SCALABLE ENERGY DISAGGREGATION VIA SUCCESSIVE SUBMODULAR APPROXIMATION

Faisal M. Almutairi[⋆]*, Aritra Konar*[†]*, and Nicholas D. Sidiropoulos*[†]

[⋆] Dept. of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN
[†] Dept. of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA

ABSTRACT

Energy disaggregation is the task of decomposing the aggregated power consumption readings of a household into its constituent parts. In this paper, we propose a supervised, non-parametric framework for energy disaggregation. We demonstrate that the problem is equivalent to maximizing a set-function subject to combinatorial constraints, which is NP–hard in its general form. A simple polynomial-time successive approximation algorithm which exploits submodularity per set-block to iteratively maximize a sequence of global lower bounds of the objective function is proposed for obtaining approximate solutions. Experiments on real data indicate the superior disaggregation performance and scalability of our approach over a state-of-the-art parametric Factorial Hidden Markov Model based framework employing convex relaxation.

1. INTRODUCTION

Reducing electricity usage in households is important for improving the efficiency of the power grid and mitigating the adverse effect of energy consumption on the environment. *Energy disaggregation* (also known as Non-Intrusive Load Monitoring (NILM)) seeks to inform residents about the power consumption of specific appliances in their household by decomposing the aggregated power of all appliances obtained from smart meter readings into its constituent parts. Accurately extracting the appliance-level power consumption helps in providing detailed and informative billing to consumers which has been shown to gain costumers' appreciation leading to reduced consumption [1].

NILM was originally proposed in [2,3], wherein a classification-based approach was adopted for disaggregation. These methods rely on classifying switching events in the aggregated signal by assigning them to the correct appliance after the switching characteristics for each appliance are learned from a training phase. The drawback of such approaches is that they do not distinguish between appliances with low power consumption. More recently, machine learning techniques have been applied to NILM. One line of work utilizes matrix/tensor factorization, where a latent factor/dictionary is learned from a training set containing the ground truth about the appliance-level consumption, and then this factor/dictionary is used in the disaggregation task [4–6]. Although these factorization-based techniques are conceptually appealing, the disaggregation accuracy depends on performing a careful training process, requiring a large amount of data to capture all the possible operating times [4], or assuming a common latent factor between training and test data [5, 6].

Another popular approach is based on using Factorial Hidden Markov Models (FHMMs), where each appliance is modeled as a HMM evolving independently of the others. The operating states of each appliance are the hidden variables and the observed aggregated power is a joint function of all hidden states [7–10]. The authors of [8] put forth an additive FHMM (AFHMM) where the observed power is the sum of the hidden states – the assumption being that the difference between two successive observations corresponds to a change in *at most one* HMM (appliance). This approach is unsupervised (i.e., the model parameters and appliance power consumptions are learned from the aggregated signal) and accounts for outliers (rarely used appliances). However, parameter identification can be an issue, and performing exact inference requires solving a challenging combinatorial optimization problem with boolean variables. As a result, the authors resort to relaxing the combinatorial constraints to obtain a convex programming problem. This relaxation-based approach can work well if the assumption that only one appliance may change state at any given 'time' (interval between subsequent power measurements) is valid. Recently, Zhong *et al.* [10] proposed a supervised model which incorporates additional information in AFHMM by constraining the total power consumed by each appliance state to a predefined value (learned from training data). The downside is that adding these extra constraints makes the model more complicated compared to basic AFHMM. As a result, the authors resort to adding extra variables and relaxing them in addition to relaxing the combinatorial constraints in order to obtain a convex optimization problem. Furthermore, while the relaxed problem can be optimally solved in polynomial-time using general-purpose convex programming solvers, this still incurs significant computational complexity for large-scale problem instances.

In this paper, we introduce a new supervised framework for energy disaggregation. We also learn device states and consumption levels from training data, but, unlike FHMMs, our approach is non-parametric, and it does not hinge on the assumption that at most one device can change state at any given time (interval) [1]. We demonstrate that the resulting combinatorial optimization problem can be posed as a constrained set-function maximization problem, which is NP–hard in its general form. A block successive approximation scheme which exploits per-block submodularity of the objective function is devised to obtain a discrete optimization analogue of the well known majorization-minimization (MM) algorithm [11–13]. This allows us to obtain a simple polynomial-time approximation algorithm that iteratively maximizes a sequence of global lower bounds of the objective function. The solution sequence generated by the algorithm is always guaranteed to be feasible for the combinatorial constraints and also features monotonically non-decreasing objective value. As opposed to the FHMM methods, it is also extremely scalable, featuring very low per-iteration complexity. Simulations conducted on real data demonstrate the effectiveness of our approach in terms of disaggregation performance and scalability

Contact: almut012@umn.edu, (aritra,nikos)@virginia.edu.

[1] This assumption is made in [8, 9], but not in [10].

compared to the state-of-the-art AFHMM method proposed in [10].

2. PROBLEM FORMULATION

For a given household outfitted with M appliances, let $\{y_t\}_{t=1}^T$ represent the sequence of aggregated power consumption readings for the entire household over a finite time horizon T. The goal of *energy disaggregation* is to infer the power consumption sequence of each appliance from the aggregated readings; i.e., we wish to breakdown y_t into M components of the form $y_t = \sum_{m=1}^M x_{m,t}$, where $x_{m,t} \in \mathbb{R}_+$ denotes the power consumption of appliance $m \in [M] := \{1, \cdots, M\}$ at time instant $t \in [T] := \{1, \cdots, T\}$. It is assumed that the power consumption profile of each appliance $m \in [M]$ can be approximated using $K_m \geq 2$ states, where each state corresponds to a particular mode of appliance operation (e.g., 'completely on', 'partially on', 'standby', 'off') with a constant power consumption $\mu_m^{(k)} \in \mathbb{R}_+, \forall\, k \in [K_m]$. Since each appliance can operate in only one state at a given t, we can represent

$$x_{m,t} \approx \boldsymbol{\mu}_m^T \mathbf{z}_{m,t}, \forall\, m \in [M], \forall\, t \in [T] \tag{1}$$

where $\boldsymbol{\mu}_m \in \mathbb{R}_+^{K_m}$ is a vector containing the power consumption values across all states for appliance m and $\mathbf{z}_{m,t} \in \{0,1\}^{K_m}$ is the state indicator vector of appliance m at time instant t (i.e., $\mathbf{1}^T \mathbf{z}_{m,t} = 1$). Hence, the aggregated power consumption at instant t can be approximated as $y_t \approx \sum_{m=1}^M \boldsymbol{\mu}_m^T \mathbf{z}_{m,t}$. Provided that the power consumption profiles $\{\boldsymbol{\mu}_m\}_{m=1}^M$ are known *apriori*, at each instant $t \in [T]$, the problem boils down to determining the state vectors $\{\mathbf{z}_{m,t}\}_{m=1}^M$. However, this is an ill-posed task, since we wish to infer the contribution of M appliances from only a single power measurement. In order to reduce problem under-determinacy, we will exploit the fact that in general, appliances change states infrequently over a short time scale. Thus, instead of performing disaggregation for each instant $t \in [T]$, we will perform the task over the entire time horizon jointly while enforcing temporal smoothness on the evolution of the state indicator vectors $\{\mathbf{z}_{m,t}\}_{(m,t=1)}^{(M,T)}$. Formally, our problem can be stated as follows

$$\min_{\{\mathbf{z}_{m,t}\}_{(m,t=1)}^{(M,T)}} \sum_{t=1}^T \Big(y_t - \sum_{m=1}^M \boldsymbol{\mu}_m^T \mathbf{z}_{m,t}\Big)^2 - \sum_{\substack{m=1,\\ t=2}}^{M,T} \lambda_m \mathbf{z}_{m,t}^T \mathbf{z}_{m,t-1} \tag{2a}$$

$$\text{s.t.} \quad \mathbf{z}_{m,t} \in \{0,1\}^{K_m}, \mathbf{1}^T \mathbf{z}_{m,t} = 1, \forall\, m \in [M], \forall\, t \in [T] \tag{2b}$$

where the first summand of the cost function represents a least-squares data fitting term, while the second summand is a smoothness-inducing regularization function where each term maximizes the sum of correlations between the state vectors associated with the m^{th} appliance for the present and previous time instants. Finally, $\lambda_m \in \mathbb{R}_+, \forall\, m \in [M]$ are appliance-specific regularization parameters.

Note that (2) is a combinatorial optimization problem with $\{0,1\}$ variables, and hence, can be equivalently expressed as the minimization of a set-function subject to set constraints. In order to simplify notation and express the problem in this form concisely, we first define $\boldsymbol{\mu} := [\boldsymbol{\mu}_1^T, \boldsymbol{\mu}_2^T, ..., \boldsymbol{\mu}_M^T]^T$, $\mathbf{z}_t := [\mathbf{z}_{1,t}^T, \mathbf{z}_{2,t}^T, ..., \mathbf{z}_{M,t}^T]^T$, $\mathbf{A} := \boldsymbol{\mu}\boldsymbol{\mu}^T$, and $\mathbf{b}_t = 2y_t\boldsymbol{\mu}$. We now represent (2) using set notation as follows: first, define $\mathcal{S}_{m,t} \subset \mathcal{V}_m := [K_m]$ to be the set of states of appliance $m \in [M]$ at time $t \in [T]$. Since each appliance can only occupy one state at a time, we have $|\mathcal{S}_{m,t}| = 1$. Next, we define the overall set of states at each time $\mathcal{S}_t := \{\mathcal{S}_{1,t}, \mathcal{S}_{2,t} \cdots, \mathcal{S}_{M,t}\}$. Note that $\mathcal{S}_t$ can be represented as a proper subset of a larger ground set $\mathcal{V} := \cup_{m=1}^M \mathcal{V}_m$ (here the union is disjoint), with a maximum of $K := \sum_{m\in[M]} K_m$ total states. In terms of $\mathcal{S}_t$, the constraint that each appliance can only occupy one state at a time can be equivalently expressed as $|\mathcal{S}_t \cap \mathcal{V}_m| = 1, \forall\, m \in [M]$. With these definitions, we can express the Boolean vector $\mathbf{z}_t$ as the indicator vector of $\mathcal{S}_t$, i.e., $\mathbf{z}_t := \mathbb{1}_{\mathcal{S}_t}$, where $\mathbf{z}_t(v) = 1, \forall\, v \in \mathcal{S}_t$ and $\mathbf{z}_t(v) = 0$ otherwise. Furthermore, each individual term of the regularization function can be expressed as $|\mathcal{S}_{m,t} \cap \mathcal{S}_{m,t-1}|, \forall\, m \in [M], \forall\, t \in [T]$, where $|.|$ denotes the cardinality of a set, and $\cap$ is the intersection between sets. Since (2) is in minimization form, temporal smoothness is enforced by equivalently maximizing the overlap of the subset of states $\mathcal{S}_{m,t}$ and $\mathcal{S}_{m,t-1}$. Putting everything together, in terms of set notation, problem (2) can be equivalently expressed as

$$\min_{\mathcal{S}\in\mathcal{I}} \Bigg\{ F(\mathcal{S}) := \sum_{t=1}^T \Big(\mathbb{1}_{\mathcal{S}_t}^T \mathbf{A} \mathbb{1}_{\mathcal{S}_t} - \mathbf{b}_t^T \mathbb{1}_{\mathcal{S}_t}\Big) - \sum_{t=2}^T \sum_{m=1}^M \lambda_m |\mathcal{S}_{m,t} \cap \mathcal{S}_{m,t-1}| \Bigg\} \tag{3}$$

where $\mathcal{I} := \mathcal{I}_1 \times \cdots \times \mathcal{I}_T$ and $\mathcal{I}_t := \{\mathcal{S}_t \subset \mathcal{V} : |\mathcal{S}_t \cap \mathcal{V}_m| = 1, \forall\, m \in [M], \forall\, t \in [T]\}$ denotes the family of subsets which allow the selection of only one state per appliance at every time instant. The proposed formulation has the following salient features: i) it is non-parametric, and ii) it allows multiple appliances to change states at a given time instant, as opposed to the FHMM-based approaches in [8, 9] which allow at most one appliance to change state at a time. While minimizing set-functions exactly is NP–hard in its general form [14, 15], in the next section, we demonstrate that the cost function of (2) exhibits a special property which allows the development of a simple polynomial-time approximation algorithm.

3. PROPOSED APPROACH

Given a ground set of N elements $\mathcal{V} := \{v_1, \cdots, v_N\}$, consider the set-function $f : 2^N \to \mathbb{R}$ which assigns a real value to any subset $\mathcal{S} \subseteq \mathcal{V}$. Notable amongst set-functions is the sub-class of *submodular* set-functions, which can be formally defined as follows.

Definition 1. [Submodular function] [16, Definition 2.1] The set function f is said to be *submodular* if and only if

$$f(\mathcal{X} \cup \{v\}) - f(\mathcal{X}) \geq f(\mathcal{Y} \cup \{v\}) - f(\mathcal{Y}) \tag{4}$$

for all $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{V} \setminus v$. That is, given any subset of elements $\mathcal{X}$, the marginal gain derived by adding an element v to $\mathcal{X}$ does not increase when we instead add v to the superset $\mathcal{Y}$. Hence, submodular functions exhibit a natural diminishing returns property.

Definition 2. [Modular function] [16, Proposition 2.1] A set function y is said to be *modular* if and only if there exists a vector $\mathbf{y} \in \mathbb{R}^N$ for all subsets $\mathcal{X} \subseteq \mathcal{V}$ such that $y(\mathcal{X}) = \mathbf{y}^T \mathbb{1}_{\mathcal{X}} = \sum_{e\in\mathcal{X}} \mathbf{y}(e)$. Note that such a function satisfies the inequality in Definition 1 with equality.

In order to establish the link between submodular functions and the problem at hand, it will be convenient to equivalently express (3) in maximization form as

$$\max_{\mathcal{S}\in\mathcal{I}^T} \Bigg\{ -F(\mathcal{S}) := \sum_{t=1}^T g_t(\mathcal{S}_t) + \sum_{t=2}^T h(\mathcal{S}_t, \mathcal{S}_{t-1}) \Bigg\} \tag{5}$$

where we have defined $g_t(\mathcal{S}_t) := -\mathbb{1}_{\mathcal{S}_t}^T \mathbf{A} \mathbb{1}_{\mathcal{S}_t} + \mathbf{b}_t^T \mathbb{1}_{\mathcal{S}_t}$ and $h(\mathcal{S}_t, \mathcal{S}_{t-1}) := \sum_{m=1}^M \lambda_m |\mathcal{S}_{m,t} \cap \mathcal{S}_{m,t-1}|$. While $-F(\mathcal{S})$ is not a submodular function, we will demonstrate that $-F(\mathcal{S}_t|\mathcal{S}_{\neg t})$ is submodular in each variable set $\mathcal{S}_t$ while fixing the other sets $\mathcal{S}_{\neg t}$.

Note that $-F(\mathcal{S}_t|\mathcal{S}_{\neg t}) = g_t(\mathcal{S}_t) + h(\mathcal{S}_t|\mathcal{S}_{t-1}) + h(\mathcal{S}_t|\mathcal{S}_{t+1})$ (except for $t = 1$ ($t = T$), where the regularization component only includes the third (second) term, respectively). Since the class of submodular functions is closed under non-negative linear combinations, it suffices to establish that each of the three terms comprising $-F(\mathcal{S}|\mathcal{S}_{\neg t})$ is submodular. First, consider $g_t(\mathcal{S}_t)$: by construction, $-\mathbf{A}$ has all its off-diagonal elements non-positive (since $\boldsymbol{\mu} \geq \mathbf{0} \Rightarrow \mathbf{A} = \boldsymbol{\mu}\boldsymbol{\mu}^T \geq \mathbf{0}$), which is both a necessary and sufficient condition for $-\mathbb{1}_{\mathcal{S}_t}^T \mathbf{A} \mathbb{1}_{\mathcal{S}_t}$ to be submodular [16, Proposition 6.3]. Meanwhile,

simple inspection reveals that the second term $\mathbf{b}_t^T \mathbb{1}_{\mathcal{S}_t}$ is modular. Hence, $g_t(\mathcal{S}_t)$ is submodular. For the regularization functions, fixing $\mathcal{S}_{t-1}$, we can express $h(\mathcal{S}_t|\mathcal{S}_{t-1}) = \mathbf{c}^T \mathbb{1}_{\mathcal{S}_t}$ (i.e., in modular form), where $\mathbf{c} := \mathbf{W}\mathbb{1}_{\mathcal{S}_{t-1}}$ and $\mathbf{W} := \text{blkdiag}(\lambda_1 \mathbf{I}_{K_1}, \cdots, \lambda_M \mathbf{I}_{K_M})$. Similarly, fixing $\mathcal{S}_{t+1}$, we obtain $h(\mathcal{S}_t|\mathcal{S}_{t+1}) = \mathbf{d}^T \mathbb{1}_{\mathcal{S}_t}$, where $\mathbf{d} := \mathbf{W}\mathbb{1}_{\mathcal{S}_{t+1}}$. Hence, each block subproblem

$$\max_{\mathcal{S}_t \in \mathcal{I}_t} \; - F(\mathcal{S}_t|\mathcal{S}_{\neg t}) \tag{6}$$

corresponds to maximizing a function that is submodular in $\mathcal{S}_t$ subject to the constraints $\mathcal{I}_t$. This motivates using a block coordinate descent (BCD) algorithm for updating the variable sets $\{\mathcal{S}_t\}_{t=1}^T$ in alternating fashion, where the update for each block requires optimally solving (6). However, (6) is an NP–hard problem in its general form, which prevents efficient solution. Nevertheless, there exist effective polynomial-time approximation algorithms for (6) in the following special cases. If the objective function of (6) is monotone, then it is well known that a simple greedy algorithm guarantees a 0.5-factor approximation for all instances of (6) [17]. A more sophisticated algorithm employing a nonlinear continuous relaxation followed by a rounding procedure was proposed in [18] which yields an improved approximation factor of $(1 - 1/e)$ for (6). As the function $-F(\mathcal{S}_t|\mathcal{S}_{\neg t})$ is not monotone (the culprit being the term $-\mathbb{1}_{\mathcal{S}_t}^T \mathbf{A} \mathbb{1}_{\mathcal{S}_t}$), this precludes us from applying these algorithms. For constrained submodular maximization of *non-negative*, non-monotone functions, [19] proposed a polynomial-time algorithm which guarantees a 0.385-factor approximation. Unfortunately, $-F(\mathcal{S}_t|\mathcal{S}_{\neg t})$ is not guaranteed to be non-negative either (depending upon the choice of $\{\lambda_m\}_{m=1}^M$). For such submodular functions, which are neither monotone nor non-negative, even verifying whether the maximum is larger than zero or not is NP–hard in general [20]. Hence, in its general form, (6) is inapproximable via polynomial-time algorithms.

To overcome these difficulties, we resort to modifying the BCD algorithm by maximizing a judiciously designed approximation of the objective function during each block iteration. Our approach can be viewed as a a discrete optimization analogue of the popular BSUM algorithm [11], where we iteratively update $\{\mathcal{S}_t\}_{t=1}^T$ in cyclic fashion by maximizing a sequence of global lower bounds of $-F(\mathcal{S})$ in the following manner. Starting from an initial solution set $\mathcal{S}^{(0)} \in \mathcal{I}$, at each iteration $k \in \mathbb{N}$, we seek to update the set-block $t = (k \bmod T) + 1$ while keeping the other blocks fixed. As the per-block subproblem (6) is hard to solve, we utilize a discrete variant of the majorization-minimization (MM) principle proposed in [14, 15] to construct a modular set-function $m_t(\mathcal{S}_t)$ to approximate $g_t(\mathcal{S}_t)$ about the current solution set $\mathcal{S}_t^{(k)}$ such that: i) $g_t(\mathcal{S}_t) \geq m_t(\mathcal{S}_t)$, $\forall\, \mathcal{S}_t \in \mathcal{I}$ and ii) $g_t(\mathcal{S}_t^{(k)}) = m_t(\mathcal{S}_t^{(k)})$. Thus, on replacing $g_t(\mathcal{S}_t)$ by its modular lower bound $m_t(\mathcal{S}_t)$, at each iteration, we obtain a subproblem of the form

$$\max_{\mathcal{S}_t \in \mathcal{I}_t} \; \{u_t(\mathcal{S}_t|\mathcal{S}_{\neg t}) := m_t(\mathcal{S}_t) + h(\mathcal{S}_t|\mathcal{S}_{t-1}) + h(\mathcal{S}_t|\mathcal{S}_{t+1})\} \tag{7}$$

which is equivalent to maximizing a global modular lower bound of $-F(\mathcal{S})$. The upshot is that (7) can be solved optimally in $O(K)$ time. To see this, note that as $u_t(\mathcal{S}_t|\mathcal{S}_{\neg t})$ is a modular function by construction, we can equivalently write (7) as

$$\max_{\mathcal{S}_t \in \mathcal{I}_t} \; \mathbf{u}_t^T \mathbb{1}_{\mathcal{S}_t} \tag{8}$$

For the purpose of computing an optimal solution, it suffices to inspect the entries of $\mathbf{u}_t$ corresponding to each subset $\mathcal{S}_{m,t}$, and obtain the index of the largest entry, $\forall\, m \in [M]$. This can be accomplished via a simple linear scan, which requires $O(K)$ time.

Note that the applicability of the scheme hinges upon our ability to compute a modular lower bound of $g_t(\mathcal{S}_t)$ which satisfies the desired properties. This can be accomplished by utilizing the notion of submodular subdifferentials, which is formally defined as follows.

Definition 3. [Subdifferential Sets of Submodular functions] [21, Section 6.2] The subdifferential set of a submodular set-function g for a given set $\mathcal{Y} \subseteq \mathcal{V}$ is defined as

$$\partial g(\mathcal{Y}) := \{\mathbf{y} \in \mathbb{R}^N : g(\mathcal{X}) - g(\mathcal{Y}) \geq y(\mathcal{X}) - y(\mathcal{Y}), \forall\, \mathcal{X} \subseteq \mathcal{V}\}$$

Let $\mathbf{v}_{\mathcal{Y}}^g \in \partial g(\mathcal{Y})$ denote a subgradient of g at $\mathcal{Y}$. We will require to compute such a subgradient for constructing our desired modular lower bound. In order to do so, it suffices to compute any element in the set of extreme points of $\partial g(\mathcal{Y})$, which can be exactly characterized as follows.

Proposition 1. [Extreme points of submodular subdifferentials] [21, Theorem 6.11] For each $\mathcal{Y} \subseteq \mathcal{V}$, a vector $\mathbf{v}_{\mathcal{Y}}^g$ is an extreme point of $\partial g(\mathcal{Y})$ if and only if there exists a maximal chain $\mathcal{C} : \emptyset = \mathcal{S}^{(0)} \subset \mathcal{S}^{(1)} \subset \cdots \subset \mathcal{S}^{(N)} = \mathcal{V}$ which includes $\mathcal{Y}$ (i.e., $\mathcal{Y} = \mathcal{S}^{(j)}$ for some $j \in [N]$) such that the modular function $v_{\mathcal{Y}}^g$ associated with $\mathbf{v}_{\mathcal{Y}}^g$ satisfies

$$\begin{aligned} v_{\mathcal{Y}}^g(\mathcal{S}^{(i)} \backslash \mathcal{S}^{(i-1)}) &= v_{\mathcal{Y}}^g(\mathcal{S}^{(i)}) - v_{\mathcal{Y}}^g(\mathcal{S}^{(i-1)}) \\ &= g(\mathcal{S}^{(i)}) - g(\mathcal{S}^{(i-1)}), \forall\, i \in [N] \end{aligned} \tag{9}$$

In [22], Edmonds presented a greedy procedure for computing such an extreme point. Let $\boldsymbol{\pi}$ be a permutation of $\mathcal{V}$ which assigns the elements in $\mathcal{Y}$ to the first $|\mathcal{Y}|$ positions (i.e., $\boldsymbol{\pi}(i) \in \mathcal{Y}, \forall\, i \leq |\mathcal{Y}|$) [2]. Every such permutation can be shown to define a maximal chain $\mathcal{S}_{\boldsymbol{\pi}}^{(0)} \subset \mathcal{S}_{\boldsymbol{\pi}}^{(1)} \subset \ldots \subset \mathcal{S}_{\boldsymbol{\pi}}^{(N)}$ with elements $\mathcal{S}_{\boldsymbol{\pi}}^{(0)} = \emptyset$, and $\mathcal{S}_{\boldsymbol{\pi}}^{(i)} = \{\boldsymbol{\pi}(1), \boldsymbol{\pi}(2), ..., \boldsymbol{\pi}(i)\}, \forall\, i \in [N]$. Note that we have $\mathcal{S}_{\boldsymbol{\pi}}^{|\mathcal{Y}|} = \mathcal{Y}$. Using this chain, we define a vector $\mathbf{v}_{\mathcal{Y},\boldsymbol{\pi}}^g \in \mathbb{R}^N$ with entries

$$\mathbf{v}_{\mathcal{Y},\boldsymbol{\pi}}^g(\boldsymbol{\pi}(i)) = \begin{cases} g(\mathcal{S}_{\boldsymbol{\pi}}^{(1)}) & \text{if } i = 1 \\ g(\mathcal{S}_{\boldsymbol{\pi}}^{(i)}) - g(\mathcal{S}_{\boldsymbol{\pi}}^{(i-1)}), & \text{otherwise} \end{cases} \tag{10}$$

With $\mathbf{v}_{\mathcal{Y},\boldsymbol{\pi}}^g$ obtained, we define the following modular function for all subsets $\mathcal{S} \subseteq \mathcal{V}$

$$v_{\mathcal{Y},\boldsymbol{\pi}}^g(\mathcal{S}) := \sum_{e \in \mathcal{S}} \mathbf{v}_{\mathcal{Y},\boldsymbol{\pi}}^g(e) \tag{11}$$

By construction, it can be verified that $\mathbf{v}_{\mathcal{Y},\boldsymbol{\pi}}^g$ satisfies the conditions listed in Proposition 1 and thus, is an extreme point of $\partial g(\mathcal{Y})$. Furthermore, it has been shown that [23] for every $\mathcal{Y} \subseteq \mathcal{V}$, the modular function $v_{\mathcal{Y},\boldsymbol{\pi}}^g(\mathcal{S})$ satisfies the following properties: i) $v_{\mathcal{Y},\boldsymbol{\pi}}^g(\mathcal{S}) \leq g(\mathcal{S}), \forall\, \mathcal{S} \subseteq \mathcal{V}$, and ii) $v_{\mathcal{Y},\boldsymbol{\pi}}^g(\mathcal{S}_{\boldsymbol{\pi}}^{(i)}) = g(\mathcal{S}_{\boldsymbol{\pi}}^{(i)}), \forall\, i \in [N]$. Thus, the modular approximation $v_{\mathcal{Y},\boldsymbol{\pi}}^g(\mathcal{S})$ of $g(\mathcal{S})$ about the set $\mathcal{Y}$ satisfies the desired properties we listed earlier.

Hence, in our aforementioned procedure, given the current solution set $\mathcal{S}^{(k)}$, in order to update $\mathcal{S}_t$, we first generate a permutation $\boldsymbol{\pi}_t^{(k)}$ to construct a tight modular lower bound $m_t(\mathcal{S}_t) = v_{\mathcal{S}_t^{(k)},\boldsymbol{\pi}_t^{(k)}}^{g_t}(\mathcal{S}_t)$ of $g_t(\mathcal{S}_t)$ about $\mathcal{S}_t^{(k)}$ via a greedy algorithm. We thus obtain a modular global lower bound $u_t(\mathcal{S}_t|\mathcal{S}_{\neg t})$ of $-F(\mathcal{S})$ which is tight at $\mathcal{S} = \mathcal{S}^{(k)}$ and can be efficiently maximized to global optimality (via a linear scan in $O(K)$ time) at each iteration. Overall, we obtain Algorithm 1, which we term as BSMA (Block Successive Modular Approximation). Note that BSMA generates a sequence of solution sets $\{\mathcal{S}^{(k)}\}_{k \geq 0}$ which always satisfy the constraints $\mathcal{I}$. In addition, we have the following chain of inequalities

$$F(\mathcal{S}^{(k+1)}) \leq F(\mathcal{S}^{(k)}) \leq \cdots \leq F(\mathcal{S}^{(1)}) \leq F(\mathcal{S}^{(0)}) \tag{12}$$

since BSMA equivalently minimizes a locally tight upper bound of $F(\mathcal{S})$ at each iteration. Hence, BSMA monotonically reduces the cost function of (5).

[2] The remaining $N - |\mathcal{Y}|$ positions of $\boldsymbol{\pi}$ can be arbitrarily assigned.

Algorithm 1 : **B**lock **S**uccessive **M**odular **A**pproximation (BSMA)

Initialization: Set $k := 0, \mathcal{S}^{(0)} \in \mathcal{I}$.
Repeat

- Set $t = (k \bmod T) + 1$
- Generate permutation $\boldsymbol{\pi}_t^{(k)}$ using $\mathcal{S}_t^{(k)}$
- Compute modular approximation $v^{g_t}_{\mathcal{S}_t^{(k)}, \boldsymbol{\pi}_t^{(k)}}(\mathcal{S}_t)$ of $g_t(\mathcal{S}_t)$ about $\mathcal{S}_t^{(k)}$ via greedy algorithm
- Compute $\mathcal{S}_t^{(k+1)} = \arg\max_{\mathcal{S}_t \in \mathcal{I}} u(\mathcal{S}_t)$ via linear scan
- Set $\mathcal{S}_r^{(k+1)} = \mathcal{S}_r^{(k)}, \forall\, r \neq t$
- Set $k := k + 1$.

Until maximum number of iteration is reached

4. EXPERIMENTAL RESULTS

We compare the performance of BSMA against a state-of-the-art FHMM method, namely AFHMM+SAC proposed in [10]. As mentioned earlier, in this model, additional constraints are added to AFHMM to improve accuracy. These constraints encourage the total power assigned to each appliance to adhere to a power budget learned from the training phase. To implement this baseline, we use the Matlab modeling language CVX (as done in [10]). We also implemented BSMA in Matlab following the steps outlined in Algorithm 1.

For performance evaluation, we use the ECO[3] data set [24], a publicly available real dataset for NILM research. ECO contains power readings (both aggregated and plug-level for some appliances) for six Swiss households at 1 Hz frequency. Here, we focus on Household 2 since it has the largest number of appliances to be tested. Although our algorithm works perfectly with this high frequency data, we had to down-sample to 1 reading/minute because the baseline we compare to does not work with large data and requires the test set to be divided into segments.

We train both models on one week and test on another week of data. In the training phase, for AFHMM+SAC, we learn the following model parameters for each appliance: the initial and transition probabilities, the power budget and the state vector $\boldsymbol{\mu}_m$. Our algorithm only requires knowledge of $\{\boldsymbol{\mu}_m\}_{m=1}^M$, which is estimated by performing k-means clustering on the appliance consumption sequence $\{x_{m,t}\}_{t=1}^T$ as done in [10]. The number of clusters K_m for each appliance is chosen from the behavior of its power consumption sequence, which was set to be $K_m \in \{2,3,4,5\}$. The training set was used to tune the penalty parameters $\{\lambda_m\}_{m=1}^M$ for both methods[4] and to choose the number of iterations for our algorithm. Figure 1 provides a demonstration of the evolution of the cost function versus BSMA iterations when applied to the test set. For evaluation, we use two metrics: i) Root Mean Square Error (RMSE) to provide a measure that accounts for individual differences between true and predicted power at each time instant; and 2) Signal Aggregated Error (SAE) to provide a measure that indicates the error in power over a test period. The latter is a more informative metric because a) disaggregation is done over a billing period, and b) it allows for comparison amongst appliances since it is normalized. These metrics are formally defined as

$$\text{RMSE}_m := \sqrt{\frac{\sum_{t=1}^T (x_{m,t} - \hat{x}_{m,t})^2}{T}}, \tag{13}$$

$$\text{SAE}_m := \frac{|\sum_{t=1}^T x_{m,t} - \sum_{t=1}^T \hat{x}_{m,t}|}{\sum_{t=1}^T x_{m,t}} \tag{14}$$

where $x_{m,t}$ and $\hat{x}_{m,t}$ denote the true and inferred power consumption for appliance m at time t.

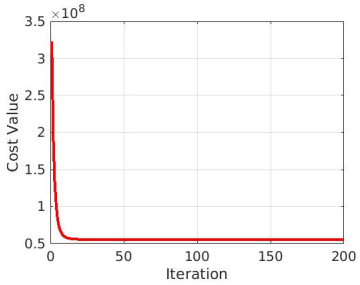

Fig. 1. Demonstration of the non-increasing cost value

Table 1. Comparing Performance

	RMSE		SAE	
Appliance	BSMA	AFHMM+SAC	BSMA	AFHMM+SAC
Tablet	**1.606**	1.748	0.995	**0.945**
Dishwasher	233.707	**209.333**	**0.592**	0.909
Air exhaust	5.246	**4.708**	**0.374**	0.989
Fridge	55.932	**48.061**	**0.120**	0.166
Entertainment	**93.297**	105.968	**0.613**	0.867
Freezer	**55.581**	65.371	**0.672**	0.900
Kettle	178.896	**172.268**	**0.920**	0.998
Lamp	27.369	**27.352**	**0.806**	0.879
Laptops	30.315	**30.049**	**0.496**	0.949
Stove	**225.028**	237.647	**0.835**	0.995
TV	**71.144**	83.114	**0.726**	0.947
Stereo	**23.108**	**23.108**	**0.185**	0.253
Average	**83.435**	84.061	**0.611**	0.817

Table 2. Comparing Running Time

	BSMA	AFHMM+SAC
Time (minutes)	**14.3**	605.3

Table 1 depicts the performance of the methods. In terms of RMSE, BSMA improves the baseline for half of the appliances and on average. In terms of SAE, our approach significantly outperforms the baseline for almost all appliances. We point out that BSMA always improves the baseline for appliances with $K_m \geq 3$ (e.g., Freezer, and Stove)–such appliances are problematic for general NILM approaches [25,26]. The overall performance of BSMA is very satisfactory considering the fact that it does not incorporate the additional information utilized by AFHMM+SAC. In Table 2, we show the timing results for both approaches when testing on a week of data with 1 reading/minute. Clearly, BSMA exhibits a vastly superior running time compared to SAC.

5. CONCLUSIONS

We proposed a non-parametric, supervised learning framework for energy disaggregation, which allows multiple appliances to change states at a given time instant. The resulting combinatorial optimization problem was posed as maximizing a general set-function, for which we devised a discrete successive approximation algorithm which exploits submodularity per set-block to iteratively maximize a sequence of global lower bounds of the objective function. The algorithm exhibits monotonically non-decreasing objective value, maintains feasibility of the generated iterates, and enjoys low per-iteration complexity. Experiments on a real NILM dataset revealed the very promising performance of our approach over a state-of-the-art FHMM method employing convex relaxation.

[3] https://www.vs.inf.ethz.ch/res/show.html?what=eco-data

[4] For AFHMM+SAC, the choice of penalty parameters dictates how stringently the power consumption constraints are enforced.

6. REFERENCES

[1] S. Darby, "The effectiveness of feedback on energy consumption," Environmental Change Institute, University of Oxford, Tech. Rep., Apr. 2006.

[2] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[3] F. Sultanem, "Using appliance signatures for monitoring residential loads at meter panel level," *IEEE Transactions on Power Delivery*, vol. 6, no. 4, pp. 1380–1385, 1991.

[4] J. Z. Kolter, S. Batra, and A. Y. Ng, "Energy disaggregation via discriminative sparse coding," in *Advances in Neural Information Processing Systems*, Vancouver, British Columbia, Canada, Dec. 2010, pp. 1153–1161.

[5] M. Figueiredo, B. Ribeiro, and A. de Almeida, "Electrical signal source separation via nonnegative tensor factorization using on site measurements in a smart home," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 2, pp. 364–373, Feb 2014.

[6] E. Elhamifar and S. Sastry, "Energy disaggregation via learning 'powerlets' and sparse coding," in *29th AAAI Conference on Artificial Intelligence*, Austin Texas, USA, Jan. 2015, pp. 629–635.

[7] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *Proceedings of the SIAM Conference on Data Mining*, Mesa, Arizona, USA, Apr 2011, pp. 747–758.

[8] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial HMMs with application to energy disaggregation," in *15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, La Palma, Canary Islands, Apr. 2012, pp. 1472–1482.

[9] M. Zhong, N. Goddard, and C. Sutton, "Interleaved factorial non-homogeneous hidden Markov models for energy disaggregation," in *The Neural Information Processing Systems workshop on Machine Learning for Sustainability*, Lake Tahoe, NV, USA, 2013.

[10] ——, "Signal aggregate constraints in additive factorial HMMs, with application to energy disaggregation," in *Advances in Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec 2014, pp. 3590–3598.

[11] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[12] F. Facchinei, L. Lampariello, and G. Scutari, "Feasible methods for nonconvex nonsmooth problems with applications in green communications," *Mathematical Programming*, pp. 1–36, 2016.

[13] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2017.

[14] M. Narasimhan and J. A. Bilmes, "A submodular-supermodular procedure with applications to discriminative structure learning," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, Jul 2005, pp. 404–412.

[15] R. Iyer and J. Bilmes, "Algorithms for approximate minimization of the difference between submodular functions, with applications," in *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, Catalina Island, CA, Aug 2012, pp. 407–417.

[16] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends in Machine Learning*, vol. 6, no. 2-3, pp. 145–373, 2013.

[17] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—part 1," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[18] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[19] N. Buchbinder and M. Feldman, "Constrained submodular maximization via a non-symmetric technique," *arXiv preprint arXiv:1611.03253*, 2016.

[20] U. Feige, V. S. Mirrokni, and J. Vondrak, "Maximizing non-monotone submodular functions," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1133–1153, 2011.

[21] S. Fujishige, *Submodular functions and optimization*, 2nd ed., ser. Annals of Discrete Mathematics. Elsevier, 2005, vol. 58.

[22] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, vol. 11, 1970.

[23] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, Jun. 1981.

[24] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, "The eco data set and the performance of non-intrusive load monitoring algorithms," in *Proceedings of the 1st ACM International Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys 2014)*, Memphis, TN, USA, Nov. 2014, pp. 80–89.

[25] S. Barker, S. Kalra, D. Irwin, and P. Shenoy, "Empirical characterization and modeling of electrical loads in smart homes," in *International Green Computing Conference Proceedings (IGCC)*, Arlington, Virginia, June 2013, pp. 1–10.

[26] N. Batra, H. Wang, A. Singh, and K. Whitehouse, "Matrix factorisation for scalable energy breakdown." in *The 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, CA, USA, Feb. 2017, pp. 4467–4473.