Sampling the Search Space of Energy Resources for Self-organized, Agent-based Planning of Active Power Provision

Jörg Bremer[1], Michael Sonnenschein[1]

Abstract

The future smart energy grid demands for new control paradigms that are able to incorporate a huge number of rather small, distributed and individually configured energy resources. In order to allow for a transition of the current central market and network structure to a decentralized smart grid, with small units pooling together to jointly trade their electricity production on specialized markets, self-organization concepts will become indispensable as an efficient management approach. In order to enable ahead of time planning of electricity that incorporates global objectives and individually constrained distributed search spaces in such highly dynamic environment, meta-models of constrained spaces of operable schedules are indispensable for efficient communication and uniform access. An essential prerequisite for building-up machine learning based domain models of individually constrained search spaces is a training set of operable example schedules. Drawing such a sample from an electricity unit's simulation model is a challenging task due to the high dimensionality of the problem. We present two computationally feasible sampling methods and analyze their complexity and appropriateness. Moreover, the embedding of these methods and the interplay of sampling and simulation in a multi agent simulation is presented.

1. Introduction

The task of ahead of time planning of active power provision will in future smart grid scenarios become much harder to solve. Due to the huge number of small, distributed and individually configured energy resources (producer, batteries as well as consumers) that have to be orchestrated, new decentralized control architectures are expected to come in place (Nieße et al., 2012; Sonnenschein et al., 2012).

Within the project Smart Nord (http://smartnord.de/), an agent based control scheme rooted in market mechanisms and discriminated in day ahead active power provision and ancillary services like frequency or voltage stability is currently developed. In this paper we will focus on the first use case.

For any planning scenario, each energy unit has to determine alternative load schedules from that it may choose. We assume that many controllable units may not only offer a set of several alternative load profiles but rather a high-dimensional search space with schedules for some given future time horizon. The term unit may refer to either a single producer, consumer or prosumer device (energy resource) or to a set of devices that is regarded as a commonly jointly ensemble of energy resources (e.g. a household). These many search spaces from all the units in a grid serve as input or as basis for decision making for several algorithms during the planning and re-scheduling phase like coalition forming (Beer, Sonnenschein & Appelrath, 2011), schedule-partition or optimization (Bremer & Sonnenschein, 2012), or continuous scheduling for grid stabilizing. These are only some examples eased by surrogate models of the search spaces for efficient communication (Bremer, Rapp & Sonnenschein, 2010) and automatic generation of decoders that enable standardized usage and integration (Bremer & Sonnenschein, 2013). Such models serve as a means for exploring the search space without actually knowing it and substitute for example computationally unfavorable unit simulations during a huge number of calls of evaluation functions during optimization. In contrast to the real (and arbitrarily implemented) simulation models, the surrogate models consist only of a

[1] University of Oldenburg, Department of Computing Science, Uhlhornsweg 84, D-26129 Oldenburg

standardized mathematical definitions and a set of defined parameters may also be communicated to other agents.

In order to build such models, representations of the feasible regions that contain the operable schedules of a unit are learned by machine learning methods from a set of example schedules. As a training set, we initially need a set of operable example schedules drawn from a simulation model. This set represents the feasible region and thus serves as a stencil for the surrogate model that later substitutes the simulation model. The encoding part that builds the model has already been scrutinized (Bremer, Rapp & Sonnenschein, 2010). Here we focus on building the training sample.

Thus, the rest of the paper is organized as follows: We start with a description of our use case and the technical environment. We then derive the necessity for sophisticated sampling methods and describe two methods and their embedding into the multi agent environment. We conclude with some results and a comparison of the methods' complexity.

2. The Active Power Planning Use Case

Prior to describing the sampling process, we sketch the use case that we aim at in the smart grid domain and derive the necessity for generating a training set of schedules. For simplicity, we restrict our description to the part of our multi agent scenario where a coalition of distributed energy resources has already formed-up to place a product on some energy market. In case of acceptance, the coalition is supposed to deliver a certain joint active power schedule for a given, future time horizon as defined by the final product definition after market-clearing. This product schedule might differ from the original plan. The schedule partition used during negotiation might have been sub-optimal for complexity reduction by using incomplete information or a reduced objective set during coalition forming. Moreover, the partition might have become infeasible due to changed conditions for grid compatibility. Anyway, a successive optimization step is necessary after market-clearing to find optimal or at least good schedules for each coalition member such that the wanted product schedule is jointly resembled by the operations of the group.

In such smart grid load planning scenarios, a scheduling algorithm (whether centralized or distributed) must know for each participating energy resource which load schedules are actually operable (satisfy all constraints) and which are not. Each energy resource has to restrict its possible operations due to a set of individual constraints. These can be distinguished into hard constraints (usually technically rooted, e.g. minimum and/or maximum power input or output) and soft constraints (often economically or ecologically rooted, e.g. personal preferences like noise pollution in the evening).

When determining an optimal partition of the schedule, exactly one alternative schedule is taken from each energy units' search space of individually operable schedules (individual feasible region) in order to assemble the desired aggregate load schedule. To make the necessary information available for each unit, the feasible region can efficiently be encoded by a support vector approach as has e.g. been demonstrated in (Bremer, Rapp & Sonnenschein 2011).

3. Modeling the Search Space

Such a model enables us to guide an arbitrary planning algorithm where to look for feasible solutions. This is done by mapping the whole domain onto the feasible region that holds the operable schedules as defined by the set of unit specific constraints. Thus an algorithm always operates with a feasible solution (Bremer & Sonnenschein, 2013). The search space model for the individually constrained feasible regions of the several units within the planning scenario consists of the following information:

1. A set of example schedules (support vectors on the surface of the enclosing manifold)
2. Associated weights (for each schedule for calculating the decision boundary of feasibility)

3. Unit parameters: max power, max cost, etc.
4. A decision function that allows for discriminating feasible schedules and for ordering schedules due to their proximity to the feasible region.

The most important feature of this model is the enabling of an automatic derivation of a mapping decoder that transforms the problem of distributed active power planning into an unconstrained one (Bremer & Sonnenschein, 2013). Such a decoder is a constraint-handling technique (Kramer, 2010) that maps the constrained problem space to some other not-restricted space where the search operates. Figure 2a shows the concept. First, a support vector approach is trained to learn an abstraction of the enclosing envelope that contains the training sample. This abstraction may already serve as a classifier that distinguishes between operable and not operable schedules. In a next step, we derive a mapping from the abstraction that maps the whole original unconstrained domain of all schedules to the region of operable schedule. In this way, we get a means that guides a search algorithm where to look for feasible solutions and the constrained problem is transferred into an unconstrained one that is much easier to solve.

Figure 1 shows the general work flow of the above sketched concept. In smart grid load planning scenarios, an optimization algorithm (whether centralized or distributed) must know for each participating energy resource which schedules are actually operable (satisfy all constraints) and which are not. These can be determined by simulating the energy unit. Individual constraints can be distinguished into hard constraints and soft constraints which are reflected in the example schedules and are thus encoded in the model.

When determining an optimal partition, exactly one alternative schedule is taken from each energy unit's search space of individually operable schedules (individual feasible region) in order to assemble the desired aggregate load schedule. At this stage the search space model substitutes the simulation model (which is private to the agent).

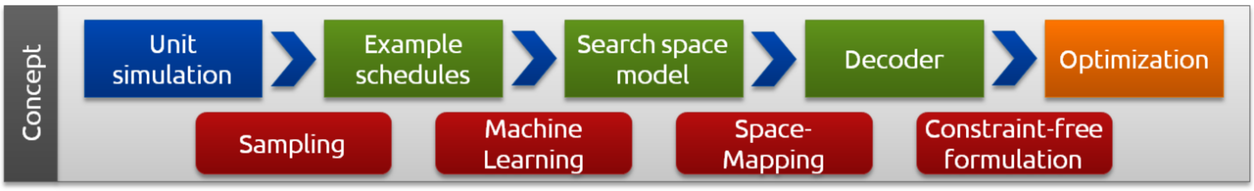


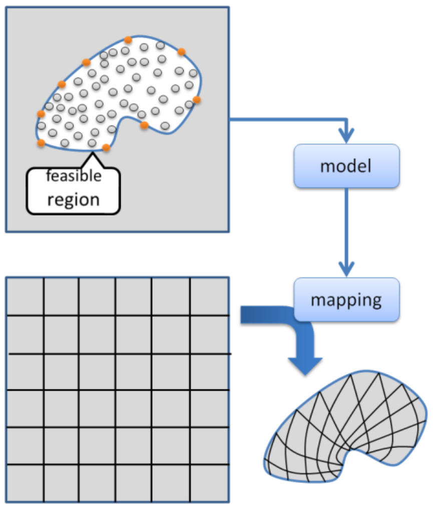
Figure 1: General workflow of the constraint-handling concept.

3.1 Sampling

The machine learning method for building the above sketched search-space-model needs a training sample of feasible schedules as input. Hence, prior to encoding, we have to draw such a sample with the help of simulation model of the energy unit. We will discuss such methods in the context of their embedding into our simulations.

3.1.1 Architecture

The architecture of the sampling system comprises the interplay of different components from different layers of our smart grid simulation. We will here focus on the part depicted in figure 2a that is concerned with sampling. On the physical layer one would usually expect the physical units (CHPs, heat pumps, etc.). During the simulations these are substituted by (usually Matlab/ Simulink) simulation models; orchestrated by the mosaik framework (Schütte, Scherfke & Sonnenschein, 2012). These are the units where we want to build the search space models of.



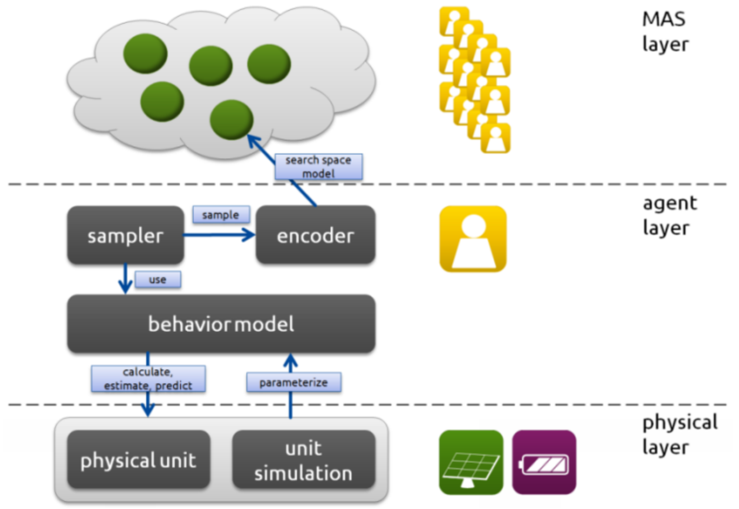


Figure 2a (left): General decoder concept. Figure 2b (right): Embedding into multi agent architecture.

Each unit is represented by an agent. In order to allow the agent to make assumptions about the future behavior of the unit, each agent possesses a behavior model for its unit. While the simulation model emulates the operations and resulting technical states of a unit, the behavior model also allows simulations about possible future operations and thus additionally supports what-if-scenarios. The algorithm that interactively queries the behavior model to generate the sample is implemented in a sampler component that in turn feeds the encoder with a set of example schedules from which the search space model is built.

On the level of the multi-agent-system these search space models are exchanged between the agents and are jointly used e.g. for negotiation.

3.1.2 Sampling Methods

Naïve sampling, i.e. guessing a schedule as a whole and then evaluating its feasibility with the help of the unit or rather its behavior model is not conductive for schedules with more than about 6 to 8 dimensions. Often, the feasible region of operable schedules is only a very small proportion of the space of all schedules. If, for example, for each time period one third of the alternative schedules are prohibited by some constraints, then the ratio of constrained and unconstrained space amounts to

$$1 : \left(\frac{2}{3}m\right)^d / m^d .$$

For the case of schedules with d=96 dimensions with m discrete power levels for each time period this would amount to approx. 1.25×10^{-17}. This would get us far too long to get one right. If the proportion of infeasible loads drops to ¼ the number of necessary tries is still 10^{12}. In the following, we compare two alternative approaches for sampling the model with sufficient efficiency to cope with the above sketched problems:

Successive Sampling: This sampling method iteratively samples shorter time horizons and constructs the final schedule by concatenating the sample parts. Hence, we currently draw samples as a successive drawing of period-wise guessing:

1. Guess a random power level for just one period (or any short enough time-horizon).
2. Validate with the help of a model of the energy unit.
3. If valid: simulate follow-up state of the unit and goto 1 for the next time period.

This approach has the advantage of leading far more likely to guesses of valid schedules. The probability P for guessing a valid schedule for a single period is already rather high. Allowing for multiple guessing (with number of tries n) results in the even higher probability (Bremer, Rapp & Sonnenschein, 2010)

$$P_{(n)} = \sum_{i=1}^{n} B(i|P,n) = \sum_{i=1}^{n} \binom{n}{i} P^i (1-P)^{n-i} .$$

Here, $P_{(n)}$ is the probability for at least one successful try out of n guesses. Successive guessing then results in an overall probability for successfully guessing a complete schedule of d time periods of

$$P_{(n)}^{d} = \left(\sum_{i=1}^{n} B(i|P,n) \right)^{d} .$$

As an example, let the probability of correctly guessing a valid power level for a single period be 0.05. Allowing for 100 guesses for each period, then the overall probability for guessing a schedule of 96 periods correctly is still $P_{(100)}^{96}$=0.5655, which is sufficiently high in contrast to 10^{-125} (chance of guessing all 96 periods right at once with probability of 0.05 for each period).

One drawback of this approach is that in this way we get a set of schedules that is dominated by the first periods. That means schedules do not have equal probability for being chosen. On the other hand, we are not primarily interested in statistical properties of the sample or the underlying density. Instead, we want to sound the geometric region where operable schedules reside in.

Figure 3 shows the interaction of this type of sampler with the unit behavior model inside an agent implementation. The procedure starts with initializing the unit behavior model with the help of parameters from the physical unit or – in case of the simulation – from its simulation model. These parameters may be directly read from the unit reflecting its current operation state or may be further projected onto a future state using the current operation schedule. This initialization defines the initial state of the unit at the starting point from whence alternative schedules are to be determined. This model with the defined starting-state is given as a reference to the sampler that uses it during the following procedure.

Each time the sampler starts with determining a new random schedule, it resets the model to the initial state. Then, a random power level for only one time period is chosen (uniformly distributed from [$P_{\min}$, $P_{\max}$]) and given to the model for validation.

In a sub-process, the model takes the power value (also works for a power vector) as a prospective load schedule and simulates the operation starting from current state and determines the follow-up state of the unit. In contrast to the unit simulation model, the behavior model is not state-less. If the follow-up state is feasible and no operation constraint is violated, the follow-up state becomes the current state of the behavior models automata. Elsewise, the operations are withdrawn and the model stays in the previous state.

Depending on the result of the validation, the sampler continues with a guess for the next period or with another try for the current period. If the maximum number of tries for one period is exceeded, the sampler gives up for the current schedule and starts with a new one.

The whole guessing procedure is repeated until the wanted number of schedules has been found.

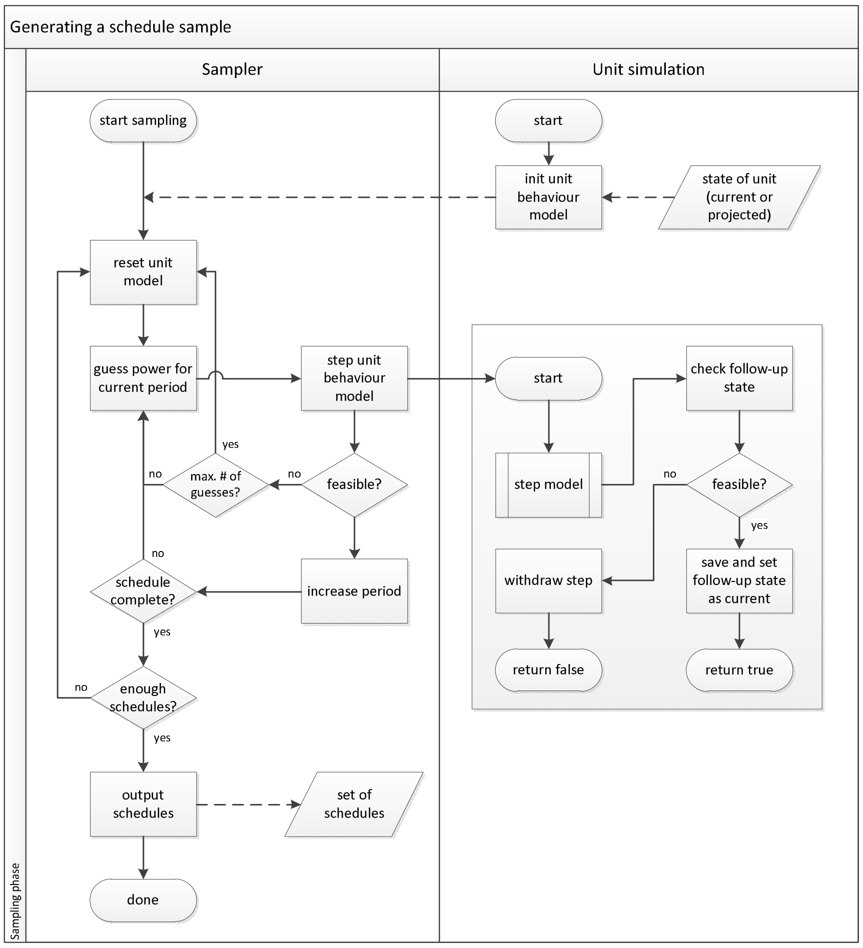


Figure 3: Interactive sampling process as dialog between sampler and behaviour model for distinguishing feasibility.

Drunken sailor sampling: As an alternative sampling scheme, we scrutinized a Monte Carlo approach that starts with an operable schedule of full length obtained from the behavior model. Usually, this schedule should be the one that results from undisturbed operation of the unit resulting from autonomous operation without any control signals or presetting from the outside. In each step, this schedule is modified at least on point in time with a random (Gaussian) mutation of the power level. If the resulting schedule is still feasible (checked by behavior model), it is added to the sample set. Else, the sampler takes one randomly chosen schedule from the bag of already found and continues the procedure of random mutations.

3.2 Results

Figure 4 shows samples drawn with different approaches. As energy unit, an under-counter boiler has been modeled by the behavior model. A traditional under-counter boiler has the simple task of keeping a small reservoir of water within a certain temperature band using a two position step control: starting the heater whenever temperature falls below lower bound and stopping above upper bound. However, in our simulations we assumed to have a device with extended control capabilities allowing a more sophisticated control for arbitrary heating as long as temperature stays within the allowed region. Nevertheless, as mean we would expect a rather simply regular oscillating behavior as for demonstration purposes we do not model any water drawings by humans. The sample that has been drawn with the naïve method shows exactly this behavior. Naturally, this would have been the best approach but it is the most computationally complex one. We have only been able to draw this sample by using a priori knowledge: Due to the specific parameterization of the unit, power is always within [0, 0.3].

On the lower left side, we see a pattern drawn with the successive approach which is dominated by the first periods and which is thus a selective sample. Both samples on the right are drawn with the drunken sailor approach (three mutations each time on top, one mutation each time on bottom). Clearly, these samples are dominated by the original undisturbed schedule and one can easily identify the tube around this first schedule.

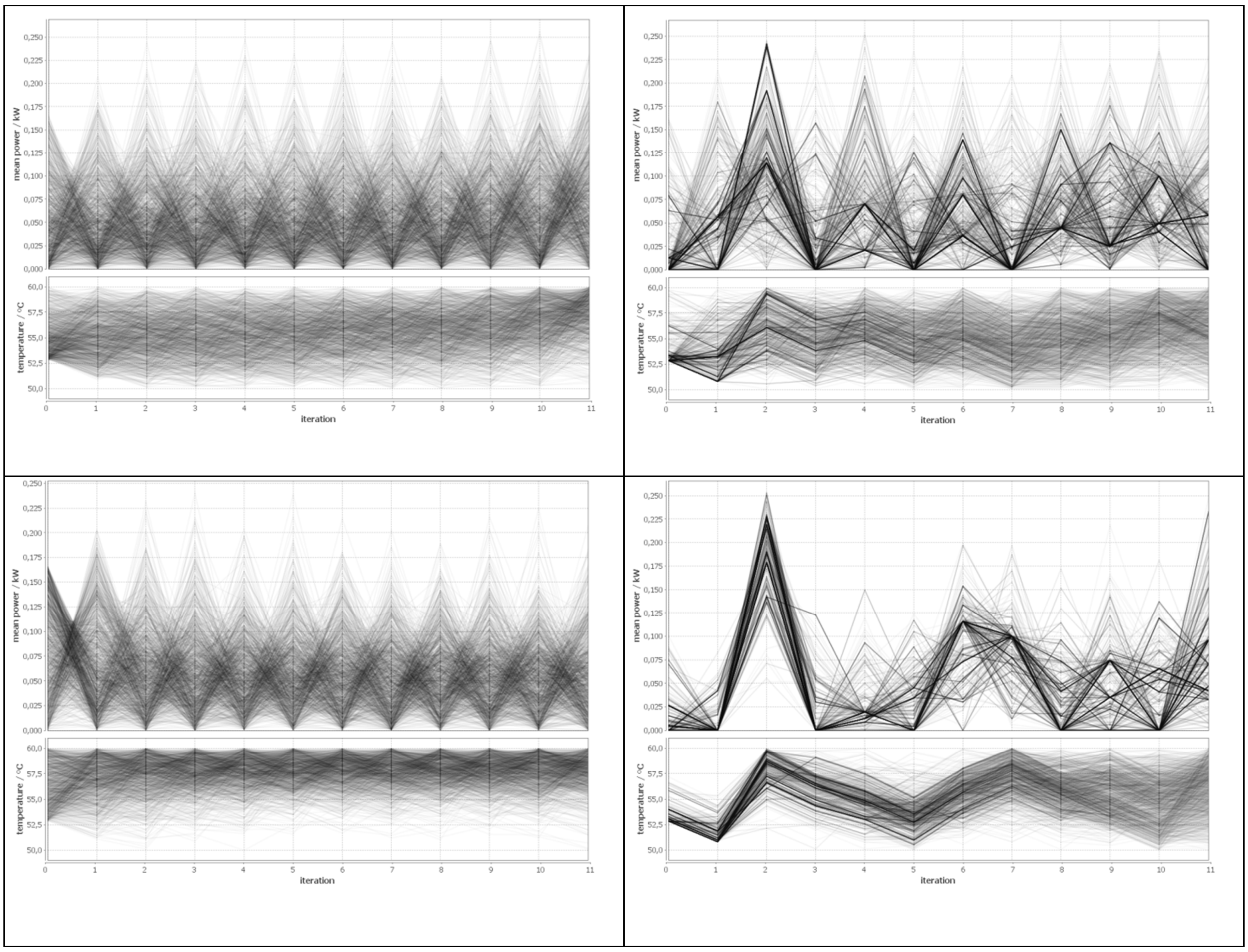


Figure 4: Results from different simulation runs.

All in all, the successive method looks most promising. Depicted are 5000 schedules each and the respectively resulting temperature profiles below. The time complexity for drawing the sample supports this assumption. Figure 5 shows a comparison of the time complexity of all methods. Successive sampling is competitively fast compared with the others but delivers the best sample. As can be seen by the temperatures in the successive sample (lower left) the sample is unbalanced but all topological traits are already there. This sample can still be improved by using adopted density functions for generating random samples but on the cost of additional calculation complexity for example with the help of kernel density estimation. In general, density estimation denotes the process of constructing an approximation function for an estimate of an unknown probability density based on a set of observed data. Several techniques for estimating the density from data sample have been developed. Building and rescaling a histogram is the most basic form of density estimation. Of course, this gives only a discrete estimate. A popular method for estimating the unknown density of a sample is the kernel density estimation (KDE) that is also known as the Parzen-Rosenblatt window method (Parzen, 1962; Rosenblatt 1956, Binder 1997):

$$\hat{f}_h(x) = \frac{1}{1nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right).$$

If $X = x_1, \ldots, x_n \in \mathbb{R}$ is an iid sample and K a symmetric (not necessarily positive) function that integrates to zero, $\hat{f}_h(x)$ gives an estimator for the underlying density of X. In general the kernel K is a continuous Lebesgue density of almost arbitrary choice. In case of a Gaussian kernel for example, a new random sample in the course of the previously described sampling process would then be generated as $x = N(x_j, h)$ with normal distribution $N(\mu, \sigma)$ with mean μ and standard deviation σ. The computational complexity for calculating (pseudo) random numbers from the estimation is thus the same as normal distributed values. For other kernels, we would have to use a random number generator for the respective distribution.

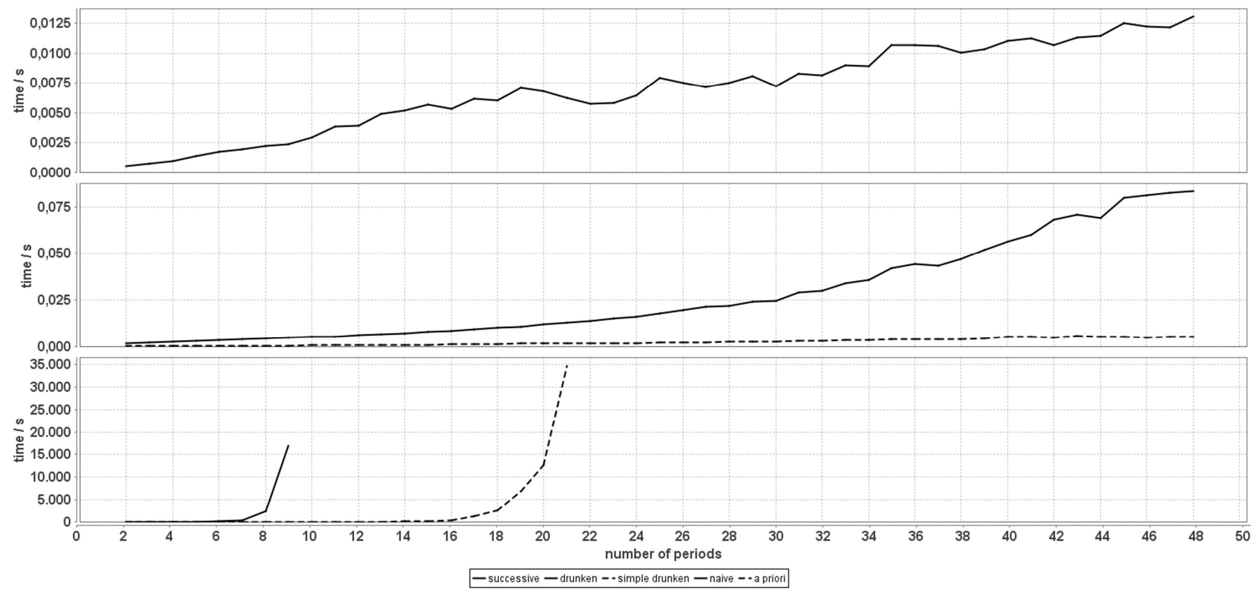


Figure 5: Comparison of runtime-behavior of the different approaches.

4. Conclusion and further work

We showed that a naïve sampling for drawing random samples as a training set from a simulation model of energy resources as input for learning search space models is computationally intractable. We presented a method for successively drawing a sample and showed its efficiency. This method is currently applied to large scenarios (approx. 50.000 electrical units) in the smart grid domain (http://smartnord.de) and is thus applied to many different unit types with differently implemented behavior models (Matlab/ Simulink, Python, Java, etc.).

An a priori estimate of the (unit model specific) distributions for generating appropriately distributed random guesses may significantly improve the sample quality. Currently, we are working on an extension that will use a kernel density estimate as an approximation of the real distribution.

Acknowledgement

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

Bibliography

Binder, K., 1997: Applications of Monte Carlo methods to statistical physics. Rep. Prog. Phys. 60(5), pp 487—559, 1997.

Beer, S., Sonnenschein, M., Appelrath, H.-J., 2011: Towards a Self-Organization Mechanism for Agent Associations in Electricity Spot Markets. In: Heiß, H.-J., Pepper, P., Schlingloff, H., Schneider, J. (Hrsg.): Informatik 2011 (Proceedings), Springer, LNCS GI-Edition, ISBN 978-88579-286-4

Bremer, J., Rapp, B., Sonnenschein, M., 2010: Support Vector based Encoding of Distributed Energy Resources' Feasible Load Spaces. IEEE PES Conference on Innovative Smart Grid Technologies Europe. Gothenburg, Sweden, October 2010, ISBN 978-1-4244-8509-3

Bremer, J., Sonnenschein, M., 2012: A Distributed Greedy Algorithm for Constraint-based Scheduling of Energy Resources. 1st International Workshop on Smart Energy Networks & Multi-Agent Systems, FedCSIS Proceeding, ISBN 978-83-60810-51-4, pp. 1285-1292

Bremer, J., Sonnenschein, M. 2013: Constraint-Handling for Optimization with Support Vector Surrogate Models: A Novel Decoder Approach. 5th International Conference on Agents and Artificial Intelligence (ICAART13). Barcelona, February 2013

Kramer, O., 2010. A review of constraint-handling techniques for evolution strategies. *Appl. Comp. Intell. Soft Comput.* 2010, Article 3 (January 2010), 19 pages

Nieße, A., Lehnhoff, S.; Tröschel, M.; Uslar, M.; Wissing, C.; Appelrath, H.-J.; Sonnenschein, M.: Market-Based Self-Organized Provision of Active Power and Ancillary Services: An Agent-Based Approach for Smart Distribution Grids. 2012 IEEE Workshop on Complexity in Engineering (IEEE COMPENG 2012). Aachen, 2012.

Nieße, A, Sonnenschein, M.: Using Grid Related Cluster Schedule Resemblance for Energy Rescheduling. SmartGreens2013. Aachen, 2013

Parzen, E., :On estimation of a probability density function and mode, The Annals of Mathematical Statistics, vol. 33, no. 3, pp. pp. 1065–1076, 1962.

Rosenblatt, M., Remarks on some nonparametric estimates of a density function, The Annals of Mathematical Statistics, vol. 27, no. 3, pp. pp. 832–837, 1956.

Schütte, S., Scherfke, S., Sonnenschein, M., 2012: Mosaik - Smart Grid Simulation API. Proceedings of SMARTGREENS 2012 - 1st International Conference on Smart Grids and Green IT Systems. ISBN 978-989-8565-09-9

Sonnenschein, M., Appelrath, H.-J., Hofmann, L., Kurrat, M., Lehnhoff, S., Mayer, C., Mertens, A., Uslar, M., Nieße, A., Tröschel, M., 2012: Dezentrale und selbstorganisierte Koordination in Smart Grids. VDE-Kongress 2012 - Intelligente Energieversorgung der Zukunft, Stuttgart