Heuristic Optimization for the Discrete Virtual Power Plant Dispatch Problem

Mette K. Petersen, Lars H. Hansen, Jan Bendtsen, Kristian Edlund, and Jakob Stoustrup

***Abstract*—We consider a virtual power plant, which is given the task of dispatching a fluctuating power supply to a portfolio of flexible consumers. The flexible consumers are modeled as discrete batch processes, and the associated optimization problem is denoted the discrete virtual power plant dispatch problem (DVPPDP). First, the nondeterministic polynomial time (NP)-completeness of the discrete virtual power plant dispatch problem is proved formally. We then proceed to develop tailored versions of the meta-heuristic algorithms hill climber and greedy randomized adaptive search procedure (GRASP). The algorithms are tuned and tested on portfolios of varying sizes. We find that all the tailored algorithms perform satisfactorily in the sense that they are able to find sub-optimal, but usable, solutions to very large problems (on the order of 10^5 units) at computation times on the scale of just 10 s, which is far beyond the capabilities of the optimal algorithms we have tested. In particular, GRASP sorted shows with the most promising performance, as it is able to find solutions that are both agile (sorted) and well balanced, and consistently yields the best numerical performance among the developed algorithms.**

***Index Terms*—Algorithms, computation time, scheduling, suboptimal control.**

I. Introduction

GLOBAL EFFORTS to reduce CO_2 emissions has driven the introduction or renewable power generation technologies into the power system. However, since solar panels and wind turbines harvest energy from sun and wind power availability becomes changeable and more difficult to predict. The smart grid was born out of the need to maintain the balance between production and consumption in this far more volatile power system. In the smart grid a communication link to the consumption side is established, such that flexible consumers like electric vehicles, heat pumps and heating, ventilation, and air conditioning (HVAC) systems can be organized and activated to follow power availability and scarcity, (see [1] and [2]).

A major challenge in developing the smart grid is the sheer size of the optimization problems involved. Solving a dispatch problem for a traditional power system with tens or hundreds of generators is a challenge, which has been researched for decades, (see [3]–[5]). Switching to the smart grid, however, will expand that problem with additional thousands or millions of units. This means that the computation time associated with determining an optimal solution is very likely to be unacceptable in practice.

A review of computation times for optimization in smart grid literature reveals that computation time is still a very real challenge. Table I summarizes problem size and computation times for ten recent scientific publications related to smart grid optimization. Obviously, computation times are highly dependent on the specific structure of the considered problem and the software and platform used for calculation. Nonetheless, Table I does give the general impression that computation times are still quite a lot longer than what one can expect to be acceptable for a fully deployed smart grid operating in real time. This is the case even though several of the cited references investigate heuristic rather than exact optimization methods.

The fastest results found are given in [8] with computation times of just 1.2 s. In [8], however, it is shown that the considered method does not scale to larger problems due to memory overload.

To substantiate the aforementioned assertion, this paper introduces the discrete virtual power plant dispatch problem in which batch processes (constant power consumption with fixed duration) must be scheduled to balance a fluctuating power supply. Similar optimization problems have been investigated in [6]–[8] and [16]. In these papers batch processes are used as simple models of electric vehicles, dishwashers, microwave ovens and more. These papers formulate objectives to schedule units to balance a fluctuating, limited or costly power supply.

After formulating the discrete virtual power plant dispatch problem it is proven formally that the optimization problem is NP-complete [17]. Motivated by this knowledge, we proceed to investigate the performance of two heuristic methods to obtain solutions, which are sub-optimal, but fast to compute. This way a feasible solution will always be available before some predefined power market gate closure. In practice, a suboptimal solution available 2 min before market gate closure is far more valuable than an optimal one available 2 min after.

The methods we will investigate are known in the literature as Hill Climber and greedy randomized adaptive search procedure (GRASP). The main reason for choosing these methods to solve the discrete virtual power plant dispatch problem is that the considered cost function, can be implemented using

Manuscript received August 8, 2013; revised January 23, 2014 and April 26, 2014; accepted June 10, 2014. Paper no. TSG-00619-2013.

M. K. Petersen is with the Department of Electronic Systems, Automation, and Control, Aalborg University, Aalborg 9210, Denmark; and also with DONG Energy, Gentofte 2720, Denmark (e-mail: mehpe@dongenergy.com).

L. H. Hansen and K. Edlund are with DONG Energy, Gentofte 2720, Denmark (e-mail: larha@dongenergy.com; kried@dongenergy.com).

J. Bendtsen and J. Stoustrup are with the Department of Electronic Systems, Automation, and Control, Aalborg University, Aalborg 9210, Denmark (e-mail: dimon@es.aau.dk; jakob@es.aau.dk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2014.2336261

TABLE I
OVERVIEW OF COMPUTATION TIMES REPORTED IN RECENT SCIENTIFIC PUBLICATIONS ON OPTIMIZATION IN SMART GRID APPLICATIONS. SIMULATION SAMPLES IS THE NUMBER OF DISCRETE TIME STEPS THAT THE CONSIDERED SIMULATION HORIZON HAS BEEN SPLIT INTO

Reference	Simulation samples	Num. units	Comp. time
[8]	144	<20	1.2 sec
[9]	24	600	30 sec
[10]	24	5	50 sec
[11]	Unclear	20858	1.1 min
[12]	288	6	3.7 hours
[13]	24	3504	6 hours
[14]	24	50	83 hours
[15]	12	3	Not stated
[16]	144	100	Not stated

delta evaluation; and both Hill Climber and GRASP are able to exploit that.

Delta evaluation means that if the cost associated with on candidate solution is known, and a new solution is obtained through a limited number of permutations of the old solution, then the cost function for the new solution can be computed only from the permutations. Using delta evaluation can only improve calculation time by a constant factor; however, this factor is usually so large, that the actual improvement is far better than algorithmic changes.

Other methods that could be considered for solving the discrete virtual power plant dispatch problem are ant swarm optimization [11], genetic algorithms [19], and tabu search [22]. However, these algorithms are all based on manipulating a set of candidate or tabu solutions. They thus have a tendency to drown in logistics as the majority of the available computation time is spent running through and updating solution sets.

The main contributions of the paper is firstly the proof that the discrete virtual power plant dispatch problem problem is NP-complete. Secondly, we also show that even though finding optimal solutions of the considered problem is indeed very challenging, highly promising results can be obtained in short time frames and for very large problems by special adaptations of the considered heuristic algorithms.

The paper is structured as follows. Firstly, Section II presents the discrete virtual power plant dispatch problem including flexibility modeling and agility. In Section III, the computational complexity of the discrete virtual power plant dispatch problem is explored in several different ways: firstly, the NP-completeness is proven formally and then the feasibility of solving the problem by use of the optimization package CPLEX [18] is explored. In Section IV, four heuristic algorithms are developed, namely uniform selection (UHC), weighted selection (WHC), GRASP random, and GRASP sorted. These algorithms are tuned and compared in Section V. Finally, Section VI summarizes general conclusions and suggestions for future work.

II. OPTIMIZATION PROBLEM

We consider a virtual power plant, which is given the task of satisfying the consumption needs of a portfolio of flexible systems (distributed energy resources) by dispatching a fluctuating power supply.

A forecast of the fluctuating power supply is denoted $P_{\text{Dispatch}}(k), k = 1, 2, \ldots, K$, and the flexible consumers are denoted local units. A portfolio of N local units is denoted $\{LU_i\}_{i=1,2,\ldots,N}$. At sample k we let $P_i(k)$ denote the power to be dispatched to LU_i, and any quantity, which cannot be dispatched to the portfolio, is denoted $\mathcal{S}(\cdot)$. The objective is to minimize the residual power, that is $|\mathcal{S}(\cdot)|$.

The optimization problem can be formulated as

$$\min_{P_i(\cdot)} \sum_{k=1}^{K} w_k |\mathcal{S}(k)| \tag{1}$$

$$s.t.$$

$$P_{\text{Dispatch}}(k) \in \mathbb{R}_+, k = 1, 2, ..., K \tag{2}$$

$$\sum_{i=1}^{N} P_i(k) + \mathcal{S}(k) = P_{\text{Dispatch}}(k) \tag{3}$$

and also subject to the dynamics and constraints of $\{LU_i\}_{i=1,2,\ldots,N}$. Here, K denotes the total simulation horizon.

The impatience weights $w_k \in \mathbb{R}$ have been added, because the forecasted power production will rarely fit exactly with the power needed to satisfy $\{LU_i\}_{i=1,2,\ldots,N}$. In practice, when this happens, a traditional power plant will have to adjust its power consumption, such that the discrepancy between supply and demand is compensated. As a rule of thumb, however, the better time the plant operator has to modify the output of a traditional power plant, the cheaper it can be done. It is therefore desirable to introduce slack as late on the simulation horizon as possible, which can be achieved by introducing $w_k, k = 1, 2, \ldots, K$ and requiring that $w_{k_1} > w_{k_2}$ if $k_1 < k_2$.

A. Flexibility Modeling

In this paper, the flexible units are modeled as batch-processes, which are characterized by constant power consumption, $\overline{P}$, a run time, K_{Run}, and a deadline, K_{End}, by which the process must be finished. Also, we let T_s denote the size of the time step. Each LU can therefore be modeled by

$$E(k+1) = E(k) + T_s P(k) \tag{4}$$

$$P(k) = \overline{P} v(k) \tag{5}$$

$$0 \leq E(k+1) \leq \overline{E} \tag{6}$$

$$E(K_{\text{End}}) = \overline{E} \tag{7}$$

$$0 \leq \sum_{l=k}^{k+K_{Run}-1} v(l) - K_{Run}\,(v(k) - v(k-1)) \tag{8}$$

where $\overline{P} \in \mathbb{R}_+$, $k = 1, 2, \ldots, K$, $v(k) \in \{0, 1\}$, $\overline{E} = \overline{P} K_{\text{Run}}$, $K_{\text{Run}} \leq K$, and $K_{\text{End}} \leq K$. Inequality (8) is the minimum runtime constraint, which ensures that if $v(k) - v(k-1)$ is one, then $v(l), l = k+1, k+2, \ldots, K_{\text{Run}} - 1$ must also all be one; that is, once the local units is activated, it must complete its consumption immediately.

We let $\{LU_i\}_{i=1,2,\ldots,N}$ denote a portfolio of N flexible consumers. Then, for each $\{LU_i\}_{i=1,2,\ldots,N}$ a solution of problem (1)–(3) consists of a set of start times, $\boldsymbol{K}_{\textbf{Start}} = (K_{\text{Start},1}, K_{\text{Start},2}, \ldots, K_{\text{Start,N}})$, one for each LU. An illustration of the discrete virtual power plant dispatch problem for batch processes is given in Fig. 1.

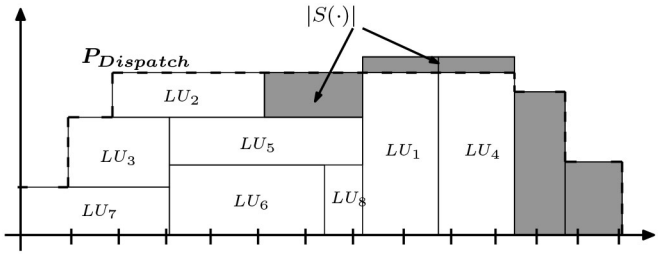

Fig. 1. Solving the discrete virtual power plant dispatch problem essentially corresponds to packing rectangles under an arbitrary profile.

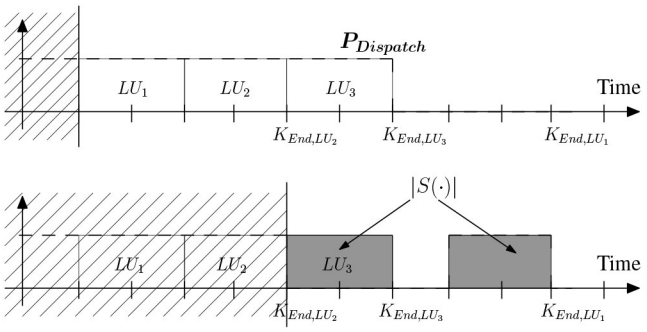

Fig. 2. As time progresses an un-agile solution can become sub-optimal when earlier projections of P_{Dispatch} turn out to be erroneous.

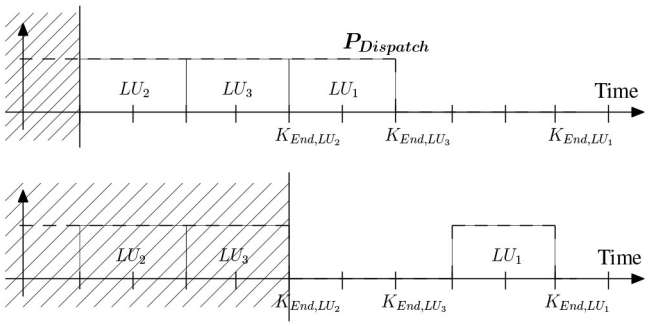

Fig. 3. As time progresses an agile solution is more likely to remain optimal even when earlier projections of P_{Dispatch} were erroneous.

B. Agility

As mentioned earlier, P_{Dispatch} is a forecast of the power production of some renewable production technology. This means that P_{Dispatch} will not correspond exactly to the power, which will actually be produced over the considered time horizon. To alleviate this issue, we want to find a solution of problem (1)–(3), which is as agile as possible. Finding an agile solution means prioritizing the most urgent tasks, namely the ones which are closest to their deadline. In this way, we will have created more maneuverability, if updated forecasts of P_{Dispatch} give very different power availability than originally projected. We also call this maximizing the agility of the portfolio (see Figs. 2 and 3).

The concept of agility is illustrated in Figs. 2 and 3 for a portfolio consisting of three units. The top illustrations in Figs. 2 and 3 both depict optimal solutions of this instance of the discrete virtual power plant dispatch problem and if the predicted progress of P_{Dispatch} is correct, then it is obviously unimportant to distinguish between the two. If, however, a significant portion of the available power is delayed as in the lower illustrations, then the top solution in Fig. 3 remains optimal, but the top solution in Fig. 2 does not. This happens because the top solution in Fig. 2 has left more maneuverability for the optimization in later time steps. Thus, in a sense introducing agility to the problem solving is an attempt to maximize the flexibility of the remaining solution space.

To find an agile solution of problem (1)–(3) the cost function is extended with a term, which adds a penalty to dispatching each LU, that is

$$f(P(\cdot)) = \min_{P_i(\cdot)} \sum_{k=1}^{K} \Big(w_k |\mathcal{S}(k)| + \sum_{i=1}^{N} w_{i,k} |P_i(k)| \Big) \tag{9}$$

s.t.

$$P_{\text{Dispatch}}(k) \in \mathbb{R}_+, k = 1, 2, ..., K \tag{10}$$

$$\sum_{i=1}^{N} P_i(k) + \mathcal{S}(k) = P_{\text{Dispatch}}(k). \tag{11}$$

The agility weights $w_{i,k}$ should then be chosen such that LU_i is dispatched before LU_j, if $K_{\text{End,i}} - K_{\text{Run,i}} < K_{\text{End,j}} - K_{\text{Run,j}}$.

To explain how agility weights are chosen, let $\{LU_i\}_{i=1,2,...,N}$ denote a set of local units sorted according to deadline minus run time. Intuitively, agility weights will penalize increasing the energy term of each LU and this penalty is proportional to the unit index. This means replacing the cost function (9) with

$$\min_{P_i(\cdot)} \sum_{k=1}^{K} \Big(w_k |\mathcal{S}(k)| + \sum_{i=1}^{N} i |E_i(k)| \Big). \tag{12}$$

However, since $E_i(k) = \sum_{l=1}^{k} T_s P_i(l)$ (12) can be written as

$$\min_{P_i(\cdot)} \sum_{k=1}^{K} \Big(w_k |\mathcal{S}(k)| + \sum_{i=1}^{N} i(K+1-k) T_s |P_i(k)| \Big)$$

and the agility weights are therefore given by

$$w_{i,k} = i(K+1-k)T_s. \tag{13}$$

C. Portfolio Generation for Simulation

Throughout this paper, we consider optimization problems on randomly generated portfolios. Each portfolio is characterized by the numbers N and K, such that *Portfolio*(N, K) denotes a randomly generated portfolio of N local units with $K_{\text{Run}} \in \{2, 3, 4, 5\}$, $\overline{P} \in \{1, 2, 3, 4\}$, and $K_{\text{End}} \in \{1, 2, \ldots, K\}$. Also, we set $T_s = 1$ in all simulations and all calculations have been performed on a standard laptop.

Fig. 4 depicts a solution of problem (9)–(11) for a *Portfolio*$(10^5, 100)$ computed by use of the algorithm GRASP Random, which is introduced in Section IV-B. It can be seen that the majority of slack is introduced toward the end of the simulation horizon as the total consumption (blue) is no longer able to follow P_{Dispatch} (black). Agility/sortedness is illustrated by green, red, cyan, and magenta lines illustrating the accumulated consumption of the first, second, third, and fourth quarter of units, respectively, when the portfolio is sorted according to deadline minus runtime.

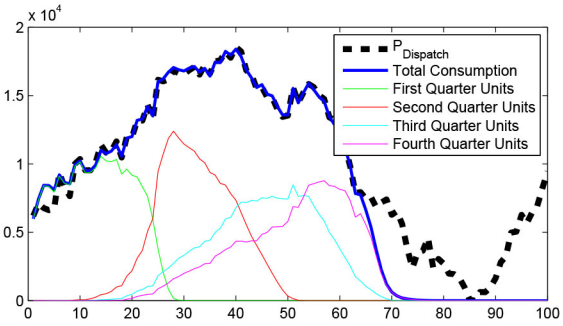

Fig. 4. Solution of problem instance of the discrete virtual power plant dispatch problem with 10^5 local units computed by use of the algorithm GRASP random introduced later in this paper.

III. Computational Complexity

In this section, we will investigate the computational complexity of the discrete virtual power plant dispatch problem. We first do this by proving that the problem is NP-Complete. Next, we attempt to solve the optimization problem via CPLEX. All calculations are performed on a standard laptop.

A. Proof of NP-Completeness

Polynomial-time reductions provide a formal means of showing that one problem is at least as hard to solve as another to within a polynomial-time factor. That is, if $L_1 \leq_P L_2$, then L_1 is not more than a polynomial factor harder than L_2 [17].

Definition 1 (Subset-Sum Problem): Let there be given a finite set $S \in \mathbb{N}$ and a target $t \in \mathbb{N}$. Is there a subset $S' \in S$ whose elements sum to t?

Lemma 1: The subset-sum problem is NP-complete.

Proof: The proof of NP-completeness of the subset-sum problem is done by formulating the $3-CNF-SAT$ (three-conjuncture-normal-form-satisfiability) language. Next it is proved that satisfiability of boolean formulas in $3-CNF-SAT$ is NP-complete. Finally, the subset-sum problem is formulated as boolean formulas in $3-CNF-SAT$, thus proving that $L_{3-CNF-SAT} \leq_P L_{\text{Subsetsum}}$. For full proof see [17]. ■

In the following, we show that a simplified version of the discrete virtual power plant dispatch problem is equivalent to the subset-sum problem. We shall refer to this reduced problem as DVPPDP for brevity.

Theorem 1: The discrete virtual power plant dispatch problem is NP-complete.

Proof: In this proof, it is shown that a subset of the class of instances of the discrete virtual power plant dispatch problem is equivalent to the subset-sum problem. By polynomial reduction this proves that the discrete virtual power plant dispatch problem is NP-complete, since the subset-sum problem is NP-complete.

Firstly, we simplify the discrete virtual power plant dispatch problem by setting agility weights to 0. This reduces problem (9)–(11) to

$$f(P_i(\cdot)) = \min_{P_i(\cdot)} \sum_{k=1}^{K} w_i |\mathcal{S}(k)| \tag{14}$$

s.t.

$$P_{\text{Dispatch}}(k) \in \mathbb{R}_+, k = 1, 2, \ldots, K \tag{15}$$

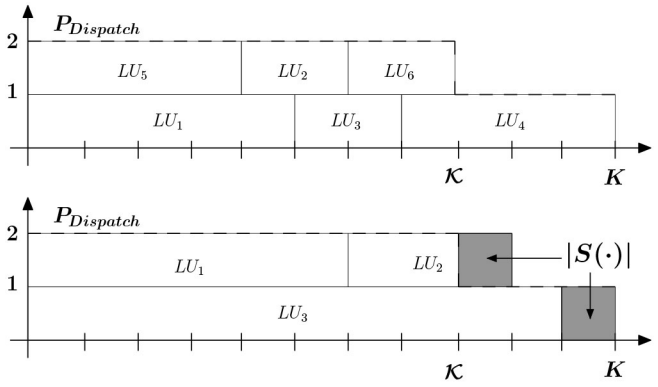

Fig. 5. For the considered instance of the discrete virtual power plant dispatch problem a solution of problem (14)–(16) such that $f(P_i(\cdot)) = 0$ can exist if and only if there also exists a subset of $K_{\text{Run},i}, i = 1, 2, \ldots, N$, which sums to $\mathcal{K}$.

$$\sum_{i=1}^{N} P_i(k) + \mathcal{S}(k) = P_{\text{Dispatch}}(k). \tag{16}$$

Flexible units are still modeled by (4)–(8).

Let $\mathcal{K} \in \mathbb{N}_+$ and $K \in \mathbb{N}_+$ be given and assume without loss of generality that $\mathcal{K} < K$. Next define portfolio $\{LU_i\}_{i=1,2,\ldots,N}$, such that $\overline{P}_i = 1, i = 1, 2, \ldots, N$, $K_{\text{End},i} = K, i = 1, 2, \ldots, N$, and $\sum_{i=1}^{N} K_{\text{Run},i} = \mathcal{K} + K$. Also, define $P_{\text{Dispatch}}(k) = 2, k = 1, 2, \ldots, \mathcal{K}$ and $P_{\text{Dispatch}}(k) = 1, k = \mathcal{K}+1, \mathcal{K}+2, \ldots, K$ (see Fig. 5).

To prove NP-completeness we formulate the following decision problem: given the discrete virtual power plant dispatch problem instance constructed above does there exist a solution of problem (14)–(16) for which $f(P_i(\cdot)) = 0$?

First observe that $\sum_{k=1}^{K} P_{\text{Dispatch}}(k) = \mathcal{K} + K$ and $\sum_{i=1}^{N} K_{\text{Run},i}\overline{P} = \sum_{i=1}^{N} K_{\text{Run},i} = \mathcal{K} + K$ as well. This means that exactly two local units must be on at any sample until sample $\mathcal{K}$ and that exactly one LU must be on at any sample after sample $\mathcal{K}$ in order for a solution with zero slack to exist. However, such a solution can exist if and only if there also exists a subset of $K_{\text{Run},i}, i = 1, 2, \ldots, N$, which sums to $\mathcal{K}$.

This, however, corresponds exactly to the subset-sum problem for the set $S = K_{\text{Run},i}, i = 1, 2, \ldots, N$ and $t = \mathcal{K}$ since $\mathcal{K}$ and K are arbitrarily chosen positive integers. Thus, if there exists a polynomial time algorithm for solving the considered instance of the discrete virtual power plant dispatch problem then this algorithm could also solve the subset-sum problem in polynomial time. It now follows from Lemma 1 that the discrete virtual power plant dispatch problem is NP-complete. ■

B. CPLEX Performance

A commonly used option for solving integer problems is to use the software package CPLEX [18]. If CPLEX is capable of solving the discrete virtual power plant dispatch problem to optimality within a reasonable time frame, then there will be no reason to apply meta-heuristic methods to the problem. In this section, we will therefore investigate how CPLEX handles the discrete virtual power plant dispatch problem.

We have tested CPLEX performance on ten *Portfolio*(25, 100) and ten *Portfolio*(50, 100) problems.

Algorithm 1: Hill Climber ($\{LU_i\}_{i=1,2,...,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,\ 2,...,K}$, n)

1: Generate initial solution $\boldsymbol{K}_{\textbf{Start},0}$ by randomly choosing feasible start samples for each local unit.
2: **Repeat**
3: Select $\boldsymbol{K}_{\textbf{Start}}{}'$ in $\nu_n(\boldsymbol{K}_{\textbf{Start},i})$ by use of **Uniform Selection** or **Weighted Selection**
4: If $f(\boldsymbol{K}_{\textbf{Start}}{}') < f(\boldsymbol{K}_{\textbf{Start},i})$ **then**
5: $\boldsymbol{K}_{\textbf{Start},i+1} = \boldsymbol{K}_{\textbf{Start}}{}'$
6: **Until** time-limit is reached

In many cases the computations are terminated with an error message stating that the computer has run out of memory and therefore no optimal solution is found. We observed that for *Portfolio*(25, 100) five of ten problems were solved with an average computation time of 8 min and for *Portfolio*(50, 100) zero of ten problems were successfully solved. Since all calculations finish due to lack of memory for 50 units and 100 samples, there is little hope that this option will scale to larger problem instances.

IV. Meta-Heuristic Algorithms

Since we have illustrated that finding optimal solutions of the discrete virtual power plant dispatch problem is indeed very challenging, we will now investigate the performance of the meta-heuristic methods Hill Climber [19], and GRASP [20].

A. Hill Climber

We first define a neighborhood associated with the discrete virtual power plant dispatch problem, which is based on the idea of an *n-move*. Next, we develop two variations of the algorithm denoted UHC and WHC.

1) Neighborhood and n-Move: Let an instance of the discrete virtual power plant dispatch problem be given, that is, a set of parameters $\overline{P}_i$, $K_{\text{End,i}}$ and $K_{\text{Run,i}}$, $i = 1, 2, \ldots, N$ and a dispatch sequence $P_{\text{Dispatch}}(k) \in \mathbb{R}, k = 1, 2, \ldots, K$. A solution of the discrete virtual power plant dispatch problem is then given by a set of feasible start times, $\boldsymbol{K}_{\textbf{Start}} = (K_{\text{Start},1}, \ldots, K_{\text{Start},N})$. The solution space is given by

$$S = \{(K_{\text{Start},1}, \ldots, K_{\text{Start},N}) \in \mathbb{N}^N \mid K_{\text{Start},i} < K_{\text{End},i} - K_{\text{Run},i}, i = 1, 2, \ldots, N\}$$

and we have that

$$|S| = \Pi_{i=1}^{N} K_{\text{End},i} - K_{\text{Run},i} \leq K^N.$$

In the discrete virtual power plant dispatch problem a neighborhood map ν_n:$S \rightarrow S^M$, defines for each solution $\boldsymbol{K}_{\textbf{Start}}$ a neighborhood set $S_n(\boldsymbol{K}_{\textbf{Start}}) \in S^M$ consisting of the set of solutions that can be obtained from $\boldsymbol{K}_{\textbf{Start}}$ by moving the start time of any n local units to feasible locations. This is called an n-move. In other words, a neighborhood map is a map of the form

$$\nu_n(\boldsymbol{K}_{\textbf{Start}}) = \{\boldsymbol{K}_{\textbf{Start}}{}' \in S \mid \boldsymbol{K}_{\textbf{Start}}{}' \text{ is obtained from } \boldsymbol{K}_{\textbf{Start}} \text{ by an } n\text{-move}\}.$$

Algorithm 2: Uniform Selection ($\{LU_i\}_{i=1,2,...,N}$, n)

1: **for** $j = 1$ **to** n **do**
2: Select LU_i from $\{LU_i\}_{i=1,2,...,N-j+1}$ with probability $\frac{1}{N-j+1}$
3: Select start sample k for LU_i with probability $\frac{1}{K_{\text{Start},i}}$
4: Set $\{LU_i\}_{i=1,2,...,N-j} = \{LU_i\}_{i=1,2,...,N-j+1} \backslash LU_k$
5: **end for**

Algorithm 3: Weighted Selection ($\{LU_i\}_{i=1,2,...,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,2,...,K}$, n)

1: **for** $j = 1$ **to** n **do**
2: Select LU_i from $\{LU_i\}_{i=1,2,...,N-j+1}$ with probability $\frac{1}{N-j+1}$
3: **for** $k = 1$ **to** $K_{\text{End},i} - K_{\text{Run},i}$ **do**
4: $v_k \leftarrow \Delta_{Cost}(i, k)$ given that $j-1$ local units have already been assigned new start samples.
5: **end for**
6: Construct $v_{negative}$ containing all negative numbers in v.
7: **if** Length($v_{negative}$) ≥ 1 **then**
8: Select start sample k for LU_i with probability $\frac{v_{negative}(k)}{\sum v_{negative}}$
9: **else**
10: Select start sample k for LU_i with probability $\frac{\frac{1}{v(k)}}{\sum \frac{1}{v}}$
11: **end if**
12: Set $\{LU_i\}_{i=1,2,...,N-j} = \{LU_i\}_{i=1,2,...,N-j+1} \backslash LU_k$
13: **end for**

2) Uniform Selection Hill Climber and Weighted Selection Hill Climber: Pseudo code for the Hill Climber method is given in Algorithm 1. The Hill Climber method first generates a random initial solution for the considered problem. Next, a solution in the neighborhood of the current solution is found. If the cost of the neighboring solution is less than the cost of the current solution, then the neighbor solution will take its place as current solution. This procedure is continued until the time limit is reached.

Two tailored versions of the Hill Climber method, denoted UHC and WHC will be investigated.

In UHC the initial solution $\boldsymbol{K}_{\textbf{Start}}{}'$ is found by choosing n local units from the portfolio with uniform probability and then selecting feasible start samples for each LU, also with uniform probability. Pseudo-code for uniform selection is given in Algorithm 2. Allowing n to be larger than one means that it is possible to accept a solution where a unit is moved to a less favorable start sample as long as other units are simultaneously moved to beneficial positions. This could help the algorithm escape a local optimum, which would not be possible if only one LU can be moved at a time.

In an alternative implementation, denoted WHC, the n local units are again chosen uniformly from the portfolio, but the start time of each LU is now chosen with probability proportional to the benefit/cost of moving the start time of the LU to each feasible time slot. Pseudo-code for weighted selection is given in Algorithm 3. If improving start times exist, we choose an improving time with probability proportional to the obtained benefit. On the other hand, if no improving

Algorithm 4: GRASP Random ($\{LU_i\}_{i=1,2,\ldots,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,2,\ldots,K}$, m, l, n)

repeat
  for $j = 1$ **to** N **do**
    for $k = 1$ **to** m **do**
      Unit List(k) $\leftarrow$ Select LU_i from $\{LU_i\}_{i=1,2,\ldots,N-j+1}$ with probability $\frac{1}{N-j+1}$.
      for $h = 1$ **to** $K_{\text{Start,Unit List}(k)} - K_{\text{Run,Unit List}(k)}$ **do**
        $v_{k,h} \leftarrow \Delta_{Cost}$(Unit List($k$), h) given that $j-1$ local units have already been assigned start samples.
      end for
    end for
    $CandidateList_{k,h} \leftarrow$ The l smallest entries in $v_{k,h}$.
    Select LU_k and start sample h from *CandidateList* with probability $\frac{1}{l}$.
    Set start time of LU_k to h and set $\{LU_i\}_{i=1,2,\ldots,N-j} = \{LU_i\}_{i=1,2,\ldots,N-j+1} \backslash LU_k$.
  end for
  Hill-Climber($\{LU_i\}_{i=1,2,\ldots,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,2,\ldots,K}$, n)
until time-limit is reached

start times exist, then we choose a deteriorating time with probability inversely proportional to the cost.

UHC and WHC are tuned and compared in Section V.

B. GRASP

A definite weakness of the developed Hill Climber methods is that the initial solution is generated by choosing local units and start time with uniform probability. This means that the solutions generated for initial use have absolutely no similarity to P_{Dispatch}. This problem is mended in GRASP.

The idea in GRASP is to construct an initial solution one element at a time by use of a greedy algorithm. The choice of next element to be added is determined by constructing a candidate list of most beneficial choices. The probabilistic element in GRASP is then introduced by randomly choosing one of the candidates in the candidate list, but not necessarily the top candidate. After an initial solution has been generated, Hill Climber is called to achieve a further improvement of the solution.

Again two versions of the algorithm have been investigated, namely GRASP random and GRASP sorted. GRASP random falls closest to the generic description of the GRASP algorithm, but as we will see later it has some challenges related to the discrete virtual power plant dispatch problem. To address this GRASP sorted is developed as well.

Pseudo-code for GRASP random is given in Algorithm 4. The idea is that m local units are chosen randomly from the portfolio and placed in the unit list. Next, the cost of starting each LU in the unit list at each feasible sample is computed and saved in v. The candidate list is then generated by choosing the l smallest elements from v. Finally, an element from the candidate list is chosen with uniform probability.

When exploring GRASP random it was discovered that for all problem sizes, GRASP random generates initial solutions, which overshoot P_{Dispatch} in the beginning of the horizon and fall lower than P_{Dispatch} toward the end of the horizon. This behavior can be explained as follows.

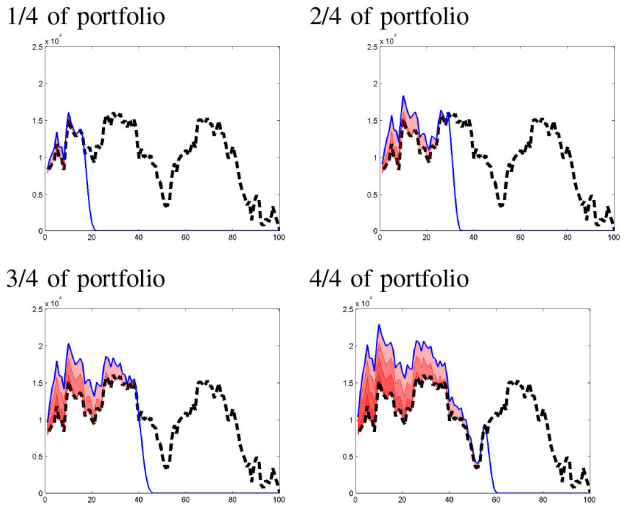

Fig. 6. GRASP random builds an initial solution one unit at a time and above it is depicted how the initial solution looks, when 1/4, 2/4, 3/4, and 4/4 of the units in the portfolio have been added. Since slack is cheaper toward the end of the horizon, GRASP random will first start units early in the horizon. When 1/4 of the portfolio has been added a decent fit with P_{Dispatch} has been obtained for sample 0–18. However, there are still units remaining in the portfolio, which have a deadline of 18 or less, and now GRASP random can only add these in such a way that the accumulated power consumption before sample 18 overshoots P_{Dispatch} even further. This means that as units are added more and more positive slack builds up at the beginning of the simulation horizon, as can be seen from the progression of the figures.

Since slack is cheaper toward the end of the horizon GRASP random will first start units early in the horizon as it builds an initial solution. At some point a decent match with P_{Dispatch} is obtained for, say, the first ten samples. However, if a LU then remains in $\{LU_i\}_{i=1,2,\ldots,N-j}$, which has deadline 10 or less, then it can only be added such that it makes the accumulated power consumption overshoot P_{Dispatch} somewhere in the first ten samples. When this has happened a number of times (see Fig. 6) the result is a consumption profile, which overshoots P_{Dispatch} in the beginning of the horizon and falls lower than P_{Dispatch} toward the end of the horizon.

To alleviate this problem, the algorithm GRASP sorted was developed. The algorithm is identical to GRASP random except that local units are not initially chosen at random, but in sorted order, starting with the n local units with the earliest deadlines. The candidate list is again generated based on the unit list and an element from the candidate list is chosen with uniform probability. Pseudo-code for GRASP sorted is given in Algorithm 5.

V. Results

Before the algorithms can be compared they must be properly tuned. When applying a meta-heuristic there will always be a trade-off between time and performance, so computation time is fixed, not a parameter. All calculations are performed on a standard laptop. The algorithms have been implemented in C# [21].

A. Tuning

To tune the algorithms, a representative training test set of R problem instances is generated and each algorithm is

Algorithm 5: GRASP Sorted ($\{LU_i\}_{i=1,2,...,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,2,...,K}$, m, l, n)

1: **repeat**
2: Sort $\{LU_i\}_{i=1,2,...,N}$ according to $K_{\text{End},i} - K_{\text{Run},i}$.
3: **for** $k = 1$ **to** m **do**
4: Unit List(k) $\leftarrow LU_k$.
5: **end for**
6: **for** $j = 1$ **to** N **do**
7: **for** $h = 1$ **to** $K_{\text{Start,Unit List}(k)} - K_{\text{Run,Unit List}(k)}$ **do**
8: $v_{k,h} \leftarrow \Delta_{Cost}(\text{Unit List}(k), h)$ given that $j-1$ local units have already been assigned start samples.
9: **end for**
10: $CandidateList_{k,h} \leftarrow$ The l smallest entries in $v_{k,h}$.
11: Select LU_k and start sample h from $CandidateList$ with probability $\frac{1}{l}$.
12: Set start time of LU_k to h and set $UnitList = UnitList \backslash LU_k \cup LU_{j+1}$.
13: **end for**
14: **Hill-Climber**($\{LU_i\}_{i=1,2,...,N}$, $\{P_{\text{Dispatch}}(k)\}_{k=1,2,...,K}$, n)
15: **until** Time limit is reached.

TABLE II
PERCENTAGE GAP AND PERCENTAGE DEVIATION FOR ALL DEVELOPED METHODS

	1.000 units		10.000 units		100.000 units	
	E	σ	E	σ	E	σ
Uniform Select.	32.4	7.7	58.6	3.6	47.8	1.5
Weighted Select.	38.0	8.6	62.5	4.5	59.3	1.7
GRASP Random	23.7	3.1	65.3	2.5	32.9	1.4
GRASP Sorted	0.6	0.4	2.5	0.3	0.7	0.5

run T times on each training data set. In order to be able to compare performance on problem instances with very different absolute values we compute the percentage gap and the percentage deviation. Since it is not feasible to determine the optimal solution of problems of the considered size, the percentage gap and the percentage deviation is computed relative to the minimum value found over all calculations on each particular problem instance.

In order to be able to compare performance on problem instances with very different absolute values, we compute the percentage gap $E = (1/S)\sum_{j=1}^{T}(((z_j - z^*)\cdot 100)/z^*)$ and the percentage deviation $\sigma = \sqrt{(1/(T-1))\sum_{j=1}^{T}\left(((z_j - z^*)\cdot 100)/z^*) - E\right)^2}$, where z^* is the optimal solution of the problem instance and j indexes the set of T candidate solutions. Since z^* is not known we will substitute the minimum value found over all calculations on the particular problem instance.

The tuning test is performed on randomly generated *Portfolio*$(N, 100)$, $N = 10^3, 10^4, 10^5$ (see Section II-C). The tuning set consists of nine instances of the discrete virtual power plant dispatch problem (sets of portfolio and P_{Dispatch}) with three portfolios of 10^3, 10^4, and 10^5 local units each. Computation time is fixed at 10 s for all runs of the algorithms and $T = 10$. Results of parameter tuning test are given in Tables III and IV where **A** = 10^3 local units, **B** = 10^4 local units, and **C** = 10^5 local units.

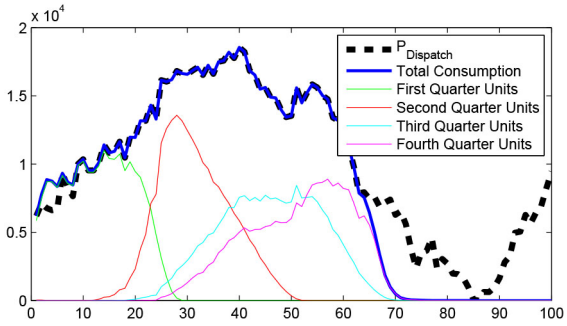

Fig. 7. Performance of uniform selection hill climber for a *Portfolio*(10^5, 100) problem.

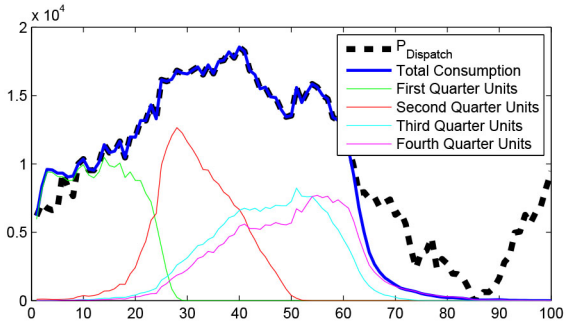

Fig. 8. Performance of weighted selection hill climber for a *Portfolio*(10^5, 100) problem.

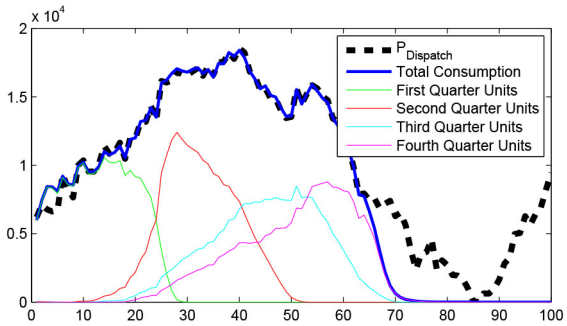

Fig. 9. Performance of GRASP random for a *Portfolio*(10^5, 100) problem.

B. Results

To finally test the algorithms a new *Portfolio*$(N, 100)$, $N \in \{10^3, 10^4, 10^5\}$ is generated with three portfolios of 10^3, 10^4, and 10^5 local units each. Computation time is still fixed at 10 s for all runs of the algorithms and $T = 10$. The performance of all four algorithms is given in Table II.

It is found that for all problem sizes the Hill Climber methods and GRASP random have very similar performance, with no clear winner. GRASP sorted, on the other hand, outperforms all the other methods by at least an order of magnitude both in terms of percentage gap and percentage deviation and for all problem sizes.

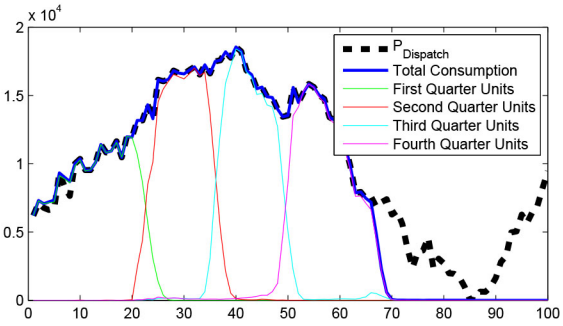

Fig. 10. Performance of GRASP sorted for a *Portfolio*(10^5, 100) problem.

TABLE III
DEVELOPED HILL CLIMBER METHODS HAVE ONE PARAMETER, NAMELY THE VALUE OF n IN $n-move$. TO PROPERLY JUDGE THE PERFORMANCE OF THE ALGORITHM A SUITABLE VALUE OF THIS PARAMETER MUST BE FOUND. THIS TABLE STATES THE BEST PARAMETER VALUE FOUND FOR BOTH HILL CLIMBER METHODS

Parameter	Test Values	Uniform Select. A	Uniform Select. B	Uniform Select. C	Weighted Select. A	Weighted Select. B	Weighted Select. C
$n-move$	{1, 5, 10, 50, 100, 500}	1	1	1	10	10	10

Figs. 7–10 depict solutions found for a *Portfolio*(100.000, 100) problem using each algorithm and the parameters given in Tables III and IV. A visual inspection confirms the general conclusions that UHC, WHC, and GRASP random have very similar performance as the curves can hardly be distinguished. However, GRASP sorted is clearly superior. It can be seen that particularly at the beginning of the simulations GRASP sorted has less slack than the other methods and GRASP sorted has furthermore found a far more sorted solution, which can be seen by the very steep slopes of the quarter lines in Fig. 10.

As demonstrated in Section III there exists no efficient strategy for determining optimal solutions of the virtual power plant dispatch problemfor large problem instances. One option would be to generate artificial, structured problem instances where the optimal solution is known. However, inherent structure in a problem could likely favor one of the methods, in particular GRASP sorted, and would thus lead to unfair comparisons.

Our best available optimal solutions are therefore the five *Portfolio*(25, 100) problem instances, which were successfully found by CPLEX in Section III-B. In Table V, GRASP sorted has been retuned for *Portfolio*(25, 100) problems and the average performance over ten solutions is given for these problems. It is found that the deviation from optimality is 6% to 13%, which is fairly good considering the short computation time.

VI. CONCLUSION

In the vision for the future smart grid, not just hundreds, but thousands or even millions of flexible consumers must be coordinated to operate in a sensible, interconnected manner. A major issue in implementing the smart grid, however, is that this must happen in real time. In this paper, we have therefore investigated computational speed of large-scale dispatch problems.

TABLE IV
DEVELOPED GRASP METHODS HAVE THE FIVE PARAMETERS LISTED UNDER "PARAMETER." THIS TABLE STATES THE BEST PARAMETER VALUE COMBINATION FOUND FOR BOTH GRASP METHODS

Parameter	Test Values	GRASP Random A	GRASP Random B	GRASP Random C	GRASP Sorted A	GRASP Sorted B	GRASP Sorted C
m	{1, 2, 3, 4}	1	1	1	4	4	4
l	{1, 2, 3}	3	3	3	1	1	2
Hill Climber-type	**Uniform** or **Weighted**	**W**	**W**	**U**	**W**	**U**	**W**
n	{1, 2, 3}	2	3	1	3	2	1
Time parameter for **Hill Climber**	{1, 2, 3, 4, 5, 8, 10, 15, 20} times the time for generating initial solution	15	15	15	15	15	4

TABLE V
OPTIMAL SOLUTIONS VALUES FOUND IN CPLEX AND SOLUTION VALUES FOUND BY GRASP SORTED FOR FIVE *PORTFOLIO*(25, 100) PROBLEMS INSTANCES

	Optimal	GRASP Sorted	Increase
Test 1	8224385	8755518	6%
Test 2	8509987	9208134	8%
Test 3	10881238	11893818	9%
Test 4	11326497	12614928	11%
Test 5	9763022	11055801	13%

Our investigations have concentrated on the discrete virtual power plant dispatch problem. Firstly, the computational complexity was investigated by proving that the problem is NP-complete and investigating the options of solving the discrete virtual power plant dispatch problem by use of CPLEX [18]. Being NP-complete the discrete virtual power plant dispatch problem is therefore at least as complex (hard) to solve as the well-known unit commitment problem [23].

We therefore developed heuristic methods for solving the problem and specifically looked at Hill Climber and GRASP. Four algorithms were developed, denoted UHC, WHC, GRASP random, and GRASP sorted. After tuning and testing, by far the best results where obtained by GRASP sorted. This method can determine solutions, which are both agile (sorted) and well balanced even for problems of 100.000 units, 100 samples and with a computation time of just 10 s.

REFERENCES

[1] A. Mohsenian-Rad, V. W. S. 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.

[2] D. S. Callaway and I. A. Hiskens, "Achieving controllability of electric loads," *Proc. IEEE*, vol. 99, no. 1, pp. 184–199, Jan. 2011.

[3] K. Edlund, J. D. Bendtsen, and J. B. Jørgensen, "Hierarchical model-based predictive control of a power plant portfolio," *Control Eng. Pract.*, vol. 19, no. 10, pp. 1126–1136, Oct. 2011.

[4] K. Aoki, T. Satoh, M. Itoh, T. Ichimori, and K. Masegi, "Unit commitment in a large-scale power system including fuel constrained thermal and pumped-storage hydro," *IEEE Trans. Power Syst.*, vol. 2, no. 4, pp. 1077–1084, Nov. 1987.

[5] A. K. Ayuob and A. D. Patton, "Optimal thermal generating unit commitment," *IEEE Trans. Power App. Syst.*, vol. 90, no. 4, pp. 1752–1756, Aug. 1971.

[6] J. Huang, V. Gupta, and Y.-F. Huang, "Scheduling algorithms for PHEV charging in shared parking lots," in *Proc. 2012 Amer. Control Conf.,* Montreal, QC, Canada, 2012, pp. 276–281.
[7] S. Chen, P. Sinha, and N. B. Shroff, "Scheduling heterogeneous delay tolerant tasks in smart grid with renewable energy," in *Proc. 51st IEEE Conf. Decis. Control*, Maui, HI, USA, 2012, pp. 1130–1135.
[8] K. C. Sou, J. Weimer, H. Sandberg, and K. H. Johansson, "Scheduling smart home appliances using mixed integer linear programming," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.,* Orlando, FL, USA, 2011, pp. 5144–5149.
[9] D. T. Phan, J. Xiong, and S. Ghosh, "A distributed scheme for fair EV charging under transmission constraints," in *Proc. Amer. Control Conf.,* Montreal, QC, Canada, 2012, pp. 1053–1058.
[10] Z. A. Vale, H. Morais, H. Khodr, B. Canizes, and J. Soares, "Technical and economic resources management in smart grids using heuristic optimization methods," in *Proc. IEEE Power Energy Soc. General Meet.,* Minneapolis, MN, USA, 2010, pp. 1–7.
[11] P. Faria, J. Soares, Z. Vale, H. Morais, and T. Sousa, "Modified particle swarm optimization applied to integrated demand response and DG resources scheduling," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 606–616, Mar. 2013.
[12] Z. Chen, L. Wu, and Y. Fu, "Real-time price-based demand response management for residential appliances via stochastic optimization and robust optimization," *IEEE Trans. Smart Grid*, vol. 3, no. 4, pp. 1822–1831, Dec. 2012.
[13] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in smart grid using heuristic optimization," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1244–1252, Sep. 2012.
[14] S. Salinas, M. Li, and P. Li, "Multi-objective optimal energy consumption scheduling in smart grids," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 341–348, Mar. 2013.
[15] P. Yi, X. Dong, A. Iwayemi, C. Zhou, and S. Li, "Real-time opportunistic scheduling for residential demand response," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 227–234, Mar. 2013.
[16] A. Subramanianz *et al.*, "Real-time scheduling of deferrable electric loads," in *Proc. Amer. Control Conf.,* 2012, pp. 3643–3650.
[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001.
[18] (2014, Jun. 18) [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/
[19] E. K. Burke and G. Kendall, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, New York, NY, USA: Springer, 2005.
[20] T. A. Feo and M G. C. Resende, "Greedy randomized adaptive search procedures," *J. Global Optim.*, vol. 6, no. 2, pp. 109–133, Mar. 1995.
[21] (2014 Jun. 18) [Online]. Available: http:www.msdn.microsoft.com
[22] S. I. Gass and C. M. Harris, *Encyclopedia of Operations Research and Management Science*. Springer, 1996.
[23] G. Xiaohong, Z. Qiaozhu, and A. Papalexopoulo, "Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming," in *Proc. IEEE Power Eng. Soc. Gener. Meet.,* Jul. 2003, pp. 1095–1100.

Mette K. Petersen received the M.Sc. degree in applied mathematics from the Technical University of Denmark, Kongens Lyngby, Denmark, in 2008, and is currently pursuing the industrial Ph.D. degree in collaboration with the Danish utility company DONG Energy, Gentofte, Denmark, and Aalborg University, Aalborg, Denmark.

Her research interests include mathematical modeling, optimization, and numerical methods.

Lars H. Hansen received the M.Sc. degree in electrical engineering in 1994, and the Ph.D. degree in mathematical modeling in 1997 from the Technical University of Denmark, Kongens Lyngby, Denmark.

From 1997 to 2001, he was a Post-Doctoral Researcher at the Wind Energy Department, Risoe National Laboratory, Roskilde, Denmark. Since 2001, he has been with industrial process control, and since 2008 he has been with smart grid research projects and wind energy projects at DONG Energy, Gentofte, Denmark. His research interests include modeling, and advanced control and numerical methods for wind power and smart grid.

Jan Bendtsen was born in Denmark in 1972. He received the M.Sc. and the Ph.D. degrees from the Department of Control Engineering, Aalborg University, Aalborg, Denmark, in 1996 and 1999, respectively.

Since 2003, he has been an Associate Professor with the Department of Electronic Systems, Aalborg University. In 2005, he was a Visiting Researcher with Australian National University, Canberra, ACT, Australia. Since 2006, he has been involved in the management of several national and international research projects, and organizing international conferences. In 2012, he was a Visiting Researcher with the University of California, San Diego, CA, USA. His research interests include adaptive control of nonlinear systems, closed loop system identification, control of energy systems, and linear parameter varying systems.

Dr. Bendtsen was the co-recipient of the Best Technical Paper Award at American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference, 2009.

Kristian Edlund received the M.Sc. Electrical Engineering (EE) degree and the Ph.D. degree in control engineering from Aalborg University, Aalborg, Denmark, in 2006 and 2010, respectively.

He is currently with the Development of Power Hub, a commercial smart-grid and flexibility product from the Danish utility company DONG Energy, Gentofte, Denmark. His research interests include the fields of mathematical optimization modeling, control of energy systems, and commodity market trading in a smart grid context.

Jakob Stoustrup received the M.Sc. degree in electrical engineering in 1987, and the Ph.D. degree in applied mathematics in 1991 from the Technical University of Denmark, Kongens Lyngby, Denmark.

From 1991 to 1996, he held several positions in the Department of Mathematics, Technical University of Denmark. From 1997 to 2013, he was a Professor of Automation and Control, Aalborg University, Aalborg, Denmark, where he has been the Head of Research for the Department of Electronic Systems since 2006. Since 2014, he has been a Chief Scientist at Pacific Northwest National Laboratory, Richland, WA, USA, leading the control of complex systems initiative for the U.S. Department of Energy. His research interests include robust control theory and the theory of fault tolerant control systems.