A Comprehensive Modelling Framework for Demand Side Flexibility in Smart Grids

Lukas Barth · Nicole Ludwig · Esther Mengelkamp · Philipp Staudt

Abstract The increasing share of renewable energy generation in the electricity system comes with significant challenges, such as the volatility of renewable energy sources. To tackle those challenges, demand side management is a frequently mentioned remedy. However, measures of demand side management need a high level of flexibility to be successful. Although extensive research exists that describes, models and optimises various processes with flexible electrical demands, there is no unified notation. Additionally, most descriptions are very process-specific and cannot be generalised.

In this paper, we develop a comprehensive modelling framework to mathematically describe demand side flexibility in smart grids while integrating a majority of constraints from different existing models. We provide a universally applicable modelling framework for demand side flexibility and evaluate its practicality by looking at how well Mixed-Integer Linear Program (MIP) solvers are able to optimise the resulting models, if applied to artificially generated instances. From the evaluation, we derive that our model improves the performance of previous models while integrating additional flexibility characteristics.

Keywords demand side management, flexibility scheduling, process modelling, smart grids.

This work was partly funded by the German Research Foundation (DFG) Research Training Group 2153 "Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation".

L. Barth
Institute of Theoretical Informatics, Karlsruhe Institute of Technology
Am Fasanengarten 5, 76131 Karlsruhe, Germany

N. Ludwig
Institute for Applied Computer Science, Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
Tel.: +49 721 608-26713
E-mail: nicole.ludwig@kit.edu

E. Mengelkamp, P. Staudt
Institute for Information Systems and Marketing, Karlsruhe Institute of Technology
Fritz-Erler-Straße 23, 76133 Karlsruhe, Germany

1 Introduction

While many societies aim at shifting their energy mix towards renewable energies, the currently implemented system relies on a centralised dispatch of electricity generation (Schleicher-Tappeser 2012). The integration of the increasing decentralised renewable energy sources into the energy system is therefore one of the two most important research fields in energy informatics (Goebel et al. 2014). High fluctuations in supply, as well as strong intra-day patterns e. g. in the case of solar energy, are challenges for a smooth integration (Denholm et al. 2010). The traditional consumer behaviour is strenuous for the power grid as it results in high peaks and low valleys of the electric load. Currently, this fluctuation is compensated by conventional steerable power plants to ensure a reliable operation of the electricity grid. As more and more intermittent renewable sources generate electricity, this balancing technique is threatened (Weidlich et al. 2012). However, the decrease in supply side flexibility of intermittent generation might be offset by an increase in demand side flexibility. Therefore, one possibility to ease the integration of *renewable energy sources* (RES) is to control the consumer demand and adapt it to the supply side (Strbac 2008). Thus, the aim of an optimal supply strategy can be reached by providing more flexibility on the demand side. For example, we might use a heat pump whenever the sun is shining instead of when it is most convenient for the consumer.

Demand side management (DSM) summarises measures that foster more flexible energy consumption (Palensky and Dietrich 2011). DSM has to be differentiated from Demand Response which deals with the incentivisation and voluntary provision of flexibility by consumers and is sometimes categorized as a subcategory of DSM (e. g. Gärttner et al. 2016; Palensky and Dietrich 2011). In this paper, we will focus on DSM. Our objective is to design a holistic modelling framework to schedule demand side flexibility. Extensive research has been done discussing DSM applications from a scheduling perspective (e. g. Petersen et al. 2013), because it can make a significant contribution to the cost-efficient integration of renewable generation (Steurer et al. 2015).

Scheduling energy loads, hence exploiting the flexibilities in the system, to enhance grid stability or reduce energy costs for the consumer is not a new idea. However, in the mathematical set-up to solve these tasks, related work employs application-specific formulations to describe the loads and their characteristics to be scheduled. This practice results in a vast amount of different modelling formulations. Additionally, most authors focus on a single application. Thus, their models are not readily transferable to new data sets or different use cases. In this context, it is especially noteworthy that demand side flexibility of private households and industrial applications exhibits very different characteristics, i. e., household appliances can usually run independently from each other while industrial processes often depend on other production steps. As the considered papers always focus on only one application, no formulation exists that integrates all of these features. This variety of formulations in the literature makes it difficult to compare the modelling approaches, their respective results and adaptability.

We present a novel comprehensive modelling framework in the field of energy informatics to represent flexibility in a household as well as in an industrial context. Based on current literature, we classify the most important characteristics of flexibility represented in various models and incorporate the majority in a single modelling framework. We combine currently existing, wide-ranging research and, to our knowledge, are the first to integrate the different approaches into a single modelling framework.

The paper is structured as follows. In Section 2, we give a short overview of existing literature concerning demand side flexibility and management. Following this, we describe common features found in the literature describing demand flexibility in Section 3. Section 4 introduces our modelling framework which is evaluated according to its performance in the following Section 5. We discuss our work in Section 6, before giving an outlook and a conclusion in Section 7.

2 Related Work

Demand Side Management (DSM) and Demand Response (DR) become increasingly important as more electricity is generated from intermittent sources. This development has been accompanied by a growing interest from researchers. A variety of authors has been dealing with demand flexibility of private households. Consequently, they ignore most characteristics of industrial loads. For example, He et al. (2013) provide a classification of household flexibility along different dimensions, while Allerding et al. (2012) focus on developing demand response for private households. Gottwalt et al. (2016) also concentrate on private households, however, they incorporate several additional restrictions. Scott et al. (2013) characterise the flexibility of individual household devices. However, the description is tailored to specific appliances and therefore not domain independent. In Fehrenbach et al. (2014) the authors show that thermal appliances and specifically the expansion of heat pump use may have the largest flexibility potential of private households. Du and Lu (2011) provide a scheduling algorithm for those thermal appliances. This work is extended by Alizadeh et al. (2015), who differentiate between curtailable thermal loads and other deferrable loads. Household behaviour with regards to the provision of flexibility and effects on electricity costs is simulated by Gottwalt et al. (2011). They conclude that saving potentials for households are moderate when compared to the investment in smart meter technology. Contrary to this result, Setlhaolo et al. (2014) come to the conclusion that a reduction of up to 25% of the electricity costs of private households is possible. The investigation by Soares et al. (2014) also considers customer dissatisfaction besides the monetary compensation.

Demand side flexibility as a means to integrate renewable generation is established by Palensky and Dietrich (2011). Other research has established that fluctuations of a low penetration of renewable generation can be offset by demand side flexibility, as for example shown by Strbac (2008). However, the authors argue that a monetary compensation is difficult to determine. Halvorsen and Larsen (2001) describe the effects of appliance endowment and additional investment on the ability to provide flexibility. A new approach for a scheduling algorithm was developed by Ströhle et al. (2014) to match uncertain supply with different demand packages to maximise total welfare. The optimal combination of private household flexibility is investigated by Gärttner et al. (2016) and extended in Gärttner (2016) to provide recommendations to flexibility portfolio aggregators.

An extensive description of characteristics of demand side flexibility beyond residential flexibility is given by Petersen et al. (2013). The authors also develop a first taxonomy for flexibility but chose not to incorporate a variety of characteristics of flexibility (Petersen et al. 2014). Paulus and Borggrefe (2011) establish that demand side management bears considerable monetary potential in energy intensive industries. Qureshi et al. (2014) develop a model to investigate economic potential of demand side management in office buildings. Ashok and Banerjee (2000) pioneer the field of industrial demand side management. Their model is specified in Ashok (2006) but leaves certain restrictions for future research. In Schilling and Pantelides (1996) we find appropriate scheduling algorithms for our problem formulation. However, as they are not specifically developed for electricity loads, individual extensions to the model are necessary. Mitra et al. (2012) and Moon and Park (2014) consider scheduling with regards to electricity costs for industrial production. Oudalov et al. (2007) use batteries to reduce demand peaks.

We present the most relevant models of demand side flexibility with regards to the restrictions and characteristics they incorporate in Table 1 using criteria presented in the next section. The aim of this paper is to integrate the features considered in the described models into one holistic modelling framework which allows to describe flexibilities across all domains, rather than developing another alternative model of demand side flexibility.

3 Modelling Flexibility

In this chapter, we describe our proposed holistic modelling framework. We incorporate the majority of features, which we found in the relevant papers in Table 1. Thus, our approach can describe and optimise flexibility independently from its domain.

The basis for our model are *jobs*, representing atomic processes that require a certain amount of electrical power during their execution. We usually associate a duration with each job. Based on these jobs, models, respectively modelling frameworks, can have various *features*, i. e., ways of representing constraints or parameters of the problem.

Table 1 summarises the papers we examined and gives an overview of the features considered. The features we address with our new formulation are indicated with check marks. Brackets indicate that we can reasonably express a certain feature *indirectly*, although we do not meet all subtleties encountered in the literature presented. Features not yet included in our modelling framework are marked with crosses. In total, we take 14 different features into account, which we describe as follows:

1. **Time Frame.** States whether the described model uses discrete or continuous time steps.
2. **Interruptible Jobs.** The model allows for *interruptible* jobs, i. e., jobs which do not have to be executed consecutively. We do not distinguish between the ability to stop jobs at any time, or at predefined time slots. *Brackets:* Models which allow for interdependent jobs (see below) enable us to split up interruptible jobs into small chunks and connect these with dependencies. This way, the original job can either be executed consecutively (if all chunks are scheduled consecutively) or with interruptions. Thus, all models supporting interdependent jobs indirectly support interruptible jobs.
3. **Storage.** The model allows to include some form of storage possibility. *Brackets:* Storage can be modelled indirectly via a special kind of dummy jobs which can be moved forward to simulate charging of the storage. The place where dummy jobs were moved away from then has more power available, simulating getting energy out of storage.
4. **Interdependent Jobs.** Jobs can have predecessors, allowing a job to be scheduled only as soon as all its predecessors are completed. Optionally, time lags can be associated with dependencies, enforcing a certain amount of time to pass between the finish of a predecessor and the earliest start of a successor.
5. **Earliest Start Time.** Jobs can be associated with an earliest start time and may not be scheduled before that time.
6. **Deadline.** Jobs can be associated with a deadline and must be scheduled such that they are finished at that time. From this, the possibility of an overall deadline directly follows.
7. **Production.** Jobs can be associated with a production output, and the whole schedule has to meet a production target. In our case, the production is fixed as we schedule each job exactly once during the time frame considered.
8. **Multiple Resources.** The model can contain more resources than electrical energy alone, and jobs may require amounts of more than one resource simultaneously.
9. **Base loads.** Uncontrollable loads, i.e., jobs that must be scheduled at a specific time, may be part of the model. *Brackets:* Base loads can be modelled indirectly if earliest start times *and* deadlines are present, or if deadlines *and* interdependent jobs are present,

Table 1: Comparison of the integrated flexibility features in related work to our modelling framework.

Reference	*Time Frame*	*Interruptible Jobs*	*Storage*	*Interdependent Jobs*	*Earliest Start Time*	*Deadline*	*Production*	*Multiple Resources*	*Base loads*	*Modes*	*Drain, Losses*	*Down-/Uptime*	*Multiple Runs*	*Ramping*
Allerding et al. (2012)	discrete	✓	✗	✗	✓	✓	✗	✗	(✓)	✗	✗	✗	✗	✗
Ashok and Banerjee (2000)	discrete	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗
Ashok (2006)	discrete	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓
Castro et al. (2002)	continuous	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗
Fink et al. (2014)	discrete	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓
Gottwalt et al. (2016)	discrete	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗
Luo et al. (1998)	discrete	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗
Mitra et al. (2012)	discrete	✗	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓
Moon and Park (2014)	discrete	(✓)	✓	✓	✗	✓	✓	✓	(✓)	✓	✗	✓	✓	✓
Oudalov et al. (2007)	discrete	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗
Petersen et al. (2013, 2014)	discrete	✗	✓[a]	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✗
Schilling and Pantelides (1996)	continuous	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗
Sou et al. (2011)	continuous	(✓)	✗	✓	(✓)	✓	✗	✗	✓	✓	✗	✓	✓	✗
This paper	discrete	(✓)	(✓)	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓

[a] Only integrated in the first paper by the authors.

by inserting dummy jobs that can only be scheduled at the specified times.

10. **Modes.** Jobs may have multiple modes, where every mode is a combination of a run time and resource requirements. Each job can have a possibly large set of parameters for all possible operation modes. Modes with less required resources usually take longer. The scheduler can decide in which mode to run a job. We assume all modes of a job to be of equal value, i.e., things like product quality do not depend on the chosen mode of a job.
11. **Drain, Losses.** Energy spent on the execution of a job may drain over time, i.e., another job which is scheduled later might need to replenish energy (and thus use more resources or take longer) if it is scheduled late.
12. **Down-/Uptime.** Jobs can be associated with a fixed amount of time where they need to be shut down after running (downtime), or a fixed amount of time that they have to be used (uptime). In contrast to the flexible description of minimum and maximum runtime, this is a fixed amount of time.
13. **Multiple Runs.** Every job can either be scheduled once or multiple times, throughout the optimisation period. Multiple runs are most useful when we choose the time horizon in such a way that we need to meet a production target. As we choose the time horizon for the optimisation period such that every job can only run once, this criterion is unnecessary. Thus, we currently abstract form integrating multiple runs in our modelling framework.
14. **Ramping.** Jobs may be associated with a ramping function of some kind, describing how resource usage slowly increases when the job is started and decreases when the job finishes. Ramping might be unnecessary if another job is executed right before the ramping job starts or directly after the job. If ramping is necessary, the runtime of a job usually increases.

4 Optimisation Model

The optimisation model derived from the modelling framework is a Mixed-Integer Linear Program (MIP). Given an instance with n jobs in which job i can be run in $m_i \geq 1$ different modes and the latest job deadline is $D_{\max}$, the decision variables consist of two groups of binary variables. A group $s_i(t)$ of variables indicating whether job i is started in time instant t and a group of variables $m_{i,j}$ indicating whether job i is run in mode j. This limits the number of decision variables to $n \cdot D_{\max} + \sum m_i$.

Table 2: Variables used in the modelling framework, with the model variables in the top, the decision variables in the middle and the derived variables in the bottom part of the table.

	Model Variables
n	Number of jobs
$\tilde{P}_i$	Base power requirement of job i
$\tilde{T}_i$	Base run time of job i
M_i	Set of mode coefficients of job i
$\phi_{i,m}$	mode coefficient for time adjustment of job i in mode m
$\psi_{i,m}$	mode coefficient for power adjustment of job i in mode m
D_i	Deadline of job i
$E_{prod}(t)$	Power available at time step t
$c(t)$	Cost function for using energy above capacity limit (i. e. production and storage)
$L_{i,j}$	Minimum time lag between job i and j, measured in time steps from the end of i to the start of j
$\tau_{i,j}$	Runtime extension coefficient for the separation of jobs i and j
Λ_i	Maximum number of ramping steps for job i
$\delta_{i,j,k}$	Number of time steps between the end of job i and the start of job j before job j must execute ramping step k before executing the actual job
$\mu_{i,k}$	Power requirement of job j's k-th ramping step
	Decision Variables
$s_i(t)$	Binary variable, becomes 1 if and only if job i starts at timestep t
$m_{i,j}$	Binary variable, indicating if job i is to be run in mode j
	Derived Variables
$\tilde{\phi}_i$	Effective time adjustment coefficient of job i
$\tilde{\psi}_i$	Effective power adjustment coefficient of job i
P_i	Power requirement of job i in its selected mode
T_i	Run time of job i in its selected mode
$\hat{P}(t)$	Total power requirement at timestep t
σ_i	Timestep in which job i starts
η_i	First timestep in which job i is finished
M	Large constant used to switch constraints on an off
$\rho_{i,k}$	Binary variable indicating whether job i must execute its k-th ramping step

The features (1. – 14.) described in Section 3 are modelled as constraints of the MIP. Overall, the number of binding constraints is quadratic to the number of jobs. Standard solvers such as Gurobi or CPLEX (Meindl and Templ 2012) can be used to solve models derived from our framework. In the following, we describe the characteristics of the modelling framework in detail, with an overview of the variables used in Table 2. The jobs can get their required power from different resources, where each resource adds $P_{r,i}$ to the overall power needed by the job. For simplicity's sake, we focus on the case of only one resource $\hat{P}(t)$ in the following.

Objective Function Instead of buying the energy from the grid, we want to change our process structure in such a way that we can produce most of our energy ourselves. Therefore, the primary goal of our modelling framework is to use the minimum possible energy from the grid by exploiting the inherent flexibility of the processes. We thus minimise the difference between self-produced electricity $E_{prod}(t)$ and the power $\hat{P}(t)$ needed to perform the desired processes. In our case we do not explicitly include storage but all previously stored energy could be added to the self-produced side of the equation. Using energy from the grid is penalised with a cost function $c(t)$. The objective function is then

$$\min \sum_t c(t) \cdot \Big(E_{prod}(t) - \hat{P}(t) \Big). \tag{1}$$

Additionally, we can also use peak shaving as a second objective to our scheduling. Minimising the peaks during our time frame would lower our overall energy costs and might make it easier to rely on renewable generation entirely even when there are only few storage capacities and production available.

$$\min \Big(\max_t \hat{P}(t) \Big) \tag{2}$$

Everything else is modelled in terms of constraints of the mixed-integer program. We now list and explain these constraints.

$$\sum_t s_i(t) = 1 \quad \forall i \tag{3}$$

$$\sum_j m_{i,j} = 1 \quad \forall i \tag{4}$$

$$\sigma_i = \sum_t t \cdot s_i(t) \tag{5}$$

$$P_i = \tilde{\psi}_i \cdot \tilde{P}_i \tag{6}$$

$$\eta_i = \sigma_i + T_i \tag{7}$$

$$\eta_i \leq D_i \tag{8}$$

$$\eta_i + L_{i,j} \leq \sigma_j \tag{9}$$

$$\tilde{\psi}_i = \sum_j m_{i,j}\psi_{i,j} \tag{10}$$

$$\tilde{\phi}_i = \sum_j m_{i,j}\phi_{i,j} \tag{11}$$

and (12) – (16)

Equation (3) ensures that each job is scheduled once during our optimisation period, with the starting time given by Equation (5) as summing over all time instances times the indicator whether the job starts in this instance results in the time instance σ_i that the job starts in. Similarly, Equation (4) ensures that for every job, exactly one mode is selected. Each job needs a power input P_i which depends on the modus m_i the job is running in and its base power $\tilde{P}_i$ (cf. Eq. (6)). The power input for some modi can also be negative i. e. we can store/ drain energy. If the energy is later used again it can be added to the overall produced energy $E_{Prod}(t)$. Before the overall schedules deadline is reached, all jobs have to be finished, with their finishing time η_i depending on the length of the job T_i (cf. Eq. (7) (8)). If the jobs are connected, the end time of the previous job and the start time of the following job need to be separated by at least their minimum time lag $L_{i,j}$ (cf. Eq. (9)). Equations (10) and (11) set the effective time and power coefficients depending on the selected mode.

We will describe the Equations (12) to (16) in detail in the following.

Interdependent Jobs Given two jobs i and j, we allow to specify a *minimum time lag* $L_{i,j}$ between i and j, specifying that j may only be started at least $L_{i,j}$ time steps after the start of i. Transformed into an MIP constraint, it looks like

$$\sigma_i + L_{i,j} \leq \sigma_j, \tag{12}$$

and directly translates to *the start of j must be at least $L_{i,j}$ time steps after the start of i.*

Time Extension for Drain and Modes Let $\tilde{T}_i$ be the *base time requirement* for Job i, and $\phi_{i,j}$ (resp. $\psi_{i,j}$) the power (resp. time) *mode coefficients* for the mode being run in. These coefficients determine how the power requirement (resp. run time) changes if mode j is selected, i.e., if $m_{i,j} = 1$. Additionally, the actual runtime may depend on one or several *drain factors* $\tau_{a,i}$. The drain factors indicate a runtime extension of i if job i is not started immediately after job a, as the energy that drained between the execution of a and i has to be replenished. In total, the resulting constraint on the runtime T_i of i is

$$T_i = \tilde{\phi}_i \cdot \tilde{T}_i + \sum_k \tau_{k,i}\left((\sigma_i - \eta_k)\right). \tag{13}$$

Here, the sum in T_i sums over all jobs k that might precede i. For jobs that do not precede i, or for which no drain is desired, $\tau_{k,i}$ should be set to zero, thereby making those terms irrelevant. Thus, the last part computes the time lag between the end of job k and the start of job i. Note that this part can never become negative, because k being a predecessor of i forces i to start only after k has finished, i.e., $\sigma_i \geq \eta_k$.

In this simplified form, the execution time extension can grow arbitrarily large. This growth is unrealistic since at some point, all energy stored during the execution of k is drained and no further replenishment is necessary. We could remedy this with a more complex constraint. However, this would exceed the scope of this example.

Ramping The ramping of job j is a series of dummy jobs describing the steps in the ramping job. Whether the λ-th ramping job must be executed is denoted by $\rho_{j,\lambda}$, where $\lambda \in \{1, \ldots, \Lambda_j\}$. Here, Λ_j is the maximum number of steps necessary to reach the power input needed for job j to start. At which ramping step we start depends on the time distance between the end of the last dependent job η_i and the start time of the job that needs ramping σ_j. We check if we execute ramping step λ by introducing one of the following constraints for every predecessor i of j

$$\rho_{j,\lambda} \cdot M \geq (\sigma_j - \eta_i - \delta_{i,j,\lambda}), \tag{14}$$

where M is a suitably large constant. Then, $\rho_{j,\lambda}$ must become 1 if the right side is larger than 0, i. e., if i and j are separated by more than $\delta_{i,j,\lambda}$ time steps. The parameter $\delta_{i,j,\lambda}$ can grow very large, however it is only relevant if a dependency to another job and ramping exist. We assume that the λ-th ramping step of job j must be executed λ time steps before the start of job

Table 3: Properties of the four sets of generated instances. Intervals $[a, b]$ indicate numbers chosen uniformly at random between a and b, inclusively.

Name	# Jobs	# Dep.	# Dep.with Drain	Net Job Slack
Jobs (Set A)	$\{50, 100, 150, 200, 250, 300\}$	$[0, 1000]$	0	$[0, 30]$
Dependencies (Set B)	200	$\{0, 100, 500, 1000, 2000, 3000\}$	0	$[0, 30]$
Drain (Set C)	200	1000	$\{0, 100, 200, 500, 900\}$	$[0, 30]$
Slack (Set D)	200	$[0, 10000]$	0	$\{1, 25, 50, 75, 100\}$
Slack w/few Dep. (Set E)	200	200	0	$\{1, 25, 50, 75, 100\}$

j. With this, the amount of power required for ramping job j at time step t can be formulated as

$$R_j(t) = \sum_{\lambda=1}^{\Lambda_j} \rho_{j,\lambda} \cdot s_j(t + \lambda) \cdot \mu_{j,\lambda}. \qquad (15)$$

Here, $R_j(t)$ becomes $\mu_{j,\lambda}$, i. e., the power for j's λ-th ramping step, if and only if j is started in time step $t + \lambda$ and $\rho_{j,\lambda}$, i.e., the indicator if the λ-th ramping step must be executed, is 1.

Total Power Requirement The total power requirement in the system at time step t is described as the sum over the power of all running jobs at time step t and the power used of the jobs currently ramping

$$\hat{P}(t) = \sum_i \left(P_i \sum_{t-T_i<t'\leq t} s_i(t') \right) + \sum_j R_j(t). \qquad (16)$$

Linearisation Some of the constraints described by us are not linear per se. See for example Equation (15), where $\rho_{j,\lambda}$ and $s_j(t)$ — both variables, not constants — are multiplied. However, for two binary variables a and b, such a multiplication can easily be linearised if the product contributes only positively to the objective function, i.e., if a solution where the product is 0 is preferred.

Let c be a third binary variable indicating whether the product $a \cdot b$ is 1. Then it is enough to introduce the constraint $c \geq a + b - 1$. We can replace $a \cdot b$ with c everywhere. If a and b are both 1, then c must be 1. In all other cases, c will be set to 0, since an optimum solution prefers the product to be 0.

5 Experimental Evaluation

We experimentally evaluate the MIP resulting from our modelling framework by generating random instances, running the MIP for 15 minutes and measuring the optimality gap, i.e., the gap between best feasible solution found and best shown lower bound. We evaluate the framework with peak shaving as objective function. This is due to cost minimization and appropriate weighting of the objectives being very problem specific and harder to generalize. We conducted all experiments on a machine with 16 Intel Xeon E5-2670 cores at 2.6 GHz and 64GB of RAM, using Gurobi 6.5 as a solver.[1]

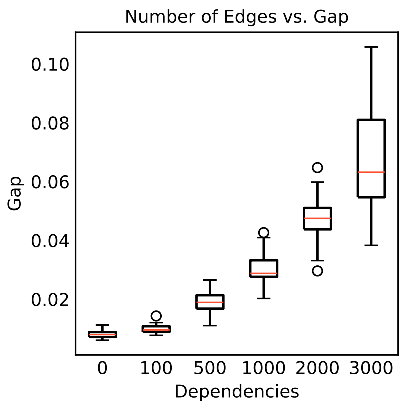
We generated five separate sets of instances. For each of the five sets, Table 3 shows the number of jobs, number of dependencies between two jobs, number of dependencies that are associated with a drain, and the (net) *slack* jobs have in the instances. The *slack* of a job is its deadline minus the release time minus the run time of the job. The slack gives an indication of the amount of freedom one has during scheduling. The *net* slack compensates for the fact that in the presence of dependencies, the earliest possible start time of a job does not just depend on the release time, but also on the start times of its predecessors. Thus, a lower bound for the earliest start time of a job is the maximum of all its predecessors' earliest start times plus their respective run times. The *net* slack takes this lower bound *and* the release time into account.

In Table 3, intervals like $[a, b]$ indicate that the value was chosen uniformly at random between a and b for every instance. A set like $\{a, b, c\}$ indicates that we generated instances for each of the values a, b and c. For each such value, we generated 30 instances, for a total of 810 instances. We set the objective for all instances to minimise the peak power requirement. The power requirement for every job has been drawn from a normal distribution with mean 5 and standard deviation 2.

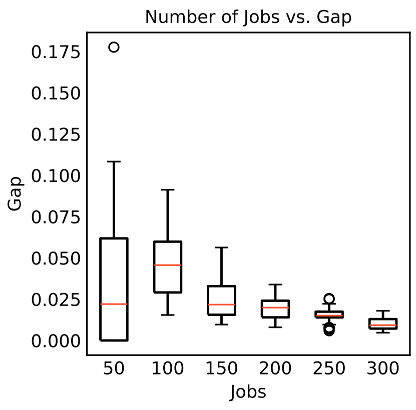
In the following, we analyse the *gap* between best found feasible solution and best lower bound. Formally, let C_{bound} be the cost of the best lower bound and C_{feasible} be the cost of the best found feasible solution, then the gap is defined as $1 - (C_{\text{bound}}/C_{\text{feasible}})$. For instances where no bound or no feasible solution was found, we set the gap to 1.

Figure 1a shows the effect of the number of dependencies on the gap achieved after 15 minutes. We can see that for up to 100 dependencies, all instances stay below a 2% gap. Even for 1000 dependencies, almost all instances can be solved up to a 4% gap. However, the

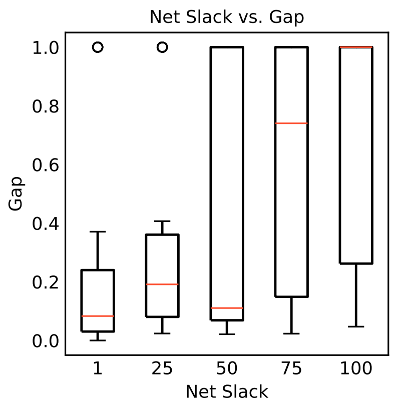
[1] http://www.gurobi.com



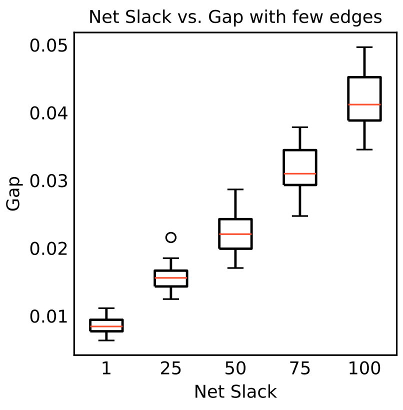
(a) Effect of varying the number of dependencies (Set B)



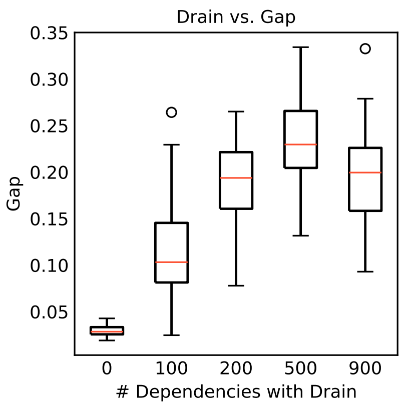
(b) Effect of varying the number of jobs (Set A)



(c) Effect of varying the amount of slack (Set D)



(d) Effect of varying the amount of slack with few dependencies (Set E)



(e) Effect of varying the number of dependencies with drain (Set C)

Fig. 1: The effect of varying different parameters in instance generation. Red lines indicate the median of all runs. The box indicates upper and lower quartile, i.e., 75% of all results lie below the upper end of the box, and 75% of all results lie above the lower end of the box. Whiskers show the extend of the remaining results, with outliers being shown as circles.

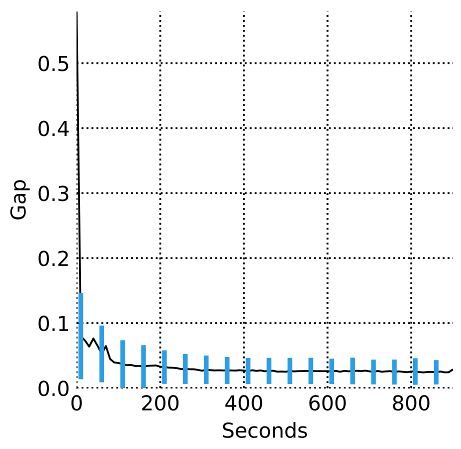


Fig. 2: Convergence speed of the MIP solutions. The black line indicates the median over all runs. The blue bars indicate the area in which 75% of all runs fall.

gap increases superlinearly with the number of dependencies.

In Figure 1b, we present the same plot for a varying number of jobs. A counterintuitive result is that while the gap first increases from 50 to 100 jobs, it decreases from there on. An explanation for this is the fact that the gap is a relative measure. As we keep adding jobs (keeping the global deadline and release time fixed), the absolute value of the optimum solution increases. A fixed (absolute) difference between the best feasible solution and the best lower bound becomes a lower gap as the optimum solution increases, which manifests here. However, note that even in the worst case, with 100 jobs, the majority of the instances could be solved to a gap of 5% or below.

Figure 1c shows the *net slack* of all jobs versus the achieved gap. It is visible that large net slacks strongly increase the computational complexity of the model.

Note that the mean duration of all jobs is 10, i.e., a net slack of 50 says that the net window of every job is already six times its duration. Furthermore, Figure 1d shows results of the same experiment where we kept the number of dependencies moderate, namely at 200. The gap again increases with the size of the net slack, however even for a net slack of 100, the gap never gets larger than 5%. Thus, a large number of dependencies combined with a lot of slack is what drives complexity here.

Regarding the effect of dependencies with drain, Figure 1e shows the gap for different numbers of dependencies with associated drains. As you can see from Table 3, we kept the number of dependencies constant at 1000 and vary the fraction of dependencies with drain. We can see that drain significantly raises the complexity of the model, even if just 10% of the dependencies are associated with drain. However, the complexity does not strictly increase with the number of dependencies with associated drain: If too many dependencies are incentivised to have jobs placed closely to each other, the flexibility in the model decreases and results improve slightly, as can be seen.

We finally take a look at the speed with which the MIP solver converges to the optimum solution in Figure 2. The black line shows the median of the achieved gap over all MIP runs at different points in time. The blue bars indicate the upper respectively lower quartiles. We can see that within the first 200 seconds, the MIP gap drops to below 10% on average. After 200 seconds further improvement is relatively slow.

Direct comparison with Petersen et al. (2014) Petersen et al. (2014) also give a MIP formulation of a problem which is a subset of the problem our framework can solve. They state that their MIP, executed on a standard laptop, was able to solve five out of twenty generated instances before hitting memory limits, and for the five solved instances, average execution time was eight minutes. We tried to generate twenty instances based on the same parameters as they did, i. e., ten instances each corresponding to their $\mathit{Portfolio}(25, 100)$ and $\mathit{Portfolio}(50, 100)$ settings. Petersen et al. (2014) define a $\mathit{Portfolio}(N, K)$ "*as a randomly generated portfolio of N local units with* $K_{\text{Run}} \in \{2, 3, 4, 5\}, \overline{P} \in \{1, 2, 3, 4\}$*, and* $K_{\text{End}} \in \{1, 2, \ldots, K\}$". A local unit is in there definition a flexible consumer, corresponding to a job in our formulation. Unfortunately, the authors do not state how P_{Dispatch}, described by $E_{prod}(t)$ in our formulation, is selected. For the given portfolio settings, the average power consumption over the (expected) optimisation period is 4.4 respectively 8.8, thus we selected $P_{\text{Dispatch}} = 5$ and $P_{\text{Dispatch}} = 9$. This should result in relatively difficult instances since, in an optimum solution, jobs must be distributed as uniformly as possible.

We solved these instances using our model on a standard laptop with 12 GB of RAM and a quadcore CPU running at 2.4 GHz. Gurobi was able to solve all instances to optimality within less than a second and a peak memory usage of less than 35 MB. The fast computation suggests that our framework indeed results in fairly tractable MIP models.

6 Discussion

We present a comprehensive modelling framework for demand side flexibility incorporating most of the characteristics from Table 1. In our implementation, we currently do not include the features multiple runs, down-/uptime and production. Without a specific production target, scheduling all jobs exactly once seems most fitting. This results in a fixed output for all possible schedules. However, we will extend the modelling framework and include the remaining flexibility constraints. Simultaneously, we plan to evaluate the representation and interdependencies of the individual constraints theoretically and with real-life case studies. In current research, it seems unclear what realistic test instances that cover a lot of possible real-life scenarios, look like. This is a topic for further research on its own.

Considering our goal to encompass as many applications as possible with our framework, it is questionable whether the cost function represented by the objective function (1) is linear in real applications, as we assumed so far. Logically, (marginal) production costs for the energy to be used would be increasing rapidly with small quantities and evening out the larger the volume. However, a realistic cost function has to be found to every case study according to the real (marginal) production costs of the case study. Thus, we use the linear cost function as a substitute and emphasise that it has to be adapted to specific use cases.

Additionally, further research should investigate the optimal degree of flexibility in production processes. In our model, flexibility has zero marginal costs. However, providing a particular level of flexibility usually incurs a certain amount of costs and resources that need to be considered. As generating and providing energy usually incurs production costs, the unused self-produced energy also needs to be further considered. Therefore, non-utilization should be penalised in the optimisation problem. A solution approach to this is the direct inclusion of energy storage capacities. Energy storage can help out by saving the otherwise unused energy for a certain amount of time. Nevertheless, storage costs will also occur and need to be considered in the optimisation problem. Currently, we are only considering costs that occur for additional consumption of electricity meaning that we minimise the absolute area difference between production and consumption.

As we have discussed before, not all flexibility aspects are yet included in the numerical model even though we consider them in the mathematical model. However, we expect the remaining characteristics to be of lower computational complexity as those that we have already incorporated. Thus, their influence on the optimality gap and runtime should be smaller than the impacts of the characteristics we already evaluated in Section 5. Furthermore, our consideration of complexity is incomplete as we have not gradually changed complexity but evaluated inherently different scenarios. A complete evaluation of our model's complexity is subject to further research.

We also point out that, for now, we use Gurobi 6.5's standard configurations to solve the Mixed-Integer Linear Programs resulting from our modelling framework. These standard configurations work adequately for our random instances. However, we point out that tuning these could lead to improved solutions. This approach might become useful in time-critical real-life implementation scenarios.

7 Conclusion

In this paper, we present and evaluate a holistic modelling framework which allows the universal representation of demand side flexibility. Thus, we address a gap in the modelling of flexibility as current research has introduced a variety of models which are suitable for specific problem instances but neglect the characteristics of demand side flexibility for other applications. After an extensive review of existing literature, we aggregate a coherent list of demand side flexibility features from research. We then create a framework to integrate them into one consistent model. After introducing the modelling framework mathematically, it is evaluated using randomly created problem instances, and the performance is measured. We measure the performance as the occurring optimality gap and show that our model performs well computationally while considering a wide range of features. We focus on the minimization of externally procured energy and peak shaving. In future work, we will consider the economic implications of providing and investing in flexibility. Our model advances current research as it can be universally used to describe flexibility for different applications and improves the comparability of optimisation algorithms.

Acknowledgements We thank one anonymous reviewer for his extraordinarily constructive comments which helped us to improve the manuscript.

References

Alizadeh, M., A. Scaglione, A. Applebaum, G. Kesidis, and K. Levitt (2015). *Reduced-order load models for large populations of flexible appliances.* In: *IEEE Transactions on Power Systems*, Vol. 30, No. 4, pp. 1758–1774.

Allerding, F., M. Premm, P. K. Shukla, and H. Schmeck (2012). *Electrical Load Management in Smart Homes Using Evolutionary Algorithms.* In: *EvoCOP*. Ed. by J.-K. Hao and M. Middendorf. Vol. 7245. Lecture Notes in Computer Science. Berlin: Springer, pp. 99–110.

Ashok, S. (2006). *Peak-load management in steel plants.* In: *Applied Energy*, Vol. 83, No. 5, pp. 413–424.

Ashok, S. and R. Banerjee (2000). *Load-management applications for the industrial sector.* In: *Applied Energy*, Vol. 66, No. 2, pp. 105–111.

Castro, P., H. Matos, and A. Barbosa-Póvoa (2002). *Dynamic modelling and scheduling of an industrial batch system.* In: *Computers & Chemical Engineering*, Vol. 26, No. 4-5, pp. 671–686.

Denholm, P., E. Ela, B. Kirby, and M. Milligan (2010). *The role of energy storage with renewable electricity generation.* Tech. rep., 1–61.

Du, P. and N. Lu (2011). *Appliance commitment for household load scheduling.* In: *IEEE transactions on Smart Grid*, Vol. 2, No. 2, pp. 411–419.

Fehrenbach, D., E. Merkel, R. McKenna, U. Karl, and W. Fichtner (2014). *On the economic potential for electric load management in the German residential heating sector–An optimising energy system model approach.* In: *Energy*, Vol. 71, pp. 263–276.

Fink, J., J. L. Hurink, and A. Molderink (2014). *Mathematical modelling of devices and flows in energy systems.* Tech. rep.

Gärttner, J. (2016). *Group Formation in Smart Grids: Designing Demand Response Portfolios.* PhD thesis. Dissertation, Karlsruher Institut für Technologie (KIT), 2016.

Gärttner, J., C. M. Flath, and C. Weinhardt (2016). *Load shifting, interrupting or both? Customer portfolio composition in demand side management.* In: *Computational Management Science*. Springer, pp. 9–15.

Goebel, C. et al. (2014). *Energy Informatics.* In: *Business & Information Systems Engineering*, Vol. 6, No. 1, pp. 25–31.

Gottwalt, S., W. Ketter, C. Block, J. Collins, and C. Weinhardt (2011). *Demand side management—A simulation of household behavior under variable prices.* In: *Energy Policy*, Vol. 39, No. 12, pp. 8163–8174.

Gottwalt, S., J. Gärttner, H. Schmeck, and C. Weinhardt (2016). *Modeling and valuation of residential demand flexibility for renewable energy integration.* In: *IEEE Transactions on Smart Grid.*

Halvorsen, B. and B. M. Larsen (2001). *The flexibility of household electricity demand over time.* In: *Resource and Energy Economics*, Vol. 23, No. 1, pp. 1–18.

He, X., N. Keyaerts, I. Azevedo, L. Meeus, L. Hancher, and J.-M. Glachant (2013). *How to engage consumers in demand response: A contract perspective.* In: *Utilities Policy*, Vol. 27, pp. 108–122.

Luo, Z., R. Kumar, J. Sottile, and J. C. Yingling (1998). *An MILP Formulation for Load-Side Demand Control.* In: *Electric Machines & Power Systems*, Vol. 26, No. 9, pp. 935–949.

Meindl, B. and M. Templ (2012). *Analysis of commercial and free and open source solvers for linear optimization problems.* In: *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS.*

Mitra, S., I. E. Grossmann, J. M. Pinto, and N. Arora (2012). *Optimal production planning under time-sensitive electricity prices for continuous power-intensive processes.* In: *Computers & Chemical Engineering*, Vol. 38, pp. 171–184.

Moon, J.-Y. and J. Park (2014). *Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage.* In: *International Journal of Production Research*, Vol. 52, No. 13, pp. 3922–3939.

Oudalov, A., R. Cherkaoui, and A. Beguin (2007). *Sizing and Optimal Operation of Battery Energy Storage System for Peak Shaving Application.* In: *2007 IEEE Power Tech*, pp. 621–625.

Palensky, P. and D. Dietrich (2011). *Demand side management: Demand response, intelligent energy systems, and smart loads.* In: *IEEE transactions on industrial informatics*, Vol. 7, No. 3, pp. 381–388.

Paulus, M. and F. Borggrefe (2011). *The potential of demand-side management in energy-intensive industries for electricity markets in Germany.* In: *Applied Energy*, Vol. 88, No. 2, pp. 432–441.

Petersen, M. K., L. H. Hansen, J. Bendtsen, K. Edlund, and J. Stoustrup (2013). *A taxonomy for modeling flexibility and a computationally efficient algorithm for dispatch in Smart Grids.* In: *2013 American Control Conference (ACC)*, pp. 1150–1156.

— (2014). *Heuristic optimization for the discrete virtual power plant dispatch problem.* In: *IEEE Transactions on Smart Grid*, Vol. 5, No. 6, pp. 2910–2918.

Qureshi, F. A., T. T. Gorecki, and C. N. Jones (2014). *Model predictive control for market-based demand response participation.* In: *IFAC Proceedings Volumes*, Vol. 47, No. 3, pp. 11153–11158.

Schilling, G. and C. C. Pantelides (1996). *A simple continuous-time process scheduling formulation and a novel solution algorithm.* In: *Computers & Chemical Engineering*, Vol. 20, S1221–S1226.

Schleicher-Tappeser, R. (2012). *How renewables will change electricity markets in the next five years.* In: *Energy policy*, Vol. 48, pp. 64–75.

Scott, P., S. Thiébaux, M. Van Den Briel, and P. Van Hentenryck (2013). *Residential demand response under uncertainty.* In: *International Conference on Principles and Practice of Constraint Programming.* Springer, pp. 645–660.

Setlhaolo, D., X. Xia, and J. Zhang (2014). *Optimal scheduling of household appliances for demand response.* In: *Electric Power Systems Research*, Vol. 116, pp. 24–28.

Soares, A., Á. Gomes, and C. H. Antunes (2014). *Categorization of residential electricity consumption as a basis for the assessment of the impacts of demand response actions.* In: *Renewable and Sustainable Energy Reviews*, Vol. 30, pp. 490–503.

Sou, K. C., J. Weimer, H. Sandberg, and K. H. Johansson (2011). *Scheduling smart home appliances using mixed integer linear programming.* In: *2011 50th IEEE Conference on Decision and Control and European Control Conference.* Piscataway, NJ: IEEE, pp. 5144–5149.

Steurer, M., M. Miller, U. Fahl, and K. Hufendiek (2015). *Enabling demand side integration–assessment of appropriate information and communication technology infrastructures, their costs and possible impacts on the electricity system.* In: *SmartER Europe.*

Strbac, G. (2008). *Demand side management: Benefits and challenges.* In: *Energy Policy*, Vol. 36, No. 12, pp. 4419–4426.

Ströhle, P., E. H. Gerding, M. M. de Weerdt, S. Stein, and V. Robu (2014). *Online mechanism design for scheduling non-preemptive jobs under uncertain supply and demand.* In: *Proceedings of the 2014 AAMAS.* International Foundation for Autonomous Agents and Multiagent Systems, pp. 437–444.

Weidlich, A., H. Vogt, W. Krauss, P. Spiess, M. Jawurek, M. Johns, and S. Karnouskos (2012). *Decentralized intelligence in energy efficient power systems.* In: *Handbook of Networks in Power Systems I.* Springer, pp. 467–486.