Competitive Algorithms for Demand Response Management in Smart Grid

Vincent Chau[1] ⋆ and Shengzhong Feng[1] ⋆ and Nguyen Kim Thang[2] ⋆⋆

[1] Shenzhen Institutes of Advanced Technology, Academy of Sciences, Shenzhen, China
vincentchau@siat.ac.cn, sz.feng@siat.ac.cn

[2] IBISC, Univ Évry, Université Paris-Saclay, 91025, Évry, France
thang@ibisc.fr

Abstract. We consider a scheduling problem which abstracts a model of demand-response management in Smart Grid. In the problem, there is a set of unrelated machines and each job j (representing a client demand) is characterized by a release date, and a power request function representing its request demand at specific times. Each machine has an energy power function and the energy cost incurred at a time depends on the load of the machine at that time. The goal is to find a non-migration schedule that minimizes the total energy (over all times).

We give a competitive algorithm for the problem in the online setting where the competitive ratio depends (only) on the power functions of machines. In the setting with typical energy function $P(z) = z^{\nu}$, the algorithm is $\Theta(\nu^{\nu})$-competitive, which is *optimal* up to a constant factor. Our algorithm is *robust* in the sense that the guarantee holds for arbitrary request demands of clients. This enables flexibility on the choices of clients in shaping their demands — a desired property in Smart Grid.

We also consider a special case in offline setting in which jobs have unit processing time, constant power request and identical machines with energy function $P(z) = z^{\nu}$. We present a 2^{ν}-approximation algorithm for this case.

1 Introduction

Electrical Smart Grid is one of the major challenges in the 21st century [20]. It is a network of electricity distribution that promotes the traffic information between producers and consumers in order to adjust the electricity flow in real time, i.e. it aims to improve the journey of the electricity through information and communication technologies in contrast of the traditional power system. It has been raised in [6] that in the US power grid, 10% of all generation assets and 25% of distribution infrastructure are required for less than 400 hours per

⋆ Research supported by NSFC (n° 61433012, U1435215) and by Shenzhen basic research grant JCYJ20160229195940462.

⋆⋆ Research supported by the ANR project OATA n° ANR-15-CE40-0015-01, Hadamard PGMO and DIM RFSI.

year, which represents roughly 5% of the time [20]. A smart grid is a power grid system that optimizes the efficiency of the power generation, distribution and consumption, and eventually the storage, of the energy in order to coordinate the electric network, from the production to the consumer. It can be noticed that the power grid may not be efficient during the peak demand hour if the management of the smart grid is not well handled. Indeed, the cost of the electricity production can be high if there is a high demand, and the electricity suppliers may charge the consumer according to the generation cost. Therefore, the cost of the electricity can be different over time, intuitively we have a lower price during off-peak hours and a higher price during peak hours. That is why *demand response management* [10] has been studied in order to overcome this problem. The goal of each user is to minimize his own cost by requesting the electricity during off-peak hours and reduce the peak load while satisfying his demand. Thus, *demand response management* is essentially beneficial to the consumers.

In fact, it can be seen as a scheduling problem. Each user is a job with the same release date and deadline in which the job cannot be schedule before the release date, nor after the deadline. Furthermore, a user is defined by an electricity demand over time which can also be represented in the scheduling problem. Finally, we have a cost that will be charged to users depending on the load at each moment. The goal is to minimize the total cost while satisfying all demands. A more formal definition is given in the next section.

In this paper, we consider a general online scheduling problem which models the demand response management and we design algorithms towards the following main purposes of Smart Grid:

– Optimizing the energy consumption.
– Enabling customer choice and letting them react rationally.

1.1 Model Definition

We consider the following scheduling problem. We are given m unrelated machines and a set of n jobs. Here, machines represents different resources in a smart grid or different electrical sub-networks. Each job represents the demand of a client and the demand is customized by the client. Specifically, each job j is characterized by its release date r_j and an *arbitrary* power request function $h_{i,j,k} : \mathbb{N} \to \mathbb{R}^+$, meaning that if a job j starts at time k in machine i then its request demand at time t is $h_{i,j,k}(t)$. We denote the *execution* of job j as $s_{i,j,k}$ if job j is processed in machine i with the starting time k. In the problem, *migration* of jobs between machines is not allowed. (In other words, a job must be executed in exactly one machine.) It is a desired property in various systems since in case of migration, the communications and storage/reloading data of jobs are costly.

Given a schedule (executions of all jobs), the total *load* at time t in machine i is $\sum_{j,k} h_{i,j,k}(t)$ where the sum is taken over all job executions in machine i. The total *energy* is defined as $\sum_i \sum_t P_i\big(\sum_{j,k} h_{i,j,k}(t)\big)$ where P_i is an energy power function of machine i. Typically, $P_i(z) = z^{\nu_i}$ for some constant parameter

$\nu_i \geq 1$. In the problem, we consider P_i as *arbitrary* non-decreasing functions, and possibly non-convex. The goal is find a feasible schedule that minimize the total energy consumption over all times.

In the paper, we consider both offline and online settings. In the offline setting, the scheduler has the full knowledge on all parameters while in the online setting, jobs arrive over time and the scheduler is aware of jobs (and their parameters) only at their arrival time. The online setting is appropriate to the dynamic nature of the demand response management.

The presented model encompasses the previous ones [11] in literature. In the latter, jobs have release date r_j, deadline d_j, processing time $p_{i,j}$ and jobs have to be processed non-preemptively. The power request of a job j is some constant h_j during its non-preemptive execution. It is captured by the model by defining

$$h_{i,j,k}(t) = \begin{cases} h_j & \text{if } r_j \leq k \leq d_j - p_{ij} \text{ and } t \in [k, k+p_{ij}], \\ 0 & \text{if } r_j \leq k \leq d_j - p_{ij} \text{ and } t \notin [k, k+p_{ij}], \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

Geometrically, in the model as shown in Equation (1), each job corresponds to a rectangle and the problem essentially consists of packing rectangles to minimize the total energy. In this case, the power request is constant from the beginning to the end of the request. For short, we call the model defined by Equation (1) *rectangle scheduling*. In our model, there is no condition on the demand (i.e., energy request) of jobs and the demands can be specifically customized by clients. Geometrically, each job in our model has an *arbitrary* (not necessarily continuous) form which represents varying power requests during its execution (See Figure 1). Hence, the model offers flexible choices to clients along the line of Smart Grid's purposes.

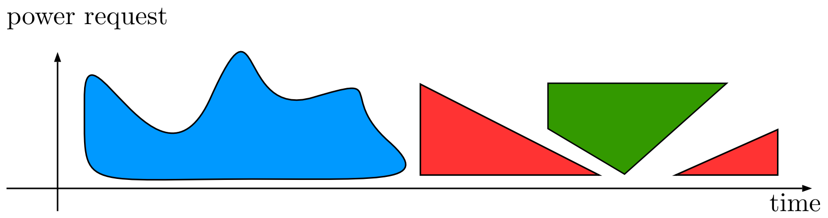

Fig. 1. Example of a schedule with arbitrary power requests during execution of jobs.

1.2 Related Works.

In this section, we summarize related works in the model of *rectangle scheduling* which, to the best of our knowledge, is the only one that has been studied so far.

Koutsopoulos and Tassiulas [11] formulated the rectangle scheduling model where the cost function is piecewise linear. They show that the problem is NP-hard, and it can further be adapted to show the NP-hardness of the general problem where the cost function is convex [5]. In the offline setting, Burcea et al. [5] gave a polynomial time algorithms for the case of unit height (i.e., unit power request) and unit width (i.e., duration of request). Furthermore, in the full version of [13] (see [14]), Liu et al. showed that the offline case, where jobs have unit processing time but with arbitrary power request, admits a $2^{\nu+1}$-approximation algorithm which is based on the results of the the dynamic speed scaling problem [1], [4], [21].

In the online setting, [9] proposed a simple greedy algorithm which is 2-competitive for the unit case and the power function is z^2. However, [13] showed that the greedy algorithm is in fact at least 3-competitive by providing a counter example. In the same paper, Liu et al. [13] considered the single machine setting in which they presented an online $\Theta\left(\log^{\nu}\left(\frac{p_{\max}}{p_{\min}}\right)\right)$-competitive algorithm where $p_{\min} = \min_j\{p_j : p_j > 0\}$ and $p_{\max} = \max_j p_j$. This is the best known algorithm (even in offline setting) where jobs have arbitrary width, arbitrary height and the power energy function is z^{ν}. Futhermore, for special cases of jobs with unit processing time, Liu et al. [13] also gave competitive algorithms. A summary of the results can be found in Table 1.

Processing time p_j	Power request h_j	Prior best-known results	Our result
unit	uniform (i.e., $h_j = h_{j'}$)	$\min\{(4\nu)^{\nu}/2+1, 2^{\nu}(8e^{\nu}+1)\}$ [13]	
	arbitrary	$2^{\nu}(8e^{\nu}+1)$ [13]	
arbitrary	arbitrary	$O\left(\log^{\nu}\left(\frac{p_{\max}}{p_{\min}}\right)\right)$ [13] $\Omega\left(\nu^{\nu}\right)$ [13]	$\mathbf{\Theta}(\nu^{\nu})$

Table 1. Summary of competitive ratios in the rectangle scheduling model with power function z^{ν} for a *single* machine. Our result holds for *unrelated machines*.

Besides, Salinas et al. [18] considered a multi-objective problem to minimize energy consumption cost and maximize some utility that can be the profit for the operator as well as for the clients. On the other hand, a related problem is to manage the load by considering different price of electricity over time [8], [16]. Recent surveys of the area can be found in [10], [15], [2].

1.3 Our contribution and Approaches

In this paper, we investigate the online and offline aspects of the problem.

Online Setting. The main result of the paper is a competitive algorithm for the problem in online setting where the competitive ratio is characterized by a notion called *smoothness* [17], [19] of the machine energy power functions. Informally, the algorithm assigns and executes each job that arrives on a machine in such a way that minimizes the marginal increase of the total cost.

In designing a competitive algorithm for the problem, we consider a primal-dual approach. The main difficulty in proving the performance of the algorithm is that all known LPs has unbounded integrality gap, even for the special case of rectangle scheduling. Intuitively, the drawback of all known LPs is that in the optimal fractional solution, jobs are fractionally assigned to machines while in the optimal integer solution, migration of jobs is not allowed. To bypass this obstacle, we consider the primal-dual framework based on configuration linear programs in [19]. The framework is presented in order to reduce the integrality gap and also to study problems with non-linear, non-convex objective functions. The approach is particularly useful since the energy power functions are non-linear. Employing the techniques from [19], we derive a greedy algorithm with competitive ratio characterized by the notion of *smoothness*, which is defined as follows.

Definition 1. *A function $f : \mathbb{R}^+ \to \mathbb{R}^+$ is* (λ, μ)-smooth *if for any sets of non-negative numbers $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$, the following inequality holds:*

$$\sum_{i=1}^{n} \left[f\Big(a_i + \sum_{j=1}^{i} b_j\Big) - f\Big(\sum_{j=1}^{i} b_j\Big) \right] \le \lambda \cdot f\Big(\sum_{i=1}^{n} a_i\Big) + \mu \cdot f\Big(\sum_{i=1}^{n} b_i\Big)$$

A set of cost functions $\{f_e : e \in \mathcal{E}\}$ is (λ, μ)-smooth *if every function f_e is* (λ, μ)-smooth.

Specifically, in the problem, assuming that all energy power functions are (λ, μ)-*smooth* for some $\lambda > 0$ and $0 < \mu < 1$, our algorithm is $\lambda/(1-\mu)$-competitive. For energy power functions of forms $P_i(z) = z^{\nu_i}$, they are $\big(O(\nu^{\nu-1}), \frac{\nu-1}{\nu}\big)$-smooth where $\nu = \max_i \nu_i$. That leads to the competitive ratio $O(\nu^\nu)$ which improves upon the best-known $\Theta\left(\log^{\nu}\left(\frac{p_{\max}}{p_{\min}}\right)\right)$-competitive algorithm where $p_{\min} = \min_j \{p_j : p_j > 0\}$ and $p_{\max} = \max_j p_j$. Our competitive ratio is not only independent on the jobs' parameters but it is indeed *optimal* up to a constant factor. The matching lower bound is given by [13, Theorem 9] for a single machine in the rectangle scheduling model. In particular, [13] gave a lower bound which is $\frac{1}{3}\big(\log \frac{p_{\max}}{p_{\min}}\big)^{\nu}$ where $\log \frac{p_{\max}}{p_{\min}} = \nu$.

Our greedy algorithm has several interesting features toward the purposes of Smart Grid. First, the algorithm is simple and easy to implement which makes it practically appealing. Note that despite the simplicity of our algorithm, no bounded competitive ratio has been known even for the rectangle scheduling model. Second, the algorithm performance guarantee holds for jobs with arbitrary varying power requests (arbitrary forms). Apart of answering open questions raised in [13], it is particularly useful for the demand response management. Once the algorithm is publicly given and clients are charged accordingly to the marginal increase of the total energy cost, clients can *arbitrarily* customized their demand in order to minimize their payment. This property is desirable since it enables the clients to react rationally. In the side of the smart grid management, no modification in the algorithm is needed while always maintaining the competitiveness (optimality in case of typical energy functions).

Offline Setting. In offline setting, we give an improved 2^{ν}-approximation algorithm when jobs have unit processing time. This result improves upon the $2^{\nu+1}$-approximation algorithm given by Liu et al. [14] in two aspects. First, it slightly improves the competitive ratio. Secondly, our result holds for multiple (identical) machine environment. Our algorithm makes use of the approximation algorithm for scheduling problems with convex norm objective functions given by [3]. The latter is designed by solving a convex relaxation and round to an integer solution using the Lenstra-Shmoys-Tardos scheme [12].

2 A Competitive Online Algorithms

Formulation. In the model, the execution of a job is specified by two parameters: (1) a machine in which it is executed; and (2) a starting time. Note that these parameters fully represent the demand of a job, including the power request at any time t during its execution. Formally, we denote the *execution* of job j as $s_{i,j,k}$ if job j is processed in machine i with the starting time k. Recall that if the execution of a job j is $s_{i,j,k}$ then the request demand of the job at time t is $h_{i,j,k}(t)$. Let $\mathcal{S}_j$ be a set of feasible executions of job j. For example, in the rectangle scheduling model, $\mathcal{S}_j$ consists of $s_{i,j,k}$ for all machines i and starting time k such that $r_j \leq k \leq d_j - p_{ij}$. As the set of machines and times[3] are finite, so is the set $\mathcal{S}_j$ for every job j. Let $x_{i,j,k}$ be a variable indicating whether the execution of job j is $s_{i,j,k} \in \mathcal{S}_j$. We say that A is a *scheduling configuration* (configuration in short) in machine i if A is a feasible schedule of a subset of jobs. Specifically, A consists of tuples (i, j, k) meaning that the execution of job j is $s_{i,j,k}$. For a scheduling configuration A and machine i, let $z_{i,A}$ be a variable such that $z_{i,A} = 1$ if and only if for every tuple $(i, j, k) \in A$, we have $x_{i,j,k} = 1$. In other words, $z_{i,A} = 1$ if and only if the schedule in machine i follows *exactly* the configuration A. Given a scheduling configuration A, let $A(t)$ be the load (height) of the corresponding schedule at time t. We denote the energy cost of a configuration A of machine i as $c_i(A) := \sum_t P_i(A(t))$. We consider the following formulation and the dual of its relaxation.

[3] For convenience, we consider schedules up to a time T, which can be arbitrarily large but finite

$$
\begin{aligned}
&\min \sum_{i,A} c_i(A) z_{i,A} \\
&\sum_{i,k: s_{i,j,k} \in \mathcal{S}_j} x_{i,j,k} = 1 && \forall j \\
&\sum_{A:(i,j,k) \in A} z_{i,A} = x_{i,j,k} && \forall i,j,k \\
&\sum_{A} z_{i,A} = 1 && \forall i \\
&x_{i,j,k}, z_{i,A} \in \{0,1\} && \forall i,j,k,A
\end{aligned}
\qquad
\begin{aligned}
&\max \sum_j \alpha_j + \sum_i \gamma_i \\
&\alpha_j \leq \beta_{i,j,k} && \forall i,j,k \\
&\gamma_i + \sum_{(i,j,k) \in A} \beta_{i,j,k} \leq c_i(A) && \forall i, A
\end{aligned}
$$

In the primal, the first constraint guarantees that a job j has to be processed by some feasible execution (in some machine). The second constraint ensures that if job j follows the execution $s_{i,j,k}$ then in the solution, the scheduling configuration of machine i must contain the tuple (i,j,k) corresponding to execution $s_{i,j,k}$. The third constraint says that in the solution, there is always a scheduling configuration (possibly empty set) associated to machine i.

Algorithm. We first interpret intuitively the dual variables, dual constraints and derive useful observations for a competitive algorithm. Variable α_j represents the increase of energy to the arrival of job j. Variable $\beta_{i,j,k}$ stands for the marginal energy if job j follows execution $s_{i,j,k}$. By this interpretation, the first dual constraint clearly indicates the greedy behavior of an algorithm. That is, if a new job j is released, select an execution $s_{i,j,k} \in \mathcal{S}_j$ that minimizes the marginal increase of the total energy.

Formally, let A_i^* be the set of current schedule of machine i and initially, $A_i^* \leftarrow \emptyset$ for every machine i. At the arrival of job j, select an execution $s_{i^*,j,k^*} \in \mathcal{S}_j$ such that

$$s_{i^*,j,k^*} \in \arg \min_{s_{i,j,k} \in \mathcal{S}_j} \left[c_i(A_i^* \cup s_{i,j,k}) - c_i(A_i^*) \right]$$

or equivalently,

$$s_{i^*,j,k^*} \in \arg \min_{s_{i,j,k} \in \mathcal{S}_j} \sum_t \left[P_i\Big(A_i^*(t) + h_{i,j,k}(t) \Big) - P_i\Big(A_i^*(t) \Big) \right]$$

where $(A_i^* \cup s_{i,j,k})$ is the current schedule with additional execution $s_{i,j,k}$ of job j. Note that in configuration $(A_i^* \cup s_{i,j,k})$, the load at time t in machine i is $P_i\big(A_i^*(t) + h_{i,j,k}(t)\big)$. Then assign job j to machine i^* and process it according to the corresponding execution of s_{i^*,j,k^*}.

Dual variables. Assume that all energy power functions P_i are (λ, μ)-smooth for some fixed parameters $\lambda > 0$ and $\mu < 1$, then we construct a dual feasible solution in the following way. Let $A_{i,\prec j}^*$ be the scheduling configuration of machine i (due to the algorithm) prior to the arrival of job j. Define α_j as $1/\lambda$ times the the increase of the total cost due to the arrival of job j. In other words, if the algorithm selects the execution s_{i^*,j,k^*} for job j then

$$\begin{aligned}\alpha_j &= \frac{1}{\lambda}\Big[c_{i^*}(A^*_{i^*,\prec j} \cup s_{i^*,j,k^*}) - c_{i^*}(A^*_{i^*,\prec j})\Big] \\ &= \frac{1}{\lambda}\sum_t \Big[P_{i^*}\Big(A^*_{i^*,\prec j}(t) + h_{i^*,j,k^*}(t)\Big) - P_{i^*}\Big(A^*_{i^*,\prec j}(t)\Big)\Big]\end{aligned}$$

For each machine i and job j, we set

$$\begin{aligned}\beta_{i,j,k} &= \frac{1}{\lambda}\Big[c_i(A^*_{i,\prec j} \cup s_{i,j,k}) - c_i(A^*_{i,\prec j})\Big] \\ &= \frac{1}{\lambda}\sum_t \Big[P_i\Big(A^*_{i,\prec j}(t) + h_{i,j,k}(t)\Big) - P_i\Big(A^*_{i,\prec j}(t)\Big)\Big].\end{aligned}$$

Finally, for every machine i, we define the dual variable

$$\gamma_i = -\frac{\mu}{\lambda}c_i(A^*_i)$$

where A^*_i is the schedule on machine i (at the end of the instance).

Lemma 1. *The dual variables defined as above are feasible.*

Proof. By the definition of dual variables, the first constraint reads

$$\frac{1}{\lambda}\Big[c_{i^*}(A^*_{i^*,\prec j} \cup s_{i^*,j,k^*}) - c_{i^*}(A^*_{i^*,\prec j})\Big] \leq \frac{1}{\lambda}\Big[c_i(A^*_{i,\prec j} \cup s_{i,j,k}) - c_i(A^*_{i,\prec j})\Big]$$

This inequality follows immediately the the choice of the algorithm.

We now show that the second constraint holds. Fix a machine i and an arbitrary configuration A on machine i. The corresponding constraint reads

$$\begin{aligned} & -\frac{\mu}{\lambda}c_i(A^*_i) + \frac{1}{\lambda}\sum_{(i,j,k)\in A}\Big[c_i(A^*_{i,\prec j} \cup s_{i,j,k}) - c_i(A^*_{i,\prec j})\Big] \leq c_i(A) \\ \Leftrightarrow \quad & \sum_{(i,j,k)\in A}\Big[c_i(A^*_{i,\prec j} \cup s_{i,j,k}) - c_i(A^*_{i,\prec j})\Big] \leq \lambda c_i(A) + \mu c_i(A^*_i) \\ \Leftrightarrow \quad & \sum_{(i,j,k)\in A}\sum_t\Big[P_i(A^*_{i,\prec j}(t) + h_{i,j,k}(t)) - P_i(A^*_{i,\prec j}(t))\Big] \\ & \qquad \leq \lambda\sum_t P_i(A(t)) + \mu\sum_t P_i(A^*_i(t)) \qquad (2)\end{aligned}$$

where $A^*_{i,\prec j}(t)$ is the load (height) of machine i (due to the algorithm) at time t before the arrival of job j.

Observe that $A^*_{i,\prec j}(t)$ is the sum of power requests (according to the algorithm) at time t of jobs assigned to machine i prior to job j. As the power function P_i is (λ,μ)-smooth, for any time t we have

$$\sum_{(i,j,k)\in A}\Big[P_i\big(A^*_{i,\prec j}(t)+h_{i,j,k}(t)\big)-P_i\big(A^*_{i,\prec j}(t)\big)\Big] \leq \lambda P_i\bigg(\sum_{(i,j,k)\in A} h_{i,j,k}(t)\bigg)+\mu P_i\big(A^*_i(t)\big)$$

Summing over all times t, Inequality (2) holds. Therefore, the lemma follows.

We are now ready to prove the main theorem.

Theorem 1. *If all energy power functions are (λ,μ)-smooth, then the algorithm is $\lambda/(1-\mu)$-competitive. In particular, if $P_i(z)=z^{\nu_i}$ for $\nu_i\geq 1$ then the algorithm is $O(\nu^\nu)$-competitive where $\nu=\max_i \nu_i$.*

Proof. By the definitions of dual variables, the dual objective is

$$\sum_j \alpha_j+\sum_i \gamma_i=\sum_i \frac{1}{\lambda}c_i(A^*_i)-\sum_i\frac{\mu}{\lambda}c_i(A^*_i)=\frac{1-\mu}{\lambda}\sum_i c_i(A^*_i)$$

Besides, the cost of the solution due to the algorithm is $\sum_i c_i(A^*_i)$. Hence, the competitive ratio is at most $\lambda/(1-\mu)$.

Particularly, energy power functions of forms $P_i(z)=z^{\nu_i}$ for $\nu_i\geq 1$ are $\big(O(\nu^{\nu-1}),\frac{\nu-1}{\nu}\big)$-smooth for $\nu=\max_i \nu_i$. In fact, the smoothness follows (smooth) inequalities in [7], which states: for $\nu>1$ and for any sets of non-negative numbers $A=\{a_1,\ldots,a_n\}$ and $B=\{b_1,\ldots,b_n\}$, it always holds that

$$\sum_{i=1}^n\left[\bigg(a_i+\sum_{j=1}^i b_j\bigg)^\nu-\bigg(\sum_{j=1}^i b_j\bigg)^\nu\right]\leq O(\nu^{\nu-1})\cdot\bigg(\sum_{i=1}^n a_i\bigg)^\nu+\frac{\nu-1}{\nu}\cdot\bigg(\sum_{i=1}^n b_i\bigg)^\nu$$

That implies the competitive ratio ν^ν of the algorithm for power functions $P_i(z)=z^{\nu_i}$. $\square$

3 An Approximation Algorithm for Unit Processing Time Jobs

In this section, we investigate the offline case in identical machine environment where jobs have unit processing time but different power requests on different machines. Note that this corresponds to the restricted model of *rectangle scheduling*. We consider typical energy power function $P(z)=z^\nu$ for every machine and we assume that jobs need to be assigned to time-slot. This problem is proved to be NP-hard by a reduction to the 3-Partition problem even for the case where jobs have common release time and common deadline [5].

Let $\Theta = \cup_{j=1}^{n}\{r_j + a \mid a = -n, \ldots, n\} \cup_{j=1}^{n} \{d_j + a \mid a = -n, \ldots, n\}$ to be a set of time-slots. We show that it is sufficient to consider only schedules in which jobs are processed within these time-slots. In particular, this set contains $O(n^2)$ time-slots and will help to design a polynomial time approximation algorithm.

Lemma 2. *The schedules in which jobs start at a date in Θ are* dominant. *In other words, any schedule can be transformed to one in which jobs start at a date in Θ without increasing the cost.*

Due to space constraint, the proof can be found in the appendix.

The main idea is to reduce the smart grid problem to the following L_ν-*norm problem.* In the latter, we are given a set $\mathcal{J}$ of n jobs and a set $\mathcal{M}$ of m unrelated machine. Each job $j \in \mathcal{J}$ have a processing time $p_{i,j}$ if it is assigned to machine i. We define the decision variable $y_{i,j} = 1$ if the job j is assigned to machine i, and $y_{i,j} = 0$ otherwise. The goal is to minimize the following function:

$$\sqrt[\nu]{\sum_{i \in \mathcal{M}} \left(\sum_{j=1}^{n} y_{i,j} p_{i,j} \right)^{\nu}} \tag{3}$$

Lemma 3. *The problem of smart grid with unit processing time jobs and identical machines can be polynomially reduced to the L_ν-norm minimization problem on unrelated machines.*

Proof. By Lemma 2, there is a polynomial number of time-slots to which jobs can be assigned. We create a corresponding machine (i, t) for each time-slot $t \in \Theta$ and each machine i. Similarly, we create a new job $j' \in \mathcal{J}$ (in L_ν-norm problem) which corresponds to job $j \in J$ (in the smart grid problem) in the following way:

$$p_{(i,t),j'} = \begin{cases} h_j & \text{if } t \in [r_j, d_j) \\ +\infty & \text{otherwise} \end{cases} \tag{4}$$

Given a schedule for the L_ν-norm problem with cost C, we show how to build a feasible schedule for the smart grid problem with cost C^ν.

For each job $j \in \mathcal{J}$ that is assigned to machine $(i, t) \in \mathcal{M}$ in the L_ν-norm problem, we schedule this job at the time-slot t on machine i in the initial problem. By doing that, the load at any time-slot t on machine i in the initial problem equals the load of the machine (i, t) in the L_ν-norm problem. Therefore, the constructed schedule for the initial problem has cost C^ν where C is the cost of the schedule in the L_ν-norm problem. □

By Lemma 3, solving the smart grid problem with unit processing time jobs and identical machines is essentially solving the L_ν-norm problem. Hence, in our algorithm (Algorithm 1), we invoke the Azar-Epstein algorithm [3] in order to get an approximation algorithm for the latter. Roughly speaking, the Azar-Epstein algorithm consists of solving a relaxed convex program and rounding fractional solutions to integral ones using the standard scheme of Lenstra, Shmoys and Tardos [12]. Given a solution for the L_ν-norm problem, we reconstruct a feasible solution for the smart grid problem with approximation ratio of 2^ν.

Algorithm 1 Approximation algorithm for the smart grid scheduling problem with unit processing time jobs and identical machines

```
1: $\Theta = \cup_{j=1}^{n}\{r_j + a \mid a = -n, \dots, n\} \cup_{j=1}^{n} \{d_j + a \mid a = -n, \dots, n\}$
2: Let $\Pi = \emptyset$ be the set of machines and $\mathcal{J} = \emptyset$ be the set of jobs
3: for each $t \in \Theta$ and each machine $i$ do
4:     Create a machine $(i, t)$ and $\Pi \leftarrow \Pi \cup \{(i, t)\}$
5: end for
6: for each job $j$ do
7:     Create a new job $j'$ with $p_{(i,t),j'} = h_j$ if $t \in [r_j, d_j)$, otherwise we have $p_{(i,t),j'} = +\infty$
8:     $\mathcal{J} \leftarrow \mathcal{J} \cup \{j'\}$
9: end for
10: Apply the Azar-Epstein algorithm [3] on instance $(\Pi, \mathcal{J})$.
11: Build the schedule for the smart grid problem as in Lemma 3.
```

Theorem 2. *Algorithm 1 achieves an approximation ratio of* 2^ν.

Proof. By Lemma 3, we know that given an assignment of jobs for the L_ν-norm problem on unrelated machines of cost C, we can construct a schedule for the smart grid problem with a cost of C^ν in polynomial time. Thus we have $(OPT_L)^\nu = OPT_{SG}$ where OPT_L is the optimal cost of the L_ν-norm problem and OPT_{SG} is the optimal cost of the smart grid problem.

Besides, Azar-Epstein algorithm [3] is 2-approximation for the L_ν-norm problem. Therefore, we have $OPT_L \leq C \leq 2OPT_L$. Finally, by raising each term of the inequality by a power of ν, we have $(OPT_L)^\nu \leq C^\nu \leq 2^\nu(OPT_L)^\nu$, so $OPT_{SG} \leq C^\nu \leq 2^\nu OPT_{SG}$. The theorem follows. □

4 Concluding Remarks

In the paper, we have considered a general model of demand-response management in Smart Grid. We have given a competitive algorithm which is optimal (up to a constant factor) in typical settings. Our algorithm is robust to arbitrary demands and so enables the flexibility on the choices of clients in shaping their demands. The paper gives rise to several directions for future investigations. First, in the scheduling aspect, it would be interesting to consider problems in the general model with additional requirements such as precedence constraints, etc. Secondly, in the game theory aspect, designing pricing schemes that allow clients to react rationally while maintaining the efficiency in the energy consumption has received particular interests from both theoretical and practical studies in Smart Grid. Through the primal-dual view point, dual variables can be interpreted as the payments of clients. An interesting direction is to design a pricing scheme based on primal-dual approaches.

Acknowledgement. We thank Prudence W. H. Wong for insightful discussions and anonymous reviewers for useful comments that helps to improve the presentation.

References

1. Albers, S.: Energy-efficient algorithms. Commun. ACM 53(5), 86–96 (2010)
2. Alford, R., Dean, M., Hoontrakul, P., Smith, P.: Power systems of the future: The case for energy sotrage, distributed generation, and microgrids. Zpryme Research & Consulting, Tech. Rep (2012)
3. Azar, Y., Epstein, A.: Convex programming for scheduling unrelated parallel machines. In: Proc. 37th Annual ACM Symposium on Theory of Computing. pp. 331–337 (2005)
4. Bell, P.C., Wong, P.W.H.: Multiprocessor speed scaling for jobs with arbitrary sizes and deadlines. J. Comb. Optim. 29(4), 739–749 (2015)
5. Burcea, M., Hon, W., Liu, H.H., Wong, P.W.H., Yau, D.K.Y.: Scheduling for electricity cost in a smart grid. J. Scheduling 19(6), 687–699 (2016)
6. Chen, C., Nagananda, K., Xiong, G., Kishore, S., Snyder, L.V.: A communication-based appliance scheduling scheme for consumer-premise energy management systems. IEEE Transactions on smart Grid 4(1), 56–65 (2013)
7. Cohen, J., Dürr, C., Thang, N.K.: Smooth inequalities and equilibrium inefficiency in scheduling games. In: International Workshop on Internet and Network Economics. pp. 350–363 (2012)
8. Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W.: Scheduling on a single machine under time-of-use electricity tariffs. Annals OR 238(1-2), 199–227 (2016)
9. Feng, X., Xu, Y., Zheng, F.: Online scheduling for electricity cost in smart grid. In: COCOA. Lecture Notes in Computer Science, vol. 9486, pp. 783–793. Springer (2015)
10. Hamilton, K., Gulhar, N.: Taking demand response to the next level. IEEE Power and Energy Magazine 8(3), 60–65 (2010)
11. Koutsopoulos, I., Tassiulas, L.: Control and optimization meet the smart power grid: scheduling of power demands for optimal energy management. In: e-Energy. pp. 41–50. ACM (2011)
12. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical programming 46(1), 259–271 (1990)
13. Liu, F., Liu, H.H., Wong, P.W.H.: Optimal nonpreemptive scheduling in a smart grid model. In: Proc. 27th Symposium on Algorithms and Computation. vol. 64, pp. 53:1–53:13 (2016)
14. Liu, F., Liu, H.H., Wong, P.W.H.: Optimal nonpreemptive scheduling in a smart grid model. CoRR abs/1602.06659 (2016), http://arxiv.org/abs/1602.06659
15. Lui, T.J., Stirling, W., Marcy, H.O.: Get smart. IEEE Power and Energy Magazine 8(3), 66–78 (2010)
16. Maharjan, S., Zhu, Q., Zhang, Y., Gjessing, S., Basar, T.: Dependable demand response management in the smart grid: A stackelberg game approach. IEEE Trans. Smart Grid 4(1), 120–132 (2013)
17. Roughgarden, T.: Intrinsic robustness of the price of anarchy. Journal of the ACM 62(5), 32 (2015)
18. Salinas, S., Li, M., Li, P.: Multi-objective optimal energy consumption scheduling in smart grids. IEEE Trans. Smart Grid 4(1), 341–348 (2013)
19. Thang, N.K.: Online primal-dual algorithms with configuration linear programs. CoRR abs/1708.04903 (2017)
20. US Department of Energy: The smart grid: An introduction. https://energy.gov/oe/downloads/smart-grid-introduction-0 (2009)
21. Yao, F.F., Demers, A.J., Shenker, S.: A scheduling model for reduced CPU energy. In: FOCS. pp. 374–382. IEEE Computer Society (1995)

Appendix

A Missing proof from Section 3

Lemma 2. *The schedules in which jobs start at a date in Θ are* dominant. *In other words, any schedule can be transformed to one in which jobs start at a date in Θ without increasing the cost.*

Proof. It is sufficient to consider a machine and show how to transform the schedule of the machine to the new one such that each job starts at a date in Θ without increasing the cost.

Let t be the first moment where jobs that are assigned to this time-slot does not belong to Θ. We consider the maximal continuous interval from time-slot t in which every time-slot has at least one job that is assigned. If the considered interval is $[t, u)$, then the time-slot $u + 1$ is idle.

First, we observe that the length of this interval is lower or equal to n. Indeed, in the worst case, each job is assigned to different time-slot. We shift this interval, as well as the jobs, by one unit time to the right, i.e. after the shift, the interval will be $[t + 1, u + 1)$.

Three cases may occur (see Figure 2):

– we reach another job. We then consider the new maximal continuous interval and continue to shift it.
– we reach a deadline. The starting time of the interval must be in Θ since the length of the interval is at most n.
– none of the above cases, we continue to shift the interval to the right.

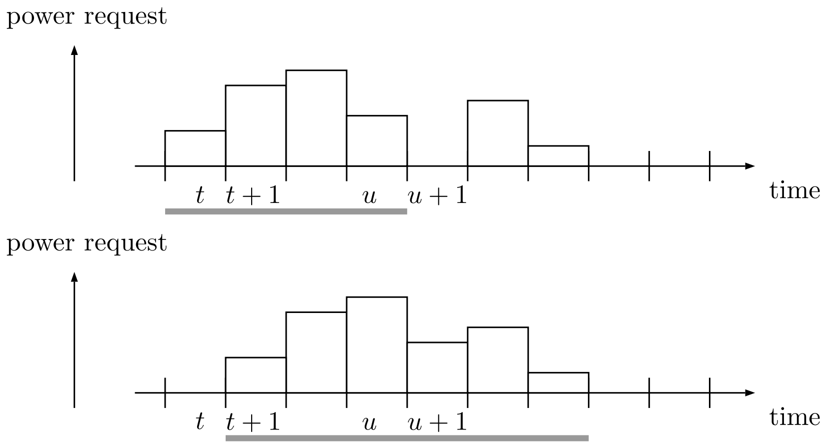

Fig. 2. Illustration of a shift of an interval. After the shift, the former interval meet another job. We then need to consider the continuous interval from time-slot $t + 1$. It corresponds to the first case in the proof of Lemma 2.

By this operation, we observe that the cost of the schedule remains the same because the costs of time-slots are independent. By doing a such modification, jobs are executed at the same way, with the same order and with the same cost, the only difference is the time-slots in which the jobs are executed. □