Proceedings of the Federated Conference on Computer Science and Information Systems pp. 551–559
DOI: 10.15439/2016F19
ACSIS, Vol. 8. ISSN 2300-5963

# Partitioning the Data Domain of Combinatorial Problems for Sequential Optimization

Christian Hinrichs, Jörg Bremer, Sönke Martens, Michael Sonnenschein
Department of Computing Science
University of Oldenburg
26129 Oldenburg, Germany
<first name>.<last name>@uni-oldenburg.de

***Abstract*—Following the long-term goal of substituting conventional power generation with cleaner energy will lead to an integration of a large share of small energy generation units imposing large problem sizes for coordination. The expected huge number of entities leads to a need for new techniques reducing the computational effort for coordination. Predictive scheduling is a frequent task in energy grid control. For a number of energy resources, schedules have to be found that fulfill several objectives at the same time. Considering day-ahead scenarios with 96-dimensional schedules imposes additional challenges to this already hard combinatorial problem. We explore the effects of reducing complexity by partitioning the data domain of the optimization problem for a sequential approach that integrates energy models for constraint handling directly into the optimization process. We explore the effects of different partitioning schemes and evaluate the trade-off between accuracy and effort with several simulation studies.**

## I. Introduction

DESPITE being environmentally friendly and sustainable, the increasing amount of renewable electricity generation has a major drawback. In contrast to conventional power plants, the generation from e. g. solar and wind power can neither be predicted with high accuracy nor scheduled precisely. Furthermore, as storage of electrical energy is a rather difficult and expensive task, balancing supply and demand in the grid in real-time is one of the most important functions of power system control centers. Thus, to incorporate renewables accordingly, methods have to be established that can compensate for the missing flexibility of those energy sources. For instance, controlling flexible loads to use electrical power in times of high availability (i. e. high wind or solar radiation) can help using renewable power more efficiently [1].

From an algorithmic perspective, the task of scheduling energy units can be seen as combinatorial optimization problem: For each unit (i. e. controllable loads and generators), an optimal schedule has to be found such that for every time interval of a predefined planning horizon, a specific amount of electrical power (positive or negative) is assigned. A combination of schedules is optimal if the aggregated power equals a target profile that is given by the use case. For example, given the inverse of a predicted feed-in time series for wind and photovoltaic power plants as target profile, an optimal schedule assignment for the controllable energy units would lead to a perfect balancing of supply and demand in the considered system at each interval of the prediction horizon. Another use case is the operation of a virtual power plant (VPP): Given a target power profile that is to be offered in an energy market, the members of the VPP must collaborate in such a way that the VPP as a whole will produce the target profile. From the outside perspective, no difference between a VPP and a classical power plant would be evident [1].

However, the schedule optimization task becomes hard to solve in the presence of device-specific restrictions. Many flexible generators and loads are controllable in principle, but at the same time have to obey specific individual constraints. For instance, a cogeneration plant (e. g. a combined heat and power plant, CHP) produces thermal and electrical power simultaneously. As the generation of those two forms of power are strictly coupled within the unit and the use of the heat is subject to further restrictions such as the size of an attached thermal buffer storage, the electrical generation is severly confined as well [2], [3]. Due to such constraints, many established optimization algorithms cannot be applied to this task. For instance, meta-heuristics like evolutionary algorithms or simulated annealing are not able to cope with constraints per se and would have to be tailored specifically for the actual use case and the involved energy units.

In [4], a method has been introduced that is able to transform a problem with restrictions into a restriction-free representation using a machine learning approach. This so-called *support vector decoder model* allows generic optimization algorithms to operate in a restriction-free representation of the constrained search space of the original optimization problem. The method has been successfully applied to the schedule optimization problem [3]. In this context, the influence of the length of the planning horizon on solution quality became apparent: Usually, the method is applied to representations of the planning horizon as a whole by interpreting feasible schedules of energy units as elements to the combinatorial problem. However, the longer the planning horizon (and the schedules, consequently), the lower the solution quality of the employed optimization algorithms. At first glance, this may seem like an inherent restriction of the problem to solve. But interestingly, preliminary experiments indicated a potentially increasing solution quality when the optimization algorithm is applied in a successive manner to sequential partitions of the planning horizon. Thus, the objective of this paper is to explore the potential benefit of partitioning the search space

978-83-60810-90-3/$25.00

of the given combinatorial problem in the data domain in combination with sequential optimization of the individual data partitions.

In Section II, the motivating optimization problem as well as the support vector decoder model are briefly recapped from previous works. Following, Section III first revisits relevant related work in the field of high-dimensionality optimization strategies before describing the introduced concept of data partitions for the considered combinatorial problem in more detail. Section IV then evaluates the approach by employing a simulation study in the aforementioned application domain. Finally, Section V concludes the paper.

## II. Methodical Background

We start with some preliminary definitions. First, let $\mathcal{U}$ be the set of DER units in the VPP and $Z_U$ be the set of operational states of unit $U$. We regard the schedule of an energy unit as a vector $\boldsymbol{p} = (p_1, \ldots, p_d) \in \mathbb{R}^d$ of mean power $p_i$ generated (or consumed) during the $i$th time interval. The starting time and the width of a time interval (today usually 15 minutes) are defined separately and have no effect on this representation. For the used support vector decoder it is advantageous to use schedules with scaled power values [5]. Scaling is done according to respective minimum ($p_{min}$) and maximum ($p_{max}$) nominal active power output (or input):

$$\begin{aligned} \rho : \mathbb{R}^d \to \mathcal{X} \subset [0,1]^d \\ \boldsymbol{p} \mapsto \boldsymbol{x} = \rho(\boldsymbol{p}), \text{with } x_i = \frac{p_i - p_{min}}{p_{max} - p_{min}}; \end{aligned} \tag{1}$$

For this paper we go with the example of predictive scheduling for active power planning in day-ahead scenarios (not necessarily 24 hours but for some given future period).

One of the crucial challenges in operating a VPP arises from the complexity of the scheduling task due to the large amount of (small) energy units in the distribution grid [6]. In the following, we consider predictive scheduling, where the goal is to select exactly one schedule $\boldsymbol{x}_i$ for each energy unit $U_i$ from a search space of feasible schedules with respect to a future planning horizon, such that a global objective function (e. g. a target power profile for the VPP) is optimized by the sum of individual contributions [7]. A basic formulation of the scheduling problem is given by

$$\delta \left( \sum_{i=1}^{m} \boldsymbol{x}, \boldsymbol{\zeta} \right) \to min \tag{2}$$

such that

$$\boldsymbol{x}_i \in \mathcal{F}^{(U_i)} \ \forall U_i \in \mathcal{U}. \tag{3}$$

In equation (2) $\delta$ denotes an (in general) arbitrary distance measure for evaluating the difference between the aggregated schedule of the group and the desired target schedule $\boldsymbol{\zeta}$. W.l.o.g., in this contribution we use the Euclidean distance $\| \cdot \|_2$. To each energy unit $U_i$ exactly one schedule $\boldsymbol{x}_i$ has to be assigned. The desired target schedule is given by $\boldsymbol{\zeta}$. $\mathcal{F}^{(U_i)}$ denotes the individual set of feasible schedules that are operable for unit $U_i$ without violating any (technical) constraint. Solving this problem without unit independent constraint handling leads to specific implementations that are not suitable for handling changes in VPP composition or unit setup without having changes in the implementation of the scheduling algorithm [8].

In [9] a so called support vector decoder has been introduced. Basically, a decoder is a constraint handling technique that gives an algorithm hints on where to look for feasible solutions. It imposes a relationship between a decoder solution and a feasible solution and gives instructions on how to construct a feasible solution [10]. For example, [11] proposed a homomorphous mapping between an $n$-dimensional hyper cube and the feasible region in order to transform the problem into an topological equivalent one that is easier to handle. In order to be able to derive such a decoder mapping automatically from any given energy unit model, [9] developed an approach based on a support vector model [5]. We will briefly describe this method.

The basic idea is to start with a set $\mathcal{X} = \{\boldsymbol{x}_i\}_n$ of feasible example schedules derived from the simulation model of an energy unit and use this sample as a stencil for the region (the sub-space in the space of all schedules) that contains only feasible schedules. The set $\mathcal{X}$ can be easily generated after a sampling method from [12]. The schedule sample is then used as a training set for a support vector based machine learning approach [13] that derives a geometrical description of the sub-space that contains the given data (in our case: the feasible schedules). Given a set of data samples, the inherent structure of the scope of action of a unit where the data resides in can be derived as follows: After mapping the data to a high dimensional feature space by means of an appropriate kernel, the smallest enclosing ball in this feature space is determined. When mapping back this ball to data space, it forms a set of contours (not necessarily connected) enclosing the given data sample. An in-depth discussion can be found e. g. in [13].

At this point, the set of alternatively feasible schedules of a unit is represented as pre-image of a high-dimensional ball $\mathcal{S}$. Figure 1 shows the situation. This representation has some advantageous properties. Although the pre-image might be some arbitrary shaped non-continuous blob in $\mathbb{R}^d$, the high-dimensional representation is still a ball and thus geometrically easier to handle (right hand side of figure 1). The relation is as follows: If a schedule is feasible, i.e. can be operated by the unit without violating any technical constraint, it lies inside the feasible region (grey area on the left hand side in figure 1). Thus, the schedule is inside the pre-image (that represents the feasible region) of the ball and thus its image in the high-dimensional representation lies inside the ball. An infeasible schedule (e. g. $\boldsymbol{x}$ in Fig. 1) lies outside the feasible region and thus its image $\hat{\Psi}_x$ lies outside the ball. But we know some relations: the center of the ball, the distance of the image from the center and the radius of the ball. Hence, we can move the image of an infeasible schedule along the difference vector towards the center until it touches the ball. Finally, we calculate the pre-image of the moved image $\tilde{\Psi}_x$ and get a schedule at the boundary of the feasible region: a

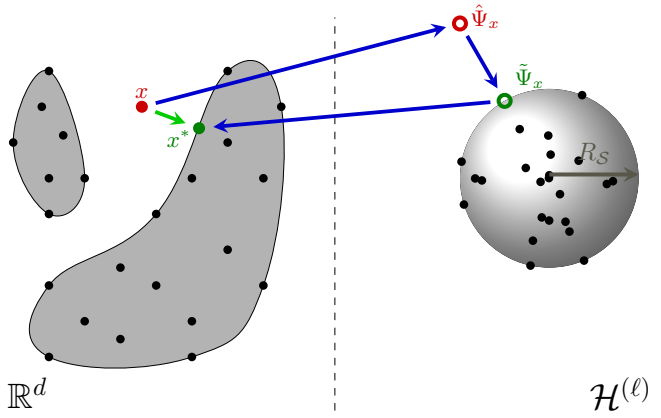


Fig. 1. General support vector model and decoder scheme for solution repair and constraint handling.

repaired schedule $\boldsymbol{x}^*$ that is now feasible. We do not need a mathematical description of the original feasible region or of the constraints to do this. The decoder that does the trick is derived directly from the training set $\mathcal{X}$ generated from the respective simulation model. More sophisticated variants of transformation are e. g. given in [4]. For a detailed description of the support vector decoder approach we refer to [4]. Formally, we have a mapping (the decoder $\gamma$)

$$\begin{aligned} \gamma : [0,1]^d &\to \mathcal{F}_{[0,1]} \subseteq [0,1]^d \\ \boldsymbol{x} &\mapsto \gamma(\boldsymbol{x}) \end{aligned} \tag{4}$$

that transforms any given (maybe in-feasible) schedule into a feasible one. Thus, we are able to transform the scheduling problem given Eq. (2) into an unconstrained formulation.

With these preliminaries in constraint handling we can now reformulate our optimization problem as

$$\delta\left(\sum_{i=1}^{m} \rho_i^{-1} \circ \gamma(\boldsymbol{x}_i), \boldsymbol{\zeta}\right) \to min, \tag{5}$$

where $\gamma_i$ is the decoder function of unit $i$ that produces feasible, scaled schedules from $\boldsymbol{x} \in [0,1]^d$ and $\sigma_i^{-1}$ scales them unit specific entrywise to correct active power values (inverse to Eq. (1)) resulting in schedules that are operable by that unit. Please note, that this is a constraint free formulation. With this problem formulation, many standard algorithms for optimization can be easily adapted as there are no constraints (apart from a simple box constraint $\boldsymbol{x} \in [0,1]^d$) to be handled and no domain specific implementation (regarding the energy units and their operation schedules) has to be integrated. Equation (5) is used as a surrogate objective to find the solution to the constrained optimization problem equation (2).

Using a decoder fairly eases the implementation of a solver because no complex constraints have to be considered. On the other hand, such a decoder may introduce additional complexity into the optimization problem by the transformation. For this reason, we scrutinized the fitness landscapes of both problems (untransformed and transformed) to gain insight into the problem structure with means from standard fitness landscape analysis [14]. Indeed, our findings indicate a slightly growing in complexity by an increased ruggedness with a growing number of local minima [15]. But, this situation can be easily countered by using a heuristics that copes well with rugged non-linear problems like Simulated Annealing (SA).

Simulated Annealing [16] is an established Markov Chain Monte Carlo Method (MCMC) for non-linear optimization. It mimics a physical cooling process. In general, MCMC methods are an effective tool for statistical sampling applied to optimization problems [17]. The basic idea is a Markov Process that samples a target probability distribution $\pi(x) = \frac{1}{z} e^{-E(x)}$ with $z$ as a problem specific normalization parameter and $E$ measuring the error of the optimization objective. Originally, the method has been mainly applied to physical problems finding a minimum energy state and thus $E$ is sometimes still written Hamiltonian $\mathcal{H}$, e.g. in [18]. We will use the term $E$. In this process a new state $\sigma_{t+1}$ is generated from $\sigma_t$ by drawing from a proposal transition distribution $Q(\sigma_{t+1}|\sigma_t)$ [19], [20]. The new state is accepted with probability

$$A(\sigma_t \to \sigma_{t+1}) = \min\left(1, \frac{\pi(\sigma_{t+1}) Q(\sigma_{t+1}|\sigma_t)}{\pi(\sigma_t) Q(\sigma_t|\sigma_{t+1})}\right). \tag{6}$$

The proposal distribution $Q$ is a free parameter and must be adjusted to the individual problem at hand. Starting from a random initial state $\sigma_0$, the process needs a while to reach equilibrium and independence from $\sigma_0$. After this burn-in phase the samples represent the target distribution $\pi$.

In systems with deep local minima the process can be trapped without escape in reasonable time. This waiting time dilemma [21] is due to a stringent requirement for equilibrium. To escape, the process must generate subsequent states with higher energy and the probability for such a move declines roughly exponentially with the energy differences that has to be overcome. Thus, the expected waiting time for such escape grows also exponentially. For high-dimensional problems like the one that we scrutinize here, this problem is even more prevalent [21]. Several techniques have been proposed to overcome the problem of getting trapped, e.g. [22], [21], [23]; one is the concept of Simulated Annealing (SA).

SA introduces a variable temperature $T$ into the target distribution: $\pi(x) = \frac{1}{z} e^{-E(\boldsymbol{x})/T}$. The effect is that the Markov Chain may escape local minima easier at a higher temperature. The general idea of Simulated Annealing is to interpret the fitness landscape of an optimization problem as a thermodynamic system with the objective function $E(\boldsymbol{x})$ denoting the error interpreted as the energy level of a proposed solution $\boldsymbol{x}$. Initially, the system is at a high temperature. During the Markov process, the system is gradually cooled down to the ground state with the global energy minimum.

Algorithm 1 shows the basic flow within our SA with integrated decoder. This integration has first been proposed in [15]. By mimicking a cooling process, temporarily worse solutions are allowed – depending on temperature and difference in solution quality – in order to escape local minima. In our approach, a solution is described by two matrices $\boldsymbol{X}_{ij}$ and $\boldsymbol{M}_{ij}$ denoting for each energy unit $i$ and for each time interval

$j$ of the schedule a scaled active power value in $[0, 1]$. In many objective scenarios, indicator values that describe the schedule with respect to different objectives might additionally be prevalent. For demonstration purposes, we stick with the single objective case here. In this sense, each row within the matrix is the schedule for one of the units. $\boldsymbol{X}$ contains schedules from the unconstrained search space (hypercube $[0, 1]^d$ not further constrained by technical issues from the units' operations). $\boldsymbol{X}$ is initialized with random values. $\boldsymbol{M}$ concurrently holds the respective feasible values generated by the support vector decoder: $\boldsymbol{M}_i = \gamma_i(\boldsymbol{X}_i)$. Thus, $\boldsymbol{M}$ always represents a feasible (scaled) solution to the problem.

$\boldsymbol{X}$ and $\boldsymbol{M}$ represent the genotype and phenotype of a solution respectively. In each iteration of the SA exactly one schedule $\boldsymbol{x}$ from $\boldsymbol{X}$ is randomly chosen and mutated. Modification is done at a randomly chosen element $x_k$ by adding a random value $p \sim N(0, 1)$:

$$x_k \leftarrow \begin{cases} x_k + p - 1 & \text{if } x_k + p > 1 \\ x_k + p + 1 & \text{if } x_k + p < 0 \\ x_k + p & \text{else.} \end{cases} \tag{7}$$

Additionally, it can be useful especially for high-dimensional schedules to allow mutations at more than one element at a time. Only this mutated schedule has to be mapped by the respective decoder in order to keep $\boldsymbol{M}$ consistent with $\boldsymbol{X}$.

The system evolves as follows: at each temperature level $T^t$ a Markov chain samples $E(\boldsymbol{x})$. $\boldsymbol{M}$ always represents a feasible, mutated solution that can be evaluated by Eq. (5). The new proposal solution part $\boldsymbol{x}^{t+1}$ is accepted (according to the Metropolis-Hastings criterion) with probability

$$A(\boldsymbol{x}^t \to \boldsymbol{x}^{t+1}) = \min\left(1, e^{\frac{-\Delta E}{T^t}}\right), \tag{8}$$

with $\Delta E = E(\boldsymbol{x}^{t+1}) - E(\boldsymbol{x}^t)$. In each iteration, temperature $T^t$ is updated with with cooling rate $\lambda \in [0, 1[$: $T^{t+1} \leftarrow \lambda \cdot T^t$.

**Algorithm 1** Basic scheme for the Simulated Annealing step (with integrated support vector decoder).

$\boldsymbol{X}_{ij} \leftarrow \boldsymbol{x}_i \sim U(0, 1)^d,\ 1 \le i \le n$
$\boldsymbol{M}_{ij} \leftarrow \gamma_i(\boldsymbol{X}_i),\ 1 \le i \le n$
$\vartheta \leftarrow \vartheta_{start}$
**while** $\vartheta < \vartheta_{min}$ **do**
  choose random $k;\ 1 \le k \le n$
  $\boldsymbol{x}^* \leftarrow \boldsymbol{X}_k$
  mutate($\boldsymbol{x}^*$)
  $\boldsymbol{M}^* \leftarrow \boldsymbol{M};\ \boldsymbol{M}^*_k \leftarrow \gamma_k(\boldsymbol{x}^*)$
  **if** $e^{-\frac{E(\boldsymbol{M}^*) - E(\boldsymbol{M})}{T}} > r \sim U(0, 1)$ **then**
    $\boldsymbol{M} \leftarrow \boldsymbol{M}^*;\ \boldsymbol{X}_k \leftarrow \boldsymbol{x}^*$
  **end if**
  $T \leftarrow \text{cooling}(T)$
**end while**

A major advantage of this approach is the anytime property: at any time, a feasible solution exists. The Markov chain may evolve in $[0, 1]^{d \cdot n}$ without taking care of technical constraints of the individual energy units. The decoder guarantees (apart from minor inaccuracies that might easily be corrected [4]) the feasibility of the solution.

## III. Partitioning the Search Space

By employing the support vector decoder approach in combination with a heuristic solver for the optimization problem as described in the previous section, we are able to solve the scheduling problem for energy units efficiently without needing to adapt any part of the process to unit-specific properties such as technical constraints. The whole process is visualized in Algorithm 2. The resulting matrix $\boldsymbol{M}$ comprises $m$ rows and $d$ columns, where the $i^{\text{th}}$ row vector represents the chosen schedule for energy unit $U_i$ (for the remaining symbol definitions refer to Section II).

**Algorithm 2** Predictive Scheduling

$m \leftarrow$ amount of energy units
$n \leftarrow$ sample size per energy unit
$d \leftarrow$ length of planning horizon
**for all** energy unit $U_i \in \mathcal{U}$ **do**
  $s_i \leftarrow$ predicted state of $U_i$ at the beginning of the planning horizon
  **repeat**
    initialize simulation model for $U_i$ with $s_i$
    simulate feasible schedule of length $d$
  **until** $\mathcal{F}^{(U_i)}$ contains $n$ feasible schedules
  scale sample $\mathcal{F}^{(U_i)}$ using $\rho_i$
  calculate support vector model $\mathcal{S}_i$
  build support vector decoder $\gamma_i$
**end for**
**return** $\boldsymbol{M} \leftarrow \left(\text{solve } \delta\left(\sum_{i=1}^{m} \rho_i^{-1} \circ \gamma(\boldsymbol{x}_i), \boldsymbol{\zeta}\right) \to min\right)$

In the considered application domain, predictive planning is commonly done for *day-ahead* planning horizons, i. e. $d$ corresponds to 24 hours with a schedule resolution of 15 minutes. In our problem formulation, this yields a 96-dimensional search space for each energy unit. Due to the curse of dimensionality [24], this may introduce significant negative effects. For instance, with larger problem dimensions, the required amount of training data for the support vector model increases exponentially [25]. This affects both the generation of feasible schedule samples via simulation, as well as learning the support vector models from these samples. Moreover, solving the optimization problem itself gets more time-consuming due to combinatorial explosion. Finally, as the support vector decoder model is based on approximation, mapping accuracy deteriorates with larger dimensions. This may lead to infeasible schedules being misleadingly recognized as feasible.

According to [26], strategies to circumvent the curse of dimensionality in such a case can be categorized as follows:

- *Decomposition:* Given that the problem is separable, decomposition subdivides the problem into smaller parts that are easier to solve.

- *Screening:* Less significant and redundant decision variables/dimensions are pruned from the problem description in order to reduce dimensionality.
- *Mapping:* The problem is mapped to a representation comprising less dimensions. For example, by exploiting correlations between variables in the original space, a mapping can be designed that yields a correlation-free space with less dimensions.
- *Space Reduction:* Using expert knowledge, parts of the search space are excluded from optimization.
- *Visualization:* An expert prunes insignificant parts of the search space using visualization techniques for high-dimensional data. In contrast to *Space Reduction*, this is done interactively during the optimization process.

For the considered support vector decoder approach, the strategies Screening, Visualization, and Space Reduction with expert's help are inappropriate, as they rely on specific knowledge about the individual problem instance to solve, which contradicts the main motivation for our approach. Because neigbouring values in the unit schedules are often quite similar (i. e. the gradient between two time intervals is usually rather small) and thus show some correlation, Mapping might be applicable. After optimization, however, the resulting low-dimensional power profile would have to be inversely mapped to a feasible high-dimensional schedule again, which would introduce further problems.

Finally, Decomposition offers a viable solution. We cannot split the problem along the $m$ axis with respect to the result matrix $\boldsymbol{M}$ in Algorithm 2 (i. e. by optimizing over disjunct sets of energy units), because in each time step along the $d$ axis, the schedule selections of *all* participating units have to be regarded in order to minimize $\delta$. On the other hand, the problem formulation might allow us to optimize over each time step along the $d$ axis independently: If the employed distance measure $\delta$ is a *metric*, it gets minimal if the individual distances along the $d$ axis are minimal. This holds true for the Euclidean distance $\|\cdot\|_2$ we are using in this paper. Therefore, from the optimization point of view, the given problem seems to be separable along the $d$ axis. Formally, we define such a *partitioning* of the search space as

$$\begin{aligned} \pi &: \mathbb{N}^2 \to \mathbb{N} \\ (d, j, k) &\mapsto l = \pi(d, j, k), \end{aligned} \tag{9}$$

where $l = \pi(d, j, k)$ denotes the length of the $j^{\text{th}}$ partition along the $d$ axis. The parameter $k$ may hold arbitrary implementation-specific values (cf. the equidistant partitioning below). For a partitioning to be valid, the concatenation of all generated partitions must yield the whole planning horizon:

$$\sum_{j=1}^{\infty} \pi(d, j, k) = d \tag{10}$$

Moreover, for convenience we require

$$\forall i: \ \pi(d, j, k) = 0 \ \Rightarrow \ \pi(d, j+1, k) = 0, \tag{11}$$

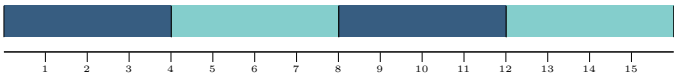


Fig. 2. Equidistant partitioning for $d = 16$ and $k = 4$.

i. e. as soon as the partitioning function yields the first zero partition, every following partition must be zero as well. Using this rather general definition of $\pi$, we may now define different partitioning strategies. For example, the *equidistant partitioning* subdivides the planning horizon into $k$ partitions of equal size

$$\pi_{\text{eq}}(d, j, k) = \begin{cases} \lceil \frac{d}{k} \rceil & \text{if } j \leq k \ \wedge \ j \leq d \bmod k, \\ \lfloor \frac{d}{k} \rfloor & \text{if } j \leq k \ \wedge \ j > d \bmod k, \\ 0 & \text{else.} \end{cases} \tag{12}$$

Figure 2 shows an example for this partitioning with $d = 16$ and $k = 4$. There are many other possible partitioning strategies, ranging from simple arithmetic fragmentations to more sophisticated strategies involving expert knowledge about the use case at hand (i. e. the structure of the target profile or the $\delta$ function). A particular promising approach is the *entropy partitioning*, which exploits the entropy in the feasible schedule samples to determine intervals of high vs. low flexibility in the units' scopes of actions, and partitions the search space accordingly. But in order to remain maximally independent from such expert knowledge, we go with the example of equidistant partitioning in the remainder of this paper.

In order to implement a partitioning scheme like e. g. the equidistant partitioning in our approach, we have to extend Algorithm 2. Special care has to be taken regarding the simulation of feasible schedules: Originally, in Algorithm 2, each simulation model was initialized with the state of the energy unit right at the beginning of the planning horizon, and was executed for $d$ time steps, such that each schedule sample exactly covers the planning horizon. Using partitions, however, schedule samples cannot be generated beforehand for the whole planning horizon. In order to identify a unit's flexibility for a certain partition, the exact state of the unit at the beginning of this partition has to be known. Thus, before being able to process a partition, we have to assign fixed schedules to the units for the preceding partition. As a consequence, the overall process ranging from schedule simulation to solving the optimization problem has to be executed for each partition separately. This ensures that, after the process finished for all partitions, the concatenated result schedules are feasible overall. On the other hand, with this approach we achieve a reduction of the design space (without expert knowledge as proposed in [26]) as every subsequent optimization process is already tackled to a fixed operational state of each unit at the beginning of a partition. The resulting process is visualized in Algorithm 3.

## IV. Evaluation

The objective of this paper is to explore the potential benefit of partitioning the search space of the given combinatorial

**Algorithm 3** Predictive Scheduling with Partioning

1: $m \leftarrow$ amount of energy units
2: $n \leftarrow$ sample size per energy unit
3: $d \leftarrow$ length of planning horizon
4: **for all** energy unit $U_i \in \mathcal{U}$ **do**
5: $s_i \leftarrow$ predicted state of $U_i$ at the beginning of the planning horizon
6: **end for**
7: $j \leftarrow 1$
8: $k \leftarrow$ implementation specific value
9: **while** $\pi(d, j, k) \neq 0$ **do**
10: **for all** energy unit $U_i \in \mathcal{U}$ **do**
11: **repeat**
12: initialize simulation model for $U_i$ with $s_i$
13: simulate feasible schedule of length $\pi(d, j, k)$
14: **until** $\mathcal{F}^{(U_i)}$ contains $n$ feasible schedules
15: scale sample $\mathcal{F}^{(U_i)}$ using $\rho_i$
16: calculate support vector model $\mathcal{S}_i$
17: build support vector decoder $\gamma_i$
18: **end for**
19: $\boldsymbol{M}^j \leftarrow \left(\text{solve } \delta\left(\sum_{i=1}^{m} \rho_i^{-1} \circ \gamma(\boldsymbol{x}_i), \boldsymbol{\zeta}\right) \rightarrow min\right)$
20: **for all** energy unit $U_i \in \mathcal{U}$ **do**
21: run simulation model for $U_i$ using schedule $\boldsymbol{x}_i$
22: $s_i \leftarrow$ predicted state of $U_i$ after running $\boldsymbol{x}_i$
23: **end for**
24: $j \leftarrow j + 1$
25: **end while**
26: **return** $\boldsymbol{M} \leftarrow \left[\boldsymbol{M}^j\right]$

problem in the data domain, followed by sequential optimization of the individual partitions. In the previous section, a partitioning framework has been introduced for this, along with a detailed description of the according optimization process chain. In order to evaluate the proposed approach with respect to the objective, a simulation study has been conducted.

### A. Simulation Setup

Following the considered example use case, we set up a simulated virtual power plant for active power planning in day-ahead scenarios, comprising CHP units with an 800 l thermal buffer store each. We used the simulation model of an EcoPower CHP as described in [3]. For each of those devices, the thermal demand for a four-family house during winter was simulated. The devices were operated in heat driven operation and thus primarily had to compensate the simulated thermal demand. Additionally, after shutting down, a device would have to stay off for at least two hours. However, due to their thermal buffer store and the ability to modulate the electrical power output within the range of $[1.3,\ 4.7]$, the devices still have some flexibility available.

For the generation of feasible schedule samples, a *successive sampling strategy* was employed: Instead of guessing whole schedules and checking feasibility afterwards (using a device's simulation model), which leads to large rejection rates, a period-wise guessing in combination with partial feasibility checks is applied repeatedly to construct feasible schedules in a successive manner, cf. [12]. Preliminary experiments indicated 200 as an adequate size for $\mathcal{F}^{(U_i)}$, so we set $n = 200$ in the present study.

The planning horizon was set to $d = 96$ time intervals, i. e. 24 hours in 15 minute resolution, which is a common use case in the application domain. As motivated in the previous section, we employ the equidistant partitioning function $\pi_{eq}$ in this study. Regarding the parameter $k$, which defines the length of the partitions and thus inversely determines the number of partitions to be generated according to (12), several experiments with $k \in [1, 96]$ have been conducted. For instance, $k = 1$ yields 96 partitions of length 1, while $k = 96$ corresponds to a single partition of length 96, i. e. no partitioning at all. This way, the influence of a partitioning on the optimization can be explored in a structured manner.

While $k$ represents the primary influence factor in our study, other parameters may cause relevant interaction effects. Here, especially the magnitude of the problem size along the $m$ axis (i. e. the number of energy units in the VPP, cf. Section III) is of particular interest, as it affects the problem complexity for each partition likewise. Similarly, different target profiles $\zeta$ have to be examined with respect to the units' available flexibilities. For example, a target profile might turn out to be easily realizable due to well matching schedule options in the units' search spaces, or vice-versa. The question arises whether this influences the potential benefit of a partitioning, and how a partitioning should be done in order to gain optimal results.

In all experiments, we used the Simulated Annealing solver as outlined in Section II. Each examined parameter configuration was simulated 100 times, so that the results can be interpreted with statistical soundness.

### B. Results

The evaluation focuses on solution quality, which is calculated as remaining error after optimization:

$$\delta\left(\sum_{i=1}^{m} \rho_i^{-1}(\boldsymbol{x}_i), \boldsymbol{\zeta}\right), \quad \boldsymbol{x}_i \in \boldsymbol{M} \tag{13}$$

where $\boldsymbol{M}$ denotes the $m \times d$ schedule matrix after all partitions have been processed (line 26 in Algorithm 3). In the following, results are visualized as box-charts, where the box spans from the upper to the lower quartile of the data. The median is shown as horizontal line within a box, whereas the whiskers span over $1.5 \times$ the interquartile range. Outliers are illustrated by circle markers.

First of all, the general influence of different $k$ values (i. e. different partition sizes) is examined. As already stated in the introduction, preliminary experiments indicated a potentially increasing solution quality when the optimization algorithm is applied in a successive manner to sequential partitions of the planning horizon. For a more thorough analysis, we conducted 100 simulations for each $k \in \{2, 8, 24, 48, 96\}$. The results are visualized in Figure 3. The optimization error clearly decreases with more and thus smaller partitions (from right to left in the figure). Comparing the extreme points, the partitioning even

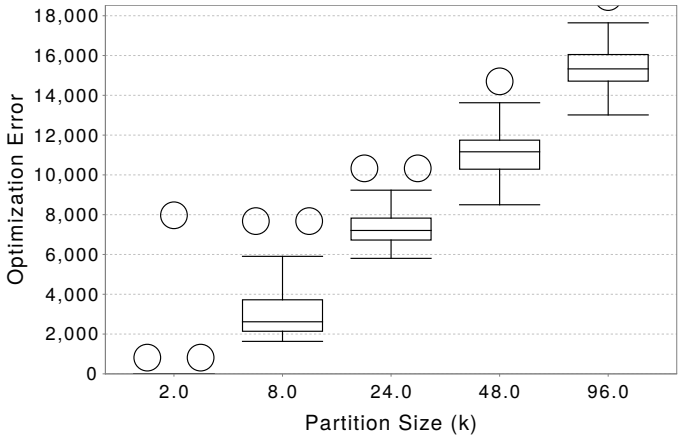


Fig. 3. Remaining optimization error for different partition sizes.

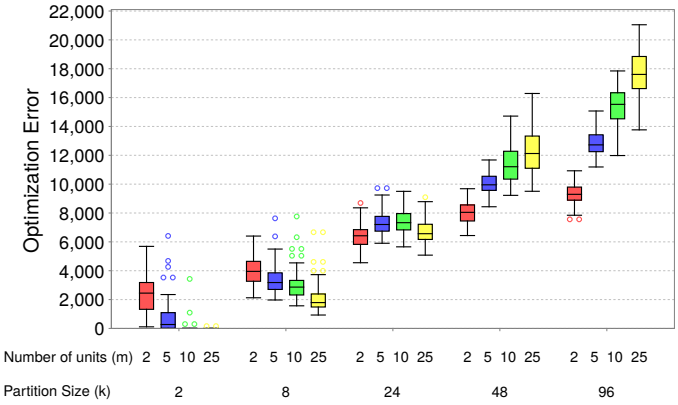


Fig. 4. Remaining optimization error for different partition sizes and varying amounts of energy units.

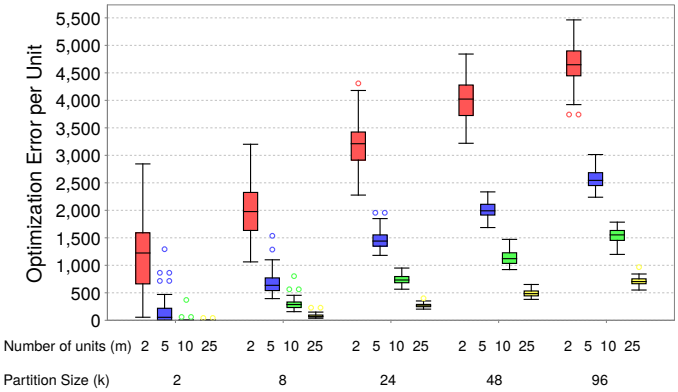


Fig. 5. Remaining optimization error per energy unit for different partition sizes and varying amounts of energy units.

allows approaching the theoretical optimum $\delta = 0$ when the partitions are generated as small as possible ($k = 2$: despite a few outliers, the box is squashed to a single line at $\delta = 0$), while the no-partitioning case yields the worst results most of the time ($k = 96$).

These results support our hypothesis strikingly, but they originate from a single experiment configuration only: On the one hand, a fixed number of energy units was involved, $m = 10$. On the other hand, the target profile $\zeta$ was generated by aggregating randomly chosen sample schedules (one for each energy unit) at the beginning of each experiment run. This way, $\zeta$ formed an "easy" target, because the energy units were able to approach it optimally in principle. In the following, we will vary this configuration in these two aspects, in order to gain more insights into the involved effects.

*1) Interaction with the Number of Energy Units:* In the considered application use-case of predictive scheduling for active power planning in day-ahead scenarios, virtual power plants may comprise different amounts of energy units, depending on e. g. regional conditions. From the optimization point of view, this corresponds to the problem size along the $m$ axis. In a partitioned setting (i. e. $k < d$), each subproblem is of size $m \times k$. Hence, $m$ affects the problem complexity for each partition likewise. To reveal possible interactions with the magnitude of $k$, the previous experiment with $k \in \{2, 8, 24, 48, 96\}$ was repeated for $m \in \{2, 5, 10, 25\}$. Figure 4 visualizes the results. Similar to Figure 3, the optimization error generally decreases with smaller partitions. Within each block, however, different effects with respect to the magnitude of $m$ are visible: For the case of small partitions, the optimization error is lower with larger values of $m$, while this trend reverses for large partitions. As this is based on the absolute error, which is naturally different for varying magnitudes of $m$, Figure 5 complementarily shows the same results against the normed optimization error with respect to the number of units, i. e. the remaining error per energy unit. Here, the trend towards a lower error for small partitions is again clearly visible, whereas the magnitude of $m$ results in a change of the slope for this trend. Concluding, $m$ seems to affect the problem complexity as a whole only, and does not seem to interact with the partition size $k$.

*2) Interaction with the Target Profile:* In active power planning, usually an application-specific target profile is given. For instance, in day-ahead energy market scenarios, a target profile would be chosen such that the economic outcome of the VPP is maximized. In contrast, in supply-demand-matching scenarios, the target profile might be e. g. a constant zero value, such that the considered set of energy units (flexible producers and consumers) can be treated as autonomous energy-wise. While it is advisable to configure VPP and target profile in a matching way, so that the latter is actually a feasible target for the former, not all target profiles are equally easy to realize.

In our study, we abstract from application-specific scenarios as follows. As a first step, a feasible target can be formed by aggregating randomly chosen sample schedules (one for each energy unit). This way, the existence of the theoretical optimum ($\delta = 0$) is guaranteed. We denote this type of target with $\zeta_0$. To generate more difficult target profiles in an easy but structured way, $\zeta_0$ can simply be shifted in magnitude:

$$\zeta_i = \zeta_0 + i \tag{14}$$

Please note that $\zeta$ is a vector, and the summation is performed element-wise, of course. Matching the size of the considered

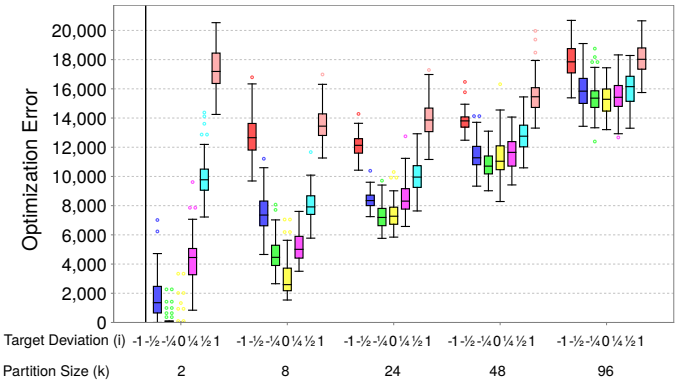


Fig. 6. Remaining optimization error for different partition sizes and varying target profile deviations.

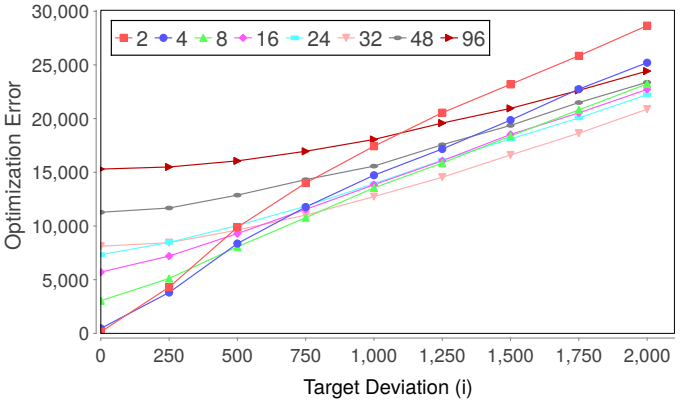


Fig. 7. Remaining optimization error for varying target profile deviations (horizontal axis) and different partition sizes (individual data series).

VPP in the present study, we choose values for $i$ between $0\,\mathrm{kW}$ and $\pm 1\,\mathrm{kW}$ in the following experiment, in order to deviate the target profile from "easy to solve optimally" towards "hard to solve optimally". Thus, as in the previous section, the original experiment with $k \in \{2, 8, 24, 48, 96\}$ was repeated for all $\zeta_i$ with $i \in \{-1, -0.5, -0.25, 0, 0.25, 0.5, 1\}$ in kW. The results are presented in Figure 6. Similar to the results from Figure 4, the general trend of better optimization results with smaller partitions is visible. The case $k = 2,\ i = -1$ is an exception. Here, the optimization was not able to find a feasible schedule at all in the available time. This is due to the very low values in the target profile in combination with a large number of partitions: Due to the independent optimization of individual partitions, the simulated CHP units stay off at the beginning of the planning horizon until the thermal buffer stores are exhausted. At that point in time, however, thermal demand exceeds the available power from the CHPs, so that no feasible schedule cannot be found anymore. With larger partitions, the effect is not present, as the optimization can act anticipatory towards feasibility (i. e. by choosing schedules that lead to a poor optimization error, but in turn form a feasible solution). This effect indicates that a strong partitioning can yield better optimization results if enough flexibility is present, but might also lead to infeasible solutions in extreme cases.

In addition to the general trend regarding the value of $k$, a u-shaped course can be seen within each configuration of the same partition size. In other words, solution quality seems to deteriorate with larger deviations from $\zeta_0$, which is not surprising at all. In order to focus on the interaction between these two effects, Figure 7 visualizes the results in a transposed way, i. e. the deviation $i$ is visualized along the horizontal axis, while the partition sizes $k$ are presented as line charts. For visualization purposes, the shown data comprises mean values only. Furthermore, in this experiment a larger amount of configurations was examined: $k \in \{2, 4, 8, 16, 24, 32, 48, 96\}$ and $|i| \in \{0, 0.25, \ldots, 2\}$. The results reveal an interesting relationship: For smaller target deviations, configurations with smaller partitions yield superior optimization results. In contrast, for larger target deviations, larger partition sizes yield better results.

In summary, the last experiment supports our previous hypothesis: With enough flexibility in a given problem configuration (in terms of feasible solution combinations with respect to the fitness function), the solver significantly benefits from a partitioning. On the other hand, in more difficult problem formulations (i. e. with less flexibility in terms of feasible solutions), the solver cannot cope with a large number of independent partitions.

## V. Conclusion

The objective of this paper was to explore the potential benefit of partitioning the search space of the given combinatorial problem in the data domain using the example of predictive scheduling in the smart grid domain. We combined the partitioning approach with a sequential optimization solving each partition successively. Simulation models of different energy units have been integrated directly in the process for handling individual search spaces and operational constraints.

Several methods to cope with the challenge of high dimensionality in optimization problems have been proposed in the past. A good overview on methods for computationally expensive black-box functions (as might be the case when using simulation models for computing objectives) is e. g. given in [26]. Our approach is a mixture of design space reduction and decomposition into sub-problems. To achieve this we have to introduce simulation models as black-boxes into the optimization process for sequentialization. Introducing this sequence of independently solvable sub-problems reduces the overall computationally effort and at the same time reduces the design space so that modeling is more accurate and optimization effort is reduced [26]. At the same time this reduction leads to a limited choice especially for later sub-problems. Sub-space parts of the design space may be missed [26]. On the other hand, with our method, we may focus on the whole sub-space at once without a need for subsequent refinement like in other methods [26].

Our results support the hypothesis of an increasing solution quality when applying the optimization algorithm in a

successive manner to sequential partitions of the planning horizon. For our experiments we mainly used a simulated annealing approach as solver although our results can be generalized to other solvers. In general, any solver benefits for partitioned data domains in predictive scheduling if a problem configuration contains enough flexibility in terms of feasible solution combinations. With decreasing flexibility, additional complexity induced by a growing number of partitions prevails.

So far all simulations have been done with scenarios regarding predictive scheduling. Additional use cases like load balancing can be easily adapted by exchanging the objective functions, as the problem structure is similar to predictive scheduling. Future work will concentrate on methods to classify the situation at hand in order to automatically decide on appropriate partition of the combinatorial problem.

## REFERENCES


[1] P. Palensky and D. Dietrich, "Demand side management: Demand response, intelligent energy systems, and smart loads," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 3, pp. 381–388, Aug 2011. doi: 10.1109/TII.2011.2158841. [Online]. Available: http://dx.doi.org/10.1109/TII.2011.2158841

[2] N. . P. F. Arteconi, A. ; Hewitt, "Domestic demand-side management (dsm): Role of heat pumps and thermal energy storage (tes) systems," *Applied Thermal Engineering, 2013, Vol.51(1-2), pp.155-165*, vol. 51, no. 1-2, p. 155. doi: 10.1016/j.applthermaleng.2012.09.023. [Online]. Available: http://dx.doi.org/10.1016/j.applthermaleng.2012.09.023

[3] J. Bremer and M. Sonnenschein, "Model-based integration of constrained search spaces into distributed planning of active power provision," *Comput. Sci. Inf. Syst.*, vol. 10, no. 4, pp. 1823–1854, 2013. doi: 10.2298/CSIS130304073B. [Online]. Available: http://dx.doi.org/10.2298/CSIS130304073B

[4] ——, "Constraint-handling for optimization with support vector surrogate models – A novel decoder approach," in *ICAART 2013 – Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, J. Filipe and A. Fred, Eds., vol. 2. Barcelona, Spain: SciTePress, 2013. doi: 10.5220/0004241100910100. ISBN 978-989-8565-38-9 pp. 91–100. [Online]. Available: http://dx.doi.org/10.5220/0004241100910100

[5] J. Bremer, B. Rapp, and M. Sonnenschein, "Encoding distributed search spaces for virtual power plants," in *IEEE Symposium Series on Computational Intelligence 2011 (SSCI 2011)*, Paris, France, 4 2011. doi: 10.1109/CIASG.2011.5953329. [Online]. Available: http://dx.doi.org/10.1109/CIASG.2011.5953329

[6] S. McArthur, E. Davidson, V. Catterson, A. Dimeas, N. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1743–1752, 2007. doi: 10.1109/TPWRS.2007.908471. [Online]. Available: http://dx.doi.org/10.1109/TPWRS.2007.908471

[7] M. Sonnenschein, C. Hinrichs, A. Nieße, and U. Vogel, "Supporting renewable power supply through distributed coordination of energy resources," in *ICT Innovations for Sustainability*, ser. Advances in Intelligent Systems and Computing, L. M. Hilty and B. Aebischer, Eds. Springer International Publishing, 2015, vol. 310, pp. 387–404. ISBN 978-3-319-09227-0. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-09228-7_23

[8] A. Nieße, S. Beer, J. Bremer, C. Hinrichs, O. Lünsdorf, and M. Sonnenschein, "Conjoint dynamic aggregation and scheduling for dynamic virtual power plants," in *Federated Conference on Computer Science and Information Systems - FedCSIS 2014, Warsaw, Poland*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 9 2014. doi: 10.15439/2014F76. [Online]. Available: http://dx.doi.org/10.15439/2014F76

[9] J. Bremer and M. Sonnenschein, "A distributed greedy algorithm for constraint-based scheduling of energy resources," in *FedCSIS*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2012. ISBN 978-83-60810-51-4 pp. 1285–1292.

[10] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, Jan. 2002. doi: 10.1016/S0045-7825(01)00323-1. [Online]. Available: http://dx.doi.org/10.1016/S0045-7825(01)00323-1

[11] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, vol. 7, pp. 19–44, 03 1999. doi: 10.1162/evco.1999.7.1.19. [Online]. Available: http://dx.doi.org/10.1162/evco.1999.7.1.19

[12] J. Bremer and M. Sonnenschein, "Sampling the search space of energy resources for self-organized, agent-based planning of active power provision," in *EnviroInfo*, ser. Berichte aus der Umweltinformatik. Shaker, 2013, pp. 214–222.

[13] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004. doi: 10.1023/B:MACH.0000008084.60811.49. [Online]. Available: http://dx.doi.org/10.1023/B:MACH.0000008084.60811.49

[14] V. K. Vassilev, T. C. Fogarty, and J. F. Miller, "Information characteristics and the structure of landscapes," *Evol. Comput.*, vol. 8, no. 1, pp. 31–60, Mar. 2000. doi: 10.1162/106365600568095. [Online]. Available: http://dx.doi.org/10.1162/106365600568095

[15] J. Bremer and M. Sonnenschein, "Parallel tempering for constrained many criteria optimization in dynamic virtual power plants," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid, CIASG 2014, Orlando, FL, USA, December 9-12, 2014*. IEEE, 2014. doi: 10.1109/CIASG.2014.7011551 pp. 51–58. [Online]. Available: http://dx.doi.org/10.1109/CIASG.2014.7011551

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. doi: 10.1126/science.220.4598.671. [Online]. Available: http://dx.doi.org/10.1126/science.220.4598.671

[17] Y. Li, V. A. Protopopescu, N. Arnold, X. Zhang, and A. Gorin, "Hybrid parallel tempering and simulated annealing method," *Applied Mathematics and Computation*, vol. 212, no. 1, pp. 216–228, 2009. doi: 10.1016/j.amc.2009.02.023. [Online]. Available: http://dx.doi.org/10.1016/j.amc.2009.02.023

[18] A. Müller, J. J. Schneider, and E. Schömer, "Packing a multidisperse system of hard disks in a circular environment," *Phys. Rev. E*, vol. 79, p. 021102, Feb 2009. doi: 10.1103/PhysRevE.79.021102. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.79.021102

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. doi: 10.1063/1.1699114. [Online]. Available: http://dx.doi.org/10.1063/1.1699114

[20] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970. doi: 10.1093/biomet/57.1.97. [Online]. Available: http://dx.doi.org/10.1093/biomet/57.1.97

[21] W. H. Wong and F. Liang, "Dynamic weighting in Monte Carlo and optimization," *Applied Mathematics. Proceedings of the National Academic of Science*, vol. 94, pp. 14 220–14 224, Dec. 1997.

[22] E. Marinari and G. Parisi, "Simulated tempering: a new Monte Carlo scheme," *Europhys. Lett.*, vol. 19, no. 6, 1992.

[23] S. Brown and T. Head-Gordon, "Cool walking: A new markov chain monte carlo sampling method." *Journal of Computational Chemistry*, vol. 24, no. 1, pp. 68–76, 2003. doi: 10.1002/jcc.10181. [Online]. Available: http://dx.doi.org/10.1002/jcc.10181

[24] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality. aide-memoire of a lecture at," in *AMS Conference on Math Challenges of the 21st Century*, 2000.

[25] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science 3512*. Springer, 2005. doi: 10.1007/11494669_93 pp. 758–770. [Online]. Available: http://dx.doi.org/10.1007/11494669_93

[26] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Structural and Multidisciplinary Optimization*, vol. 41, no. 2, pp. 219–241, 2010. doi: 10.1007/s00158-009-0420-2. [Online]. Available: http://dx.doi.org/10.1007/s00158-009-0420-2