Demand-Side Load Scheduling Incentivized by Dynamic Energy Prices

Hadi Goudarzi, Safar Hatami, and Massoud Pedram
Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089
{*hgoudarz, shatami, pedram*}*@usc.edu*

Abstract—**Demand response is an important part of the smart grid technologies. This is a particularly interesting problem with the availability of dynamic energy pricing models. Electricity consumers are encouraged to consume electricity more prudently in order to minimize their electric bill, which is in turn calculated based on dynamic energy prices. In this paper, task scheduling policies that help consumers minimize their electrical energy cost by setting the time of use (TOU) of energy in the facility. Moreover, the utility companies can reasonably expect that their customers reduce their consumption at critical times in response to higher energy prices during those times. These policies target two different scenarios: (i) scheduling with a TOU-dependent energy pricing function subject to a constraint on total power consumption; and (ii) scheduling with a TOU and total power consumption-dependent pricing function for electricity consumption. Exact solutions (based on Branch and Bound) are presented for these task scheduling problems. In addition, a rank-based heuristic and a force directed-based heuristic are presented to efficiently solve the aforesaid problems. The proposed heuristic solutions are demonstrated to have very high quality and competitive performance compared to the exact solutions. Moreover, ability of demand shaping utilizing the aforementioned pricing schemes is demonstrated by the simulation results.**

I. Introduction

Availability of affordable and sustainable electrical energy is the key to prosperity and continued socio-economic growth of nations and the world. Traditional static centralized infrastructure of electricity grid consists of: (i) a global transmission network, which transmits the electrical power generated at remote power plants through long-distance high-voltage lines to substations, and (ii) a local distribution network, which adapts and delivers electricity from the substations to end users. In this structure, the local network is adjusted to match a given load profile from the end users connected to it. The end user demands, however, vary drastically based on the day of the week and time of the day. The Power Grid must however be able to provide the worst-case level of power to the end users in order to avoid blackouts. At the same time, the electrical power consumption is rising rapidly. Without a major change in the way the Power Grid is organized, built, monitored, and utilized, the U.S. alone must invest hundreds of billions of dollars in building new power plants over the next 20 years to meet the expected growth in electrical energy consumption under worst-case demand conditions [1].

To avoid expending this large amount of capital for the expansion of the power generation capacity in the U.S. a decentralized Smart Grid, which delivers electricity from suppliers to consumers, monitors and controls the power flow in the Grid, and provides the required networks, sensing/metering gear, dynamic control, applications and databases to match the power generation to power consumption and to optimize the overall cost of electrical power delivered to the end users [2] [6]. The key idea is to shift load so that loads are diverted from a peak period to an off peak period, whereby shaving the power peaks and filling the power valleys.

In a smart grid infrastructure, utility companies can perform demand-side management by deploying a dynamic electricity pricing strategy to encourage the consumers to schedule their most power-hungry function and tasks at those times (of the week or the day) when the Grid is not overly loaded. A well designed dynamic pricing function is thus able to incentivize customers to tune their loads to the current state of the network. On the other hand, the consumers can shape their power consumption profile to minimize their electrical bill while scheduling the start times of their most power-hungry functions. This is called demand shaping. The design of energy pricing-aware scheduling algorithms is thus an important task that must be undertaken in order for the Smart Grid technology to bear fruit.

Based on the degree of security with respect to sharing the network information, a number of different scheduling strategies are introduced in [2]. The proposed distributed stochastic energy consumption scheduling algorithms utilize the available (complete or partial) knowledge to improve the overall load profile. In [3] an optimal algorithm utilizing real-time energy pricing functions is presented to find the optimal energy consumption levels for each consumer. The algorithm is implementable in a distributed manner so as to maximize the aggregate utility of all users and minimize the cost to the energy producer [3]. This work models the consumers' preferences and their energy consumption patterns in form of utility functions. Shaping the power demand to match available power supply is addressed in [4], where two market models are considered. The authors propose distributed demand response algorithms to achieve equilibrium states that maximize the social welfare.

The authors of [5] present an energy management system algorithm that learns users' behavior and schedules their tasks to perform optimal energy scheduling and allocation decisions. Because of the complexity of the solution technique, the approach presented in [5] is only applicable to cases with a small number of users and tasks. In [6], it is assumed that houses and buildings are equipped energy management controllers that control the operation of some of their appliances. The energy management controller uses both dynamic prices and user preferences to control the power usage within a single house and across multiple houses in a neighborhood. To minimize the electricity bill of cooperative users, a quasi-dynamic pricing model is presented in [7]. In this model, the electricity price comprises of two components: a time-of-use (TOU) dependent base price for kWh of energy used and a penalty term corresponding to instantaneous power consumption exceeding a TOU-dependent bound on total power consumption. Subsequently, two different methods (a deadline-driven continuous-variable method and a timeslot-based method) are presented to optimize the energy cost of a networked community of cooperative users.

References [5]-[7] are representative of the kind of work that has been reported in the literature with respect to task/appliance scheduling to minimize the electrical energy cost to the end users. None of these works, however, considers a TOU and total power consumption dependent energy pricing function with a hard bound on the instantaneous power consumption. The inconvenience cost is considered in [6]. However this reference relies on a very simple (and idealistic) energy pricing function. Similarly, reference [7] treats the instantaneous power consumption limit a soft constraint and includes its effect in the overall cost function using a penalty term. Furthermore, it does not have any notion of a target (preferred) start time for the scheduled tasks, and hence, ignores the inconvenience cost.

This paper proposes two different scheduling strategies for managing the profile of loads considering user preferences and peak power constraints. The first strategy, called *Power-Constrained, Minimum-Cost Scheduling with Fixed Prices*, assumes that the energy price is a function of TOU and that the total power consumption of the consumer is bounded. The second strategy, *Minimum-Cost Scheduling with Power-Dependent Variable Prices*, assumes that the electrical energy price is a function of TOU and the total power usage of the consumer. Both strategies are formulated and two efficient heuristics are proposed for solving them. Furthermore we present branch and bound algorithms (designed to have a low average time complexity) to find the optimal solutions to set the golden result against which the performance of the proposed heuristics is measured. All the proposed algorithms consider two different costs paid by users: electricity energy price and inconvenience cost.

The system model, energy price function, and overall cost function are presented in section II of the paper. Sections III and IV present provably optimal and heuristic solutions for task scheduling problems subject to (a) TOU-dependent energy pricing function subject to a constraint on total power consumption; and (b) scheduling with TOU and total energy consumption dependent pricing function for electricity consumption, respectively. Experimental results are reported in section V and paper is concluded in section VI.

II. System Model, Energy Price and Cost Functions

In this paper, we consider task scheduling in households, buildings and warehouses to minimize a composite cost function comprised of an *electricity price* paid by the tenants for their electrical energy usage plus an *inconvenience cost* incurred by the tenants if and when their jobs are scheduled outside a preferred timing window. In the rest of the paper we use the term *facility* to represent any residential, commercial or industrial facilities with one or more electricity-consuming tenants, occupants or workers, but a *unified electricity bill*.

In this paper, we assume that there are a number of tasks in each facility that should be executed daily. These tasks are identified by index i. The set of task indexes is denoted by K={1, …,N}. For each task $i \in K$, the earliest start time (s_i^e), the latest end time (e_i^l), and the duration of task (d_i) are specified. At each step of the task scheduling process, there are a set of assigned tasks K_a and a set of unassigned tasks K_u such that $K_a \cup K_u = K$ and $K_a \cap K_u = \emptyset$. Tasks are assumed to be non-interruptible i.e., they executed from the scheduled start times (s_i) until they are completed. Moreover, each task consumes electricity according to a known power dissipation profile. This means that if the start time and duration of task i are denoted by s_i and d_i, respectively, then the power consumption of task i will follow a known profile of $p_i(t)$ as t ranges from s_i to $s_i + d_i$. Notice that function $p_i(t)$ is a stationary function in the sense that its value is independent of the scheduled start time of the task. The value of the task power consumption function $p_i(t)$ for times outside the execution time of the task is zero.

Each task has a preferred start time. This start time, which lies in the range $[s_i^e, e_i^l - d_i]$, is denoted by s_i^p for task i. This preference is captured by assigning an *inconvenience cost* ($I_i(s_i)$) to each task which is scheduled in a time different from s_i^p. The inconvenience cost function $I_i(t)$ for task i is thus a function that assumes zero value for a start time equal to s_i^p and non-negative values for all other start times.

We assume that the *electricity price function* ($C(t)$) is pre-announced by the utility company just before the start of the day; furthermore, the price function is not changed until possibly the next day. If, however, the utility company is unable or unwilling to provide the electricity price function for the entire day and, for example, chooses to pre-announce the price function for the next hour just prior to the beginning of the hour, then the users may utilize a history-based price function prediction algorithm (which is straight-forward to design but falls outside the scope of this paper) to obtain an expected electricity price function for the whole day. This is important for energy cost-aware scheduling in small facilities because there may not be enough tasks or enough flexibility to achieve meaningful reductions in cost if the length of the *scheduling epoch* is too short. In large facilities with a multitude of users/occupants generating many tasks, a finer granularity scheduling epoch (e.g., in order of hours) will be just fine. Without loss of generality, in the remainder of this paper, we assume that the length of the scheduling epoch is 24 hours.

We also assume that the electricity price at each time instance is fixed and independent of the total amount of power consumption in the facility, i.e., $C(t)$ is only a function of time of the day and not the total power consumption at that time. This electricity price function is used in some of the previous works, e.g., [6] and [7]. Task scheduling based on this kind of electricity price model will reduce the overall electrical bill by moving the bulk of the tasks to time slots during the day where the electricity prices are low. Unfortunately, however, this can create large peaks of power consumption during such low-cost time slots, which can burden the Power Grid supplying power to the facility. In the worst case, all facilities in a given utility service area, will schedule their tasks to run in the same set of low-cost timing slots, thereby, causing a potential power delivery failure (blackout or brownout) on the Grid. To contain this potential disastrous effect, the utility company sets a peak power consumption limit for each facility. This limit may vary for different facilities and at different times of the day at the sole discretion of the utility company. In the remainder of this paper, we shall use $P^p(t)$ to denote the upper bound on the peak instantaneous power drawn by the target facility at time t. TOU-dependent (but otherwise, fixed) electricity prices are assumed in section III while a price model with both TOU and total power consumption dependence is assumed in section IV.

In this paper, we assume a *slotted time* model, i.e., all system cost parameters and constraints as well as scheduling decisions are provided for discrete time intervals of constant length. The scheduling epoch is thus divided into a fixed number of equal-sized time slots (a day is divided into T time slots, each of duration D). Tasks can be launched only at the beginning of one of these time slots and will be completed at the end of the slots.

III. Power-Constrained Load Scheduling for TOU-Dependent Energy Pricing Function

In this section we provide both the optimal solution (albeit with exponential worst-case complexity) and an efficient heuristic solution for the problem of task scheduling with TOU-dependent energy pricing function subject to a constraint on total power consumption. The objective function is to minimize the energy cost plus the inconvenience cost as defined in section II.

Let $cost_i^{s_i}$ denote the total cost of assigning task i to start at s_i. Clearly,

$$cost_i^{s_i} = I_i(s_i) + \sum_{t=s_i}^{s_i+d_i} C(t) p_i(t - s_i) D \quad (1)$$

The first term in (1) is the inconvenience cost of the task whereas the second term corresponds to the aggregated electricity cost of the task in the facility.

Problem 1: Power-Constrained, Minimum-Cost Scheduling with Fixed Prices, or PMSF for short) **--** Minimize the total electrical energy cost plus the inconvenience cost in a facility through task scheduling, i.e.,

$$Min \sum_{i=1}^{N} cost_i^{s_i} \quad (2)$$

s.t.

$$\sum_{i=1}^{N} p_i(t - s_i) \le P^p(t), \quad \forall t \quad (3)$$

$$s_i^e \le s_i \le e_i^l - d_i, \quad \forall i \quad (4)$$

Constraint (3) captures the peak power limit constraint in each time slot while constraint (4) appropriately bounds the start time for each task.

Theorem 1: The PMSF problem is at least as hard as the Generalized Assignment Problem (GAP) [9].

Proof: If the duration of tasks (d_*) is only one time slot and constraint (4) is relaxed to only finding a single time slot to schedule the task, solution of the PMSF problem can solve the MINGAP with the following efficient transformation: (i) Cost of assigning task i to start time (s_i) is $C(s_i)D + I_i(s_i)$ and (ii) Power capacities of different time slots are set to $P^p(t)$.■

Theorem 1 shows that the PMSF problem is an NP-Hard problem. Multi-slot tasks can make this minimization problem harder because assignment of tasks to a start time can cause capacity violation in more than one time slot and this complicates the proper solution for this problem.

Note that because the MINGAP problem is reducible to the PMSF problem, determining whether an instance of the PMSF problem has a feasible solution is also an NP-complete problem.

A. Branch and Bound –Based Scheduling Solution

To solve the PMSF problem optimally, a Branch and Bound (B&B) algorithm based on both lower and upper bounding of remaining costs is used as explained next. The proposed B&B algorithm recursively (and implicitly) enumerates all possible solutions. In particular, at each decision (branching) point, a task is picked according to the adopted *branching strategy* (see description below) and assigned to one of the least-cost available time slot for the selected task. This scheduling step must be repeated for all remaining (unscheduled) tasks in order to generate a complete solution. Now then, at each decision point, a *lower bound* on the total cost (already accrued plus minimum remaining) is calculated (see below for how this lower bound is calculated.) Furthermore, an *upper bound* on the total cost (already accrued plus actual remaining) is calculated using a highly effective heuristic algorithm (called rank-based scheduling, see below for a description.) If the lower bound cost is greater than the cost of the best complete solution found previously or if it is equal to the upper bound cost, then the remaining scheduling steps are skipped and the next available time slot assignment for the selected task is explored. The best found solution at the end of this process will be the globally optimum solution to the PMSF problem.

The proposed branching strategy is based on the difference of the best possible assignment (with respect to power limit constraints) and second best possible assignment of the tasks. The intuition is that the task that exhibits the largest cost increase if it is not assigned to its optimum time slot is given the highest priority so that it is scheduled first when the power constraint has not yet eliminated too many of the available time slots.

Branching Strategy:
For each unassigned task $i \in K_u$ do:
Calculate $cost_i^{t_i}$ from (1) respecting constraints of PMSF
Calculate Δ_i, which is the difference between the two smallest $cost_i^{s_i}$ values where $s_i^e \le t_i \le e_i^l - d_i$
Branching is performed on task i^* where

$$i^* = \arg\max_i \{\Delta_i\} \quad (5)$$

To calculate the cost lower bound at each decision point in the B&B search tree, the solution of PMSF with relaxation of constraint (3) is used. Because of this relaxation, PMSF can be efficiently and optimally solved with the following greedy algorithm. Note that here some tasks have already been assigned to time slots, i.e., we start with known sets of K_u and K_a.

Cost Lower Bound Algorithm:
Initialize the lower bound cost as $\sum_{i \in K_a} cost_i^{s_i}$
While $K_u \neq \emptyset$ do
Pick a task $i \in K_u$
Calculate $cost_i^{s_i}$ for each start time s_i respecting constraint (4)
Find the start time s_i^* that minimizes $cost_i^{s_i}$. The start time s_i^* is the best possible slot assignment for task i
Set $K_u = K_u - \{i\}$ and $K_a = K_a \cup \{i\}$ and add $cost_i^{s_i^*}$ to the lower bound cost

In fact in order to find the best possible time slot to assign a task, because we have made no assumptions about the form of the electricity price function or the inconvenience cost function, minimization of (2) subject to (4) is a constrained nonlinear (potentially non-convex) mathematical program, which is best solved by full enumeration of all possible time slot assignments for task i. (as is done in the proposed greedy algorithm.)

Cost Upper Bound Algorithm (Rank-based Scheduling):
Initialize the Upper bound cost as $\sum_{i \in K_a} cost_i^{s_i}$
While $K_u \neq \emptyset$ do
Apply the *Branching Strategy* to pick a task $i \in K_u$
Find the start time s_i^* that minimizes $cost_i^{s_i}$
Schedule task i at s_i^* and set $K_u = K_u - \{i\}$ and $K_a = K_a \cup \{i\}$ and add $cost_i^{s_i^*}$ to the upper bound cost
Set Flag = TRUE if $\forall j \in K_u$, task j can be scheduled to start at some slot s_j without violating (3) else FALSE
If (! Flag)
Set $K_a = K_a - \{i\}$ and $K_u = K_u \cup \{i\}$ and exclude start time s_i^* from possible start times of task i and subtract $cost_i^{s_i^*}$ from the upper bound cost

In *Rank-based scheduling,* a task is scheduled only if it does not prohibit other tasks from being scheduled at any start time. Even with this policy, it is possible that *Rank-based Scheduling* reaches a point that some of the tasks are not schedulable because of the previous assignment. To overcome this problem, all un-schedulable tasks at the end of the first iteration of the proposed heuristic are scheduled at their best possible time slots (while checking the condition that their slot assignment does not prohibit other tasks from being scheduled) and then second iteration of the proposed heuristic is executed for all the remaining tasks. While there is no guarantee that this iterative approach will converge and produce a complete scheduling solution, in practice and for all examples tried, the approach converges in one or at most three iterations if a feasible solution exists.

Computational complexity of this heuristic is $O(ATN^2 \log(TN))$ where A denotes the number of iterations of the heuristic to find a solution. It can be seen that the complexity can be decreased by calculating the outcome of assigning tasks to different start times and updating them whenever needed.

B. Rank-based Scheduling Solution

Since the proposed exact B&B algorithm has an exponential worst case time complexity, we have found that the aforesaid Rank-based scheduling algorithm is highly effective and very efficient in practice. In fact it can be used to solve the complete scheduling problem by passing $K_u = K$ and $K_a = \emptyset$ to it.

IV. Load Scheduling for TOU- and Power-Dependent Energy Pricing Function

A. System Model

In section III, task scheduling problem to minimize the total energy cost plus the inconvenience cost in the system was considered when there is an instantaneous power consumption upper bound in the facility. In this section, this upper bound constraint on the power consumption is relaxed and another electricity price model (called TOU-and power consumption dependent energy price) is considered. Under this electricity price model, energy price at each time slot is not a fixed value; instead it varies as a function of the total power consumption of the facility at that time. This means that the TOU-and power consumption dependent electricity price model is a two-dimensional electricity price in which the energy price for the facility depends on the time of day and the amount of power consumed at that time.

Problem 2: Minimum-Cost Scheduling with Power-Dependent Variable Prices, or MSPV for short) -- The cost minimization problem is as follows:

$$Min\ D \sum_{t=1}^{T} \mathbb{C}\left(t, \sum_{i=1}^{N} p_i(t-s_i)\right) \sum_{i=1}^{N} p_i(t-s_i) + \sum_{i=1}^{N} I_i(s_i) \qquad (6)$$

subject to constraint (4). Note that $\mathbb{C}(t,P)$ denotes the energy price at time t and total power consumption of P on that time.

The price function presented in the MSPV problem is assumed to be monotonically increasing. This characteristic means that a facility cannot incur a lower energy price by consuming more energy. It can be seen that MSPV is an integer nonlinear programming problem. In general there is no efficient solution for this problem. Using nonlinear optimizers can help to solve the problem for small set of unknowns but their run time is not acceptable for large number of unknowns.

B. *Branch and Bound –Based Scheduling Solution*

To find the exact solution for the MSPV problem, the branch and bound technique presented in section III is used by appropriately revising it to adopt the energy price model. More precisely, at each branching point, whenever an energy price is needed to calculate the best time slot for the selected task, the energy price function value on the power consumption of the previously assigned tasks is used.

Similar to the previous case, this exact algorithm finds the optimal task scheduling policy at the cost of exponential worst-case computational complexity. Moreover, the average computational complexity of this exact solution is more than that of the exact solution of the PMSF problem because the bounding technique (lower bound on the total cost) is less effective when the energy price is not fixed given TOU and rather varies based on the total power consumption at that time.

C. *Force Directed Scheduling Solution*

A heuristic inspired by Force-Directed Scheduling (FDS), which is one of the significant scheduling techniques in high-level synthesis [8], is applied to find a good solution with a low computational complexity.

FDS is a technique used to schedule directed acyclic task graphs so as to minimize the resource usage under a latency constraint (or with appropriate modification to the basic FDS algorithm, known as FDLS, minimize the latency under resource constraints.) The algorithm consists of three steps: (i) determine the range of available time slots for scheduling a task; (ii) Create a distribution graph, which captures the resource pressures on each time slot, by assuming that a task is equally likely to be scheduled to start in any feasible time slot; (iii) A metric called *force* is utilized to maximize resource utilization. By repeatedly assigning tasks to various time slots and calculating the force associated with the choice, several force values will be available. We choose the assignment with the lowest force value, which also balances the concurrency of tasks (i.e., reduces resource pressure.)

In the context of our problem formulation, the goal is to schedule all tasks so that their total energy cost plus the inconvenience cost is minimized. Range of possible start times for each task is determiend directly in this problem. Moreover, same as FDS algorithm, distribution graph (also referred to as the initial state) is created for this problem by assuming that a task is equally likely to be scheduled to start in any feasible time slot.

There are two different forces in this system to capture both parts of the total cost: (i) energy cost force, and (ii) inconvenience cost force. To model the concurrency in the system, each time slot (also referred to as control step in high level synthesis) is modeled by a spring. Similarly, the inconvenience cost is modeled with rubber bands in the system. Forces in the rubber bounds are not related to the masses but it depends on the displacement.

Spring and rubber band constants are determined based on the specified objective function and current scheduling solution. The FDS scheduler tries to minimize the total forces in the system which means minimizing the objective function of the problem by iteratively changing the scheduling solution.

The reader recalls that strain (force) in springs is linearly related to the stress (displacement) applied to the spring per well-known Hooke's Law. The initial value of the spring constant is determined from the corresponding distribution graph calculated in step (ii). In this approach, tasks are modeled as masses in the system. The displacement of a spring corresponding to time slot i is given by the change in the probability of assigning some task to that time slot (i.e., due to tentatively scheduling that task in some available slot.) In other words, as expected, moving masses to different locations produces different forces in the system. Moreover, rubber band constants are fixed and equal to the inconvenience cost of each task in each time slot. Assigning a task to a start time produces a displacement of one for a rubber band with rubber band constant equal to the inconvenience cost of that task on that start time. Figure 1 shows an example FDS-based scheduling setup in this problem.

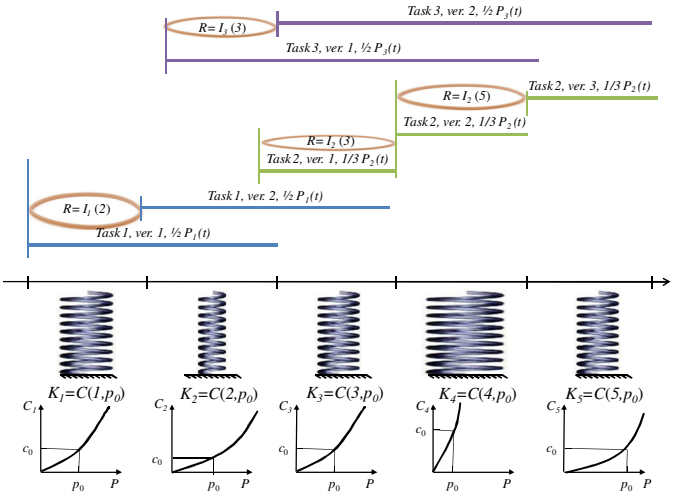

Figure 1. An example of modeling the MSPV problem as force-directed scheduling.

K_* and R denote the springs and rubber bands constants in the model. In this figure, tasks 1, 2 and 3 have two, three and two possible start times, respectively. It can be seen that the initial state is generated by probabilistically placing each task in every possible start time. For instance, task 1, which has two possible start times, is assigned to these two start times and its power consumption value is multiplied by ½ for each assignment. Summation of the weights in each time slot is used to calculate the spring's constant associated with each time slot. This spring constant is set to the value of the energy price function on that time and summation of the power probability of the tasks on that time. Moreover, in this system, a rubber band is associated with each version of each task. These rubber bands are added to model the inconvenience cost of scheduling each task to start at a time slot other than its preferred start time. Rubber band's constant for each version of tasks is equal to the inconvenience cost of assigning the task on that start time and initial displacement of each rubber band is equal to the probability of assigning each task to each start time. Assigning a task to its preferred start time does not have an inconvenience cost and no rubber band is associated for this case.

We want to move from the initial state (distribution graph) to the final state. In the final state, each task is exactly scheduled to start on a known start time. We want this final state to have the least total force in the system because the total force in the system represents the total energy cost of the facility owner in one day. So we need to change the initial state to a state in which each task has a certain start time and the total force in the system is minimized.

Force of a move in this system is defined as the difference between the total force in the system after the move and before the move. For example, scheduling a task in time slot s_i moves the full weight of the task to this time slot while simultaneously eliminating all the weight contribution of the task put to other start times. Thus, the self force for this move is defined as:

$$F_i^{s_i} \equiv \left(\sum_{t=s_i}^{s_i+d_i}(Dp_i(t)K_t) + I_i(s_i)\right) - \left(\sum_{t=S_i^p}^{e_i^l}\left(\sum_{\tau=S_i^p}^{e_i^l}\frac{(Dp_i(\tau-t)K_{\tau-t})}{e_i^l-d_i-S_i^p}\right) + \sum_{t=S_i^p}^{e_i^l-d_i}\left(\frac{I_i(t)}{e_i^l-d_i-S_i^p}\right)\right) \quad (7)$$

where K_t is the spring's constant associated with time slot t. The first term in (7) is the total forces in the system due to assigning task i to start time s_i while the second term is the summation of the forces in the system by probabilistically assigning task i to all possible start times. Note that every possible re-assignment of tasks creates a force that can be calculated based on (7). For example, if the selected move is assigning task i to two possible start times uniformly (still not the final state for this task), the force can be calculated by (8).

$$F_i^{t_1,t_2} = \frac{1}{2}F_i^{t_1} + \frac{1}{2}F_i^{t_2} \quad (8)$$

To move from the initial state to the final state, a gradual movement is considered: (i) a move from the initial state to a state where each task is assigned to at most two start times; and (ii) a move from this state to the final state.

For each movement, the move associated with a task with the least force is selected and done and then other tasks are considered. To improve the time complexity of the algorithm, only forces affected by the previous move are updated. Figure 2 shows the pseudo code for this task scheduling algorithm.

Algorithm *Force Directed Task Scheduling* ()
// Schedule tasks to minimize the total cost of facility
Create the distribution graph assuming that a task is equally likely to be scheduled to start in any feasible time slot.
Initialize forces for each version of the tasks by (7).
Initialize forces for each pair of 2 versions of the tasks by (8).
K_a=∅ and K_u=K;
Add tasks with only one possible start time to K_a and delete from K_u;
While (K_u≠∅) {
  i^* = arg min ($F_i^{t_1,t_2}$); (similarly find t_1 and t_2 for i^*)
  Remove all versions of task i^*, K_u=K_u∪{ i^*} and K_a=K_a-{ i^*};
  Add 2 versions of task i^* with weight of ½ for each to t_1 and t_2;
  Update springs' constants;
  FOR (i∈ K_u){
    If (task i has been affected by changing springs' constants){
      Update forces for each version of task i;
      Update forces for each pair of 2 versions of task i;}}}
Initialize forces for each version of the tasks by (7).
K_a=∅ and K_u=K;
While (K_u≠∅) {
  i^* = arg min ($F_i^{s_i}$); (similarly find s_{i^*} for i^*)
  Remove all versions of task i^*;
  Schedule task i at time slot s_{i^*}, K_u=K_u∪{ i^*} and K_a=K_a-{ i^*};
  Update the springs' constants;
  FOR (i∈ K_u){
    If (task i has been affected by changing springs' constants)
      Update the forces for each version of task i;}}
Calculate the total Cost;

Figure 2. Pseudo code for Force directed task scheduling.

V. Simulation Results

To demonstrate the effectiveness of the proposed algorithms, cases corresponding to the two aforesaid pricing models are examined. In these simulations, duration of a time slot is set to one hour. For this reason, the minimum duration of a task is also set to one hour, and the durations of tasks are integer multiples of one hour. Moreover, power consumption of the tasks is determined with a granularity of one hour.

For the first energy pricing model (TOU-dependent energy pricing function subject to a constraint on total power consumption), we randomly set the energy price for each hour of the day according to a uniform distribution with a mean value of 15¢/KWh and a range of 10 to 20¢/KWh. We generate a task workload that would violate this power limit if each task were simply scheduled at its best possible time slot. As stated in section II, this makes the problem hard.

For all of the B&B simulation results, if the time required to find the final solution is more than 3 hours, the B&B algorithm is halted and no result is reported.

Tasks in the facility are generated arbitrarily with average duration of 4 hours. Power consumptions of tasks are determined based on power consumption data about real appliances. Inconvenience costs are set arbitrarily to capture the cost of scheduling tasks to non-preferred time slots. The average inconvenience cost for one time slot deviation from the preferred time is set to 2.5¢. The number of possible start times of the tasks is set arbitrarily using a normal probability distribution function with mean of 4 hours and variance of 3 hours.

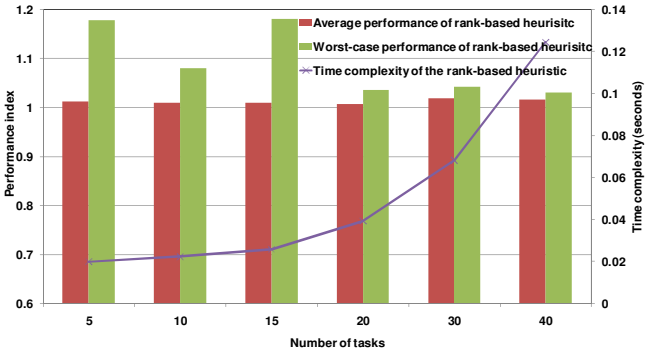

Figure 3. Performance and time complexity of the rank-based heuristic with different number of tasks.

We define the performance index of solution A with respect to solution B as the ratio of the cost of A to the cost of B. Typically solution B is the minimum cost solution obtained by the B&B method. Therefore smaller indices approaching unity are more favorable to A.

Figure 3 shows the worst case and average performance indices of the proposed rank-based algorithm (for over 10000 cases) with respect to the optimal B&B solution. Moreover, the average time complexity of the algorithm is reported in this figure.

It can be seen that the worst-case performance index of the rank-based scheduling is improved as the number of total tasks in the facility grows. Moreover, the average performance index of the rank-based scheduling is no worse that 1.02 for all cases. Note also that the rank-based scheduling was always able to find a feasible solution if one exists (i.e., if the B&B method can find it.)

We next report results for the second pricing model (Power- and TOU-dependent energy pricing function.) To show the effectiveness of the force-directed task scheduling in this case, a step-wise pricing function is considered. For each time slot, values of the step prices are determined arbitrarily increasing.

TABLE I. PERFORMANCE AND TIME COMPLEXITY OF THE FORCE-DIRECTED HEURISTIC AND GREEDY SOLUTION WITH DIFFERENT NUMBER OF TASKS.

# of tasks	Average performance of the FDS Alg.	Average performance of greedy solution	Worst-case performance of the FDS Alg.	Time complexity of the heuristic(s)
5	1.007	1.272	1.048	0.076
10	1.013	1.396	1.046	0.104
15	1.012	1.491	1.024	0.138
20	1.006	1.322	1.012	0.159
30	B&B not finished	1.182[1]	B&B not finished	0.182
40	--	1.176	--	0.227
50	--	1.153	--	0.300

Table I shows the worst-case and average performance indices of the force-directed scheduling heuristic with respect to the B&B solution. We also implemented a greedy solution for this problem. In this greedy solution, tasks are picked in an arbitrary manner and scheduled to start at the best possible time slot based on the TOU-dependent component of the energy pricing function and the total

[1] In case of not finding a solution from B&B method, performance of the greedy method is calculated with respect to the force-directed heuristic solution

power consumption of all previously scheduled tasks whose execution time overlaps with the time slot in question. The table also reports the average performance index of this greedy solution compared to the best solution found. Moreover, the average time complexities of the FDS based algorithm is reported in the table.

It can be seen that the performance index of the force-directed scheduling algorithm compared to the B&B solution is improved as the number of tasks grows. Interestingly, this means that the force-directed scheduling does better as the number of scheduled tasks that overlap each other increases. Moreover, it can be seen that performance index of the greedy solution gets worse than that of the force-directed heuristic solution (by more than a factor of 1.15.)

As can be seen in Figure 3 and Table I performance indices of the proposed heuristics are close to the optimal solution while their time complexities are significantly lower and certainly acceptable even for online decision making in response to changes in the dynamic energy prices on an hourly or even shorter time basis.

To capture the effect of the proposed pricing models and algorithms in shaping the energy consumption at the facilities, we consider a scenario with 10 facilities. Each facility has 100 tasks to schedule and the preferred start times, inconvenience cost functions, duration of tasks, and possible start times for each task have been specified.

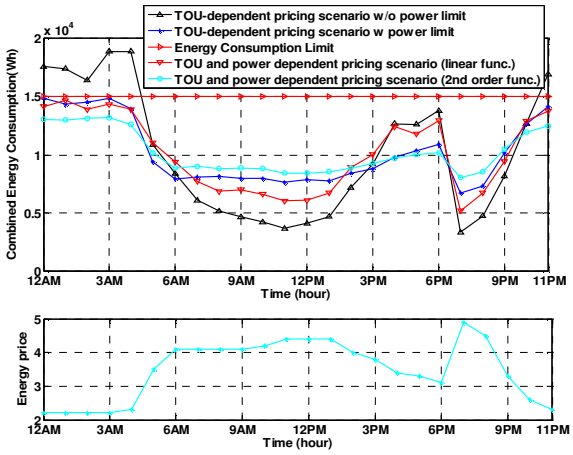

Figure 4. (a) Combined energy consumption of 10 houses in different scenarios, (b) Energy price for fixed energy price scenarios.

For this scenario, three different energy pricing models are examined: First, TOU-dependent energy prices without any power limit (specified at the hourly granularity); Second, TOU-dependent energy prices with a fixed power limit for all hours of the day; and Third, both TOU and total power consumption dependent energy prices.

For the first energy pricing model, a real hourly pricing scheme from [10] is used. This pricing function is shown in Figure 4-b. For the second pricing model, the power consumption limit for each facility is set to 1.5KW. To produce a TOU and total power consumption dependent energy price, two different functions are used. In the first case, a linear pricing function is used. This linear function is assumed to cross two fixed points. The first point is at zero power consumption and price is equal to zero. The second point's power consumption is 750W and price for this point is the energy price specified for that hour by the TOU-dependent energy price. In the second case, a second order energy price function is used. Also in this case, energy price function is assumed to cross the previous fixed points.

Figure 4 shows the combined energy consumption of these facilities under different pricing models and the energy price that we used for this simulation.

To compare different solutions, we define a figure-of-merit to measure the uniformity of energy consumption over time. The flatness is defined as:

$$Fm = \bar{E}T / \sum_{t=1}^{T} |E_t - \bar{E}| \quad (9)$$

where E_t and $\bar{E}$ denote the energy consumption in time t and the average energy consumption in a day, respectively. Larger flatness value means the energy consumption profile is closer to constant energy consumption. Table II shows flatness value, *Fm*, for different solutions.

TABLE II. FLATNESS VALUES FOR DIFFERENT SCENARIOS.

Scenarios	Flatness value(*Fm*)
TOU-dependent scenario without power limit	2.0977
TOU-dependent scenario with power limit	4.2186
TOU and power dependent scenario (1sr order)	3.4857
TOU and power dependent scenario (2nd order)	6.6396

It can be seen that adding power limit increase the flatness value with respect to the base scenario. Moreover, increasing the order of the energy-price function in TOU and total power consumption dependent pricing function increases the flatness value because by increasing the order of pricing function, facilities try to avoid using a large amount of energy in each time slot even if the base price for that time is cheap. Also it is important to note that we can change the flatness value (reduce the peak in energy consumption) or shape the electricity demand by changing the energy-price function form.

VI. CONCLUSION

Two algorithms were presented to schedule tasks in facilities based on various dynamic pricing models for the electricity. The paper described optimal (based on branch and bound algorithms) and heuristic solutions for TOU-dependent energy pricing with fixed power limit and TOU and power consumption dependent energy pricing. Simulation results depicted the near-optimal performance of the proposed heuristic algorithms. Furthermore results demonstrated the ability of the proposed policies for reshaping the energy consumption profile of the consumers.

Acknowledgement: This work was sponsored in part by a grant from the National Science Foundation.

REFERENCES

[1] L. D. Kannberg, D. P. Chassin, J. G. DeSteese, S. G. Hauser, M. C. Kintner-Meyer, R. G. Pratt, L. A. Schienbein, and W. M. Warwick, "GridWise™: The Benefits of a Transformed Energy System," PNNL-14396, Pacific Northwest National Laboratory, Richland, Sep. 2003.

[2] S. Caron and G. Kesidis, "Incentive-based Energy Consumption Scheduling Algorithms for the Smart Grid," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[3] P. Samadi, H. Mohsenian-Rad, R. Schober, V. Wong, and J. Jatskevich, "Optimal real-time pricing algorithm based on utility maximization for smart grid," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[4] L. Chen, S. Low, and J. Doyle, "Two market models for demand response in power networks," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[5] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[6] S. Kishore, L. V. Snyder, "Control Mechanisms for Residential Electricity Demand in SmartGrids," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[7] S. Hatami and M. Pedram, "Minimizing the Electricity Bill of Cooperative Users under a Quasi-Dynamic Pricing Model," *Proc. of Smart Grid Communications* (*SmartGridComm*) *Conference*, 2010.

[8] P.G. Paulin, J.P. Knight. Force-directed scheduling for the behavioral synthesis of ASICs. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Jun 1989.

[9] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Chichester, U.K.: Wiley, 1990.

[10] http://www.powersmartpricing.org/tool