Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 1285–1292

ISBN 978-83-60810-48-4

A Distributed Greedy Algorithm for Constraint-based Scheduling of Energy Resources

Jörg Bremer
Department of Computing Science
Carl von Ossietzky University
Oldenburg, Germany
joerg.bremer@uni-oldenburg.de

Michael Sonnenschein
Department of Computing Science
Carl von Ossietzky University
Oldenburg, Germany
michael.sonnenschein@uni-oldenburg.de

***Abstract*—The current upheaval in the electricity sector is leading to distributed generation schemes and new grid structures. At the same time, this change is heading for a paradigm shift in controlling energy resources within the grid. Pro-active scheduling of active power within a (from a controlling perspective) loosely coupled group of distributed energy resources demands for distributed optimization methods that take into account the individual feasible region in local search spaces. We propose a method that uses support vector based black-box models for constructing feasible regions for automated, local solution repair during scheduling and combine it with a distributed greedy approach for finding an appropriate partition of a desired target schedule into operable schedules for each participating actor.**

I. Introduction

IN ORDER to allow for a transition of the current central market and network structure of today's electricity grid to a decentralized smart grid, an efficient management of numerous distributed energy resources (DER) will become more and more indispensable. Integrating a continuously rising number of renewable resources means controlling individually configured and rather small devices in order to cope with stochastic feed-in effects.

We consider in general producers that are supposed to pool together with likewise distributed electricity consumers and prosumers (like batteries) in order to jointly gain more degrees of freedom in choosing load schedules. In this way, they become a single controllable entity with sufficient market power. In order to manage such a pool of DER, the following distributed optimization problem has to be frequently solved: A partition of a demanded (by market) aggregate schedule has to be determined in order to fairly distribute the load among all participating DER. Optimality usually refers to local (individual cost) as well as to global (e.g. environmental impact) objectives in addition to the main goal: Resemble the wanted overall load schedule as close as possible.

In order to choose an appropriate schedule for each participating DER, an optimization algorithm must know, which schedules are actually operable and which are not. Depending on the type of DER, different constraints restrict possible operations. The information about individual local feasibility of schedules has to be spread appropriately in (distributed) optimization scenarios, in order to evaluate objectives globally in distributed search spaces. For this purpose, meta-models of constrained spaces of operable schedules have been shown indispensable for efficient communication [1]. Such models can be seen as black-box representations of the feasible region of an optimization problem related to scheduling scenarios. Such models are also used for efficiently evaluating constraints during the optimization procedure for cases where determining the feasible region has comparably high computational costs.

Real world problems like this scheduling problem often face nonlinear or combined constraints. The set of constraints defines the shape of a region within the search space (the hypercube defined by parameter limits) that contains all feasible solutions. This region is called feasible region and might be arbitrary shaped or even be discontinuous.

At the same time, support vector machines and related approaches have been shown to have excellent performance when trained as classifiers for multiple purposes, especially real world problems. As a use case related to describing the region where some given data resides in, Tax and Duin developed the support vector domain description (SVDD) as a one-class support vector classification approach that is capable of learning the region that is defined by some given training data [2] and has therefore been harnessed for example by [3] as a model for the feasible region in the smart grid domain.

What we will now add to these two worlds is a new approach for integrating constraints that are modeled by a support vector classifier into distributed optimization in a way, that allows for an efficient navigation within the feasible region. The basic idea is to construct a mapping from the whole, unconstrained domain of the problem (the hypercube) to the feasible region to be able to automatically repair an infeasible solution during optimization. In this way, the problem is transferred into an unconstrained one by transferring any arbitrary solution into a nearby feasible one. All we will need for constructing this mapping is the set of support vectors together with the associated weights that make up the black-box model.

The rest of the paper is organized as follows: We start with a discussion of related approaches and the background of the optimization problem that is considered throughout this paper. Then, we define the mapping function that is used within our greedy algorithm for scheduling. We conclude with results from several simulation runs.

II. Related work and problem background

Within the framework of today's (centralized) operation planning for power stations, different heuristics are already harnessed. Examples from the research sector are for instance shown in [4] or in [5]. This task of (short-term) scheduling of different generators is also known as unit commitment problem and assigns (in its classical interpretation) discrete-time-varying production levels to generators for a given planning horizon [6]. It is known to be an NP-hard problem [7]. Determining an exact global optimum is, in any case, not possible until ex post due to uncertainties and forecast errors. In practice the software package BoFIT is often used, harnessing a mixed integer model with operational constraints as an integral part of the implementation of the model [8]. This fact makes it hard to exchange operational constraints in case of a changed setting (e.g. a new composition of energy resources) of the whole generation system.

Coordinating a pool of distributed generators and consumers with the intend to provide a certain aggregated load schedule for active power has some objective similarities to controlling a virtual power plant (VPP). Within the smart grid domain the volatile character of such group has additionally be taken into account. On an abstract level, approaches to control groups of distributed devices can be roughly divided into centralized and distributed scheduling algorithms.

Centralized approaches have long time dominated the discussion [9], not least because a generator may achieve slightly greater benefit if optimization is done from a global, omniscient perspective [10]. Centralized methods are discussed in the context of static pools of DER with drawbacks and restrictions regarding scalability and particularly flexibility.

Recently, distributed approaches gained more and more importance. Different works proposed hierarchical and decentralized architectures based on multi-agent systems and market based computing [11], [12]. Newer approaches try to establish self-organization between actors within the grid [13]–[15].

Especially for optimization approaches in smart grid scenarios, black-box models for encoding the feasible region with the set of operable schedules have been developed [1]. Encoding of a schedule's individual cost may also be easily embedded into the model [16].

The relatively new support vector approach uses support vector meta-models for black-box optimization scenarios with no explicitly given constraint boundaries. In general, various classification or regression methods could be harnessed for creating such models for the boundary [17]. There are two main reasons for using such an approach:

1) Substituting computational costs for evaluating the constraints by a comparatively easy check through the model.
2) Efficient communication in distributed environments due to the small footprint of the model.

Besides, the smart grid domain serves also as example for scenarios with (at least partly) unknown functional relationships of the constraints. The feasible region can sometimes only be derived with lacking full knowledge on hidden variables or intrinsic relations that determine the operability of a electric device and therewith the feasible region. The authors therefore have their model learned by a support vector data description approach from a set of operable (feasible) examples.

In a related approach, [18] used a two-class SVM for learning operation point and bias of a line in a power grid for easier determining an optimal way back to stable grid conditions in case of a failure.

In this paper, we will consider the following optimization problem for a given group (consumers, producers and/ or prosumers) of DER: A schedule for a given future time horizon is requested (e.g. via an electricity market mechanism) and is supposed to be jointly operated by the group. Thus, a partition of the requested target schedule has to be determined in order to fairly distribute the load among all participants. For the sake of simplicity, we will consider optimality as a close as possible adaption of the aggregated (sum of individual loads) schedule to the requested one. Optimality usually refers to additional local (individual cost) as well as to global (e.g. environmental impact) objectives. When determining an optimal partition of the schedule for load distribution, exactly one alternative schedule is taken from each DER's search space of individual operable schedules in order to assemble the desired aggregate schedule.

Therefore, the optimization problem is to find any combination of schedules (one from each DER with $\mathcal{X}$ as the set of possible choices) that resembles the target schedules $\ell_{\mathcal{T}}$ as close as possible, i.e. minimize the Euclidean distance ($\|\cdot\|$) between aggregated and target schedule:

$$\|\sum_i x_i - \ell_{\mathcal{T}}\| \to \min, \tag{1}$$

such that

$$x_i \in \mathcal{X}_i.$$

The following section will explain our approach for solving this optimization problem with individual acting DER in a distributed approach. At this, a schedule for d time intervals will be geometrically interpreted as a point in $\mathbb{R}^d$.

III. Algorithm

A. The feasible region

Each DER has to serve the purpose it has been built for and this purpose may usually be achieved in different alternative ways. For example, it is the main purpose of a μCHP (combined heat and power generator) to deliver enough heat for varying heat demands in a household at every moment in time. Nevertheless, heat usage is usually decoupled from heat production by use of a thermal buffer store. Thus, different production profiles may be used for generating the heat. This leads, in turn, to different respective electric load profiles that may be offered as alternatives to a scheduling controller.

Each DER offers a set of operable schedules for a given (future) time horizon. We see a schedule as a data vector $x \in$

$\mathbb{R}^d$, with the number of periods d. For each period the i-th element of x describes the respective amount of electric energy produced or consumed in this period or respectively the mean active power output or input for this period.

An operable schedule in this context means that such a schedule might be operated by the DER without violating any technical constraint. Moreover, we consider additional non-technical constraints that may restrict the possible operations of a DER. Constraints can be distinguished into hard (usually technically rooted) and soft constraints (often economically or ecologically rooted or subject to personal preferences).

Examples for hard constraints are:

- Minimum and/or maximum power input/output
- Integrated amount of energy produced over the given time frame
- Restrictions on thermal buffer storage
- Achieve intended purpose

Examples for soft constraints are:

- Costs (e.g. fuel costs) for operating a certain schedule
- Environmental performance
- Personal preferences (e.g. noise pollution in the evening)

These constraints can be interpreted geometrically. Without any constraint, the whole hypercube $[0,1]^d$ (active power between 0 and 100%) would be the region of feasible schedule. With each constraint, a different part (region) of the hypercube falls of the feasible region, because the respective schedules are not operable due to the constraint. Only the finally remaining region (hypercube minus superposition of all regions prohibited by constraints) is the feasible region of the DER. Only from this region, schedules might be taken during optimization.

It has been shown in [1] that the feasible region of operable schedules of a DER is not necessarily a convex polytope nor a single connected region. For this reason, concavity and clusters have to be taken into account, too. Such considerations have led to black-box models based on machine learning approaches that may

- capture the topological traits of the feasible region as a compact description of the set of all operable schedules.
- be easily communicated as a standardized description within distributed optimization scenarios.
- ease the calculation of the feasibility of a solution during optimization.

Before we discuss a new way of using this model during optimization, we will briefly discuss the basic idea of the model approach.

B. Support vector black-box model for constraints

We will describe the black-box model for the set of feasible schedules for a DER as it has been developed in [1] based on a one-class support vector data description (SVDD). The goal of building such a model is to learn the feasible region of the schedules of a DER by harnessing SVDD to learn the enclosing boundary around the set of operable schedules. This task is achieved by determining a mapping $\Phi : \mathcal{X} \subset \mathbb{R}^d \to \mathcal{H}, x \mapsto \Phi(x)$ such that all data from a given region $\mathcal{X}$ is mapped to a minimal hypersphere in some high- or indefinite-dimensional space $\mathcal{H}$. Originally, this model is used as a classifier that allows for distinguishing operable and not operable schedules during optimization without explicit knowledge about the constraints that restrict the operations of the DER.

The minimal sphere with radius R and center a in $\mathcal{H}$ that encloses $\{\Phi(x_i)\}_N$ can be derived from

$$\|\Phi(x_i) - a\|^2 \leq R^2 + \xi_i \quad \forall i \tag{2}$$

with $\|\cdot\|$ denoting the Euclidean norm and incorporating slack variables $\xi_i \geq 0$ that introduce soft constraints for sphere determination.

After introducing Lagrangian multipliers and further relaxing to the Wolfe dual form, the well known Mercer's theorem [19] may be harnessed for calculating dot products in $\mathcal{H}$ by means of a Mercer kernel in data space: $\Phi(x_i) \cdot \Phi(x_j) = k(x_i, x_j)$. In order to gain a more smooth adaption, it is known [20] to be advantageous to use a Gaussian kernel: $k_{\mathcal{G}}(x_i, x_j) = e^{-\frac{1}{2\sigma^2}\|x_i - x_j\|^2}$ instead of for instance polynomial kernels.

Putting it all together, the equation that has to be maximized in order to determine the desired sphere is:

$$W(\beta) = \sum_i k(x_i, x_i)\beta_i - \sum_{i,j} \beta_i \beta_j k(x_i, x_j). \tag{3}$$

Maximizing (3) is a problem of quadratic programming (QP) [21], which is known to be of cubic computational complexity $\mathcal{O}(N^3)$ with sample size N [22]. For this reason, the adoption of a technique called sequential minimal optimization [23] (SMO) is used for solving Eq. 3. SMO breaks up the large QP problem for SVM training into a series of smallest possible subproblems which can be solved analytically. In future, if real-time constraints might be involved, SVM training may be done incrementally with an online learning algorithm [24]. In this way, working on the data points one by one, it becomes possible to break the process with the so far reached result if a deadline for answering is approaching.

The result (weight vector β) represents the center a of the spere in terms of an expansion into $\mathcal{H}$:

$$a = \sum_i \beta_i \Phi(x_i). \tag{4}$$

The distance R of the image of an arbitrary point $x \in \mathbb{R}^d$ from $a \in \mathcal{H}$ can be calculated in $\mathbb{R}^d$ by:

$$R^2(x) = \ 1 - 2\sum_i \beta_i k_{\mathcal{G}}(x_i, x) + \sum_{i,j} \beta_i \beta_j k_{\mathcal{G}}(x_i, x_j). \tag{5}$$

Finally, the radius $R_{\mathbb{S}}$ of the sphere $\mathbb{S}$ is determined by the distance to a of an arbitrary support vector as these are mapped right onto the surface. Thus the original feasible region is now modeled as

$$\{x \in \mathbb{R}^d | R(x) \leq R_{\mathbb{S}}\}. \tag{6}$$

The model that has to be communicated consists of the set s of support vectors and respective weights from β as this is all

that is needed for reconstructing the boundary that encloses the feasible region. From β only the non zero components for the support vectors are necessary. We denote this reduced weight vector with w.

We will now use this model in a different way. We are interested in having a means of finding a nearby feasible schedule next to an arbitrary given schedule. For this purpose, we will harness a function that maps the d-dimensional unit hypercube (representing arbitrary schedules in a scaled scenario) onto the feasible region. In this way, any (in-)feasible schedule will be converted into a feasible one. The construction of this mapping is described in the next section.

C. Constructing solutions from the model

For our use case, we need a procedure that generates a nearby and feasible solution from any given (likely not feasible) schedule. Near in this context means that the distance in solution space between given and near feasible solution is small. This task can be achieved by constructing a mapping that maps every infeasible point from input space into or onto the feasible region. We have tested both approaches. Here, we describe the more general case of mapping into the feasible region that includes the specialized case. In general, this mapping can also be used for transforming the whole optimization problem into an unconstrained one.

Let $\mathcal{F}$ denote the feasible region within the domain of some given optimization problem bounded by an associated set of constraints. It is known, that pre-processing the data by scaling it to $[0,1]^d$ leads to better adaption [25]. Considering optimization problems in the energy sector, rescaling of the domain to $[0,1]^d$ leads to some advantages [3]. For this reason, we here consider scaled domains, too, and denote with $\mathcal{F}_{[0,1]}$ the likewise scaled region of feasible solutions. We want to construct a mapping

$$\begin{aligned} \gamma : [0,1]^d &\to \mathcal{F}_{[0,1]} \subseteq [0,1]^d \\ x &\mapsto \gamma(x) \end{aligned} \tag{7}$$

that is able to map the unit hypercube $[0,1]^d$ onto the d-dimensional region of feasible solutions that is bounded by a set of arbitrary (maybe nonlinear) constraints. We achieve this mapping as a composition of three functions:

$$\gamma = \Phi_\ell^{-1} \circ \Gamma_a \circ \hat{\Phi}_\ell. \tag{8}$$

We will define these three functions step by step. Instead of directly mapping to $\mathcal{F}_{[0,1]}$ we will go through the kernel space. We start with an arbitrary point $x \in [0,1]^d$ from the unconstrained d-dimensional hypercube and map it to an ℓ-dimensional manifold in kernel space that is spanned by the images of the support vectors $s_1 \dots s_\ell$. After drawing this mapped point towards the sphere in order to pull it into the image of the feasible region, we look for the pre-image of the moved image to get a point from $\mathcal{F}_{[0,1]}$. We will now look at each step in more detail.

1) Mapping x to the support vector induced subspace $\mathcal{H}^{(\ell)}$ with an empirical kernel map: Let

$$\begin{aligned} \Phi_\ell : \mathbb{R}^d &\to \mathbb{R}^\ell, \\ x &\mapsto k(.,x)|_{\{s1,\dots,s_\ell\}} \\ &= (k(s_1,x),\dots,k(s_\ell,x)) \end{aligned} \tag{9}$$

be the empirical kernel map w.r.t. the set of support vectors $\{s_1,\dots,s_\ell\}$. If Φ_ℓ is modified to

$$\hat{\Phi}_\ell : x \mapsto K^{-\frac{1}{2}}(k(s_1,x),\dots,k(s_\ell,x)) \tag{10}$$

with $K_{ij} = k(s_i,s_j)$ being the kernel Gram Matrix, then function Eq. 10 maps points x, y from input space to $\mathbb{R}^\ell$, such that $k(x,y) = \hat{\Phi}_\ell(x) \cdot \hat{\Phi}_\ell(y)$ (cf. [19]).

With $\hat{\Phi}_\ell$ we are now able to map arbitrary points from $[0,1]^d$ to some ℓ-dimensional space $\mathcal{H}^{(\ell)}$ that contains a projection of the sphere. Again, points from $\mathcal{F}_{[0,1]}$ are mapped into or onto the projected sphere, outside points go outside the sphere and must be moved in $\mathcal{H}^{(\ell)}$ towards the center in the next step in order to draw them into the image of the feasible region.

2) Re-adjustment in kernel space: In general, in kernel space $\mathcal{H}$ the image of the region is represented as a hypersphere $\mathbb{S}$ with center a (Eq. 4) and radius $R_\mathbb{S}$. Points outside this hypersphere are not images of points from $\mathcal{X}$, i.e. in our case, points from $\mathcal{F}_{[0,1]}$ are mapped (by Φ) into the sphere or onto its surface (support vectors), points from outside $\mathcal{F}_{[0,1]}$ are mapped outside the sphere. Actually, using a Gaussian kernel, Φ maps each point into a n-dimensional manifold (with sample size n) embedded into infinite dimensional $\mathcal{H}$. In principle, the same holds true for a lower dimensional embedding spanned by ℓ mapped support vectors and the ℓ-dimensional projection of the hypersphere therein.

We now want to pull points from outside the feasible region into that region. As we do have rather a description of the image of the region, we draw images of outside points into the image of the region, i.e. into the hypersphere; precisely into its ℓ-dimensional projection. For this purpose we use

$$\Gamma_a(\hat{\Psi}_x) = \tilde{\Psi}_x = a + \frac{(\hat{\Psi}_x - a) \cdot R_\mathbb{S}}{R_x \cdot \mu} \tag{11}$$

to transform the image $\hat{\Psi}_x$ produced in step 1) into $\tilde{\Psi}_x$ by drawing $\hat{\Psi}_x$ into the sphere. In this way, each image is re-adjusted proportional to the original distance from the sphere and drawn into the direction of the center. For the distributed optimization procedure described in this paper, we set $\mu = 1$ to draw points right onto the surface; which is obviously the shortest way to feasibility. After this procedure we have $\tilde{\Psi}_x$ which is the image of a point from $\mathcal{F}_{[0,1]}$ in terms of a modified weight vector $\tilde{w}^{\Gamma_a}$.

3) Finding an approximate pre-image: As a last step, we will have to find the pre-image of $\tilde{\Psi}_x$ in order to finally get the wanted mapping. A major problem in determining the pre-image of a point from kernel space is that not every point from the span of Φ is the image of a mapped data point [19]. As we use a Gaussian kernel, none of our points from kernel space can be related to an exact pre-image except for trivial

expansions with only one term [26]. For this reason, we will look for an approximate pre-image whose image lies closest to the given image using an iterative procedure after [27]. In our case (Gaussian kernel), we iterate x^* to find the point closest to the pre-image and define approximation $\tilde{\Phi_\ell^{-1}}$ by equation

$$x^*_{n+1} = \frac{\sum_{i=1}^{\ell}(\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x^*_n\|^2/2\sigma^2} s_i)}{\sum_{i=1}^{\ell}(\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x^*_n\|^2/2\sigma^2})}. \quad (12)$$

As an initial guess for x^* we take the original point x and iterate it towards $\mathcal{F}_{[0,1]}$. As this procedure is sensitive to the choice of the starting point, it is important to have x as a fixed starting point in order to ensure determinism of the algorithm. Empirically, x has showed up to be a useful guess.

Finally, we have achieved our goal to map an arbitrary point from $[0,1]^d$ into the region of feasible solutions described merely by a given set of support vectors and associated weights: x^*_n is the sought after image under mapping γ of x that lies in $\mathcal{F}_{[0,1]}$.

This model and mapping may be used in different ways during optimization. Among them are:

- Repair of infeasible solutions.
- Transformation of an constrained to an unconstrained optimization problem by mapping the whole search space into the feasible region.
- Computational easy classification of an solution's feasibility.
- Compact communication of large sets of operable schedules.

Here, we will harness the capability of repairing infeasible solutions for a distributed optimization approach.

D. The distributed greedy algorithm

With the above sketched preliminaries, we are now able to define our optimization algorithm. In order to pay attention to the ongoing decentralization of electricity grid control, it seems way more promising to design the optimization process distributed, too. In addition, the chances for success in finding an exact solution are rather low due to problem size, what makes a heuristic most suitable.

In this sense, we propose the following greedy algorithm for approximately solving optimization problem Eq. 1. In our scenario, we assume one type of agent: the control agent of a single energy resource with the following responsibilities/ capabilities:

- Simulating the underlying physical device in order to determine operable example schedules.
- Calculating support vector based black-box modelling.
- Determining the schedule for one's own physical device that minimizes the overall loss.
- Participation in optimization.

The procedure for optimizing the aggregated schedule is now the one depicted in Fig. 1. Within a group of agents $\mathcal{A}$, one agent is randomly chosen to start the procedure. Here, we assume an agent to be in charge of controlling a DER and to participate in the distributed procedure of determining

$\mathcal{A} \leftarrow$ List of all agents
if is initiator **then**
  $S \leftarrow zeros(n, d)$
else
  $S \leftarrow$aggregated schedule
  $S_{new} \leftarrow \gamma(T - (S - S_a))$
  $S \leftarrow S - S_a + S_{new}$
  if no stop criterion met **then**
    choose random agent $A \in \mathcal{A}$
    send message with S to A
  else
    publish solution S
  end if
end if

Fig. 1. Greedy algorithm that each agent repeatedly executes for successive solution improvement starting from a zero solution S with S denoting the aggregated overall solution and S_a denoting the individual current contribution of the agent.

schedules for each DER such that the aggregated schedule best fits a given objective schedule. This initiator initializes the solution with all values to zero. Then, solution improvement begins. The agent adds up all schedules (known from the solution object) from all other agents. This is equivalent to subtracting one's own schedule from the aggregated solution. In a next step, the difference δ of this sum to the desired target schedule is determined. This difference δ represents the optimal schedule for the current agent in the following sense: if he would be able to deliver this schedule, the target could be reached exactly. Therefore, the agent now determines the nearest schedule to δ that is actually operable by the device. This nearest schedule can be easily calculated with the help of the mapping γ that has been described in the previous section. Function γ maps an arbitrary schedule (in our case difference schedule δ) into the region of feasible schedules and delivers the respective operable schedule that is nearest to δ, because it uses the shortest trace to the feasible region to move a point.

In this way, each DER chooses a schedule that is a compromise of being feasible (automatically ensured by mapping γ) and doing one's own best in bringing forth the overall solution towards the wanted adaption to the target schedule as much as possible each time when it is the respective agents turn.

As a stop criterion, we chose a maximum number of iterations at which the term iteration refers to one execution of the procedure in Fig. 1 by one agent.

By one after another, the overall solution (the aggregated schedule) is successively improved. We have chosen to activate the agents in a random order, but a round robin approach may also do if each agent knows about his successor. In this way, the algorithm is distributed and sequential as only one agent has the token to work at a time. If the objective is to adapt to a given target schedule, the only information that has to be passed around (or made globally available) is the aggregated overall solution (as sum of all local solutions) and the desired target schedule. This is sufficient as each agent may remember

his own local schedule that has been determined previously. All other information can be determined by local information.

Clearly, the actual optimization is distributed but sequential. But, the most time consuming part – namely the computation of mapping γ – can be done in advance and fully parallel, what in turn allows for faster optimization afterwards without a need for considering constraints anymore.

IV. Simulation results

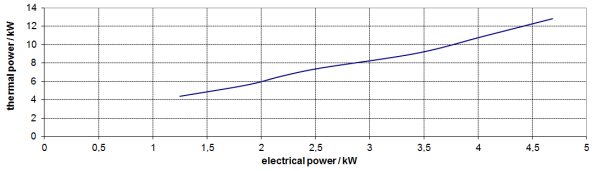


Fig. 2. Relationship between electrical and thermal power for an EcoPower CHP; modified after [28].

So far, we have tested our approach on several simulated energy resources. Among them are: co-generation devices with thermal buffer store and a simulated residential thermal energy demand as well as simulated controllable cooling devices. We will here focus on results from CHP. All simulations have been done with power scaled to $[0, 1]$. All simulations incorporating a μCHP also encompass the simulation of the respective household that is heated by this μCHP. This implies a simulation of the respective heat demand, heat use, different weather conditions or heat losses by thermal diffusion processes.

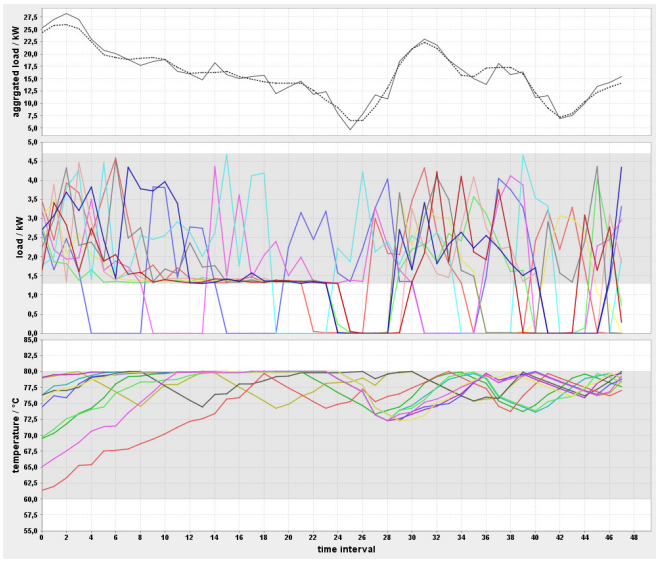


Fig. 3. Optimization result for a winter day scenario with 10 chp (EcoPower with randomly initialized storage charging) for a time horizon of 48 15-minute intervals.

For our simulations, we used the model of a modulating CHP-plant with the following specification:

- Minimum electrical power: 1.3 kW,

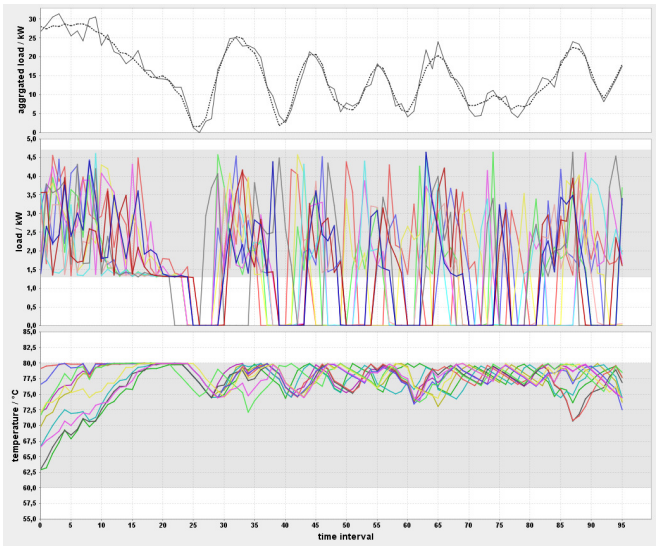


Fig. 4. Optimization result for a spring day scenario with 10 chp (EcoPower with randomly initialized storage charging) for a time horizon of a whole day in 15-minute intervals.

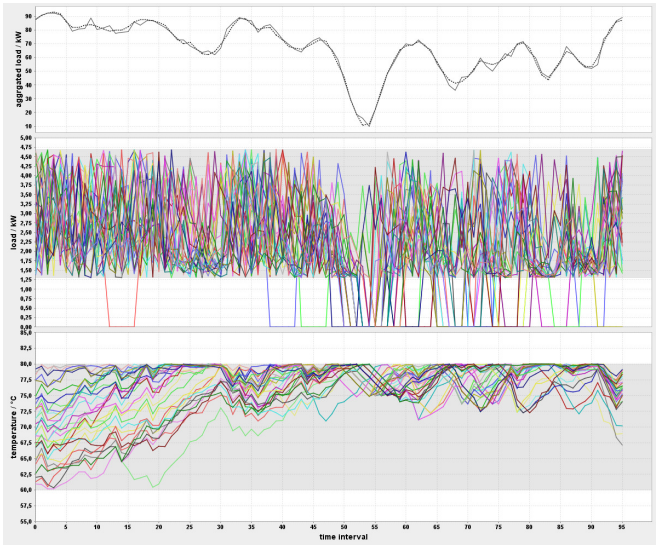
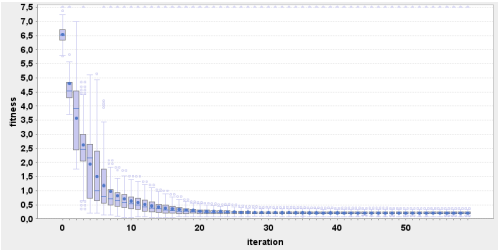


Fig. 5. Optimization result for a scenario with 30 chp for 96 15-minute intervals. This amounts to a 2880-dimensional search space.

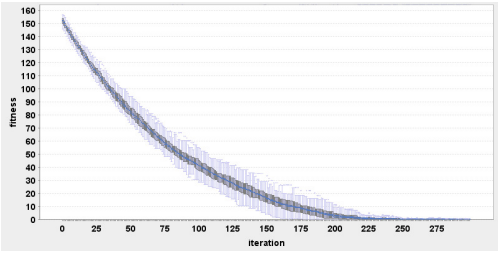
- Maximum electrical power: 4.7 kW,
- Minimum thermal power: 4 kW,
- Maximum thermal power: 12.5 kW,
- After shut down, a device has to stay off for at least 2 h.

The relationship between electrical (active) power and thermal power was modeled after Fig. 2. In order to gain more degrees of freedom for varying active power, each CHP is equipped with an 800 ℓ thermal buffer store. Thermal energy consumption is simulated by a model of a detached house with its several heat losses (heater is supposed to keep the indoor temperature on a constant level) and randomized warm water drawing for gaining more diversity among the devices.

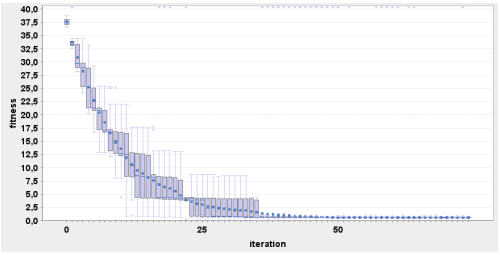
For each simulated household, we implemented an agent



(a)



(b)



(c)

Fig. 6. Convergence for different scenarios: 6(a): 5 CHP and, 8 periods; 6(b): 100 CHP, 8 periods, 6(c): 10 CHP, 96 periods.

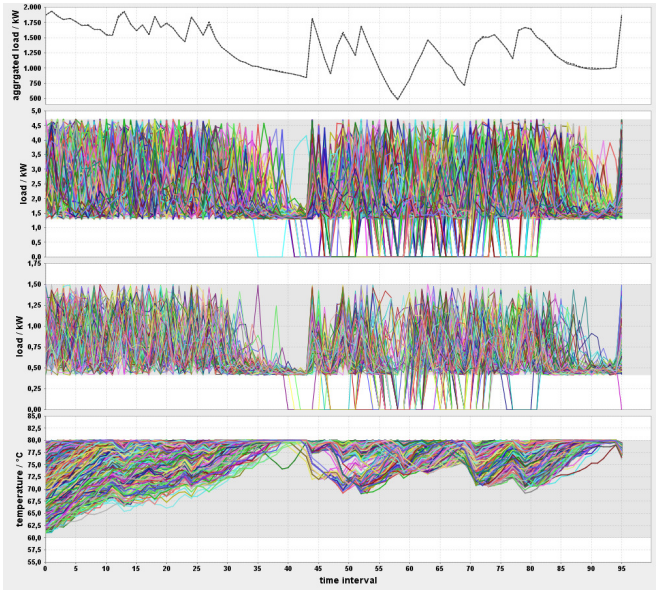


Fig. 7. Optimization result for a scenario with 750 chp for 96 15-minute intervals. This amounts to a 72000-dimensional search space.

TABLE I
CPU TIME FOR ALGORITHM AND SIMULATION REGARDING DIFFERENT PROBLEM SIZES. QUALITY AS MEAN EUCLIDEAN DISTANCE IN kW FOR n_A AGENTS, k ITERATIONS AND SCHEDULES OF d INTERVALS OF 15 MIN.

d	n_A	k	t_{sim} / s	t_{opt} / s	QUALITY
8	10	75	4.71 $\pm$ 0.23	0.006$\pm$0.008	0.054$\pm$0.023
8	100	750	45.2 $\pm$ 0.74	0.061$\pm$0.009	0.045$\pm$0.02
32	100	250	382.59 $\pm$ 27.24	0.049$\pm$0.008	1.05$\pm$1.09
96	10	750	251.4 $\pm$ 4.5	0.498$\pm$0.127	0.049$\pm$0.08

capable of simulating the CHP (and surroundings and auxiliary devices) on a meso-scale level with energy flows among different model parts but no technical details. All simulations have so far been done with a time resolution of 15 minutes for different forecast horizons. We have run several test series with each CHP randomly initialized with different buffer charging levels, temperatures and water drawing profiles.

Fig. 3 shows a result (solid line in the top chart) for a group of 10 CHP that try to reach a given objective schedule (dashed line). The resulting schedules for each single CHP are depicted in the middle chart with the allowed active power band for modulation highlighted in grey. The bottom chart shows the temperatures in the thermal buffer store resulting from operating the respective electrical schedules; again with the allowed range highlighted in grey.

The desired objective schedule has been randomly chosen. These schedules have been generated in a way that they are of a reasonable magnitude order according to the capabilities of the optimized CHP, but without any guarantee that a perfect adaption might be achievable.

Fig. 4 shows a similar simulation run, but for a time horizon of a whole day. Fig. 5 shows the result for optimizing a larger bunch of 30 CHP. As might have been expected, the result schedule gets closer to the target schedule if more CHP are involved. This is mainly due to the availability of more degrees of freedom for the system as a whole.

As a next step, we scrutinized the speed of convergence and convergence behaviour of our algorithm. Fig. 6 shows the result of some measuring series. It is noticeable that the fitness (the difference between aggregated and target solution) strictly decreases notwithstanding the uncoordinated, heuristic character of the approach. The number of necessary iterations is acceptably small, what can also be seen in Table I, where some mean CPU time results (Java implementation on Core 2, 3 Ghz) for different scenarios are listed. The simulation time t_{sim} reflects the time that is necessary for the whole simulation including the preceding calculations of the set of feasible schedules for each agent, for the calculation of all support vector models and all mapping functions γ on a single machine. These calculations would in a distributed productive system be done in parallel. The time necessary for the mere optimization is comparably small. In order to be able to simulate larger scenarios, we are currently thinking of distributing the simulation, too.

Finally, Fig 7 shows a result from a larger mixed scenario with two different types (in power magnitude) of CHP. Having

different DER in a scenario, often leads to a better adaption to the target schedule as has also been seen in similar scenarios with a mixture of CHP and refrigerators e.g. in [3].

V. Conclusion

We have presented a new approach for distributed optimization and control of distributed energy resources for smart grid scenarios with a large number of controllable entities. This approach is based on two new methodologies:

- A strategy for handling constraints in distributed optimization scenarios that may also be used for finding nearby feasible solutions by harnessing a learned model of the feasible region.
- A well scaling greedy algorithm for harnessing that strategy during the search for an optimal partition of the requested schedule for different DER.

We have demonstrated that the greedy heuristics scales well with the number of participating devices because the most expensive calculations may be done in parallel in advance by each controllable device. We are now starting the development of a distributed simulation environment as systematic testbed for further algorithms that integrate the sketched mapping procedure for further optimization solutions and use cases from the smart grid domain.

Acknowledgment

The Lower Saxony research network 'Smart Nord' acknowledges the support of the Lower Saxony Ministry of Science and Culture through the Niederschsisches Vorab grant programme (grant ZN 2764).

References

[1] J. Bremer, B. Rapp, and M. Sonnenschein, "Support vector based encoding of distributed energy resources feasible load spaces," in *IEEE PES Conference on Innovative Smart Grid Technologies Europe*, Chalmers Lindholmen, Gothenburg, Sweden, 2010.

[2] D. M. J. Tax and R. P. W. Duin, "Data domain description using support vectors." in *ESANN*, 1999, pp. 251–256.

[3] J. Bremer, B. Rapp, and M. Sonnenschein, "Encoding distributed search spaces for virtual power plants," in *IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011)*, Paris, France, 4 2011.

[4] Y. Mao and M. Li, "Optimal reactive power planning based on simulated annealing particle swarm algorithm considering static voltage stability," in *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation - Volume 01*, ser. ICICTA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 106–110. [Online]. Available: http://dx.doi.org/10.1109/ICICTA.2008.427

[5] W. Xiong, M.-j. Li, and Y.-l. Cheng, "An improved particle swarm optimization algorithm for unit commitment," in *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation - Volume 01*, ser. ICICTA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 21–25. [Online]. Available: http://dx.doi.org/10.1109/ICICTA.2008.363

[6] J. Pereira, A. Viana, B. Lucus, and M. Matos, "A meta-heuristic approach to the unit commitment problem under network constraints," *International Journal of Energy Sector Management*, vol. 2, no. 3, pp. 449–467, 2008.

[7] X. Guan, Q. Zhai, and A. Papalexopoulos, "Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming," vol. 2, 2003, p. 1100 Vol. 2.

[8] T. Franch, M. Scheidt, and G. Stock, "Current and future challenges for production planning systems," in *Optimization in the Energy Industry*, ser. Energy Systems, J. Kallrath, P. M. Pardalos, S. Rebennack, M. Scheidt, and P. M. Pardalos, Eds. Springer Berlin Heidelberg, 2009, pp. 5–17.

[9] M. Tröschel and H.-J. Appelrath, "Towards reactive scheduling for large-scale virtual power plants." in *MATES*, ser. Lecture Notes in Computer Science, L. Braubach, W. van der Hoek, P. Petta, and A. Pokahr, Eds., vol. 5774. Springer, 2009, pp. 141–152.

[10] W. Gatterbauer, "Economic efficiency of decentralized unit commitment from a generator's perspective," in *Engineering Electricity Services of the Future*, M. Ilic, Ed. Springer, 2010.

[11] R. Kamphuis, C. Warmer, M. Hommelberg, and K. Kok, "Massive coordination of dispersed generation using powermatcher based software agents," May 2007.

[12] K. Kok, Z. Derzsi, J. Gordijn, M. Hommelberg, C. Warmer, R. Kamphuis, and H. Akkermans, "Agent-based electricity balancing with distributed energy resources, a multiperspective case study," *Hawaii International Conference on System Sciences*, vol. 0, p. 173, 2008.

[13] A. Kamper and A. Esser, "Strategies for decentralised balancing power," in *Biologically-inspired Optimisation Methods: Parallel Algorithms, Systems and Applications*, ser. Studies in Computational Intelligence, M. R. A. Lewis, S. Mostaghim, Ed. Berlin, Heidelberg: Springer, Juni 2009, no. 210, pp. 261–289. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01262-4

[14] R.-C. Mihailescu, M. Vasirani, and S. Ossowski, "Dynamic coalition adaptation for efficient agent-based virtual power plants," in *Proceedings of the 9th German conference on Multiagent system technologies*, ser. MATES'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 101–112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2050592.2050603

[15] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings, "Agent-based control for decentralised demand side management in the smart grid," in *AAMAS*, 2011, pp. 5–12.

[16] J. Bremer, B. Rapp, and M. Sonnenschein, "Including Environmental Performance Indicators into Kernel based Search Space Representations," *Information Technologies in Environmental Engineering (ITEE 2011)*, 2011.

[17] O. Kramer, "A review of constraint-handling techniques for evolution strategies," *Appl. Comp. Intell. Soft Comput.*, vol. 2010, pp. 3:1–3:19, January 2010.

[18] M. Blank, S. Gerwinn, O. Krause, and S. Lehnhoff, "Support vector machines for an efficient representation of voltage band constraints," in *Innovative Smart Grid Technologies*. IEEE PES, 2011.

[19] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola, "Input space vs. feature space in kernel-based methods," *IEEE Transactions on Neural Networks*, vol. 10(5), pp. 1000–1017, 1999.

[20] A. Ben-Hur, H. T. Siegelmann, D. Horn, and V. Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[21] A. Tavakkoli, M. Nicolescu, M. Nicolescu, and G. Bebis, "Incremental svdd training: Improving efficiency of background modeling in videos," in *Signal and Image Processing*, P. Cristea, Ed. Calgary, Canada: Acta Press, 2008.

[22] C. S. Chu, I. W. Tsang, and J. T. Kwok, "Scaling up support vector data description by using core-sets," in *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, vol. 1, 2004, pp. 430–435.

[23] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*. MIT press, 1999, pp. 185–208.

[24] P. Laskov, C. Gehl, S. Krüger, and K. Müller, "Incremental support vector learning: Analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, pp. 1906–1936, 2006.

[25] P. Juszczak, D. Tax, and R. P. W. Duin, "Feature scaling in support vector data description," in *Proc. ASCI 2002, 8th Annual Conf. of the Advanced School for Computing and Imaging*, E. Deprettere, A. Belloum, J. Heijnsdijk, and F. van der Stappen, Eds., 2002, pp. 95–102.

[26] J. Kwok and I. Tsang, "The pre-image problem in kernel methods," *Neural Networks, IEEE Transactions on*, vol. 15, no. 6, pp. 1517–1525, 2004.

[27] S. Mika, B. Schölkopf, A. Smola, K. R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Proceedings of the 1998 conference on Advances in neural information processing systems II*. Cambridge, MA, USA: MIT Press, 1999, pp. 536–542.

[28] B. Thomas, *Mini-Blockheizkraftwerke: Grundlagen, Gerätetechnik, Betriebsdaten*. Vogel Buchverlag, 2007.