [19] to illustrate the effects of different frequency settings on its power consumption.

We first fix the core frequency to 400 MHz and 1000 MHz respectively and scale the memory frequency from 400 MHz to 1000 MHz with a step size of 100 MHz. Figure 1(a) and 1(b) show the normalized power of six GPU kernels with those settings. The power scaling behaviors could be either simple or complicated among different GPU kernels as well as different frequency settings.
profiling becomes available in a convenient way. We adopt CUDA version 8.0 in our work since it has good compatibility among a wide range of GPU generations and also provides sufficient supports for new features of recent GPU architectures. CUDA 8.0 provides nvprof [21] which can capture more than 50 performance counters for analyzing kernel performance. We choose 13 major counters listed in Table 2 which significantly affect the average power. achieved_occupancy and eligible_warps_per_cycle indicate the utilization of stream-multiprocessors on GPUs. dram_read_transactions and dram_write_transactions mean the number of DRAM transactions happened during kernel execution while l2_read_transactions and l2_write_transactions mean the number of L2 cache transactions. Since shared memory is also widely used in GPU kernels for performance optimization, we also include shared_load_transactions and shared_store_transactions. branch_efficiency and cf_executed reflect the divergence level of control flow while the rest three float_count_* evaluate the workload of single/double precision floating-point operations.

However, it might cause large errors if such metrics data are directly adopted to statistical methods. First, since most of the profiling metrics represent the total number of instructions or transactions of the corresponding events, they could have extreme different magnitudes if the original GPU kernels have different workloads. Second, even for the same kernel, the distribution among different types of instructions can be considerably uneven. To better take advantage of a data-driven model, we should conduct careful feature pre-processing according to not only the variable value itself but also the characteristic of the feature.

Figure 1: Power scaling behavior under different frequency settings. The upper two figures show the normalized power consumption of different GPU kernels when increasing memory frequency with fixed core frequency. The below two figures show the normalized power consumption of different GPU kernels when increasing core frequency with fixed memory frequency.

For example, the approximately linear correlations between power and frequency can be observed from Figure 1(b) while there exist knobs for some applications like Convolution (convS) and vector addition (VA) in Figure 1(a). Furthermore, some kernels have different sensitivity to two frequency domains. These phenomena happen to Matrix Multiplication with global memory (MMG) and with shared memory (MMS). Their power consumptions have very few changes when scaling up core frequency with fixed 400 MHz memory frequency, which differs from the linear increments in 1000 MHz memory frequency setting. Then we fix the memory frequency to 400 MHz and 1000 MHz respectively and scale the core frequency from 400 MHz to 1000 MHz. Figure 1(c) and 1(d) show that MMG and MMS have linearly increasing power consumptions with scaling up either core or memory frequency this time. Other kernels also display different scaling behaviors from the core-frequency-fixed cases.

The power scaling behaviors could differ with respect to both frequency settings and GPU kernel characteristics. Thus, we would like to use statistical methods to estimate the power consumption under different frequency settings of the target GPU kernel.

4 POWER ESTIMATION WITH CORE AND MEMORY FREQUENCY SCALING

4.1 Feature Selection

Thanks to the programmability provided by Compute Unified Device Architecture (CUDA), not only the software developments become simpler on modern NVIDIA GPUs, but also the performance

4.2 Power measurement

To measure the power consumption of the tested GPU kernels, we use the tool nvidia-smi [22], which is a command line utility based on top of NVIDIA GPU driver API designed for the management and monitoring of NVIDIA GPU devices. We choose the sampling frequency at 1 read per second so that the impact of nvidia-smi on the application performance is negligible. We also add ten seconds of sampling before and after the kernel execution as safeguards. We revise each tested application to let them run sufficient iterations so that the GPUs are running for at least 10 minutes and generate more than 600 power samples. It is also well-known that the GPU temperature can also affect the runtime power consumption. Our current work focuses on the effect of frequency scaling, hence we control the GPU temperatures between 45 °C to 55 °C through fan speed adjustment. To verify that our temperature range does not bring obvious variance to the sampling results, we conduct significance test with $t$-distribution on the power samples of each kernel and achieve 95% confidence interval. The result also suggests the repeatability of our experiments. We will leave the investigation of the impact of GPU temperature on power consumption as our future work.

4.3 Power modeling with SVR

Some researchers established statistical models to estimate the GPU runtime power consumption. Those models include square linear regression (SLR) [1], support vector regression with linear kernel (SVR) [15], etc. Most of them attempt to fit a linear relationship


<table>
<thead>
<tr>
<th>Metrics</th>
<th>Description</th>
</tr>
</thead>
</table>
| achieved occupancy               | Ratio of the average active warps per active cycle to the maximum number of |}


tightly so that the model can explain the contribution of each input variable. However, since the GPU architectures are becoming more and more sophisticated, simple linear interpreters become insufficient to model the relationship between the extracted kernel features and the power consumptions, which is also validated by our motivation examples in Section 3.2.

Support Vector Regression (SVR) is a classical supervised learning algorithm for data regression. Mathematically, the basic idea is to predict \( y \) given the input \( x \) with the function \( f \) in Eq. (2) by solving the optimization problem in Eq. (3). \( \langle \omega, x \rangle \) represents the kernel function used in the model, which helps map the original input data to a higher dimension. This is useful when exploiting the non-linear correlation between the input variables and the output. Thus, we apply SVR in modeling the average GPU runtime power consumptions. The concrete algorithms can be found in [2].

\[
\begin{align*}
    f(x) &= \langle \omega, x \rangle + b, \omega \in \mathbb{R}^d, b \in \mathbb{R} \quad (2) \\
    \min_{\omega,b} & \frac{1}{2} \| \omega \| \\
    \text{s.t.} \quad & y - (\langle \omega, x \rangle - b) < \epsilon \quad (3)
\end{align*}
\]

- **Pre-processing of the input features**: Instead of directly using the profiling metrics as input data of the SVR model, we conduct two special pre-processing on them. First, since some performance counters represent the number of different types of operations and are typically very large integers, we normalize this type of metrics, including (a) dram_read_transactions, (b) l2_read_transactions, (c) shared_load_transactions, (d) dram_write_transactions, (e) l2_write_transactions, (f) shared_store_transactions, (g) flop_count_dp, (h) flop_count_sp and (i) flop_count_sp_special, by dividing each dimension by the summation of them. Since our objective is to predict the average runtime power consumption, what we should concern is the relative workload information of different units on the GPU, which can be retained with that normalization. Second, to introduce the effects of frequency scaling, we multiply all the transaction and operation variables by the ratio of the relative frequency domain to the baseline frequency setting. For example, global memory speed is influenced by the memory frequency while l2 cache and shared memory speed is influenced by the core frequency. The baseline frequency setting is 400MHz core frequency and 400MHz memory frequency.

- **Model Selection**: We have tried different kernel functions including linear kernel, gaussian kernel and polynomial kernel. We also use grid search to find the optimal hyper-parameters of each kernel, which is not listed here due to space limitation. Finally we choose the polynomial kernel with degree three. We use lib-SVM [3] as the implementation and set Epsilon, which is half the width of epsilon insensitive bend, to be 0.1 and OutlierFraction, which is the expected fraction of outliers, to be 0.2. To evaluate the generalization capability of the model, we adopt ten-fold cross validation.

5 PERFORMANCE EVALUATION

We apply the proposed SVR-based power prediction model with our special feature pre-processing (denoted "pp_SVR") to 16 GPU kernels from CUDA SDK 8.0 [20] and 3 from Rodinia [5], which are listed in Table 3, to evaluate its accuracy. In the ten-fold cross validation, the data set is separated evenly to ten subsets. For each subset, we use mean squared error (MSE) to assess the model accuracy trained by the other nine subsets. The details of the hardware platform are listed in Table 4.

To demonstrate the effects of feature pre-processing, we implement a naive SVR model (denoted "naive_SVR") which directly takes the raw performance counters and the frequency settings as input. Besides, we also compare our model to multiple linear regression (denoted "MLR") from Nagasaka et al. [18] on our pre-processing features.

5.1 Training and Evaluation Samples

We first run nvprof on all the tested GPU kernels under the configuration of 400MHz core frequency and 400MHz memory frequency and collect all the required performance counter data. Then we measure the power consumption of each kernel under both core and memory frequency scaling from 400MHz to 1000MHz with a step size of 100MHz so that all the data power of 49 frequency settings are collected. Combined with the profiling data, we finally obtain 931 samples for SVR modeling.

We randomly select 40 frequency pairs and pick out the corresponding samples to train the SVR model and record the model as well as the MSEs of the cross validation and the mean absolute percentage error (MAPE) of the final model on the remaining testing samples. We call it one training group and repeat it one hundred times so that a total of one hundred training groups are obtained.
We than MLR. Besides, we observe that our special feature treatment naive nearly 30% errors. However, the average MAPEs of pp tested kernels and even some kernels like kernels in terms of MAPE. MLR results in 12.5% MAPE across all the our experiments, SVR-based methods outperform MLR for all the using our proposed pp available validation frequency se.

5.2 Experimental Results

We first analyze the distributions of profiling metrics among all the kernels. As Fig. 2 demonstrates, the kernels have various partitions of different types of metrics. It seems that the relationship between these profiling metrics and power consumption scaling is sophisticated. For example, MMS has larger partition of shared memory transactions while the global memory transactions take majority of MMG. However, their power scaling behaviors seem to be close according to Figure 1. This phenomenon indicates the necessity of using machine learning methods.

Then we demonstrate the model accuracy average across all available validation frequency settings of each tested GPU kernel using our proposed pp_SVR, naive_SVR and MLR in Figure 3. In our experiments, SVR-based methods outperform MLR for all the kernels in terms of MAPE. MLR results in 12.5% MAPE across all the tested kernels and even some kernels like QuasiG and TR can have nearly 30% errors. However, the average MAPEs of pp_SVR and naive_SVR are 3.08% and 4.92% respectively, which are much better than MLR. Besides, we observe that our special feature treatment helps improve the SVR-based model accuracy upon most GPU kernels, especially reducing over 10% prediction errors for QuasiG.

To further evaluate model accuracy and stability, the mean square error (MSE) of the leave-out validation subset of each training group is adopted. The average MSE is 0.797 W with 5.5 \times 10^{-3} variance. These results indicate the high accuracy and decent stability of our SVR model. Based on them we can conduct further validation on the testing samples.

We aggregate the absolute precision errors of the testing samples and use a heatmap shown in Figure 4 to illustrate the results. Each entry in Figure 4 is the average error of a certain frequency domain. We observe that our model also achieve great accuracy of no more than 4.5% errors for each frequency pairs.

5.3 Energy Conservation

Figure 5: Energy Conservation achieved most by ground-truth and modeling.

We evaluate the energy efficiency of our model as shown in Figure 5. Our model achieves the best energy efficiency by considering both the ground-truth and modeling results.
5.3 Energy Conservation

Combining our power estimation model and the performance model proposed by [24], we predict the energy consumptions under different frequency settings for each kernel by multiplying the time and power consumption. Then we can find the one that achieves the best energy efficiency. We take the energy consumption under 1000 MHz for both core and memory frequency as the baseline. Then we compare the best energy efficiency achieved in measurements with those predicted by our model. Figure 5 illustrates the results. First, to conserve most energy, the highest core and memory frequency, which drives the program run fastest in general, is rarely the best choice. Second, except CG, Hist and MMG, the modeling results are basically similar to the ground truth. With the prediction from our model, we finally achieve up to 36.8% energy saving and an average of 13.2% among 19 tested GPU kernels.

6 CONCLUSION

In this paper, we derive an SVR-based prediction model to estimate the average power consumptions of different core and memory frequency settings. Our approach provides not only good stability but also decent accuracy. The model takes the profiling data of a given kernel as input to estimate the average power consumptions under new frequency combinations which do not appear in the training data. We show that our model can achieve 3.08% MAPE across up to 2.5x both core and memory frequency scaling. Combined with a precise performance estimation model, we also show that using GPU frequency scaling alone can save an average of 13.2% energy consumption across 19 GPU kernels.

There exist several directions to further improve our prediction model. First, we could further analyze the final SVR model to see how the model performs on different input performance features, which can help us understand the contributions of each individual feature. Second, GPU temperature is also an important factor affecting the runtime power. Third, our current work focuses on frequency scaling only. However to incorporate voltage scaling into our model is also an important issue.

ACKNOWLEDGMENTS

The authors would like to thank all the reviewers for their insightful comments and valuable suggestions. This work is partially supported by research grant HKBU FRG2/14-15/059 and Shenzhen Basic Research Grant SCI-2015-SZTIC-002.

REFERENCES


Black-box Solar Performance Modeling: Comparing Physical, Machine Learning, and Hybrid Approaches

Dong Chen and David Irwin
University of Massachusetts Amherst
{dongchen,deirwin}@umass.edu

ABSTRACT

The increasing penetration of solar power in the grid has motivated a strong interest in developing real-time performance models that estimate solar output based on a deployment’s unique location, physical characteristics, and weather conditions. Solar models are useful for a variety of solar energy analytics, including indirect monitoring, forecasting, disaggregation, anonymous localization, and fault detection. Significant recent work focuses on learning “black box” models, primarily for forecasting, using machine learning (ML) techniques, which leverage only historical energy and weather data for training. Interestingly, these ML techniques are often “off the shelf” and do not incorporate well-known physical models of solar generation based on fundamental properties. Instead, prior work on physical modeling generally takes a “white box” approach that assumes detailed knowledge of a deployment. In this paper, we survey existing work on solar modeling, and then compare black-box solar modeling using ML versus physical approaches. We then i) present a configurable hybrid approach that combines the benefits of both by enabling users to select the parameters they physically model versus learn via ML, and ii) show that it significantly improves model accuracy across 6 deployments.

1. INTRODUCTION

The penetration of intermittent solar power in the grid is rising rapidly due to continuing decreases in the cost of solar modules. For example, the installed cost per Watt (W) for residential photovoltaics (PVs) decreased by 2 × from 2009 to 2015 (from ~$8/W to ~$4/W) [17]. As a result, the return on investment for “going solar” in many locations is now less than five years [24]. In addition, a variety of financing options are now available that lower the barrier to installing solar systems by enabling users to avoid incurring large upfront capital expenses, e.g., by leasing their roof space or entering into a long-term power purchase agreement. However, this increasing solar penetration is placing pressure on grid operators, which schedule generators to maintain a balanced supply and demand. Even when aggregated across many deployments over a large region, solar generation is more stochastic than aggregate demand, since changes in cloud cover (the primary weather metric that affects aggregate solar output) are inherently more localized and stochastic than changes in temperature (the primary weather metric that affects aggregate demand).

The increasing impact of solar on the grid has motivated a strong interest in developing custom performance models that estimate a deployment’s real-time solar output based on its unique location, dynamic and static physical characteristics, and weather conditions. Solar performance models are useful for a variety of energy analytics, including indirect solar monitoring [16], solar forecasting [9, 33], “behind the meter” solar disaggregation [28, 22, 13], anonymous localization [14], and fault detection [19, 7]. Significant recent work focuses on learning “black box” models, primarily in the context of forecasting [9, 33], using machine learning (ML) techniques. Black-box approaches are attractive because they use only historical energy and weather data for training. Thus, utilities and third-parties that remotely monitor tens of thousands of solar deployments, e.g., via smart meters and other sensors, can directly apply black-box techniques at large scales to vast archives of data.

Interestingly, these ML techniques are often “off the shelf” and do not leverage well-known physical models of solar generation based on fundamental physical properties. Instead, prior work on physical modeling generally takes a “white box” approach that assumes detailed knowledge of a deployment and its location, such as the number of modules and their size, tilt, orientation, efficiency, nominal operating cell temperature, wiring, inverter type, etc. White-box physical models translate this information into the parameters the models require. The PV Performance Modeling Collaborative documents a variety of white-box modeling methods [32], and has implemented them as part of the pvlib library [8]. This approach typically decouples the different effects on solar generation and models them separately. For example, different models exist for estimating ground-level irradiance versus estimating a deployment’s efficiency at converting this irradiance to power. The former applies physical models to local or remote sensing data, e.g., ground-level pyranometers or satellites, to estimate irradiance, while the latter applies physical models to estimate the efficiency of converting this irradiance to power. Note that our work focuses on accurately modeling the real-time output of existing solar deployments under current conditions, and not the potential output of future solar deployments. Many tools exist, such as PVWatts [2] and SAM [5], that estimate solar potential using white-box models.

Prior work also leverages stochastic ML techniques to estimate irradiance, and then uses white-box models for estimating conversion efficiency [23]. An example of such a white-box tool is Plant-Predict. Unfortunately, while these white-box approaches have high accuracy, gathering this information at large scales for millions of small-scale deployments is infeasible. As a result, these tools are typically only used for utility-scale solar farms. Of course, while less well-studied, black-box physical modeling using the same fundamental properties is also possible: as we discuss, it simply requires determining the model parameters by finding the values that best fit the data. Such physical modeling generally requires much less data to calibrate (akin to training) than ML modeling, as the physical models embed detailed information about the relation-

1https://plantpredict.com/
Hybrid Solar Performance Modeling

We implement the ML, physical, and hybrid modeling approaches above and evaluate their accuracy across 6 solar deployments with widely different characteristics. We show that the hybrid approach significantly improves the accuracy of the pure ML and physical approach. In addition, we evaluate multiple variants of our hybrid approach by selectively adding more parameters with physical models. We show that the accuracy of the hybrid model incrementally improves as we model more of the input features using physical models.

2. BACKGROUND

While there is significant prior work on ML-based solar modeling, most of it is in the context of solar forecasting, as detailed in recent surveys [9, 33, 23] that cite well-over one hundred papers on the topic. Unfortunately, this prior work generally conflates modeling and forecasting, and thus does not evaluate them separately. In addition, these forecasting approaches often implicitly embed assumptions about their specific problem variant, such as its temporal horizon, temporal resolution, spatial horizon, i.e., forecasting one solar deployment versus many deployments, spatial resolution, performance metrics, weather data, and deployment characteristics. These variants are generally not relevant to solar modeling, which simply estimates solar output (at some resolution) given a set of known conditions, e.g., the location, weather, and time. As a result, extracting a solar performance model from prior work on ML-based forecasting is non-trivial. Thus, for our pure ML-based technique we instead adapt a technique originally proposed for solar disaggregation, which focuses on separating solar generation from total system energy that also includes consumption [28]. However, instead of applying the technique to disaggregate such “net meter” data, we use it to model pure solar data. The technique has been patented by Bidgely, Inc. [29] and is in production use [21].

As discussed in Section 1, prior work on physical modeling generally takes a white-box approach [15, 10]. Our approach to black-box physical modeling is similar to these white-box approaches, in that it uses the same well-known physical models, but instead of directly measuring the necessary input parameters for a deployment, we infer them by searching for values that best fit the available data.

2.1 Black-box ML-based Modeling

Prior work on ML-based black-box solar modeling has the same broad characteristics. Since solar generation varies based on weather conditions, input features include a variety of weather metrics that are publicly available, e.g., from the National Weather Service (NWS) or Weather Underground, such as temperature, dewpoint, humidity, wind speed, and cloud cover. Note that all approaches assume a deployment’s location, and thus its weather is well-known. The dependent output variable is often the raw solar output. Given historical weather data and raw solar output, a variety of supervised ML techniques, e.g., regression, neural nets, Support Vector Machines (SVMs), can learn a model that maps the weather metrics to raw solar output. However, since solar generation potential varies significantly each day and over the year, this approach requires learning a separate model for each time period [31]. This significantly increases the training data required to learn an accurate model, as each sub-model requires distinct training data.

To reduce the size of the training data, ML-based modeling can normalize the input and output variables, such that it can use each datapoint to learn a single model [20]. Our pure ML-based approach normalizes these variables without using detailed physical models of the system [28, 29]. In particular, the approach normalizes the output variable by dividing the raw solar power by the solar capacity, defined as the system’s maximum generation over some previous interval, which it calls the solar intensity. While the prior work does not specify this interval, in this paper, we divide by a solar deployment’s maximum generation over a year. In addition, the approach also adds the time of each datapoint to the input features along with the time of sunrise and sunset. The time information en-
ables the model to automatically learn the solar generation profile. For example, a time closer to sunrise or sunset will have a lower solar intensity, even in sunny clear sky conditions, compared to a time closer to solar noon. The approach then uses a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel to learn a model from the training data. SVM-RBF is common in solar modeling, since it attempts to fit a Gaussian curve to solar data and solar profiles are similar to Gaussian curves [31, 28, 11]. Figure 1 depicts a typical solar profile and its best fit Gaussian curve. As the figure shows, the Gaussian curve fits well in the middle of the day, but diverges at the beginning and end of each day.

Note that the approach above is completely data-driven and does not incorporate any physical models of solar generation, other than the insight that solar curves vary over time and are similar in shape to Gaussian curves. While the approach requires multiple months of training data to learn an accurate model, the authors claim that the normalization enables them to train the model on different solar deployments that they test on, since all solar profiles exhibit the same Gaussian shape. In fact, this model was developed for solar disaggregation, where solar data from the deployment under test is unknown, thus requiring the model to be trained using data from separate deployments. Of course, as we show, the model is more accurate when trained data from the deployment under test due to physical differences between deployments that affect solar output.

### 2.2 Black-box Physical Modeling

Our approach to physical modeling leverages several well-known relationships that govern solar generation. Note that our approach is adapted from an approach we proposed in recent work [13]. However, our prior work, as above, focuses on solar disaggregation of net meter data and not solar performance modeling of pure solar data. Our physical model leverages existing models that estimate the clear sky solar irradiance at any point in time at any location based on the Sun’s position in the sky. Many clear sky irradiance models have been developed over the past few decades with varying levels of complexity [26]. There are multiple libraries available that implement these models [3, 1] with the simplest models requiring as input only a location, i.e., a latitude and longitude, and time. The output is then the expected clear sky irradiance (in kW/m²) horizontal to the Earth’s surface. This is the maximum solar energy available to a solar module to convert to electricity.

Of course, solar modules cannot convert all the available solar energy into electricity. Their efficiency is based on the type of module, e.g., poly- versus mono-crystalline, as well as their orientation and tilt. The simple well-known equation below describes the amount of power a solar module generates based on its tilt ($\beta$) and orientation ($\phi$) relative to the Earth’s surface, and the Sun’s zenith ($\Theta$) and azimuth ($\alpha$) angles (which are a direct function of the location and time [12]). Assuming clear skies, $I_{\text{incident}}$ is the clear sky solar irradiance, and $k$ is a module-specific parameter that is combines conversion efficiency (as a percentage) and module size (in m²). Similar expressions exist for deployments that track the sun, or consist of multiple modules with different tilts and orientations.

$$ P_s(t) = I_{\text{incident}}(t) \cdot k \cdot [\cos(90 - \Theta) \cdot \sin(\beta) \cdot \cos(\phi - \alpha) + \sin(90 - \Theta) \cdot \cos(\beta)] $$

(1)

White-box models can directly measure the module angles, size, and efficiency. While black-box models cannot directly measure these values, given the relationships above, it can search for these parameters via curve fitting. In particular, $P_s(t)$ follows the equation above and $I_{\text{incident}}(t)$ is known from existing clear sky models. To search, we can set the tilt and orientation to their ideal values (a tilt equal to the location’s latitude and a south-facing orientation in the northern hemisphere), and then conduct a binary search for the $k$ that both minimizes the Root Mean Squared Error (RMSE) with the observed data and represents a strict upper bound on the data, as we know generation should never exceed the maximum dictated by the clear sky irradiance. After fitting $k$, we then conduct a similar binary search for orientation and tilt. We iterate on the search until the parameters do not significantly change. In prior work, we have shown that this searching method results in highly accurate values for $k$ and the orientation and tilt angles [13].

The model found above assumes that $k$ is static and never changes. However, module efficiency changes over time based on numerous dynamic conditions, such as temperature, rain, snow, humidity, dust, etc. In particular, the effects of temperature on module efficiency are well-known, and are described by a variety of physical models. The simplest model is the Nominal Operating Cell Temperature (NOCT) model, which specifies the cell temperature based on the ambient air temperature and the cell temperature at 1kW/m² in 25°C. For every degree increase (or decrease) in $T_{\text{cell}}$, module efficiency drops (or rises) by roughly a constant percentage, which varies between modules, but is $\sim 0.5\%$ per degree Celsius.

To account for temperature effects, we can re-calibrate our model by adjusting the original value of $k$ above based on the temperature at each datapoint using the equation below, where $T_{\text{baseline}}$ is the temperature at the datapoint that is closest to the upper bound solar curve in the model above. Note that the relationship between cell temperature and air temperature is a constant. While efficiency varies strictly based on cell temperature, the cell temperature’s relationship to air temperature differs only by an additive constant, which cancels out when subtracting two cell temperatures (leaving only the air temperature below). The baseline temperature should represent the coldest point in the year that has a clear sky. Again, we search for the value of $c$ that minimizes the RMSE with the observed data but remains a strict upper bound on the data.

$$ k'(t) = k \cdot (1 + c \cdot (T_{\text{baseline}} - T_{\text{air}}(t))) $$

(2)

The adjustment above represents a temperature-adjusted clear sky solar generation model. Of course, skies are not always clear, such that the solar irradiance that reaches Earth is much less than the clear sky solar irradiance. The amount of cloud cover is the primary metric that dictates the fraction of the maximum solar irradiance that reaches the ground. As above, there are numerous well-known physical models [30, 34] that translate cloud cover into a clear sky index, which is the solar irradiance that reaches the Earth’s surface divided by the clear sky solar irradiance [25]. For example, one well-known cloud cover model is below [4].

$$ I_{\text{incident}}/I_{\text{clearsky}} = (1 - 0.75n^{0.34}) $$

(3)

Here, $I_{\text{incident}}$ represents the solar irradiance that reaches the Earth, $I_{\text{clearsky}}$ represents the solar irradiance from the clear sky model, and $n$ represents the fraction of cloud cover (0.0-1.0). This cloud cover (or sky condition) is typically measured in oktas, which
represents how many eighths of the sky are covered in clouds, ranging from 0 oktas (completely clear sky) through to 8 oktas (completely overcast). The sky conditions reported by the NWS translate directly to oktas [6]. For example “Clear/Sunny” is <1 okta, “Mostly Clear/Mostly Sunny” is 1-3 oktas, “Partly Cloudy/Partly Sunny” is 3-5 oktas, “Mostly Cloudy” is 5-7 oktas, and “Cloudy” is 8 oktas. While the sky condition reported by the NWS (and other sources) is a rough measure of cloud cover, more accurate measures can be extracted from satellite images [18]. However, this is non-trivial and these measures are not reported by weather sites.

Thus, using the equation above we can adjust the output of our physical model by multiplying the solar output in our temperature-adjusted model above by the fraction $I_{\text{incident}}/I_{\text{clearsky}}$. Note that, while Equation 3 is in terms of solar irradiance and not solar power, the ratio of observed solar power to maximum solar generation potential after the temperature adjustment (from Equation 1) are equivalent, since the effect of the physical characteristics cancel out. Recent work refers to this value as the clear sky photovoltaic index [16]. We could continue to adjust our model downwards based on physical models for other conditions, such as humidity, air velocity, and dust buildup [27]. Unfortunately, similar types of simple models are not readily available for these parameters.

One benefit of the physical model above is that it requires very little data to calibrate. In the limit, it requires only two datapoints during clear skies with a significant difference in temperature. In recent work, we show that physical models of clear sky generation (without the cloud cover adjustment) built with only two days of data have similar accuracy to those built with a year’s worth of data [13]. However, unlike the ML-based models, our physical model is necessarily custom to each deployment based on its unique location, tilt, orientation, efficiency, and sensitivity to temperature. Our physical model also does not account for shade from surrounding structures, e.g., buildings and trees, or multi-module systems with different tilts, orientations, and efficiencies that are wired together, e.g., in series, parallel, or a combination. While accounting for these effects in the physical model is possible, it would significantly increase its complexity. In contrast, the ML-based model is capable of inherently incorporating these effects into its model.

3. A BLACK-BOX HYBRID MODEL

The black-box ML and physical solar performance models from the previous section have both benefits and drawbacks. The ML model generally requires months of training data to build an accurate model. As we show, while we can train the pure ML model on data from one set of solar deployments, and then use it to model a separate set of solar deployments, this significantly decreases the model’s accuracy, since the approach does not take into account different physical system characteristics, e.g., tilt, orientation, size, and efficiency. In contrast, while our physical model requires little data to calibrate, it is generally less accurate than the ML model in practice because it i) depends on coarse measurements of cloud cover that are often inaccurate and ii) does not incorporate the effect of other conditions that degrade output, such as additional weather metrics, complex multi-panel characteristics, dust and snow buildup, and regular shading patterns from nearby structures. Thus, to leverage the benefits of both approaches, we present a configurable hybrid approach that combines both approaches.

Our hybrid approach first builds a physical model of solar output, as in Section 2.2, based on a deployment’s location, tilt, orientation, size, efficiency, and any other relevant parameters where physical models exist. The approach then trains a ML classifier, similar to the one in Section 2.1, that includes as input features any relevant parameters not included in the physical models. However, a key difference relative to Section 2.1 is that the dependent output variable is not the raw power normalized by the (static) solar capacity, but is instead the raw power normalized by the generation potential from the physical model above. Thus, the dependent output variable represents the additional percentage reduction in solar generation beyond that estimated by the physical model due to the parameters in the ML model. For example, the physical model might estimate a solar output of 1kW based on the current location, time, temperature, and cloud cover. However, based on the other metrics, the ML model may then estimate the actual output to be 80% of this 1kW output. In this case, the labeled data in the training set for the ML model would include any input features that are not physically modeled with an output variable of 0.80.

Thus, our hybrid model estimates solar output by multiplying the estimated output from the physical model by the fraction specified in the ML model. Note that, when the physical model includes only the metrics that affect module efficiency, e.g., tilt, orientation, size, and temperature, this ratio represents the clear sky (photovoltaic) index [16]. Our hybrid approach is configurable because we can either model input features with physical models, or using the ML model. For example, in our evaluation, we examine different hybrid variants that physically models different sets of parameters.

Note that, since the physical model is already a function of time, our ML classifier does not need sunrise, sunset, or current time as input features, unlike the pure ML model from Section 2.1. In addition, by specifying our output variable as a function of the physical model, its normalization naturally takes into account the physical differences between solar deployments. Thus, as with our pure ML model, our hybrid approach can accurately train its ML model on data from one set of solar deployments, and then apply it to a separate set of deployments with widely different physical characteristics. Of course, for any new deployment, we would still need to calibrate a physical model of the system, as described in Section 2.1. However, as we discuss, this only requires a minimal amount of data. In some sense, our physical model captures how efficiently a solar deployment translates the available solar irradiance into electricity, while our ML model captures how much solar irradiance actually reaches the module. As we show in recent work [13], the latter is primarily due to weather effects that are general and not dependent on specific physical deployment characteristics.

In this paper, we use the same classifier (SVM-RBF) in our hybrid ML model as we do in the pure ML model [28] to provide a direct comparison. More sophisticated ML modeling techniques could potentially learn the physical models above from training data without requiring the manual identification we perform in our hybrid approach. However, for systems, such as solar deployments, where the physical effect from a subset of inputs on a dependent output variable is well-known, and independent of the other inputs, it is more straightforward to simply calibrate the input directly from the data using the model. As we show, this approach significantly increases accuracy using straightforward ML techniques.

4. IMPLEMENTATION

We implement the ML-based, physical, and hybrid black-box solar performance models using python. We use the scikit-learn machine learning library to implement our ML-based models. We implement the pure ML-based model as specified in prior work [28, 29] using the same input features, dependent output variable, and SVM-RBF kernel. In particular, we use one hour resolution weather metrics (from Weather Underground) including the sky cover, dewpoint, humidity, temperature, and windspeed. We translate the sky cover string into a cloud cover percentage using the standard okta translation [6]. Our physical model leverages the
where we train a ML model for each deployment using its historical data, while the middle and bottom graphs train a ML model on 4 separate homes (not in the set of six) and then apply that same model to each of these 6 homes. The top two graphs compute MAPE over each day (across a year of data), while the bottom graph computes it from 10am-2pm. Note that the physical model requires no training; we include it in all the graphs for comparison.

The experiment shows that the physical model performs significantly worse than all the models that use ML. This is primarily due to i) the coarseness and imprecision of the cloud cover metric, and ii) that it cannot account for conditions that do not permit physical modeling, including the effect of other weather metrics [27]. As part of future work, we are leveraging various satellite images to better quantify real-time cloud cover, such as via the HELIOSTAT method [18], which should improve the results of the analytical model. Unfortunately, an accurate and precise cloud cover metric is not available via common weather services and APIs. In contrast, the pure ML approach can inherently incorporate such effects and achieves a significantly higher accuracy in all cases. Importantly, though, the hybrid model, even without including temperature and cloud cover, significantly improves on the pure ML approach. For example, for deployment’s #3 and #5 in the top graph, the improvement is over a 30% reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.

In addition, as we incorporate more physical parameters into the hybrid model, the more accurate the model becomes. This is most evident when shifting temperature from the ML model to the physical model, which results in another significant decrease in MAPE in all cases. Further, even though cloud cover is a coarse and imprecise metric, by incorporating it into the physical model (along with temperature), we again observe a slight reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.

PySolar [3] library for computing the clear sky irradiance at any location and time, which it uses to find the tilt, orientation, size, efficiency, and temperature coefficient that best fits the data. Our basic hybrid ML model uses the same weather metrics as with the pure ML-based model [28, 29], and thus does not include temperature and cloud cover as part of the physical model. We implement two other hybrid variants: one that physically models temperature and thus takes it out of the training set of input features, and one that physically models both temperature and cloud cover, which also takes cloud cover out of the ML model’s training set.

We evaluate the accuracy of each model on data from 6 rooftop solar deployments at different locations with widely different physical characteristics. Since our weather data has one-hour resolution, we use average power data at one-hour resolution in our evaluation. We examine model accuracy using two different training scenarios, where we train the ML models (both pure and hybrid) using data from either i) the same deployment we test on or ii) different deployments than we test on. In the former scenario, we perform cross-validation across one-year of data to split the dataset into a training and testing set (in a 2:1 ratio). In the latter case, we train the ML model using one year of data from 4 other deployments, and then apply the model to estimate solar output over one year from the 6 deployments. Since, due to Figure 1, the Gaussian fit is most inaccurate during the morning/evening, we evaluate accuracy over both the entire day and over mid-day between 10am and 2pm.

Finally, we quantify model accuracy using the Mean Absolute Percentage Error (MAPE), as follows, between the ground truth solar energy ($S$) and the solar energy estimated by our models ($P_t$) at all times $t$. A lower MAPE indicates higher accuracy with a 0% MAPE being a perfect model.

$$MAPE = \frac{100}{n} \sum_{t=0}^{n} \left| \frac{S - P_t}{S} \right|$$

5. EXPERIMENTAL EVALUATION

Figure 2 quantifies model accuracy for the 6 buildings with rooftop deployments in our test set across multiple scenarios. Buildings #1-#6 are located in Pennsylvania, Texas, New York, Arizona, Washington, and Massachusetts, respectively. The deployments have a wide range of sizes: buildings #1-#6 consist of 110, 16, 93, 36, 17, and 30 solar modules, respectively, with a standard size of 165cm×99cm which typically have a rated capacity of ~230-330 based on the module type. The top graph is the scenario where we train a ML model for each deployment using its historical data, while the middle and bottom graphs train a ML model on 4 separate homes (not in the set of six) and then apply that same model to each of these 6 homes. The top two graphs compute MAPE over each day (across a year of data), while the bottom graph computes it from 10am-2pm. Note that the physical model requires no training; we include it in all the graphs for comparison.

The experiment shows that the physical model performs significantly worse than all the models that use ML. This is primarily due to i) the coarseness and imprecision of the cloud cover metric, and ii) that it cannot account for conditions that do not permit physical modeling, including the effect of other weather metrics [27]. As part of future work, we are leveraging various satellite images to better quantify real-time cloud cover, such as via the HELIOSTAT method [18], which should improve the results of the analytical model. Unfortunately, an accurate and precise cloud cover metric is not available via common weather services and APIs. In contrast, the pure ML approach can inherently incorporate such effects and achieves a significantly higher accuracy in all cases. Importantly, though, the hybrid model, even without including temperature and cloud cover, significantly improves on the pure ML approach. For example, for deployment’s #3 and #5 in the top graph, the improvement is over a 30% reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.

In addition, as we incorporate more physical parameters into the hybrid model, the more accurate the model becomes. This is most evident when shifting temperature from the ML model to the physical model, which results in another significant decrease in MAPE in all cases. Further, even though cloud cover is a coarse and imprecise metric, by incorporating it into the physical model (along with temperature), we again observe a slight reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.

In addition, as we incorporate more physical parameters into the hybrid model, the more accurate the model becomes. This is most evident when shifting temperature from the ML model to the physical model, which results in another significant decrease in MAPE in all cases. Further, even though cloud cover is a coarse and imprecise metric, by incorporating it into the physical model (along with temperature), we again observe a slight reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.

In addition, as we incorporate more physical parameters into the hybrid model, the more accurate the model becomes. This is most evident when shifting temperature from the ML model to the physical model, which results in another significant decrease in MAPE in all cases. Further, even though cloud cover is a coarse and imprecise metric, by incorporating it into the physical model (along with temperature), we again observe a slight reduction in MAPE. Significant, although slightly lesser, improvements are also apparent in the middle graph. The reason for this reduction stems from normalizing the output variable of the hybrid approach’s ML model based on a custom physical model of the deployment’s output over time, rather than a static capacity value as in the pure ML model.
proach, since the Gaussian fit is much more accurate in the middle of the day. However, our hybrid approach significantly improves upon the pure ML model when incorporating the physical models for temperature and cloud cover (even during the mid-day hours in the bottom graph), especially for deployments #3, #5, and #6.

Overall, our results indicate that the hybrid approach achieves much better accuracy than either the pure ML or pure physical approach in all cases. In addition, by training the ML model on separate deployments than we test on, the hybrid model requires only a small amount of training data (as few as two datapoints) from the system under test to calibrate an accurate model.

The model error of our black-box approach is likely higher (~15-25) MAPE than that of highly-tuned white-box approaches. However, a direct comparison is difficult as prior work uses a wide range of error metrics. In many cases, these metrics are not normalized, and thus vary based on a deployment’s capacity. In addition, the variability of weather at a location also affects model accuracy. For example, solar performance models are likely to be more accurate in San Diego, where there is little variation in weather, compared to Massachusetts where weather has more day-to-day and season-to-season changes. As part of future work, we plan to incorporate more accurate estimations of ground-level irradiance from visible satellite imagery, such as offered by SolarAnywhere and SoDa. We expect this to significantly improve accuracy relative to the coarse cloud-cover metrics in standard weather data.

6. CONCLUSION

This paper surveys and compares different approaches to black-box solar performance modeling. We compare a pure ML model from prior work [28, 29], a black-box physical model based on well-known relationships in solar generation [13], and a configurable hybrid approach that combines the benefits of both by achieving the most accurate results with little historical data. Our results motivate using physical models when relationships are well-known, and leveraging ML to quantify the effect of unknown relationships. Our black-box solar modeling has applications to a wide range of solar analytics, which we plan to explore as part of future work. Finally, our methodology is potentially generalizable to other complex physical systems where the physical effect from a subset of inputs on a dependent output variable is well-known, and independent of other inputs, which have an unknown effect.

Acknowledgements. This research is supported by NSF grants IIP-1534080, CNS-1405826, CNS-1253063, CNS-1505422, and the Massachusetts Department of Energy Resources.

7. REFERENCES

Battery Swapping Assignment for Electric Vehicles: A Bipartite Matching Approach

Pengcheng You, Youxian Sun  
Zhejiang University  
Hangzhou, China 310027  
{pcyou,yxsun}@iipc.zju.edu.cn

John Pang, Steven Low  
California Institute of Technology  
Pasadena, California 91125  
jzpang,slow}@caltech.edu

Minghua Chen  
Chinese University of Hong Kong  
Hong Kong, China  
minghua@ie.cuhk.edu.hk

ABSTRACT
This paper formulates a multi-period optimal station assignment problem for electric vehicle (EV) battery swapping that takes into account both temporal and spatial couplings. The goal is to reduce the total EV cost and station congestion due to temporary shortage in supply of available batteries. We show that the problem is reducible to the minimum weight perfect bipartite matching problem. This leads to an efficient solution based on the Hungarian algorithm. Numerical results suggest that the proposed solution provides a significant improvement over a greedy heuristic that assigns EVs to nearest stations.

1 INTRODUCTION
EVs are large loads that can add significant stress to electricity grids, but they are also flexible loads that can help mitigate the volatility of renewable generation through smart charging. EV charging however takes a long time. It is not suitable for commercial vehicles, such as taxis, buses, and ride-sharing cars, that are on the road most of the time, the opposite of most private cars. An alternative EV refueling method is battery swapping where an EV swaps its depleted battery for a fully-charged battery at a service station\(^1\). This can be done in a few minutes. Several such electric taxi programs are in pilot in China [2].

1.1 Literature review
The literature on scheduling of EV battery swapping is small. In [4] the operation of a battery charging and swapping station is modeled as a mixed queuing network, consisting of an interior closed queue of batteries going through charging and swapping, and an exterior open queue of EV arrivals. Using this model, [3] proposes an optimal charging policy. An optimal assignment problem is formulated in [5, 6] that assigns to a given set of EVs best stations to swap their batteries. The assignment aims to minimize a weighted sum of generation cost and EVs’ travel distance by jointly optimizing power flow variables. The problem focuses on spatial optimization over power grid operation for a single time slot during which the set of EVs is fixed.

1.2 Summary
This paper investigates the battery swapping assignment for EVs and takes into account temporal optimization where EVs arrive over several time slots. Specifically, we adopt a discrete time model. In each time slot, a centralized operator optimally assigns stations to a set of EVs that need battery swapping. Consider the optimal station assignment problem at time slot 1 where stations are assigned in a way that minimizes both the total EVs’ cost to travel to their assigned stations and the total congestion (battery shortage) levels at these stations. The problem is a binary program with strong temporal and spatial couplings. We show that it is polynomial-time solvable by reducing it to the standard minimum weight perfect bipartite matching problem. This leads to a solution based on the Hungarian algorithm for bipartite matching problems.

2 PROBLEM FORMULATION
Consider a group of EVs that swap their depleted batteries for fully-charged ones at stations assigned by a central operator. Time is slotted with a constant length. Fix the current time slot as time slot 1 of the time horizon \(\mathbb{T} := \{-T_m + 1, \ldots, 0, 1, \ldots, T_m\}\), and let \(\mathbb{T}^+ := \{1, \ldots, T_m\}\). \(T_m\) is a constant which we will interpret later. Suppose there is a set \(J := \{1, \ldots, J]\) of stations that provide battery swapping service for EVs. At the current time slot 1, let \(\mathbb{I} := \{1, \ldots, I\}\) be the set of EVs that require battery swapping. Our goal is to optimally assign a station \(j \in J\) to each EV \(i \in \mathbb{I}\), such that a weighted sum of aggregate EV cost and station congestion is minimized.

2.1 Variables, states, and constraints
Let \(M := (M_{ij}, i \in \mathbb{I}, j \in J\) represent the current station assignment to EVs, where \(M_{ij} = 1\) if station \(j\) is assigned to EV \(i\) and \(M_{ij} = 0\) otherwise. We require that only one station be assigned to each EV, i.e.,

\[
\begin{align*}
\sum_{j \in J} M_{ij} &= 1, & i & \in \mathbb{I} \\
M_{ij} &\in \{0, 1\}, & i & \in \mathbb{I}, j \in J
\end{align*}
\]

(1)

Note that we also use \(M_{ij}(t)\), \(t = -T_m + 1, -T_m + 2, \ldots, 0\), to represent past assignments, which are given.

Let \(\tau_{ij}(t)\) estimate the arrival time of EV \(i\) if it starts to travel at time slot \(t\) from its location to station \(j\). It captures the time-dependent traffic conditions and also corresponds to an optimal routing, thus we can readily obtain the associated travel distance, denoted by \(d_{ij}(t)\). We also define \(\tau_{ij}^{-1}(t)\) as the inverse function of \(\tau_{ij}(t)\), i.e., \(\tau_{ij}^{-1}(t)\) is the time slot when station \(j\) was assigned to EV \(i\) that arrives at time slot \(t\). For brevity, let \(\tau_{ij} := \tau_{ij}(1)\) and \(d_{ij} := d_{ij}(1)\), which are available by resorting to, say, Google Maps, and their explicit modeling goes beyond the scope of this paper.

Now we interpret \(T_m\) as the maximum travel time of an EV to reach a station, i.e., \(T_m := \max_{i,j} (\tau_{ij}(t) - t + 1)\). The assignments before \(-T_m + 1\) are summarized in \(n_j^0\), and the states of stations after \(T_m\) will not be directly affected by the current assignment.

Let \(n_j(t)\) denote the number of available (fully-charged) batteries at station \(j\) at the end of time slot \(t\), which is the station state. In

\[\]
particular, \( n_j(0) \), i.e., the current number of available batteries at station \( j \), is observed and given. Hence \( n_j(t) \) increases by 1 when a battery at station \( j \) becomes fully-charged, and decreases by 1 when a fully-charged battery is removed by an EV (battery swapping time is ignored):

\[
\begin{align*}
n_j(t) &= n_j(t - 1) + c_j(t) - \sum_{i \in \mathcal{P}} M_{ij}(\tau_{ij}^{-1}(t)) \\
&\quad - \sum_{i \in \mathcal{I}} M_{ij} \cdot 1(t = \tau_{ij}), \quad t \in \mathbb{T}^+ 
\end{align*}
\]

where \( c_j(t) \) is the number of batteries that become fully-charged at station \( j \) in time slot \( t \) (which is known a priori), and \( \mathcal{P} \) is the set of all past EVs that were assigned stations during the time interval \([-T_m + 1, 0] \). \( 1(x) \) is an indicator function for the predicate \( x \). The third and fourth terms on the right-hand-side of (2) summarize the impacts of past assignments and the current one, respectively. The second and third terms are both given while the fourth one is to be decided. Note that \( n_j(t) \) can be negative. For instance, \( n_j(t) = -3 \) means there will be no available battery at the end of time slot \( t \), but 3 waiting EVs.

An EV can only be assigned a station within its driving range, i.e.,

\[
d_{ij}M_{ij} \leq rs_j, \quad i \in \mathcal{I}, j \in \mathcal{J} \tag{3}
\]

where \( r \) is the driving range per unit state of charge and \( s_j \) denotes the state of charge of EV \( i \).

### 2.2 Optimal station assignment problem

The system cost has two components. First, a cost \( \alpha_{ij} \) is incurred if station \( j \) is assigned to EV \( i \), thus the cost of EV \( i \) is \( \sum_{j \in \mathcal{J}} \alpha_{ij}M_{ij} \). For example, \( \alpha_{ij} \) can be a weighted sum of EV \( i \)'s travel distance and time from its current location to station \( j \). Second, as explained above, \((\neg n_j(t))^+\) is the number of waiting EVs at the end of time slot \( t \), where \((x)^+ := \max\{x, 0\}\). Let \( n_\mathcal{I} := (n_j(t), j \in \mathcal{J}, t \in \mathbb{T}^+) \) be the vector of station states. We are interested in the following optimal station assignment problem:

\[
\begin{align*}
\min_{\mathcal{M}, n} & \quad \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \alpha_{ij}M_{ij} + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} (-n_j(t))^+ \tag{4} \\
\text{s.t.} & \quad (1), (2), (3)
\end{align*}
\]

which minimizes the weighted sum of aggregate EV cost and station congestion, subject to EVs’ driving ranges.

### 3 POLYNOMIAL-TIME SOLUTION

The optimal station assignment problem (4) is a binary program with temporal coupling (2) and spatial coupling implied in station congestion. It can however be solved efficiently.

**Theorem 3.1.** The optimal station assignment problem (4) is solvable in polynomial time.

We prove the theorem in the following two steps.

#### 3.1 Reformulation as MILP

Note that all the constraints in (4) are linear in the variables \((\mathcal{M}, n)\). The only nonlinearity is \((\neg n_j(t))^+\), which can be removed by introducing auxiliary variables \( v_j(t), j \in \mathcal{I}, t \in \mathbb{T}^+ \) to replace aggregate station congestion by \( \sum_j v_j(t) \) and requiring \( v_j(t) \) to satisfy the linear constraints \( v_j(t) \geq 0 \) and \( v_j(t) \geq -n_j(t) \). Hence (4) is an MILP.

To reformulate it into a more convenient form, denote the number of available batteries at station \( j \) over \( \mathbb{T}^+ \) observed at time slot 1 before the current decision \( M \) is made by:

\[
\tilde{n}_j(t) := n_j(0) + \sum_{k=1}^t (c_j(k) - \sum_{i \in \mathcal{P}} M_{ij}(\tau_{ij}^{-1}(k))), \quad t \in \mathbb{T}^+
\]

It is a known constant determined by past assignments. The evolution of \( n_j(t) \) in (2) then reduces to

\[
n_j(t) = \tilde{n}_j(t) - \sum_{i \in \mathcal{I}} M_{ij} \cdot 1(t \geq \tau_{ij}), \quad t \in \mathbb{T}^+ \tag{5}
\]

which is decoupled across time slots, because \( \tilde{n}_j(t) \) and the indicator function in (5) remove the dependency of \( n_j(t) \) on \( n_j(t - 1) \).

The interpretation of \( M_{ij} \cdot 1(t \geq \tau_{ij}) \) in (5) is as follows. If station \( j \) is assigned to EV \( i \) at time slot 1, then it will arrive at time slot \( \tau_{ij} \), thus removing one available battery from station \( j \) for time slot \( \tau_{ij} \) and every time slot afterwards. For each station \( j \in \mathcal{J} \), define an arrival matrix \( A_j \in \{0, 1\}^{\mathcal{T} \times \mathcal{l}} \) such that its \((t, i)\) entry is

\[
A_j(t, i) := 1(t \geq \tau_{ij})
\]

Finally, let \( \mathcal{I} \) denote the set of \( M \) with \( M_{ij} = 0 \) if station \( j \) is outside EV \( i \)'s driving range, i.e., \( d_{ij} > rs_j \), and put the above together, then (4) is equivalent to the following MILP:

\[
\begin{align*}
\min_{\mathcal{M}, \mathcal{V}, v \geq 0} & \quad \sum_{j \in \mathcal{J}} \alpha_{ij}M_{ij} + \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} v_j(t) \\
\text{s.t.} & \quad \sum_{i \in \mathcal{I}} M_{ij} = 1, \quad i \in \mathcal{I} \\
& \quad v_j(t) \geq -\tilde{n}_j(t) + \sum_{i \in \mathcal{I}} A_j(t, i)M_{ij}, \quad j \in \mathcal{J}, t \in \mathbb{T}^+ \tag{6}
\end{align*}
\]

#### 3.2 Reduction to Bipartite Matching

We now show that the MILP (6) can be further reduced to the minimum weight perfect bipartite matching problem, which is well known to be polynomial-time solvable.

Define a bipartite graph \( \mathcal{G} = (\mathcal{A} \cup \mathcal{B}, \mathcal{E}) \), where \( \mathcal{A} \) and \( \mathcal{B} \) are the bipartition of the vertex set and \( \mathcal{E} \subseteq \mathcal{A} \times \mathcal{B} \) is the set of edges that are endowed with given weights \( \omega := (\omega_{ab}, a \in \mathcal{A}, b \in \mathcal{B}, (a, b) \in \mathcal{E}) \). Without loss of generality, we assume \( \mathcal{G} \) is complete and balanced as we can add infinite-weight edges and dummy vertices as necessary. Let \( N := |\mathcal{A}| = |\mathcal{B}| \). The standard minimum weight perfect matching problem defined on \( \mathcal{G} \) is

\[
\begin{align*}
\min_x & \quad \sum_{(a, b) \in \mathcal{E}} \omega_{ab}x_{ab} \\
\text{s.t.} & \quad \sum_b x_{ab} = 1, \quad a \in \mathcal{A} \\
& \quad \sum_a x_{ab} = 1, \quad b \in \mathcal{B} \\
& \quad x_{ab} \in \{0, 1\}, \quad a \in \mathcal{A}, b \in \mathcal{B} \tag{7}
\end{align*}
\]

where \( x := (x_{ab}, a \in \mathcal{A}, b \in \mathcal{B}) \). Hence an instance of the bipartite matching problem (7) such that an optimal solution to (7) yields an optimal solution to (6).

Let \( \mathcal{A} := \bigcup \mathcal{I} \bigcup \mathcal{I} \) is the set of EVs that were previously assigned stations, but have yet to have their batteries swapped (either on the way or waiting at stations). We restrict the matchings of EVs in \( \mathcal{I} \)
only with batteries at their originally assigned stations, as captured in (6). \( \mathbb{D} \) is the set of dummy EVs if necessary to make \( \mathbb{A} \) and \( \mathbb{B} \) balanced.

Let \( \mathbb{B} := \bigcup_{j \in \mathbb{I}} \mathbb{B}_j \cup \mathbb{D} \). \( \mathbb{D} \) is the set of available batteries at station \( j \), including not only the currently available batteries, but also those that will become available in \( \mathbb{T}^+ \). The time slot when battery \( b \in \mathbb{B}_j \) becomes available is denoted as \( \rho_b \), and \( \rho_b = 0 \) for the currently available batteries. \( \mathbb{D} \) is the set of dummy batteries if necessary to make \( \mathbb{A} \) and \( \mathbb{B} \) balanced.

Note that \( n_j(t) \) can be negative in (6). We have to make up the shortfall when \( \left| \mathbb{I} \cup \mathbb{I} \right| > \left| \mathbb{B}_j \right| \) for station \( j \) by adding dummy batteries. More precisely, \( \mathbb{D} := \bigcup_{j \in \mathbb{I}} \mathbb{D}_j \), where \( |\mathbb{D}_j| = \max(\left| \mathbb{I} \cup \mathbb{I} \right| - |\mathbb{B}_j|, 0) \). Then \( \mathbb{I}^d \) with \( |\mathbb{I}^d| = \left| \bigcup_{j \in \mathbb{I}} (\mathbb{B}_j \cup \mathbb{D}_j) \right| - \left| \mathbb{I} \cup \mathbb{I} \right| \) is added to maintain balance between \( \mathbb{A} \) and \( \mathbb{B} \).

The nonnegative weight \( \omega_{ab} \) of the match \((a, b)\) corresponds to the incremental cost added to the objective of (6) if station \( j \) which battery \( b \) belongs to is assigned to EV \( a \). Note that \( \omega_{ab} := \max(\rho_b - \tau_{ab}, 0) \). Here \( \max(\rho_b - \tau_{ab}, 0) \) is the time length for which EV \( a \) has to wait until battery \( b \) becomes available. If \( \rho_b > \tau_{ab} \), \( \omega_{ab} := \infty \).

Case 1: \( a \in \mathbb{I}, b \in \mathbb{B}_j \). Set \( \omega_{ab} := \sigma_{ab} + \max(\rho_b - \tau_{ab}, 0) \). Here \( \max(\rho_b - \tau_{ab}, 0) \) is the time length for which EV \( a \) has to wait until battery \( b \) becomes available. If \( \rho_b > \tau_{ab} \), \( \omega_{ab} := \infty \).

Case 2: \( a \in \mathbb{I}, b \in \mathbb{D}_j \). EVs matched with dummy batteries will wait until the end of \( T^a \) after their arrivals. Hence \( \omega_{ab} := \sigma_{ab} + (T + 1 - \tau_{ab}) \).

Case 3: \( a \in \mathbb{I}, b \in \mathbb{B}_j \). EVs stay original assigned stations. If station \( j \) is originally assigned to EV \( a \), \( \omega_{ab} := \max(\rho_b - \tau_{ab}, 0) \); otherwise, \( \omega_{ab} := \infty \). No EV cost is included.

Case 4: \( a \in \mathbb{I}, b \in \mathbb{D}_j \). Likewise, if station \( j \) is original assigned to EV \( a \), \( \omega_{ab} := \max(\rho_b - \tau_{ab}, 0) \); otherwise, \( \omega_{ab} := \infty \).

Case 5: \( a \in \mathbb{I}, b \in \mathbb{B}_j \). Dummy EVs do not really exist, and have no impact on the match result. Thus we have \( \omega_{ab} := 0 \).

Case 6: \( a \in \mathbb{I}, b \in \mathbb{D}_j \). Likewise, \( \omega_{ab} := 0 \).

From above, the parameters of (7) and \( \omega_{ab}, a \in \mathbb{A}, b \in \mathbb{B} \) can be computed in time of \( O(N^2) \) given an instance of (6). On the other hand, if we have an optimal matching \( x^* \) for (7), an optimal assignment is straightforward:

\[
M^*_{ij} = \sum_{b \in \mathbb{B}_j} x^*_{ib}, \quad i \in \mathbb{I}, j \in \mathbb{I}
\]

which is obtainable in time of \( O(N) \).

Hence the optimal station assignment problem (4) is reduced to the minimum weight perfect bipartite matching problem (7), which is solvable in polynomial time of \( O(N^3) \) by the well-known Hungarian algorithm [1]. This proves Theorem 3.1.

4 NUMERICAL RESULTS

We illustrate with a case study of \( I = 25 \) EVs and \( J = 3 \) stations. Fix \( T = 6 \), and other parameters are randomly generated, given which \( \left( \bar{n}_j(t), j = 1, 2, 3, t = 1, 2, \ldots, 6 \right) \) is attainable, as the red dash lines show in Fig. 1(a).

The proposed approach efficiently computes an optimal assignment; see Fig. 1(a) for how the number of available batteries at each station evolves after the assignment. Batteries at station 1 in the first half of the time horizon are almost fully utilized to avoid unduly congesting stations 2 and 3. In the second half, all stations run out of batteries. Then the optimal assignment strikes a compromise between the least EV cost and the latest time of arrival. For this test case, a 49.20% improvement is achieved by the proposed approach compared with a heuristic that assigns each EV to its nearest station.

We check the computational efficiency of the proposed approach by scaling up the number of EVs that require battery swapping while fixing other parameters with \( J = 10 \). The computation time required to run our algorithm on a normal laptop PC is shown in Fig. 1(b).

REFERENCES

Distributed Lagrangian Method for Tie-Line Scheduling in Power Grids under Uncertainty

Thinh T. Doan Subhonmesh Bose Carolyn L. Beck

ABSTRACT

System operators (SOs) manage the grid and its assets in different parts (areas) of an interconnected power network. One would ideally seek to co-optimize the grid assets across multiple areas by solving a centralized optimization problem. Gathering the dispatch cost structures and the network constraints from all areas for a centralized solution remains difficult due to technical, historical, and sometimes legal barriers. Motivated by the need for a distributed solution architecture for multi-area power systems, we propose a distributed Lagrangian algorithm in this paper. We establish convergence rates for our algorithm that solves the deterministic tie-line scheduling problem as well as its robust variant (with policy space approximations). Our algorithm does not need any form of central coordination. We illustrate its efficacy on IEEE test systems.

1 INTRODUCTION

Motivated by the question of optimal tie-line scheduling in electric power systems, we study a distributed algorithm to solve an optimization problem of the form

\[ P: \begin{align*}
    &\text{minimize} & & \sum_{i=1}^{n} f_i(x_i, y_i), \\
    &\text{subject to} & & (x_i, y_i) \in S_i, \tag{1a} \\
    & & & \sum_{i=1}^{n} A_i y_i \leq b, \tag{1b}
\end{align*} \]

over the variables \( x_i \in \mathbb{R}^{N_i^x}, y_i \in \mathbb{R}^{N_i^y} \) for each \( i = 1, \ldots, n \), where \( \mathbb{R} \) denotes the set of real numbers. Each node \( i \) is only privy to \( f_i, A_i, b, \) and \( S_i \), where

\[ f_i: \mathbb{R}^{N_i^x+N_i^y} \rightarrow \mathbb{R}, \quad A_i: \mathbb{R}^{N_i^x\times N_i^y} \rightarrow \mathbb{R}, \quad b \in \mathbb{R}^N, \quad S_i \subset \mathbb{R}^{N_i^x+N_i^y}. \]

Here, each node is an agent with computational capabilities and can only communicate with other nodes that are connected to it in an undirected graph \( \theta \). We make the following assumptions.

Assumption 1. (i) \( \theta \) is connected. (ii) \( f_i \) is jointly convex in its arguments. (iii) \( S_i \) is compact and convex. (iv) Slater’s condition holds for \( P \). (v) Agent \( i \) can optimize \( f_i \) over \( S_i \).

A Lagrangian method [3] has been widely used to obtain a decentralized framework to solve problem \( P \). However, this method requires a central coordinator to update and distribute the Lagrangian multiplier to the agents. In this paper, we present an alternate approach that does not require any central coordination. Hence, it prescribes a truly distributed framework to solve \( P \). In particular, we make use of and generalize the recently developed distributed Lagrangian method in [4, 13] in Section 2, and summarize its convergence properties in Section 3. It has come to our attention that our approach is similar to that in [5]. An in-depth comparison is relegated to future work.

Problem \( P \) can model the tie-line scheduling problem in electric power systems. To motivate the application, notice that different system operators (SOs) control parts of an interconnected power network and their associated grid assets over a geographical footprint, we call an area. Tie-lines are precisely the transmission lines that interconnect different areas. One would ideally solve a joint optimal power flow problem to compute the minimum cost dispatch across the entire power network. Such a solution requires one to aggregate dispatch costs -- often collected through bids and offers from a wholesale market organized by the SO -- as well as the network parameters within each area. For technical, historical, and legal barriers, such an aggregation is untenable. Distributed approaches are more suited to the task. Moreover, tie-lines are generally underutilized today. As is noted in [12], optimal utilization (e.g., using our algorithm) can lead to substantial monetary savings.

We will identify variables \( x_i \) as the dispatch within each area that only concerns the SO of area \( i \) (denoted henceforth as \( SO_i \)). The power flows over the tie-lines connecting areas \( i \) and \( j \) will be modeled as linear maps of \( y_i \) and \( y_j \), where the constraint in (1b) will enforce power flow constraints over the tie-lines. Finally, \( f_j \) will model the dispatch cost and \( S_j \) will model the network and generation constraints within area \( i \).

Tie-line schedules are typically fixed with a lead time prior to power delivery. The exact demand and supply conditions are unknown during the scheduling process. In Section 4, we first ignore such uncertainty and reduce the deterministic tie-line scheduling problem to an instance of \( P \), and numerically demonstrate our algorithm on a two area power system. Then, we model the uncertainty in demand and supply and formulate the question of robust tie-line scheduling. The robust counterpart with the so-called affine decision rule approximation, again reduces to an instance of \( P \). The robust problem is motivated by the need to optimize tie-line flows against the uncertainty from variable renewable resources such as wind and solar energy.

1.1 Related literature

Two different threads of the literature inform our work: the theory on distributed algorithms, and the application area of tie-line scheduling in power systems. A distributed approach to solve a joint optimization problem similar to that in \( P \) via the so-called Lagrangian method can now be found in standard texts, e.g., in [3]. They construct a Lagrangian function for the joint optimization problem, and sequentially update the primal and the dual variables. And, they rely on a central coordinator to communicate the current dual variables, e.g., in [1].

The application of distributed optimization to the tie-line scheduling problem in power systems goes back to the foundational work of [8]. In that paper and those that followed, e.g., [9] and the references therein, the authors have adopted the popular Lagrangian
We propose Algorithm 1 to solve the tie-line scheduling problem is the primal decomposition approach. Rather than utilizing a Lagrangian, they directly update two sets of primal variables in a sequential fashion, those that are native to the dispatch within each area and the remainder that define the power flows over the tie-lines. Again, a coordinator mediates between the SOs to update the latter set of variables; see for example [6, 10].

## 2 THE DISTRIBUTED LAGRANGIAN ALGORITHM

We propose Algorithm 1 to solve P. This algorithm combines a consensus step with a projected distributed sub-gradient algorithm applied to the Lagrangian dual problem of P. To describe the dual problem, we need additional notation. Define $P^y (S_i)$ as the projection of $S_i$ on the $y_i$ coordinates. Then, consider the parametric optimal cost function

$$F_i (y_i) := \min_{x_i} f_i (x_i, y_i),$$

for $y_i \in P^y (S_i)$, and its Fenchel conjugate

$$F_i^* (u) := \max_{y_i \in P^y (S_i)} \left\{ u^T y_i - F_i (y_i) \right\}. $$

The dual problem of $P$ is then equivalent to solving

$$DP : \min_{\lambda \in \mathbb{R}^N} \sum_{i=1}^n F_i^* (-\lambda^T A_i) + \lambda^T b_i. \tag{2}$$

Here, $\lambda \in \mathbb{R}^N$ is the Lagrange multiplier associated with the coupling constraint (1b). In the sequel, assume $W \in \mathbb{R}^{n \times n}$ is a doubly stochastic matrix\(^1\) with strictly positive diagonals that conforms to the sparsity pattern of $W$, i.e., the positive entries of $W$ define the connectivity of $W$. Recall that each agent knows the vector $b$ in (1b). Define $b_i := b/n$ for each $i = 1, \ldots, n$. Finally, let $\epsilon$ be a small positive number, and $\{ \epsilon (k) \}_{k=0}^{\infty}$ be a nonnegative sequence of step-sizes that parameterizes our algorithm.

**Algorithm 1** Distributed Lagrangian Method for solving P.

1. Initialize:

   $k \leftarrow 1.$

   Agents $i = 1, \ldots, n$ initialize $\lambda_i (1) \in \mathbb{R}^N.$

2. do

   3. For each $i = 1, \ldots, n$, agent $i$ executes:

      4. Communicate with neighbors to compute $v_i (k) \leftarrow \sum_{j=1}^n W_{ij} \lambda_j (k).$

      5. $(x_i (k+1), y_i (k+1)) \leftarrow \text{argmin}_{(x_i, y_i) \in S_i} \left\{ f_i (x_i, y_i) + v_i^T (k) (A_i y_i - b_i) \right\}.$

      6. $\lambda_i (k+1) \leftarrow [v_i (k) + \alpha (k) (A_i y_i (k+1) - b_i)]^+$.

      7. $k \leftarrow k + 1.$

6. while $\| \lambda_i (k) - v_i (k) \|_2 > \epsilon.$

\(^{1}\)A matrix with nonnegative entries is said to be doubly stochastic if and only if all its rows and columns sum to one.

The algorithm makes use of the following notation. If $a$ is any vector, $a^T$ denotes its transpose, $a^\top$ denotes its projection on the nonnegative orthant, and $\| a \|_2$ denotes its Euclidean norm.

**Remark.** The classical Lagrangian approach requires a central coordinator to maintain and update a global dual multiplier $\lambda$. To make the classical approach truly distributed, one can maintain local copies of $\lambda$’s and leverage an inner consensus loop to compute the global $\lambda$. We improve upon such a method by concurrently updating the local copies of the multipliers and the primal variables in steps 5 and 6 in Algorithm 1.

## 3 CONVERGENCE PROPERTIES

We denote the local Lagrangian-type function $L_i$ for each node $i$ as

$$L_i (x_i, y_i, \lambda_i) = f_i (x_i, y_i) + \lambda_i^T (A_i y_i - b_i).$$

The first result finds conditions on the step sizes $\{ \alpha (k) \}$ that makes the local copies of the dual variables with each agent converge to an optimizer of the dual problem.

**Theorem 3.1.** Suppose Assumption 1 holds. If the nonincreasing nonnegative sequence $\{ \alpha (k) \}$ satisfies

$$\sum_{k=1}^{\infty} \alpha (k) = \infty, \text{ and } \sum_{k=1}^{\infty} \alpha^2 (k) < \infty, \tag{3}$$

then $\{ x_i (k), y_i (k), \lambda_i (k) \}$ from Algorithm 1 satisfies

(a) $\lim_{k \to \infty} \lambda_i (k) = \lambda^*$ is an optimizer of $DP$ for each $i = 1, \ldots, n.$

(b) $\lim_{k \to \infty} \sum_{i=1}^n L_i (x_i (k), y_i (k), \lambda_i (k))$ is the optimal value of $P$.

One can choose $\alpha (k) = 1/k$ to satisfy (3). To present our next result, let $q (\lambda) = \sum_{i=1}^n q_i (\lambda)$, and $q^*$ denote the value of $q$ at an optimal solution of $DP$. The next result shows that $q$ evaluated at a time-averaged $\alpha$-weighted local copies of the dual variables (with each agent) converges to the optimal value $q^*$ for a particular choice of step-sizes. And, the difference of $q$ at this ‘average’ $\lambda$ from $q^*$ scales as $O \left( \ln (k) / \sqrt{k} \right)$.

**Theorem 3.2.** Suppose Assumption 1 holds. If $\alpha (k) = 1/\sqrt{k}$ for $k \geq 1$, then $\{ \lambda_i (k) \}$ from Algorithm 1 satisfies

$$q (z_i (k)) - q^* \leq \frac{1}{\sqrt{k}} \left( \psi_1 n + \frac{\psi_2 + \psi_3 \ln (k)}{1 - \sigma_2 (W)} \right), \tag{4}$$

for positive constants $\psi_1, \psi_2, \psi_3$, where $\sigma_2 (W)$ denotes the second largest singular value of $W$, and $z_i (k) := \sum_{\ell=1}^k \alpha (\ell) \lambda_i (\ell) / \sum_{\ell=1}^k \alpha (\ell)$ for each $i = 1, \ldots, n$.

## 4 TIE-LINE SCHEDULING IN MULTI-AREA POWER SYSTEMS

In this section, we apply the distributed Lagrangian method to the tie-line scheduling problem. We only present the case for $n = 2$ areas. Generalization to $n > 2$ areas is straightforward.

The deterministic problem with known net demands. Consider two areas, labelled 1 and 2. For each $i = 1, 2$, let the power network in area $i$ be comprised of $N_i$ internal buses, and $N_b$ boundary buses. The internal buses (nodes) are the ones that do not have tie-lines connected to them; the boundary buses (nodes) are the ones that do. Assume that each internal bus has a dispatchable generation resource that supplies $g_i \in \mathbb{R}^{N_i}$. Each internal bus also has a net
The adjustable robust variant with uncertain net demand. We next present the case where tie-line flows are optimized to minimize the maximum aggregate costs across the areas. The maximum cost is computed by varying the net demand \( \xi_1 \) over the polytope

\[
\Sigma_1 := \{ \xi_1 \in \mathbb{R}^{N_1} : \mathcal{D}_1 \xi_1 \leq d_1 \}, \quad i = 1, 2.
\]

Recall that \( y_i \in \mathbb{R}^{N_1+1} \) equals the vector of voltage phase angles at the boundary buses. To fix the tie-line flow schedule, we therefore assume that \( y_i \)'s are fixed prior to the time of power delivery, and allow \( x'_i \)'s to be square-integrable maps of \( y_i \) and \( \xi_1 \). Searching over square-integrable maps can be challenging. Hence, we restrict the search over affine decision rules of the form \( x_i(y_i, \xi_1) = x^y_i + X^\xi_1 \xi_1 \), and optimize over \( x^y_i \) and \( X^\xi_1 \). With this approximation, the robust tie-line scheduling problem becomes

\[
\begin{aligned}
\text{minimize} & \quad x^y_i, X^\xi_1, y_i \\
\text{subject to} & \quad S^y_i x^y_i + S^\xi_1 y_i + S^\xi_1 \xi_1 \leq t_i,
\end{aligned}
\]

for all \( \xi_1 \in \Sigma_1 \), \( i = 1, 2 \), \( y_1 - y_2 = 0 \), \( y_1, y_2 \in \mathcal{Y} \). Robust enforcement of the constraints in (7a) and (7b) render the problem semi-infinite. Duality theory of linear programming can be leveraged to reduce (7) into a standard linear program that is an instance of \( P \), e.g., see [2]. We omit the details for brevity.

## References


A First Look at Power Attacks in Multi-Tenant Data Centers

[Extended Abstract]

Mohammad A. Islam  Shaolei Ren  Adam Wierman
University of California, Riverside  University of California, Riverside  California Institute of Technology

ABSTRACT

Oversubscription increases the utilization of expensive power infrastructure in multi-tenant data centers, but it can create dangerous emergencies and outages if the designed power capacity is exceeded. Despite the safeguards in place today to prevent power outages, this extended abstract demonstrates that multi-tenant data centers are vulnerable to well-timed power attacks launched by a malicious tenant (i.e., attacker). Further, we show that there is a physical side channel — a thermal side channel due to hot air recirculation — that contains information about the benign tenants’ runtime power usage. We develop a state-augmented Kalman filter that guides an attacker to precisely time its power attacks at moments that coincide with the benign tenants’ high power demand, thus overloading the designed power capacity. Our experimental results show that an attacker can capture 53% of all attack opportunities, significantly compromising the data center availability.

1. INTRODUCTION

Multi-tenant data centers are shared data center facilities where tenants house their physical servers while the data center operator is responsible for managing the non-IT support systems such as power distribution and cooling. They are a very important segment of data centers and adopted widely by many industry sectors. For example, Apple has 25% of its servers in multi-tenant data centers. As of today, there are nearly 2,000 multi-tenant data centers in the U.S., consuming five times the energy of Google-type data centers altogether.

To accommodate the fast growing Internet and cloud-based services, multi-tenant data center operators are under a constant pressure to expand and/or build new facilities. However, data center infrastructure is very expensive to build because of its high availability requirement, costing around US$10 ~ 25 per watt of IT critical power delivered (cooling power is separate from IT power) and taking up more than 1.5 times of the total electricity cost over its lifespan. In addition, long-time-to-market and local power grid constraints also pose significant hurdles for increasing capacity in multi-tenant data centers.

Consequently, to maximize the utilization of expensive data center infrastructures, multi-tenant data center operators commonly oversubscribe the power infrastructure by selling capacity to more tenants than can be supported. Even owner-operated data centers, such as Facebook, oversubscribe power infrastructures to defer/reduce the need of building new capacities. The industry standard oversubscription ratio is 120%, and recent studies have begun to suggest even more aggressive oversubscription [3].

Power oversubscription increases capacity utilization and significantly reduces capital expenses. But, it comes with a dangerous consequence of overloading the designed capacity (a.k.a., power emergencies) when the power demand of multiple tenants peaks simultaneously. In fact, even short-term overloads over a few minutes can lead to tripped circuit breakers and costly data center outages (e.g., Delta Airline’s data center power outage resulted in a US$150 million loss [1]).

Power infrastructure redundancy is very common to ensure a high availability in today’s data centers. While it can safeguard the data center against power emergencies by taking over some overloads, redundancy protection is lost during an emergency. For example, if with an emergency, a fully-redundant Tier-IV data center can experience an outage when either the primary or redundant infrastructure fails; otherwise, an outage occurs only when both the primary and redundant infrastructures fail. The loss of infrastructure redundancy can significantly increase the outage risk of a fully redundant Tier-IV data center by 280+ times [2]. Even though emergencies only occur for 5% of the time, the availability of a Tier-IV data center can be downgraded to that of a Tier-II data center, which means a nearly 50% capital loss for the data center operator.

The severe consequences of power emergencies have led to the development of various power capping techniques such as CPU throttling. Nonetheless, the lack of control over tenants’ servers renders such techniques inapplicable in multi-tenant data centers. Concretely, a power emergency may still occur, even though all the tenants are using power within their purchased capacities due to the operator’s oversubscription decision. Multi-tenant data center operators, therefore, have taken other precautions by imposing restrictions on each tenant’s “normal” power usage to be below a certain percentage (e.g., 80%) of its subscribed capacity, which limits frequent and/or constant usage of a tenant’s full capacity. This contractual constraint effectively reduces the probability of simultaneous peaks of tenants’ power usage, keeping the risk of power emergencies at a very low level. Hence, despite oversubscription, power infrastructure in multi-tenant data centers is considered safe.

Contributions. This extended abstract summarizes our recent work [4]. Our goal is to highlight that, despite the safeguards in place today, multi-tenant data centers are vulnerable to well-timed malicious power attacks that can cause a huge financial loss for the data center operator as well as affected tenants. More specifically, a malicious tenant (i.e., attacker), which can be the competitor of the target data center and does not run any useful workloads, can

Copyright is held by author/owner(s).
We demonstrate that, by leveraging the knowledge of the layout of the target data center (through a maintenance visit of its own servers) and computational fluid dynamics (CFD), the attacker can obtain a rough idea of the heat recirculation process and use a state-augmented Kalman filter to extract the hidden information about benign tenants’ power usage contained in the thermal side channel. Although the attacker’s CFD analysis only provides limited and imprecise knowledge of the actual heat recirculation process, our experiments show that the thermal side channel can assist the attacker with successfully capturing 54% of all the attack opportunities with a precision rate of 53%, significantly threatening the data center availability.

It is also important to note that there might also exist other side channels. For example, a high response time of benign tenants’ services may indicate a high server utilization and power usage, but response time is also affected by multiple factors irrelevant of power and many tenants do not even have any user-facing services for the attacker to exploit. Further, a data center has a complex internal wiring topology (e.g., “wrapped” for N+1 redundancy) that is unknown to the attacker, and hence inferring benign tenants’ power usage from the shared data center power distribution system can be challenging. In any case, we make the first effort to exploit a side channel — thermal side channel, which can complement other side channels (if any) and assist the attacker in timing its attack more accurately.

In conclusion, the key novelty of our work is that it is the first study to consider an adversarial setting in multi-tenant data centers— well-timed power attacks by exploiting a thermal side channel. There are a small but quickly expanding set of papers [5, 7] that attempt to create malicious virtual machines to overload the power capacity in an owner-operated data center. In sharp contrast, our work exploits a unique co-residency thermal side channel to launch well-timed power attacks in a multi-tenant data center where an attacker controls physical servers and can easily inject a high power load to create severe power emergencies.

2. EXPLOITING A THERMAL SIDE CHANNEL

In this section, we exploit a thermal side channel to guide the attacker to time its attacks against the shared power infrastructure, significantly compromising the data center availability.

2.1 Threat Model

We consider that an attacker shares the power infrastructure capacity $C$ with other benign tenants. The attacker can increase its power to its maximum subscription capacity by running CPU-intensive workloads. We consider an attack successful if $p_a + p_b > C$ for at least $L$ minutes, where $p_a$ and $p_b$ are the attacker’s and benign tenants’ power. In our evaluation, we use $L = 5$ minutes, which is sufficient to trip a circuit breaker. The attacker can be a competitor of the target multi-tenant data center and wants to cause a million-dollar loss due to compromised data center availability by spending only a small fraction in its capacity subscription and server costs. Note that we do not consider hiding advanced weapons or physically tampering with the power infrastructures for attacks.

2.2 A Thermal Side Channel

We observe that the physical co-location of the attacker and benign tenants in shared data center spaces results in a prominent thermal side channel which carries benign tenants’ power usage information. Concretely, almost all the power consumed by a server is converted into heat, and due to the lack of complete heat containment in many data centers (see Fig. 3), some of the hot air can travel to other servers (a.k.a. heat recirculation) and increases their inlet temperatures. Thus, an attacker can easily conceal temperature sensors in its servers to monitor the inlet temperatures, extracting information of benign tenants’ runtime power usage.

A naive strategy for the attacker is to look at server inlet temperature and launch power attacks whenever the inlet temperature is sufficiently high. But, even with the same power consumption, a server closer to the attacker can cause a greater temperature increase at the attacker’s server inlet than a server that is farther away. Hence, a high temperature reading at the attacker’s inlet does not necessarily indicate a high aggregate power consumption of benign tenants.

**Figure 1: Illustration of opportunity and power attack.**

intentionally increase power to its peak/subscribed capacity in an attempt to create power emergencies, when it detects an already high utilization of the shared capacity. Importantly, the total cost incurred by the attacker, such as server and power capacity costs, is only small fraction (between 1.44% and 15.88%, as shown by [4]) of the total financial loss borne by the operator and benign tenants.

To illustrate this point, we show in Fig. 1 a 24-hour power trace by four representative tenants. These tenants run web and data analysis workloads. The total capacity is 200kW and sold as 240kW following industry standard oversubscription of 120%. The attacker subscribes to 30kW and increases its power to full capacity for 10 minutes whenever the aggregate power approaches the capacity limit. Consequently, we see multiple power emergencies, while the attacker only occasionally peaks its power and meets its contractual constraint.

While power attacks are dangerous, attack opportunities only exist intermittently due to the fluctuation of benign tenants’ aggregate power usage, as illustrated in Fig. 1. Thus, a key question is: how does the attacker detect an attack opportunity?

Keeping a constant high power will capture all attack opportunities, but it is not allowed by operator and can lead to the attacker’s eviction. Because of intermittency of attack opportunities, randomly attacking is not likely to be successful either. Further, even though some coarse windows of attack opportunities (e.g., possibly peak traffic hours) can be identified and help the attacker locate the attack opportunities within a smaller time frame, the actual attack opportunity is short-duration and may not last throughout the entire coarse window. Thus, the attacker needs to precisely time its attacks to coincide with other benign tenants’ high power usage. Nonetheless, this may seem impossible, as the attacker does not have access to the operator’s power meters to monitor the benign tenants’ power usage at runtime.

We observe that the physical co-location of the attacker and benign tenants in shared data center spaces results in a prominent thermal side channel which carries benign tenants’ power usage information. Concretely, almost all the power consumed by a server is converted into heat, and due to the lack of complete heat containment in many data centers (see Fig. 3), some of the hot air can travel to other servers (a.k.a. heat recirculation) and increases their inlet temperatures. Thus, an attacker can easily conceal temperature sensors in its servers to monitor the inlet temperatures, extracting information of benign tenants’ runtime power usage.

A naive strategy for the attacker is to look at server inlet temperature and launch power attacks whenever the inlet temperature is sufficiently high. But, even with the same power consumption, a server closer to the attacker can cause a greater temperature increase at the attacker’s server inlet than a server that is farther away. Hence, a high temperature reading at the attacker’s inlet does not necessarily indicate a high aggregate power consumption of benign tenants.
2.2 Heat Recirculation at Rack Level

Although a detailed server-level heat recirculation model is nearly impossible, we can create a zone-based linear heat-recirculation model based on the widely-used CFD analysis. By considering all servers in a zone, we can simulate heat recirculation. In fact, according to a survey done by Uptime Institute on 1000+ data centers [6] and shown in Fig. 3, almost 80% of the data centers have at least 25% of their racks without any heat containment and 20% of the data centers have no heat containment at all. Without heat containment, some hot air can travel a few meters to other servers and increases their inlet temperatures, constituting a thermal side channel that conveys some (noisy) information of benign tenants’ power usage to the attacker.

2.3 Estimating Benign Tenants’ Power

Because of the non-uniform impact of different server racks, a high server inlet temperature does not mean that the benign tenants’ power usage is also high. Thus, a model for the heat recirculation process is crucial to extract the benign tenants’ power usage.

As the attacker does not know all the details of the data center, having a detailed server-level heat recirculation model is nearly impossible. Thus, we create a zone-based linear heat-recirculation model for the attacker based on the widely-used CFD analysis, by considering all servers in a zone have a uniform impact on the attacker’s sensors. This zonal consideration significantly reduces the complexity of modeling heat recirculation, but naturally comes at the cost of inaccuracy. Nonetheless, the zone-based model suffices to detect attack opportunities.

Based on a zone-level model, we develop a Kalman filter to estimate benign tenants’ runtime power usage, which is hidden in the thermal side channel. Although the attacker’s zone-level model only provides a limited view of heat recirculation and can deviate from the actual process, our experiments show that the attacker can still estimate the benign tenants’ aggregate power with a high accuracy (e.g., only 3% error on average). This is partly because the attacker only needs to track the benign tenants’ power variations and know the aggregate value.

2.4 Attack Strategy

To launch its attacks, the attacker sets a triggering threshold on its estimate of aggregate power. The attacker waits for $T_{\text{wait}}$ time to see if the estimate remains high to avoid attacking during a transient power spike. When the estimate surpasses the triggering threshold for $T_{\text{wait}}$, the attacker starts attacks and keeps its power high for a predetermined time of $T_{\text{attack}}$ minutes. In addition, to comply with the contract, the attacker does not re-attack by cooling down for at least $T_{\text{hold}}$ minutes, even though it may detect a consecutive attack opportunity.

2.5 Experimental Evaluation

We conduct a CFD analysis to simulate heat recirculation processes in a multi-tenant data center and show the summary of power attacks in Fig. 4(a). We set $T_{\text{attack}} = 10$, $T_{\text{wait}} = 1$ and $T_{\text{hold}} = 10$ minutes in our evaluation. With a lower triggering threshold, the attacker will attack more frequently, detecting more attack opportunities and meanwhile launching more unsuccessful attacks. Thus, as shown in Fig. 4(b), this results in a higher true positive rate (percentage of attack opportunities captured), but a lower precision rate (percentage of successful attacks among all the launched attacks). To keep power attacks under 10% of the total time, the attacker can set its triggering threshold at 101%, resulting in a true positive rate of 54% and a precision rate of 53%.

We only include our key findings in this extended abstract, while details on CFD, Kalman filter, and other results are available in [4].

Acknowledgement

This work was supported in part by the U.S. NSF under grants CNS-1551661, CNS-1565474, and ECCS-1610471.

3. REFERENCES

Load-side Frequency Regulation with Limited Control Coverage

Extended Abstract

John Z. F. Pang, Linqi Guo and Steven, H. Low
California Institute of Technology, Pasadena, CA91106, USA
{jzpang,guo,low}@caltech.edu

ABSTRACT
Increasing renewable energy increases uncertainty in energy systems. As a consequence, generator-side control for frequency regulation, impacted by the slow reaction of generators to meet urgent needs, may no longer suffice. With increasing integration of smart appliances which are able to sense, communicate and control, load-side control can help alleviate the aforementioned problem as it reacts fast and helps to localize disturbances. However, almost all existing methods for optimal load-side control require full information control coverage in the system. Framing the problem as an optimization problem and applying saddle-point dynamics, we obtain a control law that rebalances power and asymptotically stabilizes frequency after a disturbance. We generalize previous work to design a controller which only requires partial control coverage over all nodes, yet still achieves secondary frequency control. We verify these results via simulation.

1 INTRODUCTION
Frequency regulation aims to keep frequency of a power system close to its nominal value. Frequency deviates with demand deviations, and propagates through the network, leading to potential blackouts. Previously, generator-side control suffices for frequency regulation, and is typically classified into three stages: (i) Primary - stabilization, (ii) Secondary - fast, expensive recovery and (iii) Tertiary - slow, cheaper recovery. As renewables become increasingly utilized, demand fluctuates more due to the lack of predictability in yield of renewables. This problem will continue to grow as states aim for 50% renewables by 2030 [9], and also since alternatives, e.g. battery storage programs, are cost-ineffective, with break-even costs estimated to be $197/kW [6].

Controllable loads, e.g. smart household appliances, have the ability to decrease energy imbalance, with simulations and real-world demonstrations highlighting their potential [4]. However, in a large scale grid, not every node have controllable loads capable of supporting frequency regulation, and therefore it is important to consider situations where the load control coverage is not full.

1.1 Literature Review
The use of primal-dual dynamics for control started out with Kelly [5] and Low [7] in network resources and [3] for stability of primal-dual dynamics. It has been recently used in the power systems community for frequency control by [13] with prior work [14] identifying power system swing equations as a primal-dual dynamic, setting the foundation for a series of work in this area (e.g. [8, 12]). There is a larger body of work on distributed frequency control (see e.g. [11] and references therein for recent examples) that does not employ the primal-dual framework.

1.2 Summary
In this work, we focus on load-side participation for secondary frequency control. In particular, assuming that the power or frequency imbalance can be reported through a distress signal in real-time, we show that a limited coverage of nodes participating in load control can cooperate to yield secondary frequency regulation. Simulation is performed on a non-linear system, demonstrating the effectiveness of our control. Our contributions can be summarized as follows:

(1) Primary Regulation under limited control coverage.
(2) Secondary Regulation under perfect disruption prediction.
(3) Bounding the gap between nominal frequencies and stabilized frequencies under errors in such predictions.

2 NETWORK MODEL & PRELIMINARIES
We adopt the model of Zhao et al. [13]. The power network is described by a graph $\mathcal{G} = (N, E)$ where $N = \{1, \ldots, |N|\}$ is the set of buses and $E \subseteq N \times N$ is the set of transmission lines $(i, j)$, such that if $(i, j) \in E$, then $(j, i) \notin E$. We also consider a communication graph $\mathcal{G}_c = (N, E_c)$ induced by the set of nodes $A \subseteq N$. We partition the buses as $N = G \cup L$ and use $G$ and $L$ to denote the set of generator and load buses respectively. Generator nodes $i \in G$ generates electrical power and may also have loads. Load nodes $i \in L$ have only loads. The subset $A \subseteq N$ corresponds to active loads, i.e. nodes with load control, and requires communication with their neighbors. We identify the set of nodes with frequency sensitive load with $F$.

As in [13], we have that the power system dynamics as follows:

\[
M_i \omega_i = P_i^n - (1_{i \in A} d_i + 1_{i \in F} \dot{d}_i) - \sum_{e \in E} C_e P_e, \quad \forall i \in G \quad (1a)
\]

\[
0 = P_i^n - (1_{i \in A} d_i + 1_{i \in E} \dot{d}_i) - \sum_{e \in E} C_e P_e, \quad \forall i \in L \quad (1b)
\]

\[
\dot{P}_{ij} = B_{ij}(\omega_i - \omega_j), \quad \forall (i, j) \in E \quad (1c)
\]

where $d_i$ denotes an aggregate controllable load and $\dot{d} := D_i \omega$ denotes an aggregate uncontrollable but frequency-sensitive load and $M_i$ is the generator’s inertia. $P_i^n$ is the mechanical power generated or used and $P_{ij}$ is the real power flow from $i$ to $j$. $C_e$ are the elements of the incidence matrix $C \in \mathbb{R}^{|N| \times |E|}$ of the graph $\mathcal{G}$ defined as $C_{ei} = -1$ if $e = (i, j) \in E$, $C_{ei} = 1$ if $e = (j, i) \notin E$ and $C_{ei} = 0$ otherwise. The incidence matrix $C^T$ for the communication graph is similarly defined. We refer the reader to Zhao et al. [13] for a detailed motivation of the model. We are interested in the situation where the system is originally at an equilibrium, i.e. when $\dot{\omega} = \dot{P}_{ij} = 0$, and then the system is perturbed locally.
3 STABILIZATION UNDER LIMITED CONTROL COVERAGE

In this section, we design a distributed control mechanism that rebalances the system while driving the frequency back to its nominal value, even when not all nodes participate in load control.

We define an optimal load control (OLC) problem as follows:

\[
\begin{align*}
\min_{d, \hat{P}, P, R} & \quad \sum_{i \in A} c_i(d_i) + \frac{1}{2} \sum_{(i,j) \in E} d_i^2 \\
\text{s.t.} & \quad P_{im}^n - (1_{i \in A} d_i + 1_{(i,j) \in E} d_j) = \sum_{e \in E} C_{ie} P_e, \quad \forall i \in N \\
& \quad \hat{P}_i^m + \hat{P}_i - d_i = \sum_{e \in E} C_{ie} R_e, \quad \forall i \in A
\end{align*}
\]

(2a)

(2b)

Unlike [8] and [13], we do not assume that the load control coverage is full, nor do we assume that every node is communicable and admit frequency sensitive loads. \( \hat{P}_i \) is a constant which serves as a prediction to the disruption in the power flow. \( R_e \) is a virtual line flow. We make the following assumptions on the OLC, i.e., (i) cost functions \( c_i \) are strictly convex and twice continuously differentiable; and (ii) OLC is feasible, implying Slater’s condition since constraints in OLC are linear [1].

Let \( \nu_i \) be the Lagrange multiplier associated with (2a) and \( \lambda_i \) for (2b). The dual of this problem can then be written as:

\[
\Phi(v, \lambda, d, \hat{P}, P, R) = \max_{\lambda, \nu} \min_{d, \hat{P}, P, R} \mathcal{L}(d, \hat{P}, P, R, v, \lambda)
\]

(3)

Differentiating with respect to \( d_i \) and \( \hat{d}_i \), the minimizer satisfies:

\[
c_i'(d_i) = \nu_i + \lambda_i, \quad \forall i \in A \text{ and } \hat{d}_i = D_i v_i, \quad \forall i \in F
\]

which suggests that controllable loads should be set via:

\[
d_i(v_i, \lambda_i) = (c_i')^{-1}(\nu_i + \lambda_i)
\]

(5)

The maximum in (3) can only be attained if \( \nu_i = \nu_j \) for all \( i, j \in N \) and \( \lambda_i = \lambda_j \) for all \( i, j \in C_i^e \) where \( C_i^e \) is a connected component of the virtual network. Substituting these back into (3) implies that the dual of OLC can be equivalently written as the maximization of:

\[
\max_{\nu, \lambda} \min_{d, \hat{P}, P, R} \mathcal{L}(d, \hat{P}, P, R, v, \lambda) = \max_{\nu, \lambda} \min_{v, P, R} \mathcal{L}(d, \hat{P}, P, R, v, \lambda)
\]

(6)

***Lemma 3.1.*** The function in (6) is separable, i.e., \( \Phi(v, \lambda) = \sum_{i \in N} \Phi_i(v_i, \lambda_i) \) where:

\[
\Phi_i(v_i, \lambda_i) = v_i \left( P_{im}^n - 1_{i \in A} d_i - 1_{(i,j) \in E} v_j - \sum_{e \in E} C_{ie} P_e \right), \quad \forall i \in G
\]

\[
0 = P_{im}^n - 1_{i \in A} d_i - 1_{(i,j) \in E} v_j - \sum_{e \in E} C_{ie} P_e, \quad \forall i \in L
\]

\[
\lambda_i = \eta_i \left( P_{im}^n + \hat{P}_i - d_i - \sum_{e \in E} C_{ie} R_e \right), \quad \forall i \in A
\]

\[
\hat{P}_i = \hat{P}_i(v_i - v_j), \quad \forall i \in A
\]

(7a)

(7b)

(7c)

(7d)

where \( M_i^{-1} \) and \( B_{ij} \) with \( \omega_{ij} \), with parameters \( \eta_i \) and \( \alpha_i \), determining how much weight we place on information from the virtual variables.

Consider the Lagrangian of the dual of the OLC problem:

\[
\mathcal{L}_D(v_G, \nu_L, \lambda, \pi) = \sum_{i \in N} \Phi_i(v_i, \lambda_i) - \sum_{(i,j) \in E} \pi_{ij}^N(v_i - v_j) - \sum_{(i,j) \in E} \pi_{ij}^L(\lambda_i - \lambda_j)
\]
To amend for the partiality of the primal-dual dynamics we used, consider the partial Lagrangian:

$$\mathcal{L}_D(P,R,v_G,\lambda) = \max_{v_L} \mathcal{L}_D(P,R,v_G,v_L,\lambda)$$

(8)

and by considering the same constraints as before, \(v_L\) takes on its unique maximizer value. By using the Envelope Theorem, we can then compute the partial derivatives of (8) with respect to the variables \((P,R,v_G,\lambda)\), and end up with the same equations as (7). By the following lemma\(^2\), we then know the proposed dynamics is asymptotically stable.

**Lemma 4.1.** For a \(C^1\) function \(F(x,z) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}\), if

1. \(F\) is globally convex in \(x\) and linear in \(z\),
2. for each \((x,z) \in \text{Saddle}(F)\), if \(F(x,z_\ast) = F(x,z)\), then \(F(x,z) \in \text{Saddle}(F)\),

then \(\text{Saddle}(F)\) is globally asymptotically stable under the saddle point dynamics and convergence of trajectories is to a point.

With the above results, we state the main theorem of this work:

**Theorem 4.2.** Assume the physical graph \(\mathcal{G} = (\mathcal{N},\mathcal{E})\) is connected, \(A \neq \emptyset\), \(F \neq \emptyset\), and \(L \subseteq F \cup A\). The system with control steps as defined in (7) asymptotically converges to an equilibrium point where primary frequency control is attained.

If in addition, an exact estimate \(\mathcal{P}\) of the power step change is available, secondary frequency control can be attained.

In the case of inaccurate predictions of the power step change, the performance of the dynamics can be bounded as:

$$|\mathcal{V}^\ast| < \frac{\varepsilon}{\sum_{i \in F} D_i}$$

where \(\varepsilon\) is the error in prediction.

5 NUMERICAL ILLUSTRATIONS

To demonstrate the performance of our load control to aid secondary control, we apply our control on the IEEE 39-bus test system. Unlike the analyzed linear model, the simulation adopts a nonlinear set-up, including e.g. nonlinear governor dynamics and power flows. We pick out a subset of nodes randomly for load control, and choose one node randomly to add a step increase of 1pu (based on 100MVA) of its current load. We do not limit the amount of load control the nodes can utilize, and illustrate that secondary frequency control is attained. The results are illustrated in Figure 1.

6 CONCLUSION

In this work, we propose a control design for frequency control based on a linearized model of the swing equations. By exploiting the amenability of the primal-dual dynamics, part of the dynamics can be designed to match power system dynamics. The control steps and virtual variables are updated based on the values of the physical variables. We prove that under some mild conditions, the system provably converges with an accurate prediction of power disruption, and bounds on the performances are provided with respect to inaccuracies in predictions. We implement our control steps on a non-linear simulation, demonstrating the performance of our control.

A full version of this work can be found in [10].

\(^2\)\(C^1\) functions are continuously differentiable functions and \(\text{Saddle}(F)\) are the sets of saddle points of the function \(F\).