Multi-Objective Optimal Energy Consumption Scheduling in Smart Grids

Sergio Salinas, Ming Li, and Pan Li

***Abstract*—A major source of inefficiency in power grids is the underutilization of generation capacity. This is mainly because load demand during peak hours is much larger than that during off-peak hours. Moreover, extra generation capacity is needed to maintain a security margin above peak load demand. As load demand keeps increasing and two-way communications are enabled by smart meters (SMs), demand response (DR) has been proposed as an alternative to installing new power plants in smart grids. DR makes use of real-time schemes to allow users to modify their load demand patterns according to their energy consumption costs. In particular, when load demand is high, energy consumption cost will be high and users may decide to postpone certain amount of their consumption needs. This strategy may effectively reduce the peak load demand and increases the off-peak demand, and hence could increase existing generation capacity utilization and reduce the need to install extra generation plants. In this paper, we consider a third-party managing the energy consumption of a group of users, and formulate the load scheduling problem as a constrained multi-objective optimization problem (CMOP). The optimization objectives are to minimize energy consumption cost and to maximize a certain utility, which can be conflicting and non-commensurable. We then develop two evolutionary algorithms (EAs) to obtain the Pareto-front solutions and the ϵ-Pareto front solutions to the CMOP, respectively, which are validated by extensive simulation results.**

***Index Terms*—Energy consumption scheduling, evolutionary algorithms, multi-objective optimization.**

I. Introduction

IN POWER GRIDS, generation capacity is required to meet peak-hour load demand plus a security margin. However, according to recent studies, the average utilization of the generation capacity is below 55% [1]. This leads to inefficient operation of power grids because a portion of generation plants is largely unused or underutilized, but must still be maintained and supervised to guarantee its reliability. On the other hand, as energy demand, and peak load demand as well, continue increasing, additional generation capacity will be needed to accommodate future load demand, which requires a large investment and might lead to even lower utilization.

Recently, the smart grid (SG) has been proposed as a new type of electrical grid to modernize current power grids to efficiently deliver reliable, economic, and sustainable electricity services. One of the key features of the SG is the replacement of conventional mechanical meters with smart meters to enable two-way communications between users and grid operators. Using the communication infrastructure of the SG, it is possible to shape the users' load demand curves by means of demand response (DR) strategies. One promising DR strategy is real-time pricing (RTP), where utility companies charge users with a price that varies according to the generation cost, i.e., the higher the generation cost, the higher the price. The advantage of RTP is threefold. First, users may reduce their energy consumption when the price is high, and hence lower their electric bills. Second, peak-hour load demand can be reduced, thus reducing the redundant generation capacity needed to meet reliability requirements. Third, off-peak load demand can be increased, which can increase the utilization of the available generation capacity.

Most current research on real-time pricing focuses on how to optimally schedule all users' energy consumption given their predefined energy demand. In particular, Mohsenian-Rad *et al.* [2] propose an autonomous load scheduling algorithm based on cooperative game theory, where each user is a player and their load schedules are the strategies. Agarwal and Cui [3] propose a load scheduling noncooperative game among users that can be reduced to a congestion game. In both studies, the single optimization objective is to minimize the electric bill of the users, while the reduction of the peak-hour consumption is considered as a desirable secondary effect. Moreover, Samadi *et al.* [4] propose an auction based scheme where users provide their utility functions and energy constraints to the utility company, who then replies with a set of prices that maximizes users' utility functions. A similar auction scheme is also proposed by Li *et al.* [5].

Notice that previous study mostly aims at a single objective, e.g., to minimize users' cost. In this paper, we formulate the load scheduling problem as a constrained multi-objective optimization problem (CMOP). Specifically, we consider a third-party managing the energy consumption of a group of smart grid users. All users submit their energy requests to the third-party, which then optimally schedules their energy consumption so that its two objectives can be satisfied. The first objective is to minimize the total energy consumption cost, while the second one is to maximize its utility measured by a certain utility function. This third party can be a company, who schedules its departments' energy consumption in order to minimize the cost and maximize its gross income. Or it can be a community manager, who schedules the residents' energy consumption so that the total energy cost is minimized and its utility (e.g., life comfortness living in this community) is maximized.

We note that these two objectives considered in this study are conflicting and non-commensurable. In the literature, evolutionary algorithms (EAs) have been proven to be effective in finding good approximations of optimal solutions to multi-objective optimization problems [6]–[11]. In particular, EAs aim to find a set of solutions that approximate the Pareto-optimal

Manuscript received April 01, 2012; revised April 10, 2012; accepted July 09, 2012. Date of publication September 28, 2012; date of current version February 27, 2013. This work was supported in part by the U.S. National Science Foundation under Grants CNS-1149786, ECCS-1128768, CNS-1147851. Paper no. TSG-00173-2012.

The authors are with the Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS 39762 USA (e-mail: sas573@msstate.edu; ml845@msstate.edu; li@ece.msstate.edu).

Digital Object Identifier 10.1109/TSG.2012.2214068

front in the objective space, which all follow two basic steps iteratively: variation and selection. Variation consists of choosing some solutions from the existing (maybe random) solutions to be combined and produce new ones. Then, selection is performed to keep the good solutions and discard the bad ones. Different ways for selecting the best solutions and storing them have been proposed in the literature. In this study, to solve the formulated CMOP, we first develop an evolutionary algorithm, called LSEA, to retrieve a set of Pareto-optimal solutions and show the trade-offs between energy consumption cost and the utility. Then, in order to further improve the algorithm efficiency, we present an ϵ-approximate evolutionary algorithm, called ϵ-LSEA, to obtain ϵ-Pareto fronts of the objective space. Extensive simulations have also been conducted to evaluate the performance of the two proposed algorithms.

The rest of this paper is organized as follows. Section II introduces system models considered in this study. We describe the constrained multi-objective optimization problem in Section III. Section IV details the proposed evolutionary algorithms for solving the CMOP. Simulation results are presented in Section V. Finally, we conclude this paper in Section VI.

II. System Model

In this section, we briefly describe smart grids, and energy cost model and utility function model in smart grids.

A. Smart Grids

Smart grids have been promoted by many governments as a way of addressing energy independence and sustainability, global warming, and emergency resilience issues [12]. In smart grids, the energy consumption of each user is monitored by a smart meter (SM), which is also capable of controlling the user's appliances (e.g., turning them on or off, adjusting their settings). Due to their communication capability, SMs also enable two-way communications between users and utility companies, via multihop wireless, wired, or hybrid networks.

In this study, we consider a third-party managing the energy consumption of a group of smart grid users. Each user submits its energy request to the third-party, e.g., 2 kilowatt-hour (kWh) between 10:00 and 18:00, before a day starts (0:00). Then, the third party optimally schedules all users' energy consumption (either locally or via cloud computing) so that its objectives can be satisfied, which are first, to minimize the total energy consumption cost, and second, to maximize its utility measured by a certain utility function. For example, this third party can be a company, who schedules its departments' energy consumption in order to minimize the cost and maximize its gross income. The third party can also be a community manager, who schedules the residents' energy consumption so that the total energy cost is minimized and its utility (e.g., life comfortness living in this community) is maximized.

B. Energy Cost Model

We discretize a day into H time slots of equal length, which are denoted by a set $\mathcal{H}$. A complete energy consumption schedule for user u ($u \in \mathcal{U}$) during one day is given by a vector $\mathbf{x}_u = [x_u^1, x_u^2, \ldots, x_u^H]$, where x_u^h is user u's energy consumption in the hth time slot, and $\sum_{h=1}^{H} x_u^h = e_u$, i.e., user u's required energy consumption during one day. Then, the total energy consumption of all users in time slot h ($1 \le h \le H$), denoted by E_h, is

$$E_h = \sum_{u=1}^{U} x_u^h$$

where $U = |\mathcal{U}|$ is the cardinality of the set $\mathcal{U}$, i.e., the number of users in this area.

Besides, we assume that the energy price functions are known to the third party. One example for such a price function is given below:

$$C_i(E_r) = a_i E_r^2 + b_i E_r, \quad \text{for} \quad 0 \le E_r < G_i^{max}$$

where E_r is the total energy consumption of all users, a_i and b_i are non-negative coefficients, and G_i^{max} is a upper bound on the energy consumption for this price function to hold.

Furthermore, in practice, the energy price function may be piecewise. In this paper, we consider a two-piece price function without loss of generality, which is composed of two functions denoted by C_1 and C_2, respectively. Assume that $a_2 > a_1$ and $b_2 > b_1$, i.e., the energy price increases even faster once the energy consumption exceeds a certain threshold. Consequently, the overall cost function of consuming E_r energy, denoted by $C(E_r)$, is shown in the equation at the bottom of the page, where $M_1 > 0$ accounts for a marginal cost. Notice that when the total energy consumption exceeds a certain threshold, i.e., $E_r \ge G_1^{max} + G_2^{max}$, the cost goes to infinity. It means that the third party is only allowed to use this much energy (i.e., $G_1^{max} + G_2^{max}$) at most, which could be a constraint to ensure the stability of the neighboring areas considered from the whole grid perspective.

C. Utility Function Model

In addition to low cost, the third party also intends to achieve high utility, which is calculated by a utility function. As mentioned before, the utility could be a company's gross income, or a community's living comfort, and so on. Usually, the utility functions are non-decreasing with respect to the consumed power, concave, and results in a zero utility value given zero power consumption [4]. For simplicity, we use the following utility function, denoted by $V(E_r)$, in this study:

$$V(E_r) = \sqrt{E_r} \tag{1}$$

$$C(E_r) = \begin{cases} C_1(E_r), & \text{for } 0 \le E_r < G_1^{max} \\ C_1(G_1^{max}) + C_2(E_r - G_1^{max}) + M_1, & \text{for } G_1^{max} \le E_r < G_1^{max} + G_2^{max} \\ \infty, & \text{for } E_r \ge G_1^{max} + G_2^{max} \end{cases}$$

where E_r is the total energy consumption of all the users. Note that the utility value may not have the same unit as the energy cost.

III. Constrained Multi-Objective Optimization Problem Formulation

In general, a constrained multi-objective optimization problem (CMOP) is defined as follows [13]:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, && i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, && j = 1, \ldots, p \\ & x_q^l \leq x_q \leq x_q^u, && q = 1, \ldots n \end{aligned}$$

where $F(\mathbf{x})$ is the set of objective functions, $g_i(\mathbf{x})$ is the set of inequality constraints, $h_j(\mathbf{x})$ is the set of equality constraints, and x_q^l and x_q^u are the minimum and maximum values of each decision variable x_q, respectively. A CMOP minimizes k objective functions *simultaneously*, where the objective functions represent (usually) competing or conflicting objectives.

In this study, we consider two objective functions, and formulate a CMOP as follows:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \left(\sum_{h=1}^{H} C\left(\sum_{u=1}^{U} x_u^h \right), \quad -\sum_{h=1}^{H} V\left(\sum_{u=1}^{U} x_u^h \right) \right)$$

$$\text{subject to} \quad \sum_{u=1}^{U} x_u^h \leq G_1^{max} + G_2^{max}, \forall h \in [1, H] \tag{2}$$

$$x_u^h \begin{cases} \geq 0, & S_u \leq h \leq T_u \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$e_u - \bar{e}_u \leq \sum_{h=1}^{H} x_u^h \leq e_u + \bar{e}_u \tag{4}$$

$$1 \leq S_u \leq T_u \leq H \tag{5}$$

In the above CMOP, the first objective function minimizes the total energy generation cost during one day, and the second objective function maximizes the utility function. Constraint (2) guarantees that in each time slot the total energy consumption does not exceed the maximum generation capacity of the system. Constraint (3) indicates that each user u has certain energy demand which needs to be satisfied between a required starting time S_u and a required stopping time T_u. Constraint (4) represents a user' tolerance of its daily energy consumption, i.e., the user is fine with consuming $e_u - \bar{e}_u$ to $e_u + \bar{e}_u$ energy in one day. Constraint (5) simply means that the starting time is no later than the stopping time for each user, which are both between time slots 1 and H.

IV. Solving CMOPs by Evolutionary Algorithms

Evolutionary algorithms (EAs) have been proven to be effective in finding good approximations of CMOPs' optimal solutions. The basic idea is to use the crossover, mutation and selection principles of Darwinian evolution to combine, modify and choose possible solutions iteratively until a good approximation of the optimal solution to a CMOP is found. Specifically, crossover and mutation are probabilistic procedures that combine solutions in order to make (possibly better) new solutions. Selection is a deterministic procedure that discards the bad solutions found so far and keeps the good ones. Besides, selection procedures are based on the solutions' fitness, which is usually assigned by an EA based on Pareto dominance and the distance to its nearest neighbors in the objective space. Before we dive into the details, we give some definitions as follows.

Definition 1: In a CMOP, a solution vector $\mathbf{x}$ is said to Pareto dominate another solution vector $\mathbf{y}$ ($\mathbf{x} \succ \mathbf{y}$), if $x_i \leq y_i$ for all $i \in [1, k]$ and there exists some $i \in [1, k]$ such that $x_i < y_i$, where k is the dimension of the solution vectors.

EAs are usually applied to unconstrained optimization problems. Some different penalty functions and definitions of dominance have been proposed in the literature to handle constraints. Penalty functions are functions of the infeasibility of a solution, where larger values are assigned to solutions farther away from the feasible space of the problem while smaller values are assigned to solutions closer to the feasible space. In this paper, we adopt the dominance definition given by Deb *et al.* [6], which takes constraints into consideration and is described below.

Definition 2: A solution vector $\mathbf{x}$ is said to constraint-dominate another solution vector $\mathbf{y}$ ($\mathbf{x} \succ \mathbf{y}$) if any of the following conditions is true:

1) $\mathbf{x}$ is feasible but $\mathbf{y}$ is not.
2) Both $\mathbf{x}$ and $\mathbf{y}$ are feasible and $\mathbf{x}$ Pareto dominates $\mathbf{y}$, as defined in Definition 1.
3) Both $\mathbf{x}$ and $\mathbf{y}$ are infeasible, but $\mathbf{x}$ has lower overall constraint violation.

After an EA is executed, several non-dominated solutions, in the Pareto sense, are obtained. Each of these solutions is a compromise between the multiple objective functions. In what follows, we first propose an evolutionary algorithm to find Pareto optimal solutions to the load scheduling problem formulated in Section III, and then develop an ϵ-approximate evolutionary algorithm to obtain ϵ-Pareto fronts of the solutions.

A. Load Scheduling With an Evolutionary Algorithm (LSEA)

An evolutionary algorithm is usually composed of several important processes, including initialization, selection, crossover, and mutation. In the following, we describe such processes, respectively.

In the beginning, N random solutions, called individuals, are created to form the initial population P_0. The initial individuals satisfy constraints (3)–(5) but may not meet constraint (2). Next, all individuals are compared to each other using the constraint-dominance definition (Definition 2) and each individual is assigned a rank according to the number of individuals by which it is dominated. For example, non-dominated individuals receive a rank of 1, individuals dominated by only one individual receive a rank of 2, and so on. Individuals with the same rank form a front. Besides, a crowding distance [6] is assigned to each individual within the same front. The crowding distance is a measure of how close an individual is to other individuals in the objective space, where a larger crowding distance indicates the individual is farther away from other individuals. Specifically, crowding distance is computed in D steps, where D is the objective space dimensionality. In each dimension d, the individuals are sorted according to their dth objective value. Then, we obtain for each individual the aggregate distance to its two adjacent neighbors with respect to the dth objective. The first and last individuals in each dimension d are assigned a crowding distance of ∞ to preserve diversity. Finally, an individual's crowding distance is calculated as its total aggregate

distances in all dimensions. Please refer to Function 1 for more details.

Function 1 Crowding Distance Assignment

Input: Individuals p_k's in front Z, objective space dimension D

1: Calculate for each individual p_k the objective values $f_{k,1}, \ldots, f_{k,D}$ in the objective space

2: Set I_k to 0 for each individual p_k

3: **for** $d = 1$ to D **do**

4: Sort individuals p_k's in Z in ascending order according to $f_{k,d}$

5: The crowding distance of the first and of the last individual are set to infinity

6: **for** $k = 2$ to the size of Z minus 1 **do**

7: $I_k = I_k + (f_{k-1,d} - f_{k+1,d})/(\max_k\{f_{k,d}\} - \min_k\{f_{k,d}\})$

8: **end for**

9: **end for**

Output: Crowding distances I_k's

Once all individuals are assigned a rank and crowding distance, the next step is to select some individuals from P_0, to create a mating pool for crossover and mutation. The selection is done using binary tournament, i.e., randomly selecting two individuals from P_0 and comparing their ranks. The individual with the smaller rank will be selected for the mating pool. If the two individuals have the same rank, then the one with larger crowding distance is selected. If both individuals have the same rank and the same crowding distance, then either one is selected with a probability of 0.5. After the mating pool is filled, the crossover process starts. Each time two random individuals are taken from the mating pool, called parents, to create two more individuals, called offsprings, with probability p_c. Then, the offspring are mutated with probability p_m. Usually, p_c is large and p_m is small. After N offspring individuals have been created, they are grouped in Q_0.

The ith $(i \geq 1)$ iteration will start by creating an aggregated population $R_i = P_{i-1} \cup Q_{i-1}$. Then all individuals in population R_i will be assigned a rank and crowding distance. Individuals with rank 1 are added to P_i. Recall P_i has a fixed size of N. If there are less than N individuals with rank 1, all individuals with rank 1 will be added to the new population P_i. To fill in the remaining spots in P_i individuals with rank 2 are considered, and so on. When the last front is considered, and its size is larger than the remaining spots, individuals with larger crowding distances will be included in P_i. All other individuals are discarded. Finally, a new offspring population is created by selecting individuals from P_i for the mating pool, as described previously, and performing crossover and mutation. When the number of iterations reaches a predefined threshold, say G, the algorithm stops and the non-dominated individuals can be extracted from P_G to form a Pareto-front.

Notice that the above description does not specify how to conduct crossover and mutation. Next, we introduce these two processes, respectively. In particular, we adopt the simulated binary crossover (SBX) [14] scheme for the crossover process. This procedure creates two offsprings, y and $\tilde{y}$, from two parents x and $\tilde{x}$ as follows. For any $u \in [1, U]$, $h \in [1, H]$, we get

$$y_u^h = \frac{1}{2}\left[(1-\beta^h)x_u^h + (1+\beta^h)\tilde{x}_u^h\right]$$
$$\tilde{y}_u^h = \frac{1}{2}\left[(1+\beta^h)x_u^h + (1-\beta^h)\tilde{x}_u^h\right]$$

where y_u^h and $\tilde{y}_u^h$ are the elements of vectors y and $\tilde{y}$, respectively, x_u^h and $\tilde{x}_u^h$ are the elements of vectors x and $\tilde{x}$, respectively, and β^h is a sample generated by a random number generator shown below:

$$\beta(v) = \begin{cases} \frac{1}{2}(2v)^{\frac{1}{\eta_c+1}}, & v \leq \frac{1}{2} \\ \frac{1}{2}\left[2(1-v)\right]^{-\frac{1}{\eta_c+1}}, & v > \frac{1}{2} \end{cases}$$

where v is a random variable uniformly distributed in [0,1], and η_c is a predefined parameter.

Besides, we perform the mutation process shown in the following. For any $u \in [1, U]$, $h \in [1, H]$, we have

$$y_u^h = x_u^h\left(\frac{1}{2} + \delta\right) \tag{6}$$

where δ is uniformly distributed between 0 and 1.

In the case that the kth decision variable of an offspring after crossover and mutation fall outside the lower and upper bounds specified in the CMOP constraints, they are reset as follows:

$$y_u^h = \begin{cases} x_u^{h,lo}, & \text{if } y_u^h \leq x_u^{h,lo}, \\ x_u^{h,up}, & \text{if } y_u^h \geq x_u^{h,up} \\ y_u^h, & \text{if } x_u^{h,lo} \leq y_u^h \leq x_u^{h,up} \end{cases}$$

We further detail the evolutionary algorithm for load scheduling in Algorithm 1, which is called LSEA.

Algorithm 1 Load Scheduling with an EA (LSEA)

Input: N

1: Create an random initial population, P_0 of size N, satisfying constraints (3)–(5) in the CMOP

2: Apply non-dominating sorting to P_0

3: Apply binary tournament to P_0 to fill mating pool

4: Crossover individuals in mating pool to fill offspring set Q_0

5: Apply mutation to Q_0

6: Set the maximum number of generations, G

7: **for** $g = 1$ to G **do**

8: $R_g = P_{g-1} \cup Q_{g-1}$

9: Apply non-dominating sorting to R_g

10: Apply binary tournament to R_g to fill mating pool

11: Apply crossover to individuals in mating pool to generate Q_g

12: Apply mutation to individuals in Q_g

13: Create P_g

14: **end for**

Output: Non-dominated individuals in P_G

B. Load Scheduling With an ϵ-Approximate Evolutionary Algorithm (ϵ-LSEA)

The evolutionary algorithm proposed above provides a dense and diverse set of solutions on the Pareto front (i.e., the Pareto

TABLE I
COST FUNCTION PARAMETERS (U: THE NUMBER OF USERS, $H = 24$)

a_1	b_1	a_2	b_2	G_1^{max}	G_2^{max}	M_1
0.2	0.3	0.4	0.6	$8U/H$	$16U/H$	1

TABLE II
PARAMETERS IN CONSTRAINTS (2)–(4)

S_u	T_u	$\bar{e}_u$
0	24	0.5kW

optimal solutions). However, a dense set of solutions may not be necessary because adjacent solutions provide similar trade-offs. In the following, we develop an ϵ-approximate evolutionary algorithm for the load scheduling problem.

We first give some definitions as follows [15].

Definition 3: Let $\mathbf{a}$ and $\mathbf{b}$ be two vectors of dimension k' in the objective space. Then $\mathbf{a}$ is said to ϵ-dominate $\mathbf{b}$ for some $\epsilon > 0$, denoted as $\mathbf{a} \succ_\epsilon \mathbf{b}$, if

$$\epsilon \cdot a_i \geq b_i \qquad \forall i \in \{1, \ldots, k'\}.$$

Definition 4: Let $\mathbf{Y}$ be the objective space and $\epsilon > 0$. Then a set $\mathbf{Y}_{\boldsymbol{\epsilon}}$ is called an ϵ-approximate Pareto front of $\mathbf{Y}$, if any vector $\mathbf{b} \in \mathbf{Y}$ is ϵ-dominated by at least one vector $\mathbf{a} \in \mathbf{Y}_{\boldsymbol{\epsilon}}$, i.e.,

$$\forall \mathbf{b} \in \mathbf{Y}, \quad \exists \mathbf{a} \in \mathbf{Y}_{\boldsymbol{\epsilon}} : \mathbf{a} \succ_\epsilon \mathbf{b}.$$

The set of all ϵ-approximate Pareto fronts of $\mathbf{Y}$ is denoted as $\mathbf{P}_{\boldsymbol{\epsilon}}(\mathbf{Y})$.

Definition 5: Let $\mathbf{Y}$ be the objective space and $\epsilon > 0$. Then a set $\mathbf{Y}^*_{\boldsymbol{\epsilon}} \subseteq \mathbf{Y}$ is called an ϵ-Pareto front of $\mathbf{Y}$ if

1) $\mathbf{Y}^*_{\boldsymbol{\epsilon}}$ is an ϵ-approximate Pareto front of $\mathbf{Y}$, i.e., $\mathbf{Y}^*_{\boldsymbol{\epsilon}} \in \mathbf{P}_{\boldsymbol{\epsilon}}(\mathbf{Y})$, and
2) $\mathbf{Y}^*_{\boldsymbol{\epsilon}}$ contains Pareto points of $\mathbf{Y}$ only, i.e., $\mathbf{Y}^*_{\boldsymbol{\epsilon}} \subseteq \mathbf{Y}^*$.

The set of all ϵ-Pareto fronts of $\mathbf{Y}$ is denoted as $\mathbf{P}^*_{\boldsymbol{\epsilon}}(\mathbf{Y})$.

The main idea of ϵ-LSEA is to choose a parent from a variable size population A, called the archive, and another parent from a fixed size population P. After crossover, the resulting offspring may be accepted into the archive depending on whether or not it ϵ-dominates any individual in A. Similarly, the offspring may be accepted into the population depending on its dominance relation to individuals in P. After a predefined number of offsprings have been generated, the solutions in the archive form a diverse ϵ-approximate Pareto front. In what follows, we explain in details the archive acceptance and population acceptance algorithms as well as ϵ-LSEA.

Regarding the archive acceptance algorithm, we adopt the selection strategy proposed by Deb *et al.* [16] to find ϵ-Pareto fronts with guaranteed convergence and diversity, which is shown in Procedure 1. This algorithm divides the two-dimensional objective space into boxes of size $\epsilon \times \epsilon$ and stores in an archive only one non-dominated solution per box on the ϵ-Pareto fronts. Using a generalized dominance relation on these boxes, the algorithm maintains a set of non-dominated boxes, and hence guaranteeing the ϵ-approximation property. In particular, Procedure 1 accepts or rejects an offspring as follows. We first identify the solutions in the archive that are dominated by the current offspring. Here, dominance relation is determined using the vector b of each solution obtained with Function 2. If the offspring dominates any solution, the dominated solution is removed and the offspring is added to the archive. When there are no box-dominated solutions in the archive, we further check two cases. First, if the offspring lies inside a box occupied by an archive solution, then the dominating solution in the Pareto-sense is kept in the archive and the dominated solution is discarded. Second, if the offspring lies inside a box where there is no archive solution, the offspring is added to the archive. Moreover, since in each box there is only one non-dominated solution, the convergence property can be guaranteed, too.

In addition, we have the following theorem [17].

Theorem 1: Let $\mathbf{Y}^{(t)} = \bigcup_{j=1}^{t} \mathbf{y}^{(j)}$, $1 \leq \mathbf{y}_i^{(j)} \leq B$, be the set of all objective vectors created by an multi-objective evolutionary algorithm and given to the selection operator defined in Algorithm 1. Then $\mathbf{A}^{(t)}$ is an ϵ-Pareto set of $\mathbf{Y}^{(t)}$ with bounded size, i.e.,

1) $\mathbf{A}^{(t)} \in \mathbf{P}^*_{\boldsymbol{\epsilon}}(\mathbf{Y^{(t)}})$
2) $|\mathbf{A}^{(t)}| \leq (\log B / \log \epsilon)^{k-1}$

Procedure 1 Selection process for ϵ-Pareto Front

Input: A, f
1: $D := \{f' \in A | box(f) \succ box(f')\}$
2: **if** $D \neq \emptyset$ **then**
3: $\quad A' = A \cup (f \backslash D)$
4: **else if** $\exists f' : (box(f') = box(f) \wedge f \succ f')$ **then**
5: $\quad A' = A \cup f \backslash f'$
6: **else if** $\nexists f' : box(f') = box(f) \vee box(f') \succ box(f)$ **then**
7: $\quad A' = A \cup f$
8: **else**
9: $\quad A' = A$
10: **end if**
Output: A'

Function 2 *box*

Input: f

1: **for all** $i \in \{1, \ldots, m\}$ **do**
2: $\quad b_i = \lfloor \log f_i / \log 1 + \epsilon \rfloor$
3: **end for**
4: $b = (b_i, \ldots, b_m)$
Output: b

Procedure 2 Population Acceptance Procedure for ϵ-LSEA

Input: population P_{g-1}, offspring q
1: Apply Function 1 to P_{g-1} to assign crowding distances CD to each population individual p
2: **if** $\exists p : q \succ p$ **then**
3: $\quad$ Replace the individual p that is dominated by the offspring q and has the smallest CD with q (or break ties randomly).
4: **else if** $\exists p : p \succ q$ **then**
5: $\quad$ Discard q
6: **else**

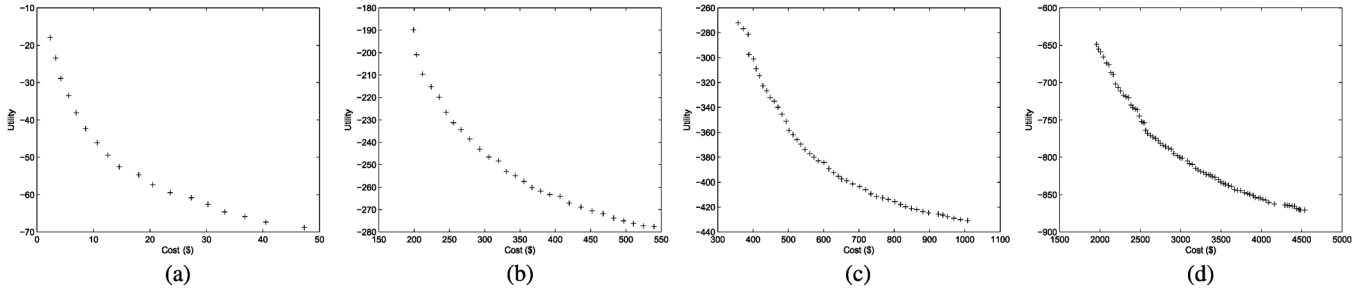


Fig. 1. Pareto front for 5, 15, 25, and 50 users respectively, using LSEA. (a) 5 users; (b) 15 users; (c) 25 users; (d) 50 users.

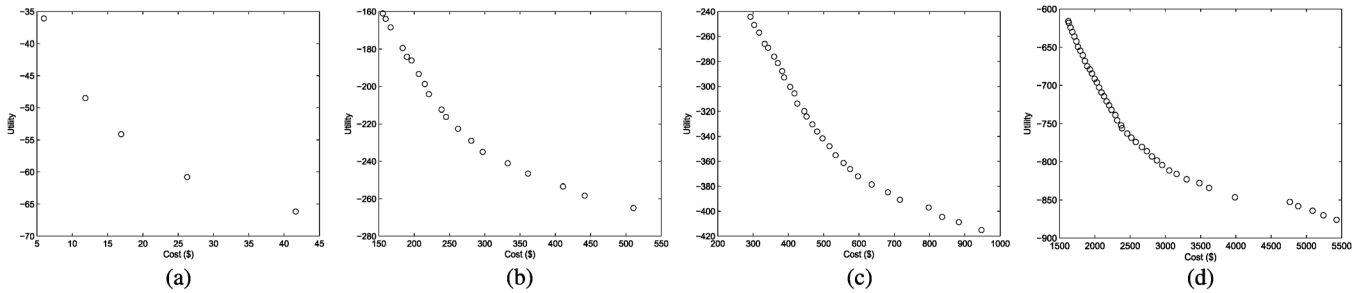


Fig. 2. ϵ-Pareto front for 5, 15, 25, 50 users, respectively, using ϵ-LSEA. (a) 5 users; (b) 15 users; (c) 25 users; (d) 50 users.

7: Replace the p with the smallest CD with offspring q (or break ties randomly).

8: **end if**

Output: P_g

Our population acceptance mechanism, detailed in Procedure 2, uses dominance relations and crowding distances to accept an offspring into the population or reject it. In particular, the algorithm works as follows. First, a crowding distance is assigned to each population individual p in P_{g-1} using Function 1. Next, it is determined if offspring q dominates any p. If it does, the algorithm replaces the dominated p that has the lowest crowding distance CD with q. In case q is dominated by any p, it is rejected. On the other hand, if q does not dominate any p and it is also non-dominated, the p with the lowest CD among all individuals in P_{g-1} is replaced by q. If several individuals have the same lowest CD, then a randomly chosen one is replaced by q. Finally, the procedure returns the updated population P_g. Notice that this procedure only compares the offspring with all members of the population P_{g-1}, rather than compare it with all members of the whole population as in Algorithm 1. This keeps the computational cost low, and the use of crowding distances maintains a well spread population.

Finally, we describe in details the ϵ-approximate evolutionary algorithm (ϵ-LSEA) for the load scheduling problem in Algorithm 2. Initially, a random population P_0 is created satisfying constraints (3)–(5) specified in the CMOP. Then, the non-dominated individuals in P_0 are copied into archive A. In the gth iteration, an individual p is randomly selected from the population P_{g-1} using binary tournament and another solution a is randomly chosen from the archive A to form the mating pool. The parent individuals, p and a, are used for crossover, and the resulting offspring q is subject to mutation. Unlike that in the previous algorithm, only one offspring q is generated per iteration. Next, offspring q is accepted or rejected from the population using Procedure 2. Lastly, Algorithm 1 is used to decide whether or not offspring q is added into the archive A. The algorithm stops after a predefined number of offsprings G have been generated. Since fewer solutions are needed to converge to the Pareto-front, this algorithm has a shorter computation time than Algorithm 1.

TABLE III
COMPLETION TIME

	LSEA		ϵ-LSEA	
Users	**Time(mins)**	**Generations**	**Time(mins)**	**Generations**
5	25	92×10^3	15	26×10^3
15	548	1×10^6	495	5×10^6
25	1312	1.5×10^6	632	13×10^6
50	10033	1×10^6	4978	14×10^6

Algorithm 2 Load Scheduling with an ϵ-Approximate EA (ϵ-LSEA)

1: Create a random initial population, P_0 of size N, satisfying constraints (3)–(5) in the CMOP

2: Copy non-dominated individuals in P_0 to A

3: **for** $g = 1$ to G **do**

4: Choose a solution p from P_{g-1} using binary tournament, and a solution a from A at random

5: Use p and a as parents to create one offspring q.

6: Apply mutation to q resulting in q'

7: Run Procedure 2 to decide if q' is included in population P_g

8: Run Procedure 1 to decide if q' is included in the achieve A

9: **end for**

Output: ϵ-Pareto fronts in A

V. SIMULATION RESULTS

In this section, we conduct simulations to evaluate the performance of the proposed two algorithms, i.e., Load Scheduling with an EA (LSEA, Algorithm 1) and Load Scheduling

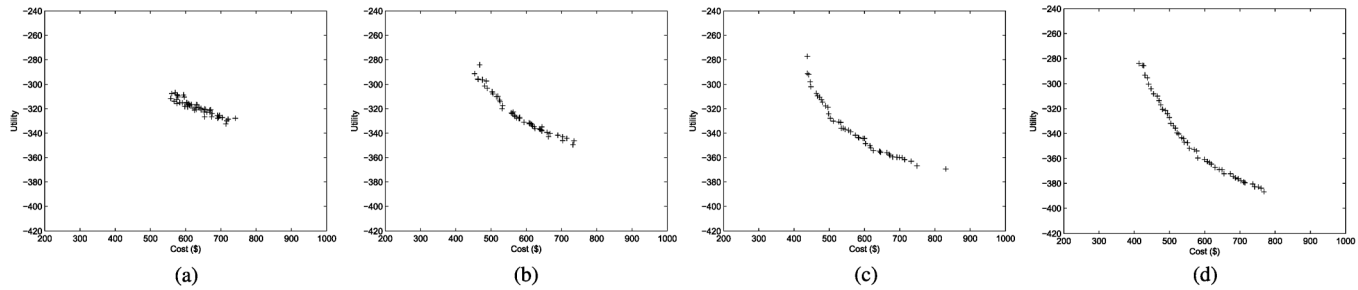


Fig. 3. Population evolution using LSEA at different generations for 25 users. (a) 15 min (5×10^3); (b) 90 min (23×10^3 Iterations); (c) 240 min (80×10^3 Iterations); (d) 600 min (185×10^3 Iterations).

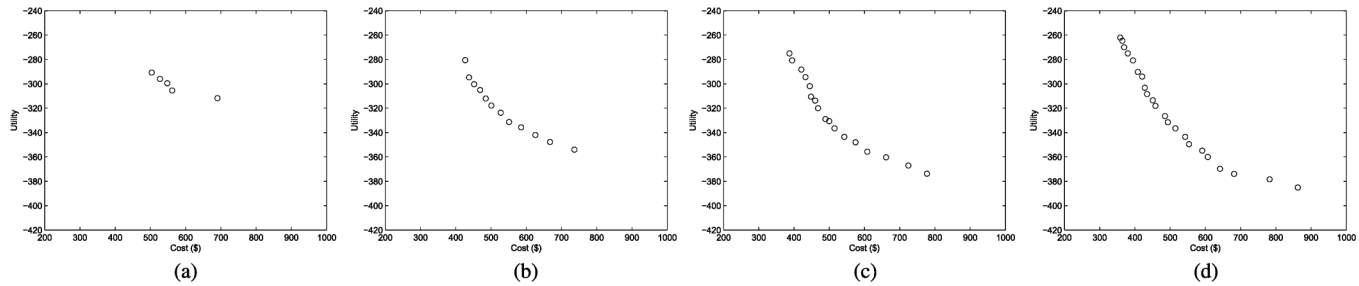


Fig. 4. Population evolution using ϵ-LSEA at different generations for 25 users. (a) 1 min (11×10^3 Iterations); (b) 15 min (152×10^3 Iterations); (c) 60 min (578×10^3 Iterations); (d) 120 min (1×10^6 Iterations).

with an ϵ-approximate EA (ϵ-LSEA, Algorithm 2), respectively. The proposed algorithms are implemented in Matalb2011b on a general purpose computer with a 3.4GHz CPU and 4GB RAM memory. The parameters for the cost function in (1) are presented in Table I, and some parameters indicated in constraints (2), (3) and (4) are given in Table II which are the same for all users. Besides, when two parents are selected for reproduction, the crossover process (SBX) will be applied with probability $p_m = 0.9$ and $\eta_c = 0$, and each offspring will mutate with probability $p_m = e^{-g/G}$, where g is the number of the current iteration and G is the predefined iteration number.

A. LSEA

We first evaluate the performance of LSEA with 5, 15, 25, and 50 users, respectively. In particular, each user has a daily energy requirement e_u, which is uniformly distributed between 0 and 24 kWh, to be scheduled throughout 24 hours. Fig. 1(a) shows the obtained Pareto-front for 5 users. Each cross in the graph represents a solution found by LSEA and its position is determined by the values of the corresponding objective functions. We can observe that the range of the cost objective goes from \$2 to \$48 and the utility function spans from 10 to 70. These solutions in objective space provide us with a wide set of trade-offs between the total energy consumption cost and the overall utility. Moreover, we notice that the Pareto-front is densely populated, i.e., adjacent solutions are very close to each other. Fig. 1(b)–1(d) show similar results for the cases of 15, 25, and 50 users, respectively.

B. ϵ-LSEA

Next, we show the performance of ϵ-LSEA with 5, 15, 25, and 50 users, respectively. The same as before, we assume that each user has a daily energy requirement e_u, which is uniformly distributed between 0 and 24 kWh, to be scheduled throughout 24 hours. As shown in Fig. 2(a), we can easily see there is an ϵ-Pareto front with only a few solutions, which can make the final decision easier. Fig. 2(b)–2(d) also show an ϵ-Pareto front that can be easily identified. Moreover, in these three cases the results are obtained using a large number of iterations. However, as we will show in the next section, in fact a lot fewer generations are enough to obtain an ϵ-Pareto front. Here, we show the results with a large number of iterations after an ϵ-Pareto front has been identified to be sure that the algorithm has converged.

Moreover, the time and the number of iterations needed for obtaining the results shown in Fig. 1 and Fig. 2 are presented in Table III. We can see that the efficiency of ϵ-LSEA is higher than that of LSEA, and the efficiency improvement gets more significant when the number of users becomes larger.

C. Convergence of LSEA and ϵ-LSEA

Finally, we compare the convergence speed of LSEA and ϵ-LSEA by looking into the evolution of the population of LSEA and of the archive of ϵ-LSEA, when the number of users is 25. Fig. 3(a)–3(d) show the progress of the population of LSEA when the running time is equal to 15, 90, 240, and 600 min, respectively. We can find that a good Pareto front can be found only after 600 min. Compared to that, we can see in Fig. 4(a)–4(d) that a good ϵ-Pareto front can be achieved after 120 min, which is much faster. Besides, considering the modest capability of the computer used to run these simulations, the third party usually would have more computing resources and thus even shorter computation time. It can also employ cloud computing to accomplish the load scheduling tasks, which would further reduce the computation time.

VI. Conclusions

In this paper, we consider a third-party managing the energy consumption of a group of smart grid users, and formulate the load scheduling problem as a constrained multi-objective optimization problem. The first objective is to minimize the total energy consumption cost, while the second is to maximize its utility measured by a certain utility function. To solve the

problem, we first develop an evolutionary algorithm, called LSEA, to retrieve a set of Pareto-optimal solutions and show the trade-offs between energy consumption cost and the utility. Then, in order to further improve the algorithm efficiency, we present an ϵ-approximate evolutionary algorithm, called ϵ-LSEA, to obtain ϵ-Pareto fronts of the objective space. Extensive simulations have also been conducted to evaluate the performance of the two proposed algorithms. We can observe that ϵ-LSEA is more efficient compared to LSEA.

REFERENCES

[1] G. Strbac, "Demand side management: Benefits and challenges," *Energy Policy*, vol. 36, no. 12, pp. 4419–4426, November 2008.

[2] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320–331, Dec. 2010.

[3] T. Agarwal and S. Cui, "Noncooperative games for autonomous consumer load balancing over smart grid," CoRR, 2011 [Online]. Available: http://arxiv.org/abs/1104.3802

[4] P. Samadi, R. Schober, and V. Wong, "Optimal energy consumption scheduling using mechanism design for the future smart grid," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Brussels, Belgium, Oct. 2011.

[5] D. Li, S. Jayaweera, and A. Naseri, "Auctioning game based demand response scheduling in smart grid," in *Online Conf. Green Commun. (GreenCom)*, Sep. 2011.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[7] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation," in *Proc. Congr. Evol. Comput. (CEC)*, Washington, DC, Jul. 1999.

[8] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength Pareto evolutionary algorithm," TIK-Rep. No. 103, 2001.

[9] J. Horn, N. Nafpliotis, and D. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput.*, Orlando, FL, Jun. 1994, pp. 82–87.

[10] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Proc. Parallel Problem Solving From Nat. Conf.*, Paris, France, Sep. 2000, pp. 839–848.

[11] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective optimization with messy genetic algorithms," in *Proc. Symp. Appl. Comput.*, Villa Olmo, Italy, Mar. 2000.

[12] U.S. Dept. of Energy, "Communications requirements of smart grid technologies," Oct. 5, 2010.

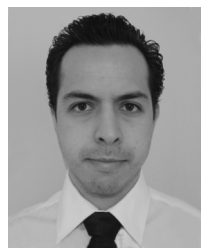
[13] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Springer, 2007.

[14] R. B. Agrawal and K. Deb, "Simulated binary crossover for continuous search space," Indian Institute of Technology, Convenor Tech. Rep., 1994, .

[15] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems. New York: Springer, 2003, vol. 535, pp. 3–38.

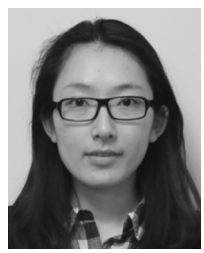
[16] K. Deb, M. Mohan, and S. Mishra, "Evaluating the Domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions," *Evol. Comput.*, vol. 13, no. 4, pp. 501–525, Winter, 2005.

[17] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb, "Archiving with guaranteed convergence and diversity in multi-objective optimization," in *Proc. Genet. Evol. Comput. Conf.*, New York, Jul. 2002.



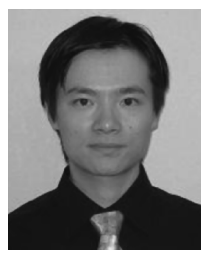
Sergio Salinas (S'06) received the B.S. degree in telecommunications engineering from Jackson State University, Jackson, MS, in 2010. He is currently working towards the Ph.D. degree at the Department of Electrical and Computer Engineering, Mississippi State University, Starkville.

His research interests include cyber-physical systems, cloud computing, and online social networks.



Ming Li (S'09) received the B.E. degree in electrical engineering from Sun Yat-sen University, China, in 2007 and the M.E. degree in electrical engineering from Beijing University of Posts and Communications, China, in 2010, respectively. She is currently working towards the Ph.D. degree in the Department of Electrical and Computer Engineering, Mississippi State University, Starkville.

Her research interests include cross-layer optimization, and security and privacy in cognitive radio networks, smart grids, and cloud computing.



Pan Li (S'06–M'09) received the B.E. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in electrical and computer engineering from University of Florida, Gainesville, in 2009, respectively.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, Mississippi State University, Starkville. His research interests include capacity and connectivity investigation, cross-layer optimization and design, and security and privacy in wireless networks, complex networks, cyber-physical systems, mobile computing, and cloud computing.

Dr. Li has been serving as an Editor for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS—Cognitive Radio Series and IEEE *Communications Surveys and Tutorials*, a Feature Editor for IEEE *Wireless Communications*, and a Guest Editor for IEEE *Wireless Communications* SI on User Cooperation in Wireless Networks. He received the NSF CAREER Award in 2012.