How much demand side flexibility do we need? Analyzing where to exploit flexibility in industrial processes

Lukas Barth
Institute of Theoretical Informatics,
Karlsruhe Institute of Technology
Karlsruhe, Germany
lukas.barth@kit.edu

Veit Hagenmeyer
Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology
Eggenstein-Leopoldshafen, Germany
veit.hagenmeyer@kit.edu

Nicole Ludwig
Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology
Eggenstein-Leopoldshafen, Germany
nicole.ludwig@kit.edu

Dorothea Wagner
Institute of Theoretical Informatics,
Karlsruhe Institute of Technology
Karlsruhe, Germany
dorothea.wagner@kit.edu

ABSTRACT

We introduce a novel approach to demand side management: Instead of using flexibility that needs to be defined by a domain expert, we identify a small subset of processes of e. g. an industrial plant that would yield the largest benefit if they were time-shiftable.

To find these processes we propose, implement and evaluate a framework that takes power usage time series of industrial processes as input and recommends which processes should be made flexible to optimize for several objectives as output. The technique combines and modifies a motif discovery algorithm with a scheduling algorithm based on mixed-integer programming.

We show that even with small amounts of newly introduced flexibility, significant improvements can be achieved, and that the proposed algorithms are feasible for realistically sized instances. We thoroughly evaluate our approach based on real-world power demand data from a small electronics factory.

CCS CONCEPTS

• **Applied computing** → *Industry and manufacturing*; *Computer-aided manufacturing*; • **Mathematics of computing** → Combinatorial optimization; • **Computing methodologies** → Motif discovery;

KEYWORDS

flexibility, smart grid, project scheduling, demand side management, demand response, motif discovery

ACM Reference Format:
Lukas Barth, Veit Hagenmeyer, Nicole Ludwig, and Dorothea Wagner. 2018. How much demand side flexibility do we need? Analyzing where to exploit flexibility in industrial processes. In *e-Energy '18: The Nineth International Conference on Future Energy Systems, June 12–15, 2018, Karlsruhe, Germany.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3208903.3208909

This is a preprint version of https://doi.org/10.1145/3208903.3208909.
e-Energy '18, June 12–15, 2018, Karlsruhe, Germany
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.

1 INTRODUCTION

It has almost become folklore within the energy research community that creating and exploiting demand-side flexibility can and should be a response to the growing amount of intermittent, non-dispatchable generation in future energy systems based on renewable energy sources. A body of literature (e. g. [4, 8, 14, 24]) looks into this from different angles, exploring and demonstrating usefulness, applicability and computational feasibility.

Many of the approaches, especially those employing centralized coordination of *demand side management* (DSM), need to solve an optimization problem that is a particular case of a project scheduling problem and can be formulated as follows. Given a set of processes, each associated with electrical power demand, and given specific flexibility for each process, run each process at the right time such that some objective regarding the total power usage is optimized. The objective, the processes, and the form of the flexibility can take many forms.

However, even though research in this field is plentiful, implementations of demand side management in practice, especially *centralized* DSM, are scarce. There are many reasons for this, not the least important of which is that much of the Smart Grid infrastructure so far only exists on paper, or that the current electricity market has insufficient incentives to provide flexibility.

We believe there are other significant roadblocks for centralized DSM. First, DSM often focuses on households, neglecting that shifting demand in an industrial context would have a stronger impact due to the amount of energy expended. This focus is mainly happening because contrary to households, opportunity costs for shifting demand in industry, i. e., changing a production schedule, can be very high, making most price-based DSM schemes (also called *Demand Response*) infeasible.

Additionally, we have found that managers collecting the data of industrial plants are not always able to specify what the flexibility of their processes is. Designing processes such that they are flexible has not been a focus in the past. Neither has analyzing and documenting the flexibility hidden in existing industrial processes. Thus, confronting plant operators with the option of DSM, we realized that many feel that this is an all or nothing choice and decide that flexibility is just not for them.

Finally, while there are several sets of household consumption data available (such as the REDD data set by Kolter and Johnson [13]), such data is very sparse in the industrial context, additionally impeding research in this area.

We hope that we can overcome all these obstacles with a different approach: Instead of solving scheduling problems where flexibility must be specified a priori per process, we suggest taking industrial plants as they are currently operated and analyzing which processes would yield the most substantial benefit if they were flexible. We hope that such an analysis can be a powerful tool for industrial plant managers in motivating and implementing DSM.

Our Contribution. We propose, implement and evaluate a new framework that takes power consumption time series of industrial processes as input and indicates which processes should be made flexible to optimize for several objectives. With this framework, we want to answer the question: How much flexibility do we need? Thus, given all the processes we have, how many must be made flexible (and by how much) to get improvements regarding the energy consumption. To the best of our knowledge, we are the first to propose this approach to DSM. Based on a recently published set of power consumption time series from an electronics factory, we show that we can achieve notable improvement even with little newly-created flexibility.

Outline. Before describing the proposed technique, we formally define the terminology and problems used throughout this paper in Section 1.1 and summarize related work in Section 2. Then, we describe our approach and its individual steps in Section 3. We evaluate our approach in Section 4 and discuss our results in Section 5, concluding the paper in Section 6.

1.1 Problem Definition

The Flexibilization Project Scheduling Problem (FPSP) is the main problem of this paper. Contrary to usual scheduling problems, we do not start with a set of jobs and ask for a schedule. Instead, we take a schedule plus some limitations on the addable flexibility as inputs. We first formally define a *schedule*:

Definition 1. *Schedule*
A schedule *is a set of n triples* $(c_i, p_i, u_i) \in \mathbb{N} \times \mathbb{N} \times \mathbb{R}$*, for* $i \in \{1, 2, \dots n\}$*. Each triple describes a* job *in the schedule. We identify the triple* (c_i, p_i, u_i) *with job i. For job i, the field* c_i *indicates the (current)* start time *of job i in the schedule,* p_i *indicates the* processing time *(or* duration*) of the job, and* u_i *specifies the amount of power that job i uses during execution.*

Note that for the sake of simplicity, in this definition and throughout this paper, all jobs have constant power demand during their execution. While this certainly is a simplification, we argue in sections 4.1 and A.3 why it is probably an acceptable simplification for many industrial processes, and describe in sections 3.4.3 and A.2 how our approach can easily be adapted to jobs with non-constant power demand, at the expense of computational complexity.

Next, we define the terms in which we talk about flexibility.

Definition 2. *Flexibilization Limits*
Given a schedule as in Definition 1, two integers $\hat{T} \in \mathbb{N}$ *and* $\hat{J} \in \mathbb{N}$ *are used to limit the amount of flexibility that may be created in the schedule. Here,* $\hat{J}$ *specifies how many jobs may be moved away from their original start times, and* $\hat{T}$ *specifies by how much time steps jobs may be moved in total.*

With this, we can now define the problem examined throughout this paper:

Definition 3. *Flexibilization Project Scheduling Problem*
Given are a schedule as in Definition 1 and flexibilization limits as in Definition 2.
Find for each job $i \in \{1, 2, \dots n\}$ *a new start time* $s_i \in \mathbb{N}$ *such that*

- *the number of jobs i for which* $s_i \neq c_i$ *is at most* $\hat{J}$
- *the total deviation from the current start times is at most* $\hat{T}$*, i.e.,* $\sum_{i=1}^{n} |c_i - s_i| \leq \hat{T}$

Any S which satisfies the conditions from Definition 3 is a *feasible* solution to the FPSP problem. Slightly misusing our notation, we also call such an S a *schedule*. While specifying $\hat{T}$ and $\hat{J}$ of course means that to apply our framework one still has to specify these limits on flexibility, flexibility does not have to be specified on a per-job basis. Therefore it becomes easy to explore what improvements we can achieve at the "cost" of what amount of flexibility — as we do in Section 4. Also, we outline in Section 6 how one can truly get rid of having an amount of flexibility as input parameter if one can estimate the costs of introducing new flexibility.

Note that so far we have only defined what a *feasible* solution is, not what makes a solution *optimal*. In fact, we explore different objective functions, the first of which models peak shaving. To do so, we need the total power usage at a certain point in time. For a feasible schedule S, let U_t be the amount of power that is used during time step t, i. e.,

$$U_t = \sum \{u_i \mid s_i \leq t \wedge s_i + p_i > t\}.$$

Definition 4. *Flexibilization Project Scheduling Problem with Peak Shaving (FPSP-PS)*
For an instance of FPSP, let $\hat{U}$ *be the maximum amount of power used by concurrently executing jobs throughout S*

$$\hat{U} = \max_t U_t.$$

The problem is then, find the feasible schedule that minimizes $\hat{U}$*.*

While peak shaving is an objective that is relevant in real-world applications, it is often combined with available *generation*; the objective is to reduce the peaks of power demand which cannot be served by the generation. To model such objectives, we introduce an amount of generation for each time step: Let $G_t \in \mathbb{R}$ units of power be available in time step t. With this, we can define the next possible objective:

Definition 5. *Flexibilization Project Scheduling Problem with Peak Shaving and Generation (FPSP-PSG)*
Given an instance of FPSP and generation G_t*, let* $\tilde{U}$ *be the maximum amount of power used by concurrently executing jobs throughout S which cannot be met by own generation (the peak* residual *load)*

$$\tilde{U} = \max_t (\max(U_t - G_t, 0)).$$

The problem is then, find the feasible schedule that minimizes $\tilde{U}$*.*

Our third and last examined objective looks at *overshoot minimization*. In this setting we try to minimize the total amount of energy that can not be served by available generation:

Definition 6. *Flexibilization Project Scheduling Problem with Overshoot Minimization (FPSP-OM)*
Given an instance of FPSP and generation G_t, find the feasible schedule that minimizes

$$\sum_{t=0}^{\infty} \max(U_t - G_t, 0).$$

Note that the mixed-integer programming approach presented in Section 3.4 is able to optimize for all these objectives. The modelling approach presented is based on the technique introduced in Barth et al. [4], where the authors show how to adapt to various objective functions.

2 RELATED WORK

The need for more flexibility in energy usage has been well established (e. g. in [24]), for an analysis of the benefits see e. g. Strbac [23] or Feuerriegel and Neumann [8]. Incentives and strategies to use more flexibility are usually summarized under the term demand side management. For our purpose, we focus on what Palensky and Dietrich [19] define as *shifting demand*. Hence, we move the jobs in time; if we decide not to run the job now, it has to run later.

The existing literature on DSM, in general, is vast and spans a significant area of applications and problems. For example, Gong et al. [9] investigate how DSM can be used for households while still preserving their privacy, a profoundly important question which also results in the mentioned lack of support by industry to participate in such measures. Additionally, DSM has been looked at not only for households but also for example for data centers e. g. by Klingert et al. [12]. However, we found the resulting schemes not to be transferable to industrial customers of the kind we consider. In contrast to our approach, Zehir et al. [28] focus on getting small customers to participate in demand response, where the machines they can change are more related to that of households and not of manufacturing customers.

Although there are studies concerned with how much flexibility can be provided by the consumers [7], even on a device level [26], and how much this flexibility is worth [1, 8, 21, 25], we are not aware of anyone investigating what amount of demand side flexibility is needed to improve the energy consumption significantly, especially not for industrial customers. Furthermore, many papers schedule flexible demands (for an extensive overview see Barth et al. [4]) without looking into how many of those demands need to be changed by the scheduler to improve the energy pattern.

Peak minimization is one of the primary goals for many applications, e. g. Liu et al. [16] and Zhao et al. [29] schedule loads to avoid specific peak demands, the first for data centers, the latter for electric vehicle charging. In our formulation, we focus on scheduling non-preemptive and deferrable loads, as do for example O'Brien and Rajagopal [18].

To find the patterns in the industrial load time series, we use motif discovery. There exist other algorithmic techniques to find starting processes and monitor appliances, most prominently non-intrusive load monitoring [11], and also e. g. Ardakanian et al. [2] and Rollins and Banerjee [22]. However, these methods are not applicable in our case, mainly due to the lack of labels in our data and thus the need to work without supervision.

Our approach also touches the field of project scheduling. Research in this area is vast, many variations of problem settings have been explored. For an extensive survey, see Weglarz [27].

3 THE FRAMEWORK

In this section, we describe all steps of our proposed framework in the order in which we apply them, as well as its evaluation. We start by describing the input data that we use for evaluation in Section 3.1. We then present a motif discovery algorithm which we use to detect the individual industrial processes from the usage data in Section 3.2. Not strictly part of our framework, but necessary for our evaluation is the generation of synthetic test data. We do this so that we can evaluate our approach on more than one data set. We explain the generation of synthetic data in Section 3.3. On the discovered processes (resp. our synthetic data), we run a model-based scheduling algorithm, which we present in Section 3.4.

For the sake of clarity, we present some terminology first. The initial data is a set of electrical consumption time series from *machines*, one time series per machine. We assume that a machine can either be running a *process* or be idle. The part of a machine's time series where a process is running is denoted a (time series) *sequence*. If several such sequences are similar, we assume they describe the same process. In this case, we say they are *occurrences* of the same *motif* (we explain this in more detail in Section 3.2). The mean motifs are then used to generate synthetic test data, which are sets of *instances*, each consisting of *jobs*.

3.1 Data

Our input data is the HIPE dataset [5], which is gathered at a small-scale electronics factory operated by the Institute for Data Processing and Electronics[1] at KIT. It is a set of time series of the apparent power in kVA of ten machines, which range from soldering furnaces to pick-and-place machines. The data was gathered over almost a year, from December 2016 to October 2017, with sub-minute resolution. See Section A.1 on how to get access to the raw data as well as the instances we create from the data set, as well as a more in-depth description of the raw data.

Some of the machines are frequently running in standby mode. Consequently, their power is above zero even while no process is running. To ease the analysis later and only consider the running processes without the standby times, we distinguish between an *active state* and a *passive state* of those machines. The active state means the machine is running a process, while the passive state means the machine is either off or in a standby mode. We determine the two states for each machine with the help of a k-means clustering algorithm, with which we cluster the power demands of each machine individually. Setting $k = 2$ leaves us with a cluster for each state. As we are only interested in the active state of the machine, we set the power demand to zero for all points in the passive cluster.

The points in time where a series goes from zero to non-zero power demand are the *start points* of a *sequence*. The sequence

[1] https://www.ipe.kit.edu/english/index.php

ends when the power demand goes back to zero, at its *end point*. To be able to compare sequences in a meaningful way later, we normalize the lengths of all sequences on a per-machine basis. For each machine, we determine the 80% quantile of the lengths of its sequences. We scale all sequences of this machine to this length for our motif discovery. We chose 80% because, on the one hand, stretching sequences comes with less data loss than compressing. On the other hand, taking the longest sequence would increase the impact of outliers.

3.2 Motif Discovery

Given the data sequences describing the machines power intake, we now use motif discovery to find recurring patterns in these sequences [17]. In the following we only summarize the algorithm briefly, for a proper introduction and mathematical detail, we refer to the works of Chiu et al. [6] and Lin et al. [15].

The motif discovery algorithm we use is based on *symbolic aggregated approximation* (SAX) of time series. SAX uses words to represent the time series at hand. To get to a word representation, we first discretize the power levels in the time series. Each discrete power level of the time series is represented by a letter. For every sequence, we then assign to every time step of the sequence the letter for the power level that the sequence has at the respective time step. The resulting string of letters forms the word by which we represent the sequence.Each sequence is thus described by one word, where the word size for each machine is the normalized length established above. The association of the discrete levels with letters is equiprobable, given a predefined alphabet size from which we draw the letters. A higher alphabet size is usually helpful if the time series at hand has larger variations.

To compare the sequences, we compare their word representations with each other. Those sequences which can be represented by similar words are considered as belonging to the same motif. Each sequence associated with a motif is then an *occurrence* of this motif, where each motif consists of at least two occurrences. In the following, we use the mean of these occurrences as the representation for each motif.

Figure 1 shows a discovered motif with all its occurrences in gray lines. The colored lines represent the mean, 20% quantile and 80% quantile of all the occurrences. These lines can give a feeling for how a *standard* process might look like for this machine. We show all discovered motifs in Section A.6.

3.3 Generation of Synthetic Instances

To evaluate the feasibility and usefulness of our proposed approach, we need many instances of the FPSP problem which emulate real-world processes. Therefore, we generate artificial instances for the FPSP problem from the motifs discovered as described in Section 3.2. In our generated instances, the start times, job durations and power requirements are statistically derived from the discovered motifs. These characteristics are key factors with regards to peak power demands during a schedule. Since we preserve these characteristics of the discovered motifs, we expect our generated instances to adequately resemble reality.

To this end, we describe every motif by three normal distributions: One energy distribution, one start time distribution, and one duration distribution. Each of these distributions is determined by fitting a normal distribution to the lengths, start times and energy consumptions of the respective motif's occurrences. The start time distribution is actually a mixture of normal distributions: We assume that the same process might have several times within a day at which it usually starts. To factor this in, we cluster the start times of every motif's occurrences (using affinity propagation) and generate a normal distribution for every cluster. We then create a mixture distribution for that motif's start times, weighting each normal distribution by the size of its cluster. However, for most motifs, only a single cluster was found.

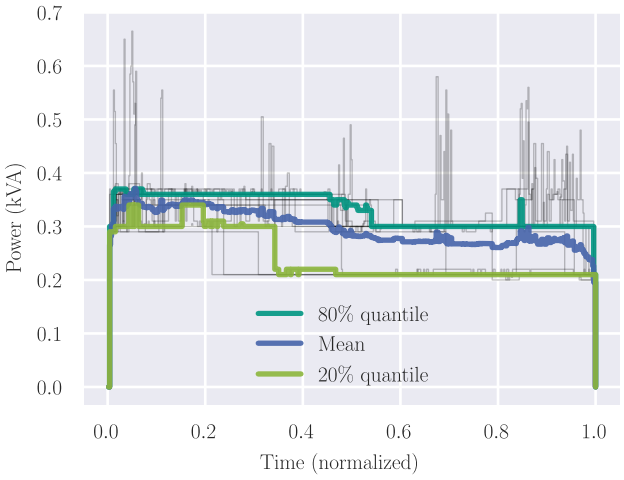


Figure 1: The discovered motif A. Each gray curve shows one occurrence, with the length of all occurrences being normalized to 1. Their shared common features, roughly represented by the mean and upper and lower quantiles, constitutes the motif.

We generate instances (i. e., sets of jobs) with a fixed number of jobs by repeatedly randomly selecting a motif (weighted by the number of the motif's occurrences) and then generating a job for this motif. For each job, we generate a duration by randomly drawing from the respective motif's length distribution. We discard any length of less than one time step. Similarly, we determine a start time for this job by randomly drawing from the respective motif's start time distribution. Finally, we randomly generate an energy consumption for the job by drawing from the motif's energy distribution. The job's power demand is set as energy consumption divided by duration.

Since drawing values from normal distributions may yield extreme results in few cases, which nonetheless are sufficient to substantially skew the complexity and results of the scheduling problem, we discard any values that deviate from the mean by more than three times the standard deviation.

3.4 Scheduling

We now describe a mixed integer program (MIP) that models and optimizes the FPSP problem. We base our MIP on the modeling technique from [4], which is intended for classic smart grid scheduling problems, i. e., assumes jobs given with fixed flexibility, expressed in terms of earliest starts, deadlines, etc. The approach is able to model many real-world processes' features encountered in literature, such as ramping, energy drain or interdependent jobs. We only give a very rough overview over the most important parts of the technique, and then describe how to extend it to cope with the FPSP problem.

The MIP technique in [4] is based on a discrete-time formulation for the Resource-Constrained Project Scheduling Problem. At the core of the model, a binary variable is created for every job and every time step at which this job could start. For every job, exactly one of its start indicator variables must be one, fixing the start time of the respective job. The start time variable of a job i is σ_i, and its finishing time is η_i. Variables for the power usage at every time step can be built using the start indicator variables.

The foundation for our modification of this technique are *window extensions*. These are a way of expressing the current start times c_i and the flexibilization limits from FPSP in terms of a "classic" project scheduling problem, which works with *release times* and *deadlines* instead.

Instead of every job having a desired start time c_i, we assume every job i to have a release time r_i (i. e., an earliest start time) and a deadline d_i (i. e., a time when the job must be finished).

We transform an FPSP problem by first setting $r_i = c_i$ and $d_i = c_i + p_i$ for all jobs i. This way, we get a problem that we can immediately plug into the MIP technique from [4], and in which every job i is forced to be started at its current start time c_i, because we have made its *window* (the time between release and deadline) small enough. In a second step, we now extend the MIP to allow for some jobs to be executed outside their window, i. e., to *extend* their window. This window extension is tailored such that it honors $\hat{T}$ and $\hat{J}$ of the FPSP instance.

In the MIP framework, the jobs' windows are enforced with the simple constraints[2]

$$\sigma_i \geq r_i \qquad \forall i$$
$$\eta_i \leq d_i \qquad \forall i.$$

We introduce two new variables per job i, namely $\overleftarrow{x_i}$ and $\overrightarrow{x_i}$, the *left window extension* and *right window extension* of job i, respectively. We then change the aforementioned constraints to

$$\sigma_i \geq r_i - \overleftarrow{x_i} \qquad \forall i \tag{1}$$
$$\eta_i \leq d_i + \overrightarrow{x_i} \qquad \forall i. \tag{2}$$

The additional constraints to uphold $\hat{T}$ are simple

$$\overleftarrow{x_i} \geq 0 \quad \wedge \quad \overrightarrow{x_i} \geq 0 \qquad \forall i, \tag{3}$$
$$\sum_{i=1}^{n} \left(\overleftarrow{x_i} + \overrightarrow{x_i} \right) \leq \hat{T}. \tag{4}$$

To also uphold the job move limit $\hat{J}$, we need to introduce a binary variable indicating that a job was *not* moved. We will call this variable $\tilde{c}_i$. We force $\tilde{c}_i$ to become zero if job i was moved with this constraint

$$\tilde{c}_i \in \{0, 1\} \qquad \forall i, \tag{5}$$
$$\tilde{c}_i \leq 1 - \frac{\overleftarrow{x_i} + \overrightarrow{x_i}}{\hat{T}} \qquad \forall i. \tag{6}$$

With this, it is easy to limit the number of moved jobs

$$\sum_{i=1}^{n} \tilde{c}_i \geq n - \hat{J}. \tag{7}$$

Note that with constraint 6, we get $\hat{T}$ as a coefficient in the constraint matrix, potentially giving us constraint coefficients of greatly varying magnitude. This can cause problems for the numeric stability of the resulting MIP model, as we notice in Section 4.4.

3.4.1 Modelling further Constraints. Real-world scenarios will most likely require more constraints to be placed on jobs than what we model in this paper. However, since the MIP framework from [4] is very flexible, many additional constraints are easy to include. As an example, the MIP framework handles dependencies between jobs (in the form of *time lags*), energy drain for postponed jobs, and hard release times and deadlines.

3.4.2 Modelling individual Move Costs. In realistic scenarios, moving some jobs in time might be way more effort than moving other jobs. The model can account for this with a slight modification: We introduce a weighting factor $w_i \in \mathbb{R}$ for every job i. We then modify Constraint 4 to:

$$\sum_{i=1}^{n} w_i \left(\overleftarrow{x_i} + \overrightarrow{x_i} \right) \leq \hat{T} \tag{8}$$

This way, moving some jobs counts stronger towards the $\hat{T}$ limit than others.

3.4.3 Modelling Fluctuating Power Demands. As noted in Section 3.3, all jobs are assumed to have constant power demand. However, as the authors outline in [4], the MIP technique can approximate jobs with non-constant power demand. This approximation is accomplished by using the feature of time lags mentioned above, i.e., constraints on the order in which jobs are scheduled. By using positive and negative time lags, it is possible to chain a series of jobs together such that they must always be executed consecutively in a fixed order and without any interruptions between them. To approximate a job with non-constant power demand, we could use such a chain, each job of the chain having constant power demand. The more jobs we chain together to represent a single process, the closer we can approximate the process' fluctuating power demand.

[2] Note that in [4], the deadline is denoted as D_i instead of d_i.

We give a detailed explanation of this technique in Section A.2 in the appendix.

4 EVALUATION

Having introduced our framework and the data with which we work, we now evaluate our approach. We start with describing the motifs we find, and the instance sets generated, before assessing each of our problem sets individually.

4.1 Discovered Motifs

Given the nine machines, we find a total of 15 motifs in our time series data. The resulting motifs for each machine can be found in Figure 18 in the appendix. We have also included an overview over our parameters used in Table 4. As we can see there, most of the motifs are block-shaped. This fact gives us reason to believe that many processes can be adequately approximated by our assumption of constant power demand. For an in-depth discussion of this assumption see Section A.3. The parameters for the motif discovery algorithm in this paper are tailored to our specific problem at hand. For example, we use a relatively small alphabet size for most machines as their variations are small. We also choose all parameters in such a way that the algorithm can classify most sequences without assigning all of them to a single motif. All sequences which are not classified as belonging to one of the motifs are classified as noise and excluded from further analysis. In future work, we might want to do an extensive evaluation of our parameter choices. However, we expect the settings we have chosen to be sufficiently useful for our problems at hand.

4.2 Instance Sets

We generate several sets of instances from the data as described in Sections 3.1 through 3.3, resembling the input data from Section 3.1 to varying degrees. We publish all the instance sets together with a description of the data format, see Section A.1.

Set PS-Nonuniform. This set is the first set of instances with which we evaluate FPSP-PS. We proceed like described in Section 3.3: Job lengths, job power demands and job start times are generated from the normal distributions fitted to the discovered motifs. Since in realistic scenarios, the optimization horizon likely is more than one day, we generate instances that span five days. The start times of our discovered motifs are times within a day. Thus, for every job, we not only pick a start time from the motif's start time distribution but also pick uniformly at random at which of the five days the job starts. One time step corresponds to five minutes. We generate instances with 150 jobs each.

Regarding the possible choices for $\hat{T}$, the amount of total job movement allowed, it seems reasonable to specify $\hat{T}$ relative to the instance size. We therefore introduce Θ and set $\hat{T} = \Theta \cdot \sum_i p_i$. Here, Θ specifies the fraction of cumulative duration that jobs may be moved. We test all of $\Theta \in \{0.005, 0.01, 0.02, 0.03, 0.04\}$. Table 1 shows statistics about the values that result for $\hat{T}$ for the various values for Θ. We see that the values for $\hat{T}$ range from about 3.5 hours to about 30 hours.

For $\hat{J}$, the number of jobs allowed to be moved, we investigate all of $\hat{J} \in \{3, 6, 9\}$, for a total of 15 different flexibilization limits.

	$\hat{T}$ (hours)	
Θ	Mean	Std. Dev.
0.005	3.6	0.20
0.01	7.2	0.39
0.02	14.3	0.77
0.03	21.5	1.16
0.04	28.7	1.54

Table 1: Statistics of $\hat{T}$ values for all possible values of Θ.

We generate 30 sets of 150 jobs each, and pair them with every flexibilization limit from above, leading to a total of 450 instances.

Set PS-Uniform. As we see in our evaluation (cf. Section 4.4), the heterogeneity in power demand between the generated jobs (arising from heterogeneous power demand in the discovered motifs) has a significant influence on the computational feasibility and the possible optimization benefits of the instances. We do not wish to bias our conclusions on the basis of this phenomenon, which may not occur in other workloads. Hence, we generate the PS-Uniform instance set in which we use a single, fixed normal distribution for all jobs' power demands (with mean 30 and standard deviation 10). Aside from this, we proceed as for the PS-Nonuniform set.

Set PSG. In the PSG set, we again explore peak shaving, but with fluctuating generation. This set corresponds to a setting where e. g. solar generation is available and one tries to minimize the peak residual load, which corresponds to FPSP-PSG.

We use a solar generation curve for one day derived from total solar generation data for Germany, Austria, and Luxembourg with quarter-hourly time resolution, which was retrieved from ENTSO-E.[3] For every quarter hour, we average the production from all summer days in the year 2016. In our instances, we set available generation (i. e., G_t) on all five days based on this curve, scaling the curve such that in total, 20% of the total energy demand in each instance is provided via solar energy. Aside from that, we generate instances as described for the PS-Nonuniform set.

Set OM. With the OM set, we evaluate an overshoot minimization objective, i. e., the FPSP-OM problem.

For the available generation, we assume that we can meet 65% of the total energy requirement in each instance by own generation. This number results from our calculations of the energy consumption and production in the summer month of BASF based on [10]. As BASF's power plants are steam-controlled, the generation in the winter months is so high that they can sell excess energy. However, during the summer months, the opposite is true, and they have to buy energy from the grid, which is more expensive. We assume the generation to be a flat curve in our calculations. For a steam-controlled power plant and a time horizon of five days, this is a realistic assumption since steam demand usually fluctuates relatively little. Formally, we set

[3] Via Open Power Systems Data:
https://data.open-power-system-data.org/time_series/

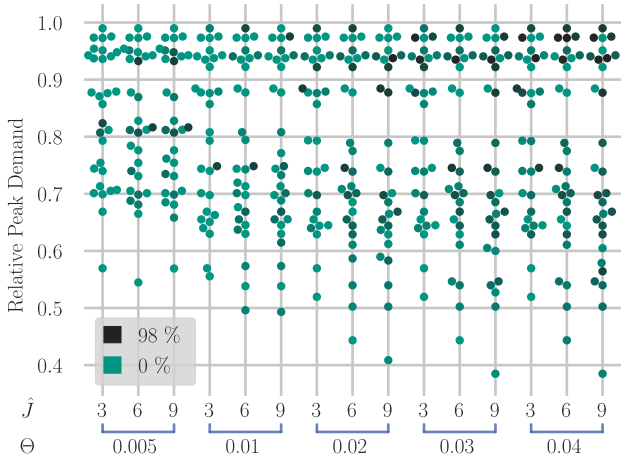


Figure 2: Relative reduction in peak demand for the PS-Nonuniform set, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y-axis indicates the change in peak demand after optimization. Color indicates the remaining MIP gap when the optimization was stopped. Results are summarized in Table 8.

$$G_t = \frac{\sum_i (p_i \cdot u_i) \cdot 0.65}{5 \cdot 24 \cdot 60/5} \qquad \forall t.$$

For this instance set, we use the objective from FPSP-OM, but proceed as for the PS-Nonuniform set otherwise.

4.3 Evaluation Environment

For all instances, we build a MIP model according to Section 3.4. We optimize every model using Gurobi 7.0. We also evaluated optimizing using CPLEX 12.8, but we achieve slightly better results using Gurobi for our models. We use a system with dual AMD EPYC 7601 CPUs (having 64 physical CPU cores) with 512 GB of RAM. We optimize every model for 45 minutes, allowing for 20 threads per solver, running 6 solvers in parallel.

MIP Gap. In the following, we report the MIP gaps after optimization together with the solution quality. The MIP gap is the difference between the best feasible solution found and the best lower bound that the solver was able to prove, divided by the best feasible solution. A large MIP gap is an indicator that further optimization could potentially find better solutions. Please note that we optimize with a focus on finding high-quality solutions.[4] Thus, we could have further reduced MIP gaps at the cost of solution quality. We performed parameter tuning via Gurobi's auto-tuning tool. However, the default settings produced the best results for us.

4.4 Evaluation of FPSP-PS and FPSP-PSG

We start by looking at the sets that represent peak shaving objectives. Figure 2 summarizes our results for the PS-Nonuniform set. Every dot represents one instance. Every column of dots represents one combination of $\hat{J}$ and Θ. The y-coordinate of the dot indicates the respective instance's peak demand after optimization divided by the peak demand before optimization, which we define as *relative peak demand*. We use the dots' colors to indicate the MIP gap achieved for the respective instance, where darker colors indicate larger MIP gaps, i. e., worse optimization, and the lightest color indicates that the instance was solved to optimality. Table 8 in the appendix reports numerical results. In Figure 2 we see that in the most extreme cases, peak demand is reduced by more than 60%. We also see that the improvement is mostly distributed evenly between 0% and about 40%, and that increasing $\hat{J}$ or Θ does not result in significant improvements. Figures 14b and 14a (in the appendix) give an insight how the peak demand for every instance changed when increasing Θ or $\hat{J}$, respectively. We find that increasing Θ yields larger peak reductions than increasing $\hat{J}$, and that improvements gradually diminish with larger values for Θ (resp. $\hat{J}$). However, the optimization gets harder with increasing Θ. Since the reported MIP gaps after optimization were large for many instances for $\Theta \in \{0.03, 0.04\}$, results may improve for those parameters if one optimizes the instances further. Figure 11 (in the appendix) shows the MIP gaps of every instance.

We perform a statistical significance test (Wilcoxon's signed-rank test[5]) on our findings. For every consecutive pair of $\hat{J}$ (resp. Θ) values, while keeping the Θ (resp. $\hat{J}$) value fixed, we compute the p-value, which indicates how likely it is that the change in improvements is random happenstance instead of an effect of altering $\hat{J}$ (resp. Θ). We report all values in Table 2.

We want to assume significance with 95% confidence, i. e., say that a change is significant if the p-value is below 0.05. However, throughout the whole of this paper, we perform a total of 88 such tests. With an error probability of 5%, we thus expect erroneously reporting significance for four tests. To compensate for this, one can apply a Bonferroni correction, which essentially means assuming significance only when the p-value is below $0.05/88 \approx 0.00057$.

We see that changing Θ from 0.02 to 0.03 and 0.04 does probably not result in significant improvements for the PS-Nonuniform set. This lack of improvement might be because the optimization problem becomes too hard, but could also be because we already achieve optimal results for many instances with $\Theta = 0.02$ (see below). Aside from that, the only non-significant change is changing $\hat{J}$ from 6 to 9 at $\Theta = 0.005$ and $\Theta = 0.04$, which is like because the little possible movement in time can already be optimally distributed over six jobs, resp. because the instances got too hard.

The very uniform distribution of improvements between 0% and 40% poses the question for validity, hence we looked into characteristics of the instances that allow for an unusually large or small improvement. We discovered that instances which allow for almost no improvement always contain jobs with very high power demands compared to all other jobs' power demands, i. e., substantial heterogeneity in power demands. This makes sense, since the job with the largest power demand (which we call the *tallest* job) is a lower bound for the overall peak power demand. The margin

[4] We set the MIPFocus parameter to 1 for Gurobi.

[5] Because we have many ties in our data, the way such ties are handled is important in our case. We use the approach suggested by Pratt [20].

	3	→	6	→	9
0.005		$< 10^{-4}$		0.00089	
↓	$< 10^{-4}$		$< 10^{-5}$		$< 10^{-5}$
0.01		$< 10^{-4}$		0.00051	
↓	$< 10^{-4}$		$< 10^{-4}$		$< 10^{-5}$
0.02		$< 10^{-4}$		0.00039	
↓	0.0018		0.014		0.0032
0.03		0.00029		0.00014	
↓	0.012		0.047		0.035
0.04		$< 10^{-4}$		0.0018	

Table 2: p-Values for the change of one parameter in the PS-Nonuniform set. Values highlighted in green indicate that changing one of $\hat{J}$ and Θ, while keeping the other one constant, results in a statistically significant change in improvements. Values in blue are significant only before Bonferroni correction.

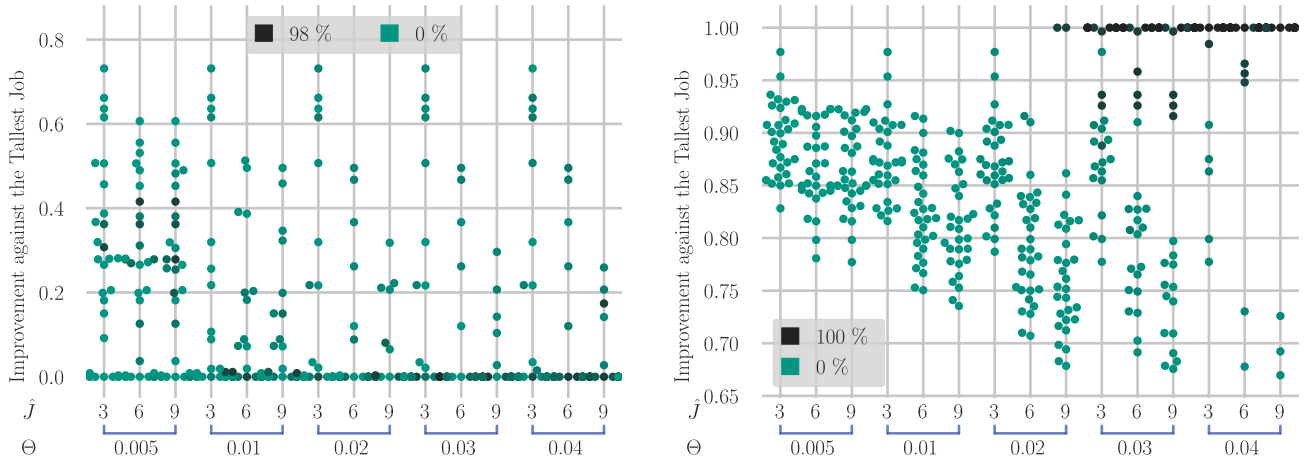
for optimization is at most the difference between the overall peak power demand and the demand of the tallest job.

We therefore also evaluate how well our optimization performs within this margin, i. e., how the difference between peak demand and demand of the tallest job changes, which we define as the *improvement against the tallest job*. Figure 3a shows the results. A value of 0 indicates that after optimization, the overall peak power demand equals the demand of the tallest job. Figure 3a shows that for the majority of instances, we are in fact able to achieve this optimum. For all other instances, we improve by at least 20% within the margin between tallest job and original peak demand.

Since the peak demand being dominated by single jobs seems like a peculiar property of our instances, we create the PS-Uniform set, where we compensate for this characteristic. Regarding the results for the PS-Uniform set, shown in Figure 4 (numerical results in Table 9 in the appendix), we see that even with power demands being drawn from a single distribution, peak demand reductions of 5% to 30% are realistic. We also see that solving these instances is a lot harder than the instances from the PS-Nonuniform set. We report the MIP gaps in Figure 5. Here, an interesting trend can be seen: While almost all instances for $\Theta \leq 0.02$ could be optimized to within 20% gap, often to optimality, the MIP gaps for $\Theta \geq 0.03$ are mostly above 60%. Thus, it seems like the computational complexity grows rapidly with $\hat{T}$ — this is certainly because of the larger solution space, but might also be exacerbated by the numerical stability issues with constraint 6 mentioned in Section 3.4.

The results for the PS-Uniform set are more tightly clustered, which makes sense since the instances are more similar to each other. For this set, we also evaluate how well we optimize within the margin between overall peak power demand and demand of the tallest job, which we report in Figure 3b. Since in PS-Uniform, the peak power demand is not dominated by single jobs anymore, this now correlates closely with the absolute improvement. We again perform significance analysis as for the PS-Nonuniform set, the results of which we report in Table 5 (in the appendix). We can see that some of the changes that were not significant for PS-Nonuniform, especially increasing Θ, have now become significant. This change supports the assumption that the non-significance in these cases for the PS-Nonuniform set is because optimal values have already been achieved for $\Theta = 0.02$.

The final instance set about peak shaving is the PSG set, in which we assume solar generation. We report the improvements in Figure 6 and numerical results as well as the effect of changing both parameters in Table 10 and Figure 16 (both in the appendix). We see that the distribution of improvements is similar to the PS-Nonuniform set. We can assume that tall jobs again have a large impact in this instance set. However, the absolute values regarding improvement are much better than for PS-Nonuniform: In most extreme cases, we are able to reduce the residual peaks by almost 80%. Most improvements are evenly distributed between 5% and 50%.



(a) Set PS-Nonuniform **(b) Set PS-Uniform**

Figure 3: Change of the difference between the peak demand and the demand of the tallest job after optimization.

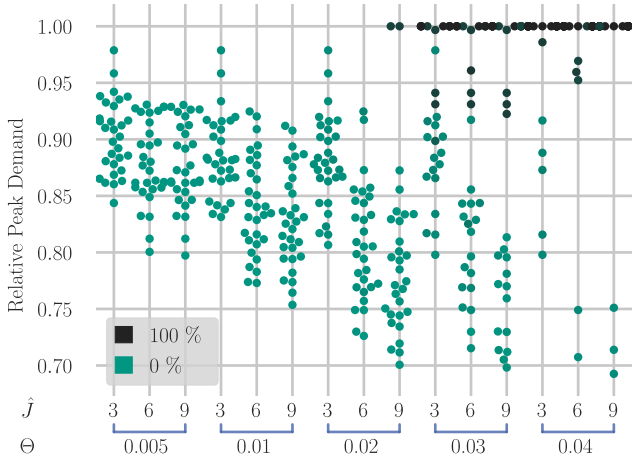


Figure 4: Relative reduction in peak demand for the PS-Uniform set, one point per instance. Columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in peak demand after optimization. Color indicates how well the instance could be optimized. Results are summarized in Table 9.

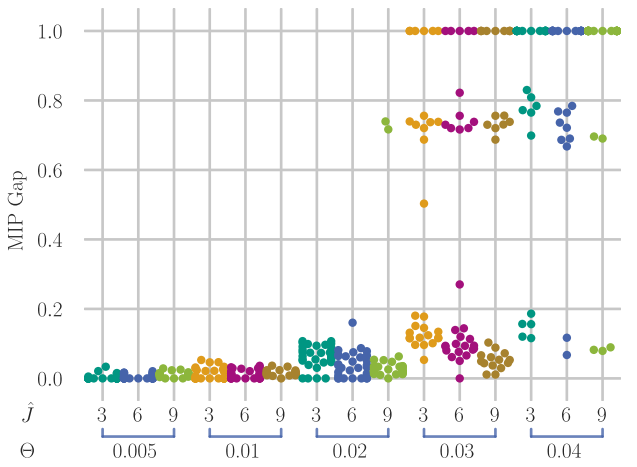


Figure 5: MIP gaps for the various settings of $\hat{J}$ and Θ in the PS-Uniform instance set. Every dot corresponds to one instance. Colors are used to distinguish the columns.

This seems plausible, since in the PSG set, (residual) peak reduction can not only be achieved by avoiding concurrent execution of jobs, but also by moving jobs towards the peaks of the solar generation curve.

We report MIP gaps in Figure 12 in the appendix, which are not worse than for the PS-Nonuniform set. Significance values are reported in Table 6. We see that almost all changes in parameter choice lead to significant improvements. Overall, our approach

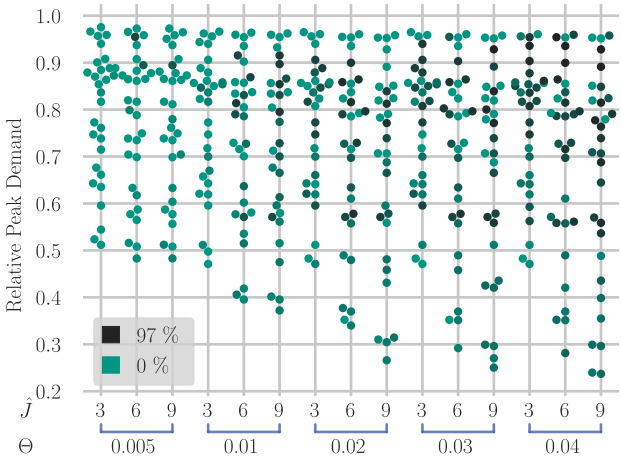


Figure 6: Relative reduction in peak demand for the PSG set, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in peak demand after optimization. Color indicates the remaining MIP gap when the optimization was stopped. Results are summarized in Table 10.

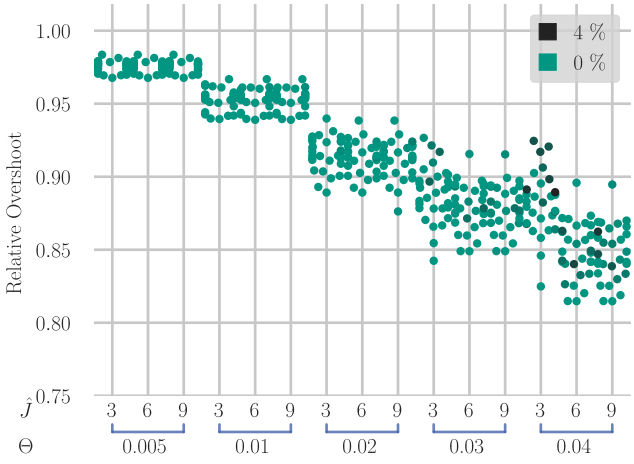


Figure 7: Results for set OM, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in overshoot after optimization. Color indicates how well the instance could be optimized. Results are summarized in Table 11. For better readability we removed one outlier at $\Theta = 0.01, \hat{J} = 3$ with a value of ca. 0.5.

seems to be able to exploit the benefits of a given generation curve without increasing the computational complexity.

4.5 Evaluation of FPSP-OM

We evaluate the overshoot minimization objective from FPSP-OM with the OM instance set, the results of which we report in Figure 7

and Table 11 (in the appendix). We see strong clustering of the results, indicating that with, e. g., $\Theta = 0.02$, one can expect the amount of energy to overshoot generation, i. e., the amount of energy that must be bought, to decrease between 6% and 12%. For FPSP-OM, apparently $\hat{T}$ plays a crucial role, while $\hat{J}$ yields only minor improvements. These observations can be seen from figures 17a and 17b. Computational complexity increases slightly with decreasing $\hat{J}$. However, the reported MIP gaps (see Figure 13 in the appendix) are drastically smaller than for our peak shaving sets. We solve all instances to at most 4% MIP gap, and in fact, solve most of them to optimality. We again do a significance analysis, reported in Table 7. Here, we see that almost all parameter changes result in significant improvements.

5 DISCUSSION

For all examined variations of the FPSP problems, we can show that with a relatively small amount of flexibility, significant improvements in the target metric can be achieved. Since all our test data is founded on real energy consumption data obtained from a factory, we assume our results to apply to real-world scenarios. However, real industrial processes come with more constraints than we were able to respect within the scope of the present paper. Since our optimization is based on an MIP framework (which supports additional constraints such as process dependencies etc.), many additional constraints should be straightforward to model.

We discovered that the possible improvements depend a lot on the heterogeneity of the process' power demands. However, even for the heterogeneous instance sets derived from our real-world data, possible improvements were promising.

For the problem variants that assume available generation (FPSP-PSG and FPSP-OM), we need to choose the amount of generation. While we could obtain a solar generation curve from real-world data, we need to fix the total amount of energy available via generation somewhat arbitrarily. However, we have no reason to believe that our approach works significantly better or worse if we choose this amount differently. We were able to show that our approach is in fact suitable to reduce the peak residual demand as well as the amount of energy that must be bought from the grid.

Another discrepancy between our test instances and real-world processes is the fact that we assume the power demand of each process to be constant over time. However, we have argued in sections 4.1 and A.3 why this is probably a reasonably close approximation of the motifs we discovered, which is why we believe that we can even use constant-demand jobs as an approximation for many real industrial processes. Furthermore, we have shown in sections 3.4.3 and A.2 how to get around this limitation.

5.1 Optimization Aspects

Regarding the computational complexity of optimizing the MIPs of our approach, we can see that the most important parameter (besides the instance size) is $\hat{T}$. This importance for the complexity is because, for a large $\hat{T}$, the MIP formulation needs to create many start-indication variables ($s_i(t)$ in the original framework), quickly increasing the size of the underlying MIP problem. Also, the numerical stability of the model probably suffers from large $\hat{T}$ values, as noted in Section 3.4.

We also discover that substantial heterogeneity of power demands, while decreasing possible peak improvements, is beneficial for the complexity of the optimization problem.

Given all that, most of the time we can optimize instances of realistic sizes, spanning a whole working week with five-minute resolution, within 45 minutes to acceptable MIP gaps. We therefore think that our approach can not just be beneficial in reality, but can also be applied to realistically sized problems.

6 CONCLUSION & OUTLOOK

We have shown a technique that can be used to guide flexibilization efforts in industrial processes. Our technique starts with consumption data obtained from the current operation of an industrial plant, and it ends with an indication which processes would be most beneficial if they were more flexible. We have shown that our technique can lead to significant improvements and should be applicable in real-world industrial processes.

Regarding our initial question, how much demand side flexibility do we need, our framework helps to understand which processes would benefit most from being flexible. Additionally, our results show that there is currently no need to flexibilize all processes. Starting with only a few small changes in the operation of the machines can already improve the energy consumption. This comparatively little necessary effort gives us hope that more flexibility in industrial processes is achievable and not a daunting prospect for any process manager. On the other hand, future work is needed to verify that our motif discovery technique really detects realistic workloads. For that, cooperation with domain experts is necessary.

In the future, we can think of several extensions of our approach. While we currently require fixed limits to be set for the amount of new flexibility to create (in terms of $\hat{T}$ and $\hat{J}$), it would be straightforward to allow for a weighting between improvement in peak demand (or overshoot) and the required new flexibility by making $\hat{T}$ and $\hat{J}$ variables and including them in the objective function. However, this requires a reasonable estimate of the (financial) costs of adding flexibility to processes compared to the costs associated with peak demand or overshoot. Also, it would be easy to price the flexibilization on a per-job basis in the objective, to account for some processes to be more costly to make flexible than others.

From an algorithmic perspective, one should look into finding efficient heuristics for optimizing problems of the FPSP family, to enable the application of our approach to large-scale industrial processes. Also, incorporating uncertainties into the model should be a future step that would be beneficial for practical relevance.

ACKNOWLEDGMENTS

This work was supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation. Thanks to the Institute for Data Processing and Electronics at the Karlsruhe Institute of Technology for supplying their data and special thanks to Simon Bischof, Holger Trittenbach, Michael Vollmer and Dominik Werle for preparing the data and helping us understand it.

REFERENCES

[1] Mirjam Ambrosius, Veronika Grimm, Christian Solch, and Gregor Zottl. 2016. Investment incentives for flexible energy consumption in the industry. In *2016 13th International Conference on the European Energy Market (EEM)*. IEEE, 1–5. https://doi.org/10.1109/EEM.2016.7521234

[2] Omid Ardakanian, Ye Yuan, Roel Dobbe, Alexandra von Meier, Steven H. Low, and Claire Tomlin. 2016. Event Detection and Localization in Distribution Grids with Phasor Measurement Units. *CoRR* abs/1611.04653 (2016).

[3] Lukas Barth, Veit Hagenmeyer, Nicole Ludwig, and Dorothea Wagner. 2018. Dataset accompanying "How much demand side flexibility do we need? - Analyzing where to exploit flexibility in industrial processes". KITOpen Repository. (2018). https://doi.org/10.5445/IR/1000082194

[4] Lukas Barth, Nicole Ludwig, Esther Mengelkamp, and Philipp Staudt. 2018. A comprehensive modelling framework for demand side flexibility in smart grids. *Computer Science - Research and Development* 33, 13 (2018), 1865–2042. https://doi.org/10.1007/s00450-017-0343-x

[5] Simon Bischof, Holger Trittenbach, Michael Vollmer, Dominik Werle, Thomas Blank, and Klemens Böhm. 2018. HIPE – an Energy-Status-Data Set from Industrial Production. In *Proceedings of ACM e-Energy (e-Energy 2018)*. ACM, New York, NY, USA. https://doi.org/10.1145/3208903.3210278

[6] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. 2003. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Ted Senator (Ed.). ACM, New York, NY, 493–498. https://doi.org/10.1145/956750.956808

[7] Reinhilde D'hulst, Wouter Labeeuw, Bart Beusen, Sven Claessens, Geert Deconinck, and Koen Vanthournout. 2015. Demand response flexibility and flexibility potential of residential smart appliances: Experiences from large pilot test in Belgium. *Applied Energy* 155 (2015), 79–90. https://doi.org/10.1016/j.apenergy.2015.05.101

[8] Stefan Feuerriegel and Dirk Neumann. 2016. Integration scenarios of Demand Response into electricity markets: Load shifting, financial savings and policy implications. *Energy Policy* 96 (2016), 231–240. https://doi.org/10.1016/j.enpol.2016.05.050

[9] Yanmin Gong, Ying Cai, Yuanxiong Guo, and Yuguang Fang. 2016. A Privacy-Preserving Scheme for Incentive-Based Demand Response in the Smart Grid. *IEEE Transactions on Smart Grid* 7, 3 (2016), 1304–1313. https://doi.org/10.1109/TSG.2015.2412091

[10] Veit Hagenmeyer, Heinz Langner, and Werner Hartwig. 2014. Eine Methode zur Bewertung der Energieversorgungssicherheit von komplexen Produktionsstätten. In *VGB-Fachtagung: Dampferzeuger, Wirbelschichtfeuerungen, Industrie- und Heizkraftwerke*. Weimar.

[11] George W. Hart. 1992. Nonintrusive appliance load monitoring. *Proc. IEEE* 80, 12 (1992), 1870–1891. https://doi.org/10.1109/5.192069

[12] Sonja Klingert, Florian Niedermeier, Corentin Dupont, Giovanni Giuliani, Thomas Schulze, and Hermann de Meer. 2015. Introducing Flexibility into Data Centers for Smart Cities. In *Smart Cities, Green Technologies, and Intelligent Transport Systems: 4th International Conference, SMARTGREENS 2015, and 1st International Conference VEHITS 2015, Lisbon, Portugal, May 20-22, 2015, Revised Selected Papers*, Markus Helfert, Karl-Heinz Krempels, Cornel Klein, Brian Donellan, and Oleg Guiskhin (Eds.). Springer International Publishing, Cham, 128–145. https://doi.org/10.1007/978-3-319-27753-0_7

[13] J. Zico Kolter and Matthew J. Johnson. 2011. REDD: A Public Data Set for Energy Disaggregation Research. In *in SustKDD*.

[14] Jungsuk Kwac and Ram Rajagopal. 2013. Demand response targeting using big data analytics. In *2013 IEEE International Conference on Big Data*. IEEE, 683–690. https://doi.org/10.1109/BigData.2013.6691643

[15] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. 2002. Finding motifs in time series. *SIGKDD '02* (2002).

[16] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. 2013. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation* 70, 10 (2013), 770–791. https://doi.org/10.1016/j.peva.2013.08.014

[17] Nicole Ludwig, Simon Waczowicz, Ralf Mikut, and Veit Hagenmeyer. 2017. Mining Flexibility Patterns in Energy Time Series from Industrial Processes. In *Proceedings. 27. Workshop Computational Intelligence, Dortmund, 23. - 24. November 2017*, Frank Hoffmann, E. Hüllermeier, and Ralf Mikut (Eds.). KIT Scientific Publishing, 13–32.

[18] Gearoid O'Brien and Ram Rajagopal. 2016. Scheduling Non-Preemptive Deferrable Loads. *IEEE Transactions on Power Systems* 31, 2 (2016), 835–845. https://doi.org/10.1109/TPWRS.2015.2402198

[19] Peter Palensky and Dietmar Dietrich. 2011. Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads. *IEEE Transactions on Industrial Informatics* 7, 3 (2011), 381–388. https://doi.org/10.1109/TII.2011.2158841

[20] John W. Pratt. 1959. Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures. *J. Amer. Statist. Assoc.* 54, 287 (1959), 655–667. http://www.jstor.org/stable/2282543

[21] Danny Pudjianto and Goran Strbac. 2017. Assessing the value and impact of demand side response using whole-system approach. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* 231, 6 (2017), 498–507. https://doi.org/10.1177/0957650917722381

[22] Sami Rollins and Nilanjan Banerjee. 2014. Using rule mining to understand appliance energy consumption patterns. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 29–37. https://doi.org/10.1109/PerCom.2014.6813940

[23] Goran Strbac. 2008. Demand side management: Benefits and challenges. *Energy Policy* 36, 12 (2008), 4419–4426. https://doi.org/10.1016/j.enpol.2008.09.030

[24] Jay Taneja. 2014. Growth in renewable generation and its effect on demand-side management. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 614–619. https://doi.org/10.1109/SmartGridComm.2014.7007715

[25] Jay Taneja, Ken Lutz, and David Culler. 2013. The impact of flexible loads in increasingly renewable grids. In *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 265–270. https://doi.org/10.1109/SmartGridComm.2013.6687968

[26] Ngoc Cuong Truong, Tim Baarslag, Sarvapali D. Ramchurn, and Long Tran-Thanh. 2016. Interactive scheduling of appliance usage in the home. In *25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. 869–875. http://eprints.soton.ac.uk/396670/

[27] Jan Weglarz. 2012. *Project Scheduling: Recent Models, Algorithms and Applications*. Springer US. https://books.google.de/books?id=uFDtBwAAQBAJ

[28] M. A. Zehir, M. H. Wevers, A. Batman, M. Bagriyanik, J. L. Hurink, U. Kucuk, F. J. Soares, and A. Ozdemir. 2017. A novel incentive-based retail demand response program for collaborative participation of small customers. In *2017 IEEE Manchester PowerTech Main*. IEEE, 1–6. https://doi.org/10.1109/PTC.2017.7981059

[29] Shizhen Zhao, Xiaojun Lin, and Minghua Chen. 2017. Robust Online Algorithms for Peak-Minimizing EV Charging Under Multistage Uncertainty. *IEEE Transactions on Automatic Control* 62, 11 (2017), 5739–5754. https://doi.org/10.1109/TAC.2017.2699290

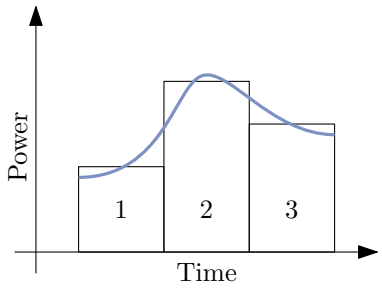


Figure 8: A fixed chain of three jobs 1, 2 and 3, which are used to approximate a job that has the blue demand curve.

A APPENDIX

A.1 Data Publication

We publish the instance sets from Section 4.2 at

https://publikationen.bibliothek.kit.edu/1000082194

This publication [3] contains

- The PS-Nonuniform, PS-Uniform, PSG and OM instance sets,
- The computational results of our optimization,
- and information on how to repeat our experiments.

Additionally, we publish the software that we used for optimization at

https://github.com/kit-algo/TCPSPSuite

Note that the raw data from which we detected our motifs is also published as the HIPE dataset [5].

A.2 MIP Extension for Job Chains

In this section, we give some details on how the MIP model presented in Section 3.4 can be extended to better approximate real-world processes (as already laid out in Section 3.4.3).

The MIP modelling technique from [4] allows to model what is called *minimum time lags* between two jobs i and j. This time lag $L_{i,j}$ specifies the number of time steps that must pass between the start of i and the start of j. Together with the start time variable σ_i for every job i, this results in constraints of the form

$$\sigma_j \geq \sigma_i + T_{i,j} \qquad \forall\ i, j.$$

An interesting aspect is that the $T_{i,j}$ may be negative. Using this, we can form fixed chains of jobs. Say we have three jobs 1, 2 and 3, which all have processing time (p_i) of 1. If we set $T_{1,2} = 1$, $T_{2,3} = 1$ and $T_{3,1} = -2$, it must hold that

$$\sigma_2 \geq \sigma_1 + 1,$$
$$\sigma_3 \geq \sigma_2 + 1,$$
$$\sigma_1 \geq \sigma_3 - 2.$$

Which results in

$$\sigma_2 = \sigma_1 + 1,$$
$$\sigma_3 = \sigma_2 + 1.$$

Such a situation is sketched in Figure 8, where the three rectangles represent the three jobs: The width of each rectangle is the job's duration and the height of the rectangle is the job's power demand. We see that the blue curve, which might represent the power demand curve of some industrial process, can be approximated more closely by this chain of jobs than if we only used one job.

When using this approach, one must pay attention to correctly encode the $\hat{T}$ / $\hat{J}$ limits in the instance. In Section 3.4, we fix every job at its start time (by setting release and deadline correctly). When using job chains, only one of the jobs of each chain must be fixed like this. Otherwise, moving a chain of k jobs by t time steps would count as k jobs regarding $\hat{J}$ and would contribute $k \cdot t$ units to $\hat{T}$. However, fixing only one job's window also fixes all other jobs in the chain.

It is important to note that this approach usually leads to a significant increase in the computational complexity of the resulting model. Tweaking the model such that this technique becomes feasible for large instances is beyond the scope of this paper.

A.3 Motif Analysis

We now analyze the discovered motifs further, especially with regard to the question whether jobs with constant power demand are a reasonable approximation of the motifs, and if not, how much better the approximation becomes when we allow to split the jobs into multiple blocks as outlined in sections 3.4.3 and A.2. Note that all discovered motifs are presented in Figure 18.

The occurrences of motifs correspond to stepwise functions: Every point in the occurrence's power demand time series results in one step in its power demand function. Let o be an occurrence. We then call the (stepwise) function mapping a point in time to the power demand of the occurrence at that time $P_o\colon [0,1] \to \mathbb{R}$ (note that occurrences are normalized, thus a point in time is in $[0,1]$). The main question is how well we can approximate these functions with other stepwise functions *of low complexity*, i. e., with few steps. Note that a job with a constant power demand corresponds to a stepwise function with exactly one step, a chain of two jobs corresponds to a stepwise function with up to two steps, and so on. Let $\tilde{P}_{o,k}\colon [0,1] \to \mathbb{R}$ be such a function with at most k steps, that tries to approximate P_o.

We need some notion of the difference between P_o and $\tilde{P}_{o,k}$. We suggest the following

$$\Delta\left(P_o, \tilde{P}_{o,k}\right) = \frac{1}{N_o} \int_0^1 \left(P_o(t) - \tilde{P}_{o,k}(t)\right)^2 \,\mathrm{d}t.$$

Here, N_o is a normalization factor to make different motifs comparable: $N_o = \int_0^1 P_o(t)^2 \,\mathrm{d}t$. We can compute the value of the integral without actually integrating, since both functions are discrete in t. This metric penalizes deviations of $\tilde{P}_{o,k}$ from P_o with a quadratic term. We assume a deviation that is large in magnitude but short in time to be worse than a deviation which is small in magnitude but long in time: That is because deviations of large magnitude might hide exactly the peaks in power demand that we are interested in reducing.

To analyze how block-shaped our motifs really are, we fitted multiple $\tilde{P}_{o,k}$ (for multiple values of k) to the P_o of every occurrence o.[6] Our first attempt is $k = 1$, i. e., a step function with exactly one step, representing jobs with constant power demand. We see the value of $\Delta(P_o, \tilde{P}_{o,1})$ in Figure 9. The x axis groups the occurrences by their motif. We sort motifs by how well they are approximable for $k = 1$.

We cannot say what values for $\Delta(P_o, \tilde{P}_{o,k})$ are good or bad: The question of what is an acceptable approximation must be answered by the person using our framework. However, we can clearly see that nine of our fifteen discovered motifs are a lot better approximated by a job with constant power demand than the remaining six. This seems intuitively correct when looking at the motifs in Figure 18.

We can also see how $\Delta(P_o, \tilde{P}_{o,k})$ changes when we go from $k = 1$ to $k = 2$, which is shown in Figure 10a. We see that the change is substantial for the six motifs that were not well approximated before. Especially motifs L, N and O seem to profit from a two-step function. When looking at these motifs in figures 18l, 18n and 18o, that seems plausible.

We see a similar effect when going to $k = 5$ (see Figure 10b) and $k = 10$ (see Figure 10c): The $\Delta(P_o, \tilde{P}_{o,k})$ values shrink gradually for the six motifs which are not very block-shaped, although im improvement is less than for going from $k = 1$ to $k = 2$.

We can thus conclude that the technique proposed in Section 3.4.3 has benefits: Being able to approximate the motifs with stepwise functions of more than one step most likely brings the results of the optimization closer to reality. Especially allowing for two jobs instead of one might be a worthwhile option. However, we can also conclude that a constant function is already a good approximation for the majority of the discovered motifs, and is not completely outlandish for the rest of the motifs either.

[6] Using a black-box SLSQP optimizer on $\Delta(P_o, \tilde{P}_{o,k})$

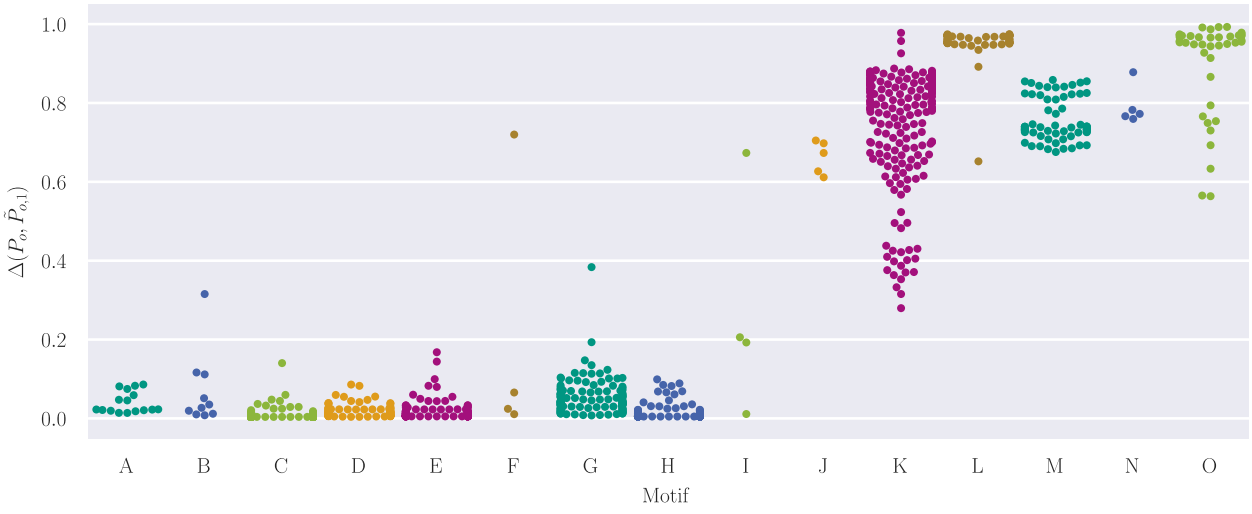
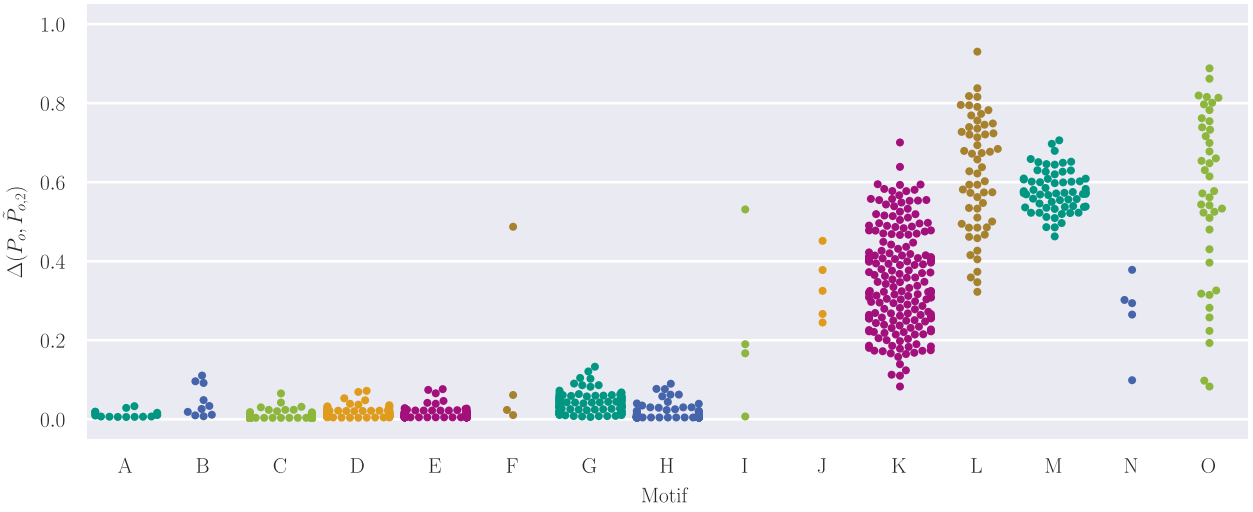
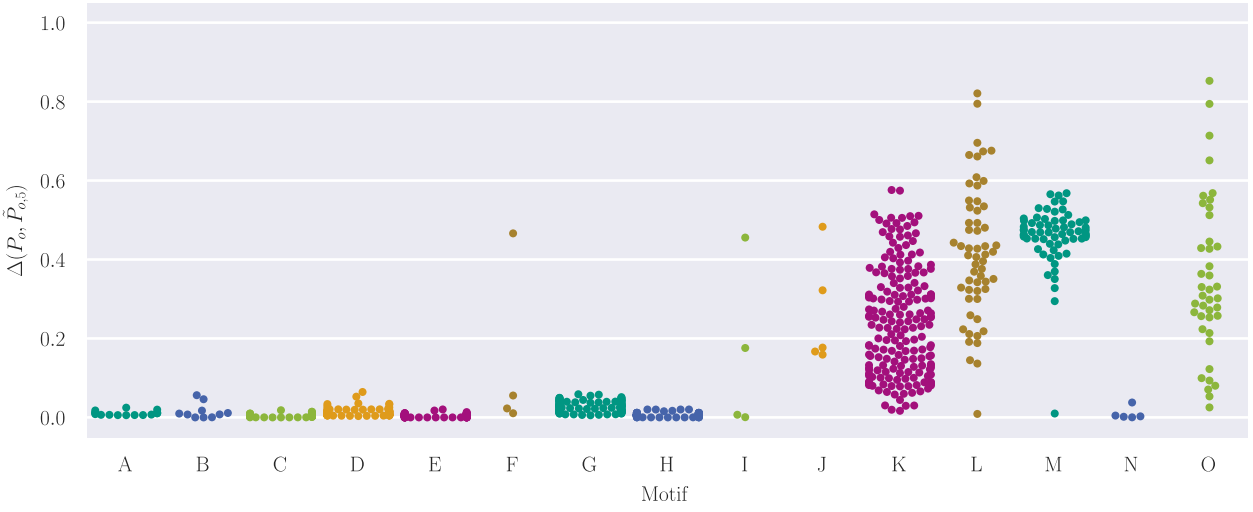


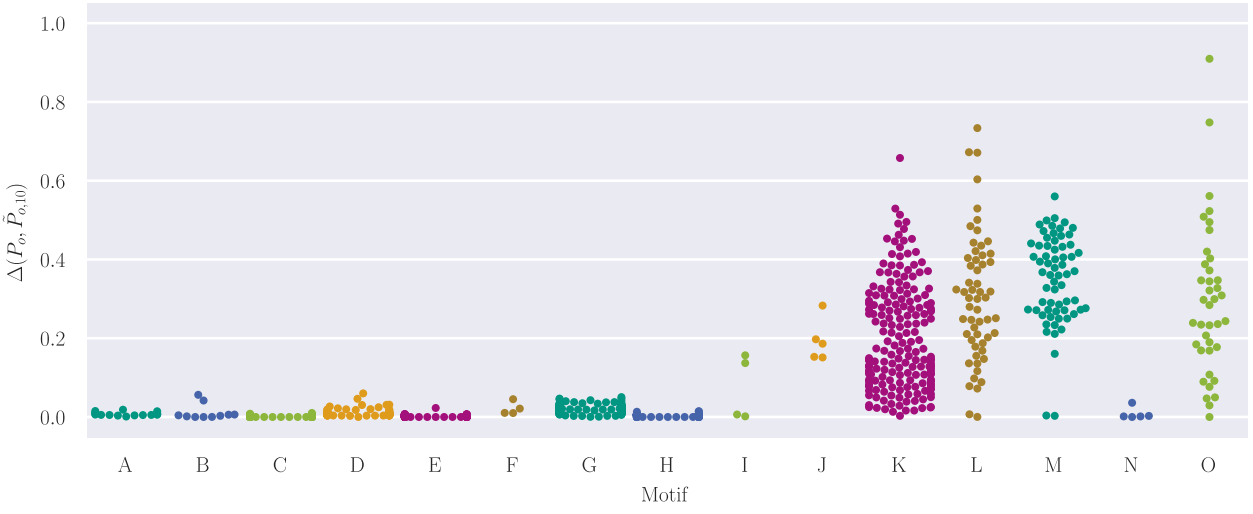
Figure 9: The difference between all occurrences' P_o an and their respective optimal $\tilde{P}_{o,1}$



(a) $k = 2$



(b) $k = 5$



(c) $k = 10$

Figure 10: The difference between all occurrences' P_o an and their respective optimal $\tilde{P}_{o,k}$

Machine	Motifs	Alphabet Size	Wordlength
AVT 01	A, D	4	505
AVT 02	J, K	4	403
AVT 03	B, M, N	4	267
AVT 04	H	4	433
AVT 05	C	4	543
AVT 06	E	4	406
AVT 08	F, L	4	211
AVT 09	G	4	500
AVT 10	I, O	4	426

Table 3: Parameter choices for the motif discovery algorithm. The alphabet size was varied between 2 and 10 words, resulting in largely the same results as presented above.

A.4 Numerical Evaluation

	3	→	6	→	9
0.005		$< 10^{-5}$		0.00089	
↓	$< 10^{-5}$		$< 10^{-5}$		$< 10^{-5}$
0.01		$< 10^{-5}$		$< 10^{-5}$	
↓	0.00025		$< 10^{-5}$		0.00033
0.02		$< 10^{-5}$		0.00036	
↓	0.0008		0.03		0.041
0.03		$< 10^{-4}$		0.086	
↓	0.00011		$< 10^{-5}$		0.00022
0.04		0.00033		0.0018	

Table 4: p-Values for the change of one parameter in the PS-Uniform set. Values highlighted in green indicate that changing $\hat{J}$ and Θ, while keeping the other one constant, results in a statistically significant change in improvements. Values in blue are significant before Bonferroni correction.

	3	→	6	→	9
0.005		$< 10^{-5}$		$< 10^{-4}$	
↓	$< 10^{-5}$		$< 10^{-5}$		$< 10^{-5}$
0.01		$< 10^{-5}$		$< 10^{-5}$	
↓	0.00037		$< 10^{-4}$		$< 10^{-5}$
0.02		$< 10^{-5}$		$< 10^{-5}$	
↓	0.00021		0.0051		0.00017
0.03		$< 10^{-5}$		$< 10^{-5}$	
↓	0.0021		$< 10^{-5}$		$< 10^{-4}$
0.04		$< 10^{-5}$		$< 10^{-5}$	

Table 5: p-Values for the change of one parameter in the PSG set. Values highlighted in green indicate that changing $\hat{J}$ and Θ, while keeping the other one constant, results in a statistically significant change in improvements. Values in blue are significant before Bonferroni correction.

	3	→	6	→	9
0.005		$< 10^{-4}$		0.00045	
↓	$< 10^{-5}$		$< 10^{-5}$		$< 10^{-5}$
0.01		0.04		0.059	
↓	$< 10^{-4}$		$< 10^{-5}$		$< 10^{-5}$
0.02		$< 10^{-5}$		0.00068	
↓	$< 10^{-5}$		$< 10^{-5}$		$< 10^{-5}$
0.03		$< 10^{-4}$		0.011	
↓	$< 10^{-4}$		$< 10^{-5}$		$< 10^{-5}$
0.04		$< 10^{-5}$		$< 10^{-5}$	

Table 6: p-Values for the change of one parameter in the OM set. Values highlighted in green indicate that changing $\hat{J}$ and Θ, while keeping the other one constant, results in a statistically significant change in improvements. Values in blue are significant before Bonferroni correction.

Θ	0.005			0.01			0.02			0.03			0.04		
$\hat{J}$	3	6	9	3	6	9	3	6	9	3	6	9	3	6	9
Min	0.57	0.54	0.57	0.56	0.5	0.49	0.52	0.44	0.41	0.52	0.44	0.38	0.52	0.44	0.38
Max	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Mean	0.84	0.83	0.83	0.81	0.78	0.78	0.8	0.77	0.76	0.8	0.77	0.75	0.8	0.77	0.75
Median	0.86	0.81	0.81	0.83	0.75	0.75	0.83	0.74	0.72	0.83	0.74	0.72	0.83	0.74	0.72
Std. Dev.	0.11	0.12	0.12	0.13	0.15	0.15	0.14	0.16	0.17	0.14	0.16	0.17	0.14	0.16	0.17

Table 7: Statistics of the change in peak demand after optimization in the PS-Nonuniform set.

Θ	0.005			0.01			0.02			0.03			0.04		
$\hat{J}$	3	6	9	3	6	9	3	6	9	3	6	9	3	6	9
Min	0.84	0.8	0.8	0.83	0.77	0.75	0.81	0.73	0.7	0.8	0.72	0.7	0.8	0.71	0.69
Max	0.98	0.93	0.93	0.98	0.92	0.91	0.98	0.92	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Mean	0.9	0.88	0.88	0.89	0.84	0.83	0.88	0.81	0.79	0.93	0.88	0.88	0.98	0.98	0.97
Median	0.91	0.87	0.87	0.88	0.84	0.83	0.88	0.81	0.78	0.92	0.84	0.93	1.0	1.0	1.0
Std. Dev.	0.032	0.036	0.036	0.035	0.042	0.043	0.04	0.05	0.071	0.066	0.099	0.12	0.056	0.068	0.085

Table 8: Statistics of the change in peak demand after optimization in the PS-Uniform set.

Θ	0.005			0.01			0.02			0.03			0.04		
$\hat{J}$	3	6	9	3	6	9	3	6	9	3	6	9	3	6	9
Min	0.51	0.48	0.48	0.47	0.4	0.37	0.47	0.34	0.27	0.47	0.29	0.25	0.47	0.28	0.24
Max	0.98	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
Mean	0.79	0.77	0.77	0.78	0.73	0.72	0.77	0.71	0.68	0.77	0.7	0.67	0.77	0.69	0.66
Median	0.85	0.81	0.8	0.83	0.79	0.75	0.82	0.76	0.72	0.82	0.76	0.72	0.82	0.76	0.72
Std. Dev.	0.14	0.15	0.15	0.14	0.17	0.18	0.14	0.19	0.21	0.14	0.19	0.22	0.14	0.2	0.22

Table 9: Statistics of the change in peak demand after optimization in the PSG set.

Θ	0.005			0.01			0.02			0.03			0.04		
$\hat{J}$	3	6	9	3	6	9	3	6	9	3	6	9	3	6	9
Min	0.97	0.97	0.97	0.52	0.94	0.94	0.89	0.89	0.88	0.84	0.85	0.85	0.82	0.81	0.81
Max	0.98	0.98	0.98	0.97	0.97	0.97	0.94	0.94	0.94	0.93	0.92	0.92	0.92	0.9	0.89
Mean	0.98	0.98	0.98	0.94	0.95	0.95	0.92	0.91	0.91	0.89	0.88	0.88	0.88	0.85	0.85
Median	0.98	0.98	0.98	0.95	0.95	0.95	0.92	0.91	0.91	0.89	0.88	0.88	0.88	0.85	0.85
Std. Dev.	0.0037	0.0037	0.0037	0.078	0.0069	0.0069	0.011	0.011	0.013	0.019	0.015	0.015	0.021	0.019	0.019

Table 10: Statistics of the change in overshoot after optimization in the OM set.

A.5 Omitted Figures

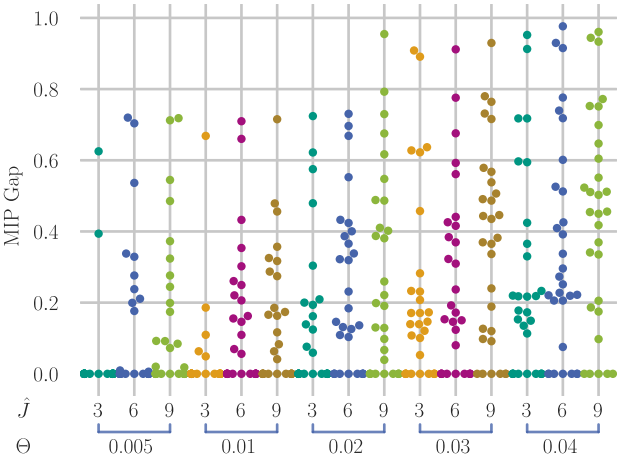


Figure 11: MIP gaps for the various settings of $\hat{J}$ and Θ in the PS-Nonuniform instance set. Every dot corresponds to one instance. Colors are used to distinguish the columns.

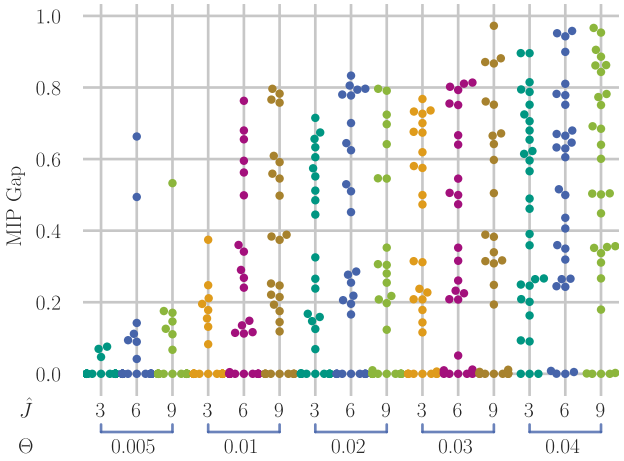


Figure 12: MIP gaps for the various settings of $\hat{J}$ and Θ in the PSG instance set. Every dot corresponds to one instance. Colors are used to distinguish the columns.

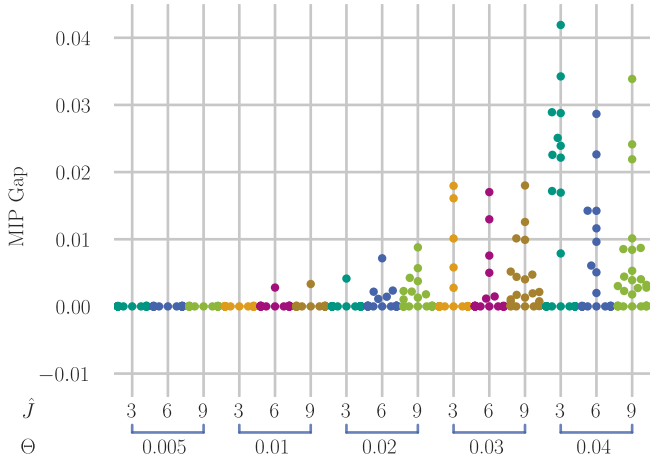
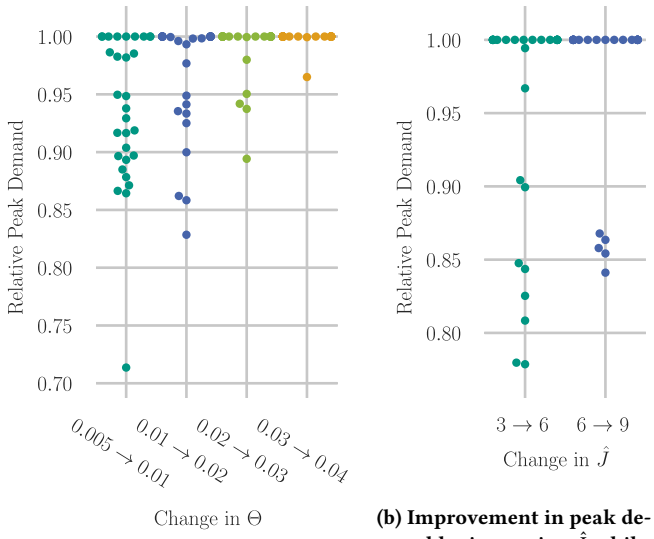


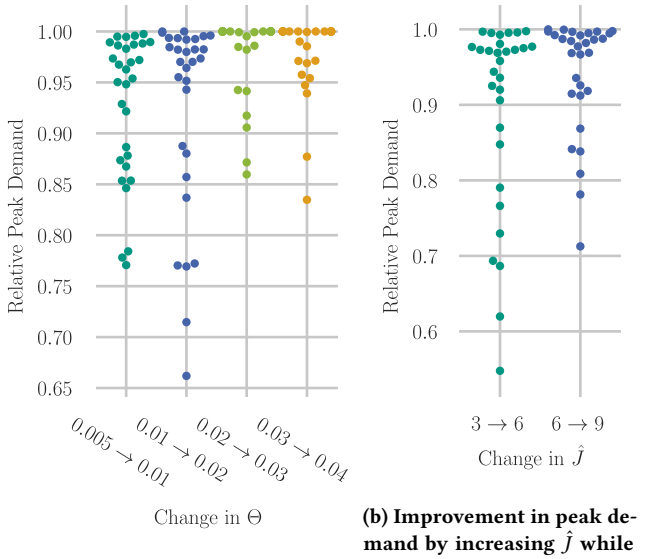
Figure 13: MIP gaps for the various settings of $\hat{J}$ and Θ in the OM instance set. Every dot corresponds to one instance. Colors are used to distinguish the columns.



(a) Improvement in peak demand by increasing Θ while keeping $\hat{J}$ constant

(b) Improvement in peak demand by increasing $\hat{J}$ while keeping Θ constant.

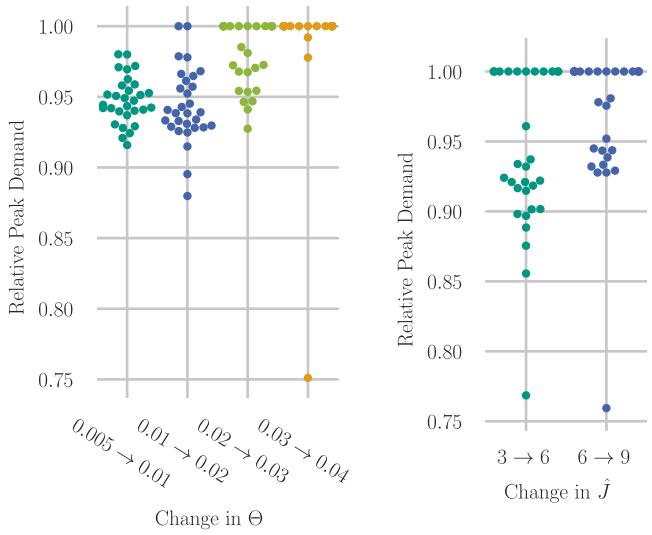
Figure 14: Additional Results for set PS-Nonuniform, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in peak demand after optimization. Color indicates how well the instance could be optimized.



(a) Improvement in peak demand by increasing Θ while keeping $\hat{J}$ constant

(b) Improvement in peak demand by increasing $\hat{J}$ while keeping Θ constant.

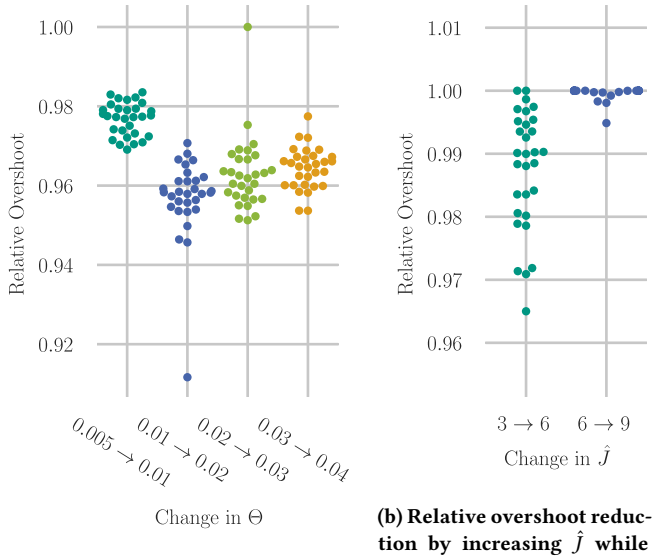
Figure 16: Additional results for set PSG, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in peak demand after optimization. Color indicates how well the instance could be optimized.



(a) Improvement in peak demand by increasing Θ while keeping $\hat{J}$ constant.

(b) Improvement in peak demand by increasing $\hat{J}$ while keeping Θ constant.

Figure 15: Additional Results for set PS-Uniform, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in peak demand after optimization. Color indicates how well the instance could be optimized.

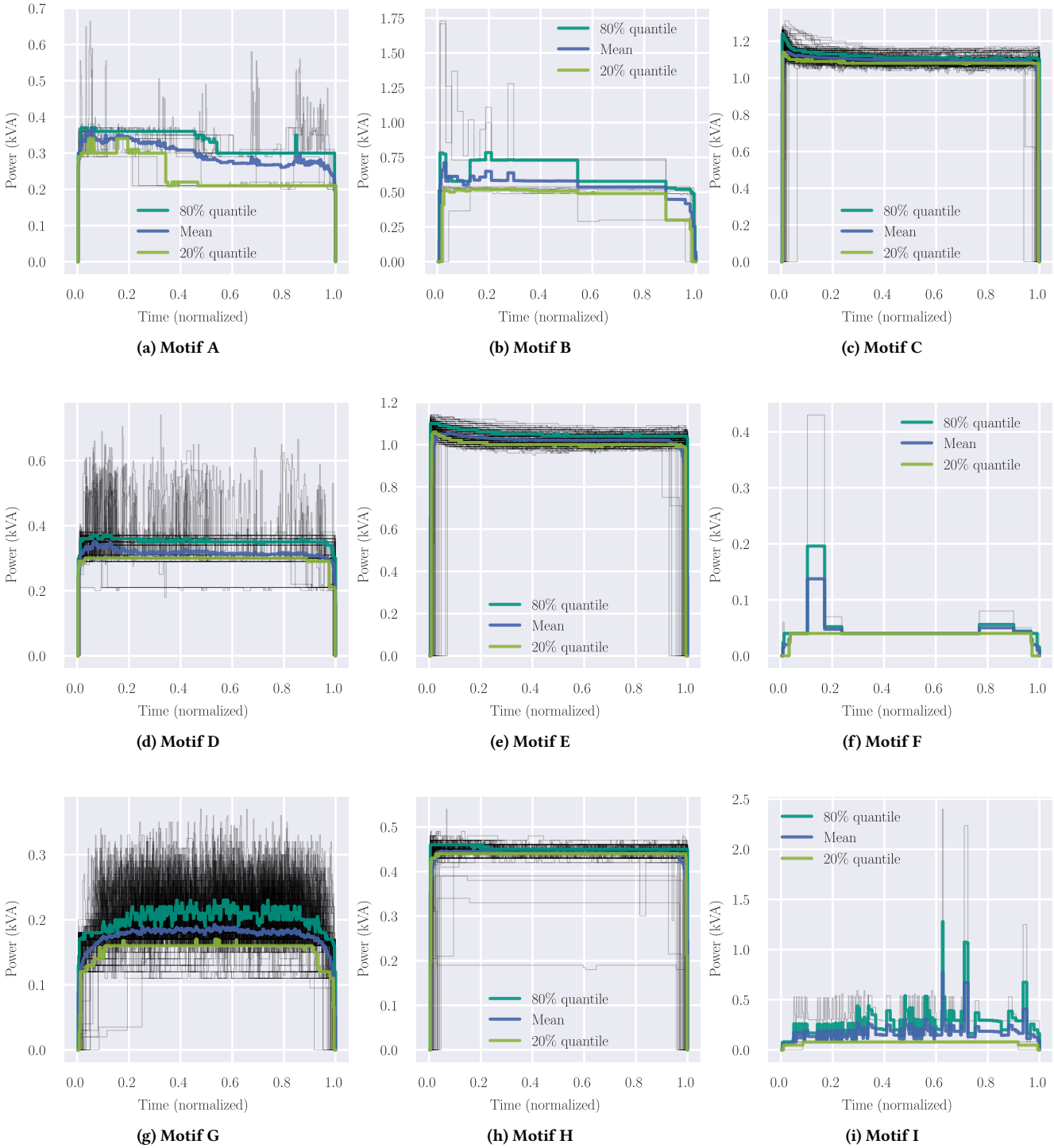


(a) Relative overshoot reduction by increasing Θ while keeping $\hat{J}$ constant

(b) Relative overshoot reduction by increasing $\hat{J}$ while keeping Θ constant.

Figure 17: Additional results for set OM, one point per instance. The columns are the different settings for $\hat{J}$ and Θ. The Y axis indicates the change in overshoot after optimization. Color indicates how well the instance could be optimized.

A.6 Discovered Motifs

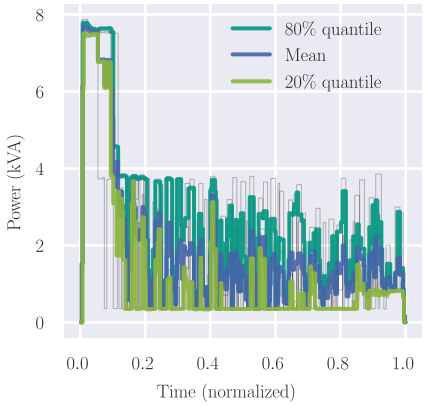


(a) Motif A (b) Motif B (c) Motif C

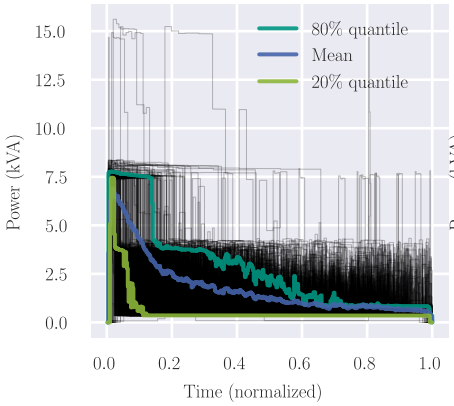
(d) Motif D (e) Motif E (f) Motif F

(g) Motif G (h) Motif H (i) Motif I

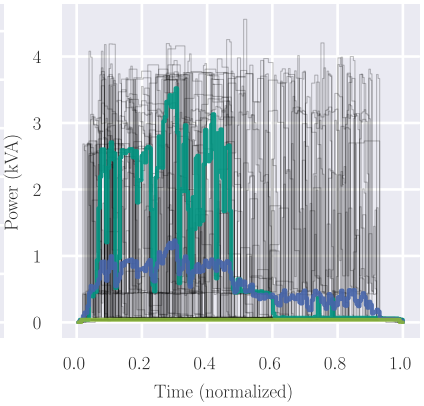
Figure 18: All discovered motifs. Each black line indicates one occurrence of the respective motif.



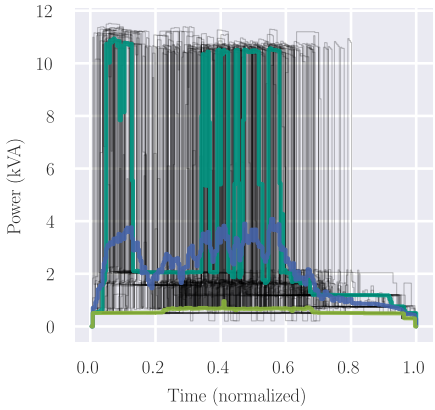
(j) Motif J



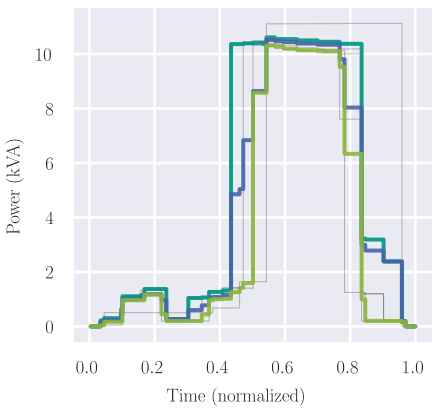
(k) Motif K



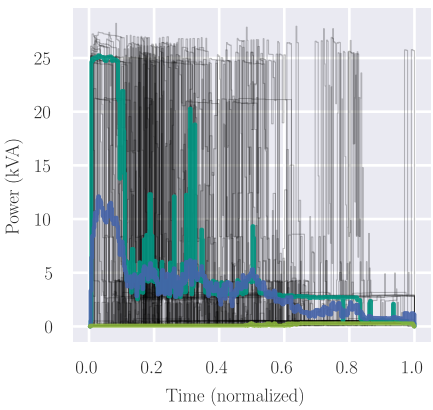
(l) Motif L



(m) Motif M



(n) Motif N



(o) Motif O