Stochastic Control of Power Supply in Data Centers

Georgios Rovatsos, Shaofeng Zou, Alejandro D. Domínguez-García and Venugopal V. Veeravalli
Department of Electrical and Computer Engineering
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Emails: rovatso2@illinois.edu, szou3@illinois.edu, aledan@illinois.edu, vvv@illinois.edu

***Abstract*— The problem of operating the power supply system of a data center in a reliable manner is studied. Two approaches are examined. The first one assumes that there is no control over the power supply system. The second one assumes that there is a control over the maximum capacity of the system load. For the second case, the problem is fitted into a dynamic programming (DP) framework. An optimal control policy is derived, however is intractable. An approximate solution is then constructed by the Perseus algorithm. This solution is compared to two easily implementable heuristic algorithms, and with the case where no system control is allowed. Numerical results show that the approximate solution has a superior performance.**

***Index Terms*— Smart grid, dynamic load control, power distribution networks.**

I. Introduction

In modern data center systems, the assurance of a high-level of reliability in supplying power to the loads is critical. In particular, it is of high importance to be able to keep the data center online by reducing the frequency of breakdowns due to outages in the redundant power distribution lines supplying the data center. This is translated to being able to tackle topology changes regarding the transmission lines that connect the power distribution system to the data center. A disruption of the function of the data center can lead to service disruptions and even to data loss which can have catastrophic results.

In this paper, we tackle the aforementioned problem and propose a load management scheme that is adaptive to line outages. We leverage measurements of power injection and phase angles provided by distribution-level phasor measurement units (PMUs). These devices generate a noisy sequence of power measurements in real-time. The goal is to exploit these measurements in an efficient way, so that a breakdown will happen rarely. This problem is closely related to the line outage detection problem studied in [1]–[3]. There, the authors exploit PMUs in the bulk power system to design quickest change detection (QCD) algorithms that detect line outages with very small delays [4]–[6]. As will be seen, a breakdown can occur due to a topology change (e.g. line outage) and for a specific set of power injection values.

In this paper, we consider a stochastic setting, i.e., we assume that there exists uncertainty, not only in the measurement, but also in the topology of the supply system. The goal is to design algorithms that will efficiently decide the maximum load capacity of the system in real-time, through acquiring observations. We study two different settings. In the first one, we assume load management is not allowed. In the second one, we assume load shedding, which corresponds to reducing the maximum system load, possible. To quantify the performance of our proposed techniques, we fit the problem into a dynamic programming framework. We then derive the optimal control policy, which, however, is intractable. We thus propose an approximate solution, by using Perseus, a partially observable Markov decision process (POMDP) approximation algorithm [7], [8]. Furthermore, we present two heuristic algorithms. All these techniques use the sufficient statistic of the dynamic program to operate the system. The sufficient statistic can be easily calculated by using a recursion. Finally, in the numerical results section, we show the superiority of our approximate solution, compared to the heuristics, and the case where no control is allowed.

This work was supported in part by the NSF under grant ECCS 14-62311.

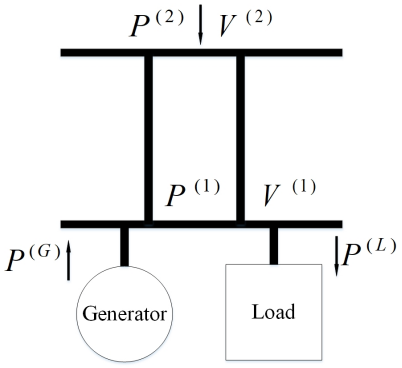

Fig. 1. A data center connected to power distribution networks using two transmission lines.

II. Power System Model

Consider the power supply system in Fig. 1, connecting the data center to a power distribution network via two parallel feeds. The power injection at each bus can be calculated as follows:

$$\begin{aligned} P^{(1)} &= \frac{V^{(1)}V^{(2)}}{X_{eq}} \sin(\theta^{(1)} - \theta^{(2)}), \\ P^{(2)} &= \frac{V^{(1)}V^{(2)}}{X_{eq}} \sin(\theta^{(2)} - \theta^{(1)}), \end{aligned} \tag{1}$$

where $P^{(i)}$ is the power injection into bus i, $V^{(i)}$ is the magnitude of the voltage at bus i, $\theta^{(i)}$ is the phase angle at

bus i, $i = 1, 2$, and X_{eq} is the equivalent reactance of the two parallel feeds (the distribution lines are assumed to be short and lossless). We then denote $\gamma \triangleq \frac{V^{(1)}V^{(2)}}{X_{eq}}$.

The value of γ can change accordingly, due to the occurrence of some event that changes the equivalent reactance of the lines (e.g., a line outage). We denote the value of γ at time instant k as γ_k. We assume that γ_k can take two values, $\gamma^{(0)}$ and $\gamma^{(1)}$, where $\gamma^{(0)}$ corresponds to the case of a single line outage, and $\gamma^{(1)}$ corresponds to the case that both lines are on. We do not consider the case when line outages happen on both lines simultaneously since the probability of such event is negligible. We assume that $\{\gamma_k\}_{k\geq 1}$ evolves according to a Markov model, as shown in Fig. 2. More specifically,

$$\mathbb{P}(\gamma_{k+1} = b \,|\gamma_k = a) = \begin{cases} \rho_0, & \text{if} \quad a = \gamma^{(0)}, b = \gamma^{(1)}; \\ 1-\rho_0, & \text{if} \quad a = \gamma^{(0)}, b = \gamma^{(0)}; \\ \rho_1, & \text{if} \quad a = \gamma^{(1)}, b = \gamma^{(0)}; \\ 1-\rho_1, & \text{if} \quad a = \gamma^{(1)}, b = \gamma^{(1)}. \end{cases}$$

Here, ρ_1 quantifies the frequency of line outages, and ρ_0 quantifies the time it takes to repair an outage. It is assumed that $\rho_1 > \rho_0$, since the time it takes to repair a line outage should be less than the time that a line can operate before an outage occurs.

We assume that $P^{(1)} = -P^{(2)} \triangleq P$, thus, we measure $P^{(1)} \triangleq P$, the value of which at time instant k is denoted by P_k. We assume that the power injection takes value in a discrete set $\mathcal{P} = \{\pm p^{(F)}, \pm p^{(F-1)}, \ldots, p^{(0)}\}$, where each value is in per-unit (p.u.) and $p^{(0)} \triangleq 0$. This is an approximation of the actual power injection values. Furthermore, we will assume that $\{P_k\}_{k\geq 1}$ evolves according to a Markov model as shown in Fig. 3. Note that the transition probabilities in this model are symmetric. This is assumed for simplicity and a solution can be easily provided for the case of arbitrary transition probabilities.

In addition to the power injection, we measure the difference between the phase angles, denoted by $\theta_k = \theta_k^{(1)} - \theta_k^{(2)}$. As a result, we can rewrite the first line in (1) as:

$$P_k = \gamma_k \sin(\theta_k). \tag{2}$$

We assume noisy power and phase angle measurements. In particular, we observe

$$\hat{P}_k = P_k + N_k^{(P)},$$
$$\hat{\theta}_k = \theta_k + N_k^{(\theta)},$$

where $N_k^{(P)}$ and $N_k^{(\theta)}$ are independent and identically distributed (i.i.d.) zero-mean Gaussian random variables, with variance σ^2.

III. Load Control Schemes

We assume that $\gamma_0 < p^{(F)} \leq \gamma_1$. If there is no line outage, the transmission line can support any power injection in $\mathcal{P}$. If there is a line outage at time k, i.e, $\gamma_k = \gamma^{(0)}$, it can lead to a breakdown if the power injection exceeds a certain level. In this case, a breakdown cost will be incurred, which is denoted by C_b. In particular, we denote the set of all such power injection values as

$$\mathcal{P}_{BR} = \{\pm p^{(F-M+1)}, \ldots, \pm p^{(F)}\},$$

where $2M$ is the number of possible power injection values that can lead to a breakdown.

To keep the data center from breakdown, we would like to design an algorithm that can detect the line outage and shed the maximum allowable load, such that the system can still operate safely, but with a lower load level.

If the load is shed, then the system operates with limited capability. In this case, the maximum allowable power injection is $p^{(F-M)}$, and only a power injection value in

$$\mathcal{P}_{SH} = \mathcal{P} \backslash \mathcal{P}_{BR} = \{\pm p^{(F-M)}, \ldots, p^{(0)}\}$$

is allowed. Thus, even with a line outage, the system is still safe, and will not break down.

More involved load-shedding mechanisms with multiple maximum load levels can be developed. In this work, we focus on the simple case with only two levels to convey the central idea.

We further penalize the action of shedding load, due to the concern that the system may choose to shed the load all the time, and then the system will not break down but is underperforming. Thus, we introduce a per-stage cost, C_d, that is paid every time instant if the system is in shed-load mode.

We denote the action taken at time k by u_k, where $u_k \in \mathcal{U} = \{0, 1\}$. Here, $u_k = 0$ denotes the action of unshedding load and $u_k = 1$ denotes the action of shedding load. Taking all the above into consideration, we have that the evolution of the underlying system follows the state-transition diagram in Fig. 4. We note that the there are also arrows from each block(state) to itself, which are omitted in the figure. As we can see from Fig. 4, there are in total five states. At time k, which state the system is in is determined by the values of γ_k and u_{k-1}. In particular, the two states on the left correspond to the case with unshed load, and the two other states on the right correspond to the case with shed load. There is also a state that the system enters upon the occurrence of a breakdown.

In this paper, γ_k can not be measured directly, and can only be inferred from noisy measurements of P_k and θ_k. Roughly speaking, we would like to design an algorithm that control the maximum allowable load depending on whether

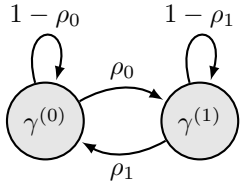

Fig. 2. Markov chain describing the evolution of γ_k.

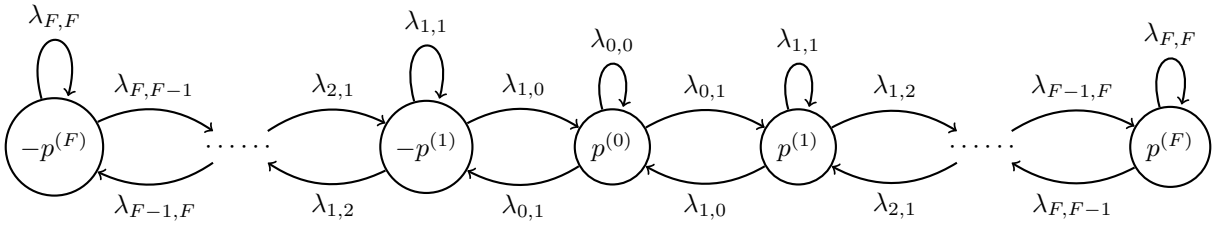

Fig. 3. Markov chain for the power injection P_k.

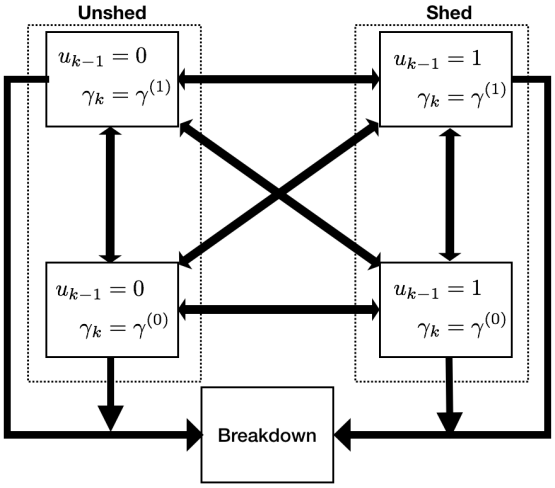

Fig. 4. System state evolution model.

there is a line outage. Specifically, if there is a line outage, load should be shed; if there is no line outage, load should be unshed (restored).

Consider a control policy $\pi = \{u_k\}_{k=0}^{\infty}$. We define the breakdown time as τ, which is random and depends on π. Then, the total average cost is defined as

$$J_\pi = \mathbb{E}\left\{\delta^\tau C_b + \sum_{k=1}^{\tau-1} \delta^k C_d \mathbb{1}_{\{u_{k-1}=1\}}\right\}, \tag{3}$$

where $0 < \delta < 1$ is the discount factor, which is used to penalize early breakdown. If $\delta = 1$, a policy that never shed load yields the smallest cost, C_b. In this case, the data center will break down after a line outage, which is not desirable. The goal is to find a control policy that minimizes the total average cost in (3).

IV. Dynamic Programming Formulation

In this section, we cast the load control problem as a dynamic programming (DP) problem. Let

$$I_k = \left[I_k^P, I_k^\theta, I_{k-1}^U\right]$$

denote the accumulated information up to time k. In particular, $I_k^P = [\hat{P}_1, \ldots, \hat{P}_k]^T$ denotes the accumulated power injection measurements, $I_k^\theta = [\hat{\theta}_1, \ldots, \hat{\theta}_k]^T$ denotes the phase angle measurements and $I_k^U = [u_1, \ldots, u_{k-1}]^T$ denotes the actions up to time $k-1$.

We first consider the finite horizon case, i.e, we restrict the stages in which decisions are made in the interval $[0, T]$. We then extend the problem to the infinite-horizon case by letting $T \to \infty$. In this paper, we assume that $M = 1$, and our arguments can be generalized to the case with arbitrary M, similarly. For the finite horizon case, the total cost is given by

$$J_\pi^T = \mathbb{E}\left\{\delta^\tau C_b \mathbb{1}_{\{\tau<T\}} + \sum_{k=1}^{\min\{T,\tau-1\}} \delta^k C_d \mathbb{1}_{\{u_{k-1}=1\}}\right\}. \tag{4}$$

Then, the cost-to-go function for the DP can be written as:

$$\begin{aligned}
&J_T^T(I_T) = 0\\
&J_k^T(I_k)\\
&= \min\Big\{C_B\Big(\mathbb{P}(\gamma_k=\gamma^{(1)}, P_k \in \{-p^{(F)}, p^{(F)}\}, u_{k-1}=0)\rho_1\lambda_{F,F}\\
&+\mathbb{P}(\gamma_k=\gamma^{(1)}, P_k \in \{-p^{(F-1)}, p^{(F-1)}\}, u_{k-1}=0|I_k)\rho_1\lambda_{F-1,F}\\
&+\mathbb{P}(\gamma_k=\gamma^{(0)}, P_k \in \{-p^{(F-1)}, p^{(F-1)}\}, u_{k-1}=0|I_k)(1-\rho_0)\lambda_{F-1,F}\\
&+\mathbb{P}(\gamma_k=\gamma^{(1)}, P_k \in \{-p^{(F-1)}, p^{(F-1)}\}, u_{k-1}=1|I_k)\rho_1\lambda_{F-1,F}\\
&+\mathbb{P}(\gamma_k=\gamma^{(0)}, P_k \in \{-p^{(F-1)}, p^{(F-1)}\}, u_{k-1}=1|I_k)(1-\rho_0)\lambda_{F-1,F}\Big)\\
&+\delta\mathbb{E}_0[J_{k+1}^T(I_{k+1})|I_k], C_d+\delta\mathbb{E}_1[J_{k+1}^T(I_{k+1})|I_k]\Big\}, \forall 0 \le k < T,
\end{aligned}$$

where $\mathbb{E}_u[\cdot]$ denotes the expectation over $(\hat{P}_{k+1}, \hat{\theta}_{k+1})$ when action u is taken. The first term in the minimization in the cost-to-go functions is the expected cost if the policy is to restore load, and the second term is the expected cost if the policy shed load.

For this dynamic programming program, it can be seen that the sufficient statistic is a vector, $\boldsymbol{b}_k$, of probabilities of the form $\mathbb{P}(\gamma_k, P_k, u_{k-1}|I_k)$, for $\gamma_k \in \{\gamma^{(0)}, \gamma^{(1)}\}$, $P_k \in \mathcal{P}$ and $u_{k-1} \in \mathcal{U}$. Therefore, $\forall 0 \le k < T$, $J_k^T(I_k)$ can be rewritten as

$$\begin{aligned}
J_k^T(I_k) \triangleq \min\Big\{&g(\boldsymbol{b}_k, 0) + \delta\mathbb{E}_0[J_{k+1}^T(I_{k+1})|I_k],\\
&g(\boldsymbol{b}_k, 1) + \delta\mathbb{E}_1[J_{k+1}^T(I_{k+1})|I_k]\Big\}.
\end{aligned} \tag{5}$$

We denote $s_k = (\gamma_k, P_k, u_{k-1})$. At each time instant, s_k takes a value from a set $\mathcal{S}$ of $|\mathcal{S}| = 8F-2$ elements. It can be shown that the sufficient statistic can be calculated by a recursive formula of the form:

$$\boldsymbol{b}_{k+1} = \phi(\boldsymbol{b}_k, u_k, \hat{P}_{k+1}, \hat{\theta}_{k+1}). \tag{6}$$

It is clear that for the s_{k+1} with its action being different from the u_k in I_{k+1},

$$\mathbb{P}(s_{k+1}|I_{k+1}) = 0. \tag{7}$$

For the s_{k+1} with its action being the same as the u_k in I_{k+1}, we have that

$$\begin{aligned}
&\mathbb{P}(s_{k+1}|I_{k+1}) = \mathbb{P}(s_{k+1}|I_k, \hat{P}_{k+1}, \hat{\theta}_{k+1}, u_k) \\
&= \frac{\mathbb{P}(s_{k+1}, \hat{P}_{k+1}, \hat{\theta}_{k+1}, u_k|I_k)}{\mathbb{P}(\hat{P}_{k+1}, \hat{\theta}_{k+1}, u_k|I_k)} \\
&= \frac{\sum\limits_{\substack{s'_k \in \mathcal{S} \\ \text{s.t. } u'_{k-1}=u_{k-1}}} \mathbb{P}(s'_k|I_k)\mathbb{P}(s_{k+1}|s'_k, I_k)\mathbb{P}(\hat{P}_{k+1}, \hat{\theta}_{k+1}|s_{k+1})}{\sum\limits_{\substack{s'_{k+1} \in \mathcal{S} \\ \text{s.t. } u'_k=u_k}} \sum\limits_{s'_k \in \mathcal{S}} \mathbb{P}(s'_k|I_k)\mathbb{P}(s'_{k+1}|s'_k, I_k)\mathbb{P}(\hat{P}_{k+1}, \hat{\theta}_{k+1}|s'_{k+1})}
\end{aligned} \tag{8}$$

where we denote $s'_k = (\gamma'_k, P'_k, u'_{k-1})$, and $s'_{k+1} = (\gamma'_{k+1}, P'_{k+1}, u'_k)$, and the last equality in (8) is due to the fact that if $u'_k \neq u_k$,

$$\mathbb{P}(s'_{k+1}, s'_k, \hat{P}_{k+1}, \hat{\theta}_{k+1}, u_k|I_k) = 0.$$

We next compute each term in (8). In particular, we first have

$$\begin{aligned}
&\mathbb{P}(s_{k+1}|s'_k, I_k) \\
&= \mathbb{P}(\gamma_{k+1}|\gamma'_k)\mathbb{P}(u_k|u'_{k-1})\mathbb{P}(P_{k+1}|P_k, u_k) \\
&= \mathbb{P}(\gamma_{k+1}|\gamma'_k)\mathbb{P}(P_{k+1}|P_k, u_k),
\end{aligned} \tag{9}$$

since $u'_{k-1} = u_{k-1}$ in the summation over s'_k. By the Bayes' rule, we obtain

$$\begin{aligned}
&\mathbb{P}(\hat{P}_{k+1}, \hat{\theta}_{k+1}|s_{k+1}) \\
&= \mathbb{P}(\hat{P}_{k+1}, \hat{\theta}_{k+1}|\gamma_{k+1}, P_{k+1}) \\
&= \mathbb{P}(\hat{P}_{k+1}|P_{k+1})\mathbb{P}(\hat{\theta}_{k+1}|P_{k+1}, \gamma_{k+1}).
\end{aligned} \tag{10}$$

Similarly, we can also compute the terms in the denominator.

Since $\boldsymbol{b}_k$ is a sufficient statistic of the DP problem, we have that $J_k^T(I_k)$ can be written as a function of $\boldsymbol{b}_k$, i.e., $J_k^T(\boldsymbol{b}_k)$. We now analyze the optimal decision rule for the infinite horizon case by letting $T \to \infty$.

Theorem 1: Let $\boldsymbol{b} = [b_1 \dots b_{|\mathcal{S}|}]$ be an element in the $|\mathcal{S}|$-dimensional probability simplex

$$\mathcal{B} \triangleq \{\boldsymbol{b} : \sum_{j=1}^{|\mathcal{S}|} b_j = 1\}.$$

The infinite-horizon cost-to-go for the DP has the following form

$$J(\boldsymbol{b}) = \min \left\{ g(\boldsymbol{b}, u) + \delta \mathbb{E}_u[J(\phi(\boldsymbol{b}, u, \hat{P}, \hat{\theta}))|b] : u = 0, 1 \right\}, \tag{11}$$

where the expectation $\mathbb{E}_u$ is over $(\hat{P}, \hat{\theta})$ for action u.

Proof: This result follows from Proposition 1 in [9, Chapter 5.1], that as $T \to \infty$,

$$J_k^T(\boldsymbol{b}) \to J(\boldsymbol{b}),$$

which is independent of k. ∎

From Theorem 1, we can obtain the optimal decision rule:

$$u_k^* = \begin{cases} 0, & \text{if } g(\boldsymbol{b_k}, 0) + \delta \mathbb{E}_0[J(\boldsymbol{b_{k+1}})|\boldsymbol{b_k}] \\ & \quad < g(\boldsymbol{b_k}, 1) + \delta \mathbb{E}_1[J(\boldsymbol{b_{k+1}})|\boldsymbol{b_k}]; \\ 1, & \text{otherwise.} \end{cases}$$

V. Perseus-based Approximate Policy

Exactly solving the Bellman equation in (11) is difficult. Thus, we resort to using numerical techniques to approximate the cost-to-go function. We use the Perseus algorithm [7], [8], which is an iterative algorithm that approximates the cost-to-go function through a set of hyperplanes. We denote the estimate of the cost-to-go function at iteration k as $J^{(k)}(\cdot)$ and the accumulated hyperplanes as $A^{(k)}$. An individual hyperplane is defined as $\boldsymbol{\alpha}_i^{(k)}$, where i is used to index the hyperplanes in $A^{(k)}$. The steps of the algorithm are the following:

1) Sample a set of belief vectors $\mathcal{B}^{(1)} \subset \mathcal{B}$. These beliefs are obtained by simulating the system with random actions. After an action is taken, a new belief vector is calculated by using the recursive formula in (8).
2) Pick a belief vector $\boldsymbol{b} \in \mathcal{B}^{(k)}$ uniformly at random. The candidate hyperplane to be added to $A^{(k)}$ is computed by:
$$\boldsymbol{\alpha} = \arg \min_{\{\boldsymbol{\alpha}_u^{\boldsymbol{b}}\}_{u \in \mathcal{U}}} \boldsymbol{b} \cdot \boldsymbol{\alpha}_u^{\boldsymbol{b}} \tag{12}$$
where
$$\begin{aligned}
&\boldsymbol{\alpha}_u^{\boldsymbol{b}} = g(\boldsymbol{b}, u) \\
&+ \delta \iint\limits_{(\hat{P}, \hat{\theta}) \in \mathbb{R}^2} f(\hat{P}, \hat{\theta}|u, \boldsymbol{b}) \arg \min_{\boldsymbol{\alpha}_i^{(k)} \in A^{(k)}} \phi(\boldsymbol{b}, u, \hat{P}, \hat{\theta}) \cdot \boldsymbol{\alpha}_i^{(k)}
\end{aligned} \tag{13}$$
and $f(\hat{P}, \hat{\theta}|u, \boldsymbol{b})$ is the probability density function of the measurements conditioned on $\boldsymbol{b}$ and u.
3) If $\boldsymbol{b} \cdot \boldsymbol{\alpha} \leq J^{(k)}(\boldsymbol{b})$, add hyperplane $\boldsymbol{\alpha}$ to $A^{(k+1)}$, and set $\mathcal{B}^{(k+1)} = \{\boldsymbol{p} \in \mathcal{B}^{(k)} : \boldsymbol{\alpha} \cdot \boldsymbol{b} > J^{(k)}(\boldsymbol{b})\}$. Otherwise, hyperplane $\boldsymbol{\alpha}' = \arg \min_{\boldsymbol{\alpha} \in A^{(k)}} \boldsymbol{\alpha} \cdot \boldsymbol{b}$ is added, and $\mathcal{B}^{(k+1)} = \mathcal{B}^{(k)} \backslash \{\boldsymbol{b}\}$.
4) If $\mathcal{B}^{(k+1)} = \emptyset$, go to step 1 using $A^{(k+1)}$ as the initial set of hyperplanes. Otherwise, go to step 2. Repeat this process until some predetermined convergence criterion is satisfied. We denote the resulting set of hyperplanes by A^*.

It should be noted that the hyperplanes that are used to approximate the cost-to-function are precomputed. Thus, the task of computing the hyperplanes does not burden the decision making process, since the approximation algorithm is run offline. After A^*, the approximate policy obtained from the Perseus algorithm is calculated as follows:

$$\begin{aligned}
&\hat{u} = \arg \min_{u \in \mathcal{U}} g(\boldsymbol{b}, u) \\
&+ \delta \iint\limits_{(\hat{P}, \hat{\theta}) \in \mathbb{R}^2} f(\hat{P}, \hat{\theta}|u, \boldsymbol{b}) \min_{\boldsymbol{\alpha}_i \in A^*} \phi(\boldsymbol{b}, u, \hat{P}, \hat{\theta}) \cdot \boldsymbol{\alpha}_i,
\end{aligned} \tag{14}$$

TABLE I

AVERAGE COSTS FOR $\delta = 0.9$.

σ^2	Algorithm			
	Non-shed	γ-heur.	P-heur.	Perseus
0.1	5.15	4.83	4.91	2.79
0.05	5.15	4.34	4.74	2.61
0.01	5.15	4.29	4.58	2.46

TABLE II

AVERAGE COSTS FOR $\delta = 0.8$.

σ^2	Algorithm			
	Non-shed	γ-heur.	P-heur.	Perseus
0.1	1.89	1.61	1.67	1.24
0.05	1.89	1.57	1.62	1.19
0.01	1.89	1.52	1.59	1.09

TABLE III

AVERAGE COSTS FOR $\delta = 0.7$.

σ^2	Algorithm			
	Non-shed	γ-heur.	P-heur.	Perseus
0.1	0.95	0.84	0.86	0.53
0.05	0.95	0.81	0.83	0.51
0.01	0.95	0.77	0.8	0.46

VI. NUMERICAL RESULTS

In this section, we demonstrate the effectiveness of our proposed algorithm, compared to a system without load shedding capability. In addition, we compare the policy obtained by the Perseus approximation algorithm with two heuristic policies when load shedding is allowed. As we see, the load shedding capability reduces the average cost significantly when the Perseus approximation algorithm is used. The heuristic algorithms provide a smaller performance gain, but do not require prior computations as the Perseus algorithm does.

For the first heuristic algorithm, we shed load if

$$\mathbb{P}(\gamma_k = \gamma^{(0)}|I_k) > 0.5,$$

otherwise, we restore load. This algorithm is referred to as γ-heuristic algorithm. Note that $\mathbb{P}(\gamma_k = \gamma^{(0)}|I_k)$ can be calculated through the components of $\boldsymbol{b}_k$.

The second heuristic algorithm boils down to thresholding the probability that the system breaks down at the next time instant, when a non-shed decision is taken, which is referred to as Probabilistic-heuristic algorithm. In particular, if a non-shed action is taken, the system is likely to break down. The probability of a breakdown at the next time instant can be calculated through $\boldsymbol{b}_k$. The algorithm decides to shed load when this calculated probability is larger than 0.5, otherwise, the decision is to move to a non-shed state. This algorithm is less conservative than the γ-heuristic algorithm.

We set $\gamma^{(0)} = 0.5$, $\gamma^{(1)} = 1$, $\rho_0 = 0.1$ and $\rho_1 = 0.01$, $C_b = 1000$ and $C_d = 0.01$. For the power injection model, we consider $F = 2$, $p^{(1)} = 0.4$ and $p^{(2)} = 0.8$. For each state, we assume uniform transition probabilities. We vary the value of the discount factor δ and the variance of the observation noise. The average costs are summarized in Tables I, II and III. In Table I, we provide the numerical results for $\delta = 0.9$.

As we can see from Table I, the heuristic algorithms provide approximately a 10% decrease in average cost compared to the results acquired when we do not allow the system to enter shed mode. Furthermore, we note that the Perseus solution provides a significant gain (approximately 50% decrease in average cost) compared to the case when doing nothing. Similar results can be observed when $\delta = 0.8$ and $\delta = 0.7$ (Table II and III).

Note that decreasing the noise variance leads to a better performance for all of the algorithms. This is to be expected, since lower variance leads to better awareness of the system state through the acquired observations. Furthermore, decreasing the discount factor δ leads to smaller cost values overall. This holds since smaller δ leads the per-stage costs to decrease faster.

VII. CONCLUSION

In this paper, we proposed a stochastic control framework based on shedding/unshedding maximum allowable load, to protect mission-critical data centers from breakdowns. We solved this problem by formulating it into a dynamic programming problem. Our numerical results show that our algorithm has a superior performance compared to the case where no control is available.

One potential direction of future interest is to generalize to the case with multiple data centers connected to one power distribution network, in which each data center controls its own load.

REFERENCES

[1] G. Rovatsos, J. Jiang, A. D. Domínguez-García, and V. V. Veeravalli, "Statistical power system line outage detection under transient dynamics," *submitted to IEEE Transactions on Signal Processing*, 2016.

[2] G. Rovatsos, X. Jiang, A. D. Domínguez-García, and V. V. Veeravalli, "Comparison of statistical algorithms for power system line outage detection," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr. 2016.

[3] Y. C. Chen, T. Banerjee, A. D. Domínguez-García, and V. V. Veeravalli, "Quickest line outage detection and identification," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 749–758, 2016.

[4] A. G. Tartakovsky, I. V. Nikiforov, and M. Basseville, *Sequential Analysis: Hypothesis Testing and Change-Point Detection*, Statistics. CRC Press, 2014.

[5] H. V. Poor and O. Hadjiliadis, *Quickest Detection*, Cambridge University Press, 2009.

[6] V. V. Veeravalli and T. Banerjee, "Quickest change detection," *Academic press library in signal processing: Array and statistical signal processing*, vol. 3, pp. 209–256, 2013.

[7] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Int. Res.*, vol. 24, no. 1, pp. 195–220, Aug. 2005.

[8] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, July 2013.

[9] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.