Active Learning for Efficient Sampling of Control Models of Collectives*

Alexander Schiendorfer[†], Christoph Lassner[‡§], Gerrit Anders[†], Wolfgang Reif[†], and Rainer Lienhart[‡]
Institute for Software & Systems Engineering, University of Augsburg, Germany[†]
Multimedia Computing and Computer Vision Lab, University of Augsburg, Germany[‡]
Perceiving Systems Department, Max Planck Institute for Intelligent Systems, Germany[§]
Email: {schiendorfer, anders, reif}@isse.de, lienhart@informatik.uni-augsburg.de
christoph.lassner@tuebingen.mpg.de

***Abstract*—Many large-scale systems benefit from an organizational structure to provide for problem decomposition. A pivotal problem solving setting is given by hierarchical control systems familiar from hierarchical task networks. If these structures can be modified autonomously by, e.g., coalition formation and reconfiguration, adequate decisions on higher levels require a faithful abstracted model of a collective of agents. An illustrative example is found in calculating schedules for a set of power plants organized in a hierarchy of Autonomous Virtual Power Plants. Functional dependencies over the combinatorial domain, such as the joint costs or rates of change of power production, are approximated by repeatedly sampling input-output pairs and substituting the actual functions by piecewise linear functions. However, if the sampled data points are weakly informative, the resulting abstracted high-level optimization introduces severe errors. Furthermore, obtaining additional point labels amounts to solving computationally hard optimization problems. Building on prior work, we propose to apply techniques from active learning to maximize the information gained by each additional point. Our results show that significantly better allocations in terms of cost-efficiency (up to 33.7% reduction in costs in our case study) can be found with fewer but carefully selected sampling points using Decision Forests.**

I. Abstracting Collective Behavior

In systems theory, models of dynamical systems capture input-output behavior necessary for a controller to be engineered. For hierarchical (multi-agent) planning and optimization tasks [1], [2], [3, Chapter 11], decisions for a collection of entities have to be made on a higher level and refined on lower levels to allow for effective task decomposition. Consider for instance a problem in swarm robotics: a team of robots jointly pulls an object to a target with only a limited number of available attachment points. If the weight of the load exceeds the capacities of the robots, a stronger yet slower robot would be used instead of a weaker fast one, reducing the team's overall speed. This change in speed would however be abrupt, visible as a jump in a function mapping weight to possible transportation speed. An agent coordinating multiple teams could make coarse decisions based on an approximate representation. Similarly, in a grid computing scenario, we are interested in calculating jobs in an energy-efficient manner. Certain workloads could require switching on additional (perhaps older) machines that abruptly increase the energy demand. We find related instances in situations where the cooperation of agents depends on concurrent access to restricted resources (e.g., devices, bandwidth, and the like). Another prominent example can be found in decentralized power management. A virtual power plant is a composition of a set of physical power plants that jointly act as one supplier, implying that bidding on scheduled productions on an energy market depends on individual control capabilities. For instance, the costs for a joint production depend on the best combination of two underlying productions providing that capacity.

We note that all these instances show a functional relationship of two variables defined for the collective that are linked combinatorially with variables of the underlying agents. If systems are further equipped with the ability to *self-organize*, i.e., to form collaboration and organization structures autonomously at runtime, we need methods and tools to support the creation of a suitable input-output model at runtime. Assuming access to certain points of the graph of the function, i.e., input-output pairs, we may construct an abstract model of the collective using a set of these points. Such approximations could be achieved by fitting a regression model or defining piecewise linear functions. Due to the combinatorial nature, it is highly desirable to avoid selecting uninformative input-output points, e.g., in areas of the function with linear slopes. Other subranges may exhibit more discontinuities or abrupt changes, making sampled points in this subrange more informative. Algorithms in *active learning* such as uncertainty sampling deal with precisely these problems [4]. The classical framework assumes a supervised learning scenario where unlabeled data is ample but the labeling task itself is time-consuming or expensive. We suggest to use this data-driven methodology to be able to benefit from the flexibility to model various unknown functions. Active learning is subdiscipline within machine learning and traditionally focused on classification tasks with some results being transferred to regression problems. Since the resulting model loses precision with respect to the original collective and relies on a set of sampled points, we refer to this kind of model creation as *sampling abstraction*.

We present an algorithm that is designed for the above-mentioned scenarios where i) a functional dependency is present, ii) this mapping is combinatorial in nature (e.g., by discrete jumps), and iii) we have access to *sample* points of the dependent variable for given inputs. This vision has previously been sketched in [5], extends previous work on abstraction algorithms [6], and is properly evaluated in this work.

*This research is partly sponsored by the research unit *OC-Trust* (FOR 1085) of the German Research Foundation.

For Review Only

In Sect. II, we formalize the requirements and assumptions based on the introductory examples and present the core idea of our algorithm. Section III instantiates this algorithm for our case study in decentralized energy management followed by a description of the active learning methods based on Decision Forests in Sect. IV. Our evaluation in Sect. V confirms our assumption that active learning significantly improves the selection of sampling points in our case study. Pointers to related work are given in the form of underlying theories and fields wherever applicable and not in a dedicated section as the authors are not aware of related methods combining active learning with discrete optimization to acquire labeled data.

II. Algorithm for Sampling Abstraction

Formally, we assume a function $f : D_A \to C_A$ with a (possibly n-dimensional) domain $D_A \subseteq \mathbb{R}^n$ that is regulated by constraints over a system of agents A and a codomain $C_A \subseteq \mathbb{R}$ representing the dependent variable. We further need a function $cand(D_A) \to 2^{D_A}$ that returns a finite set of *candidate* input points for f.[1] Additionally, we require f to be computable and f being an algorithm that calculates $f(d)$ for a given input $d \in D_A$. However, we make no use of derivatives of f since we assume no analytical form for f but only access to selected points. For the swarm robotics scenario, D_A could be $\mathbb{R}^+$ representing the weight of the load in kg, where $cand$ offers a discretization as a grid with predefined accuracy δ and C_A could be $\mathbb{R}^+$ representing the *minimal* speed (or energy demand) of the swarm. As f typically represents optimal outputs (if, e.g., many configurations of robot teams can carry the load, we seek the *fastest*), f has to be an optimization algorithm possibly dealing with NP-hard combinatorial problems. We call $(f, cand, \mathsf{f})$ a *sampling problem specification*.

Our main idea is to successively train a probabilistic regression model $\hat{f}$ with input-output pairs $(d, f(d))$ and select points $d' \in D_A$ showing the highest uncertainty next, instead of randomly (or equidistantly) selecting points, assuming a fixed "budget" of s sampling points. We write $\sigma^2(\hat{f}(d))$ for the variance associated to the probabilistic output (μ, σ^2) of the regression model $\hat{f}$ (e.g., a Gaussian Process or a Decision Forest). We assume a set of initial points from D_A to be selected and solved in order to train the regression model, e.g., by taking a random set or distribute these points equidistantly over D_A.

Algorithm 1 presents a meta-algorithm that needs to be instantiated for the concrete problem instance. In particular, the definition of suitable optimization problems, algorithms (e.g., mathematical programming, constraint programming, or heuristic optimization), and regression models guides the design of the concrete implementation. While we illustrate the principle by picking the point with the highest uncertainty in line 9, other (perhaps information-theoretic) criteria, including mutual information, are possible. We demonstrate this mapping for an optimization problem faced in distributed energy management. Since that case study shows many of the aforementioned characteristics, it serves as the basis for our empirical evaluation and implementation in Sect. V.

[1] In the context of active learning, this set is also referred to as *pool*.

Algorithm 1 Active Learning for Sampling Abstraction

Require: $(f, cand, \mathsf{f})$ is a sampling problem specification
1: **procedure** SAMPLING-ABSTRACTION($(f, cand, \mathsf{f}), s$) ▷ select s points
2: $S \leftarrow \emptyset$ ▷ Set of sampling points
3: $I \leftarrow$ INITIALINPUTS($cand(D_A)$)
4: **for all** $i \in I$ **do**
5: $o = \mathsf{f}(i)$
6: $S \leftarrow S \cup \{(i, o)\}$
7: **loop** s times
8: $\hat{f} \leftarrow$ TRAINREGRESSIONMODEL(S)
9: $\hat{i} \leftarrow \text{argmax}_{d \in cand(D_A)} \sigma^2(\hat{f}(d))$
10: $\hat{o} = \mathsf{f}(\hat{i})$
11: $S \leftarrow S \cup \{(\hat{i}, \hat{o})\}$
12: **return** S

III. Hierarchical Distributed Energy Management

Future energy systems move from systems of relatively few centrally organized units providing most of the power demanded by consumers to many highly distributed units. This calls for manageable control mechanisms [7]. To deal with the high complexity in scheduling a set of power plants in the face of uncertainties introduced by nature and technical deficiencies, hierarchical organization structures based on virtual power plants can be employed [8], [9].

A. Autonomous Virtual Power Plants

In our vision of future energy management systems [8], inner nodes of such a hierarchy are called *Autonomous* Virtual Power Plants (AVPP) and act as *intermediaries* on behalf of their subordinates. They are, in fact, single nodes representing a whole *collective*. Power plants are thus structured into *systems of systems* represented by AVPPs, that can themselves take part in other AVPPs, as shown in Fig. 1. In light of the high dynamics involved, this structure can be adapted at runtime to, e.g., maintain a balanced ratio of controllable vs. stochastic power plants in each virtual power plant. A paramount goal in energy management is to balance supply and demand to avoid deviations of the mains frequency that result in possible device damages or even blackouts. The task is to assign *schedules* to controllable power plants such that their joint output meets the *residual load*, i.e., the difference between demand and supply from stochastic (e.g., weather-dependent) sources.

More precisely, the problem to be solved constitutes a hierarchical resource allocation problem where the resource to be allocated to a set of agents maps to their scheduled contributions. The goal is to meet a predicted demand over a scheduling window $\mathcal{W}$ consisting of a finite set of time steps with a fixed resolution (typically 15 minutes, the unit of energy markets). Agents have to act *proactively*, i.e., indeed need to create schedules, since they are subject to inertia and cannot be assumed to react fast enough in case of rapidly increasing (or decreasing) demand. The calculated schedules can be interpreted as a fixed-length sequence of actions in a planning framework. With regard to the case study, we derive the minimal set of constraints from the physical requirements

For Review Only

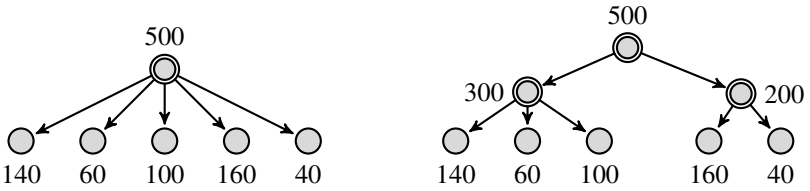


Fig. 1: A central and hierarchical solution example to a resource allocation problems. Inner nodes representing intermediaries are marked by double circles and redistribute their assigned share of the overall demand.

that power plants impose (see [10] for a discussion of the literature): i) minimal and maximal production, ii) discontinuity provided they can be switched off, and iii) functions limiting the possible change in production over a time step.

We can subsume minimal and maximal contributions as well as switching on and off by defining a sorted list of *feasible intervals* $L_a \subseteq \mathbb{R}$. A disconnectable power plant a providing in a range $0 < P_{\min} < P_{\max}$ would then be represented by $L_a = \langle [0,0], [P_{\min}, P_{\max}] \rangle$. It is imperative to be able to switch plants on and off selectively to achieve more favorable aggregate partial load behavior compared to a single conventional generator that suffers from increased costs when not operated at optimal energy conversion efficiency [11]. To allow planning for inertia in a, we define functions $\overrightarrow{P}_a^{\min}, \overrightarrow{P}_a^{\max} : \mathbb{R} \to \mathbb{R}$ that return the minimum and maximum contribution in the next time step given the momentary contribution. In the simplest case, we assume a constant maximal change ΔP:

$$\overrightarrow{P}_a^{\min}(c) \stackrel{\text{def}}{=} \max\{P_{\min}, c - \Delta P\} \quad (1)$$

$$\overrightarrow{P}_a^{\max}(c) \stackrel{\text{def}}{=} \min\{P_{\max}, c + \Delta P\} \quad (2)$$

Certainly, these functions can model richer systems, e.g., by considering a hot or cold start-up [10], as well as rates of change that map combinatorially to the underlying agents in case a itself is an intermediary. Additionally, cost functions κ_a return the minimal costs given a certain energy contribution.

We revisit the scheduling problem presented in [12] for some inner node—called intermediary λ—since the problem is solved top-down, as shown in Fig. 1.[2] With (piecewise) linearity restrictions regarding $\overrightarrow{P}_a^{\min}$, $\overrightarrow{P}_a^{\max}$, and κ_a, the problem is an instance of a mixed integer linear program (MILP). Given its assigned schedule $S_\lambda[t]$ at a given time step t, each intermediary redistributes it to its own subordinate agents A_λ until, eventually, schedules are assigned to all leaf agents, e.g., physical power plants. The allocation aims to both minimize deviations from the demand (Δ) as well as the incurred costs (Γ). Relative importance is specified by the weights α_Δ and α_Γ. The root node Λ is assigned the total demand of the environment, i.e., $S_\Lambda[t] = Env[t]$.

$$\begin{aligned} \underset{S_a[t]}{\text{minimize}} \quad & \alpha_\Delta \cdot \Delta + \alpha_\Gamma \cdot \Gamma \quad (3) \\ \text{subject to} \quad & \forall a \in A_\lambda, \forall t \in \mathcal{W}: \\ & \exists [x,y] \in L_a : x \le S_a[t] \le y, \\ & \overrightarrow{P}_a^{\min}(S_a[t-1]) \le S_a[t] \le \overrightarrow{P}_a^{\max}(S_a[t-1]) \\ & \text{with } \Delta = \textstyle\sum_{t \in \mathcal{W}} \left| S_\lambda[t] - \sum_{a \in A_\lambda} S_a[t] \right|, \\ & \text{and } \Gamma = \textstyle\sum_{t \in \mathcal{W}, a \in A_\lambda} \kappa_a(S_a[t]) \end{aligned}$$

[2]The overall central resource allocation problem is obtained if λ is the root AVPP Λ and all agents are directly controlled by it ($A_\Lambda = A$).

To achieve a reduction of complexity in the scheduling problem to be solved by the overall system, we proposed two approaches based on self-organization for hierarchical problem decomposition of the task:

- A so-called "regio-central" approach in which agents transfer formal control models to their intermediary which, at meso-level, centrally optimizes the allocation [6], [10].
- An auction-based decentralized approach [8] where agents need not submit their model but only bid on a given demand based on their private capabilities.

We borrow techniques from model abstraction [13], abstract interpretation [14], and approximation of hybrid systems [15] to obtain abstracted control models of the collectives. In particular, intermediaries approximate functional dependencies over the combinatorial production space stemming from the aggregate of their underlying agents. They do so by repeatedly sampling input-output pairs and substituting the actual functions by piecewise linear functions [6].

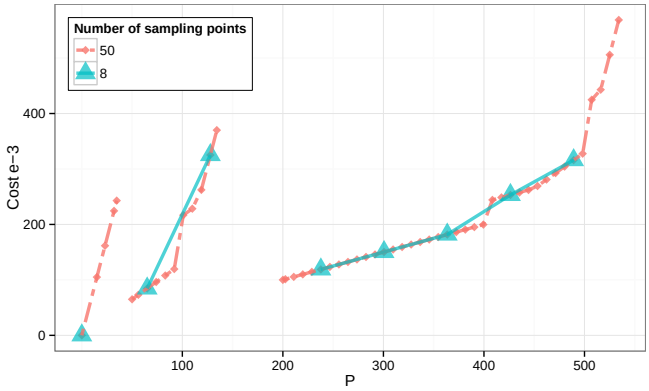
Obtaining a good abstraction of an intermediary's behavior as a compact representation of the underlying collective's combined behavior is desirable for both algorithms to make informed yet computationally efficient control decisions. In the regio-central case, one wants to simplify the generated optimization problems at meso-level by reducing the number of decision variables and constraints. In the auction-based algorithm, an intermediary could, in principle, have all agents bid simultaneously to a single auctioneer, i.e., use a super-flat hierarchy. Clearly, this auctioneer imposes a bottleneck with a rising number of agents. In a truly hierarchical setting, an intermediary ought to be aware of the physical boundaries of its subordinate agents *before* submitting bids in order to avoid inconsistencies that need to be (monetarily) punished by the organization [8].

As a simple illustration, consider that an intermediary better not bid for a contribution greater than 200 if it is comprised of two underlying agents with a maximal contribution of 100 each. Even if this trivial summation means no computational effort, additional constraints from limited rates of change and disconnectability make the problem harder. In fact, an intermediary needs to solve an optimization problem quite similar to Eq. (3) in order to calculate optimal bids for a given demand. Similarly to the "regio-central" approach, abstraction has further to be applied to obtain a simplified model of the intermediary that its own superior may use to compute bids.

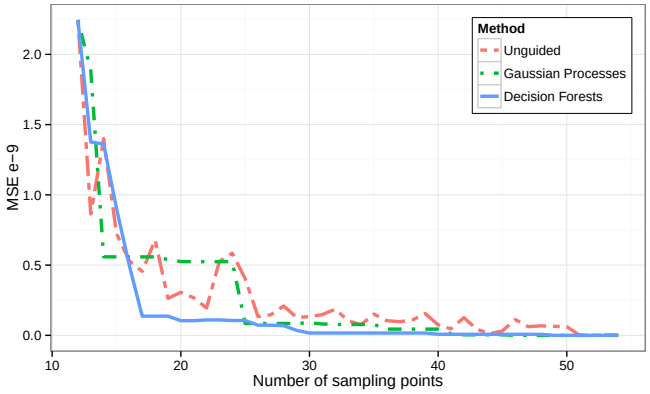
B. Obtaining Abstracted Models

Having motivated the need for abstraction techniques, we briefly revisit our existing approach [6] to discuss improve-

For Review Only



(a) Accuracy affected by the number of sampling points selected.



(b) MSE values for different sampling methods and point counts.

Fig. 2: Selecting the "right sampling points" is crucial for good accuracy. The cost function κ_v is only defined over the (discontinuous) domain $L_v = \langle [0,0], [15,35], [50,135], [200,535] \rangle$ [5].

ments using instances of the algorithm presented in Sect. II. An essential abstraction consists of finding the possible contributions of an intermediary by combining the lists of feasible intervals of its subordinate agents. Assume agent a_1 can contribute in $[x_1, y_1]$ and agent a_2 in $[x_2, y_2]$. Their combined contribution must be in $[x_1, y_1] \oplus [x_2, y_2] = [x_1 + x_2, y_1 + y_2]$. The operator $\oplus$ naturally extends to lists of intervals by combining in a Cartesian fashion and merging overlaps. Handling lists is in fact required by our application since power plants might have to contribute a positive amount $P_{\min}$ *if* they are switched on and 0 otherwise.

With regard to the optimization problem in (3), an intermediary λ controlling another intermediary λ' has to ask questions such as "What is the minimal cost for λ' to contribute x?" or "What is the maximal or minimal next contribution of λ' given the momentary contribution y"? Both questions provide examples for functions $f_A : D_A \to C_A$ that rely on the individual capabilities of agents in A. They are indeed required to take the places of κ_λ, $\overrightarrow{P}_\lambda^{\min}$, and $\overrightarrow{P}_\lambda^{\max}$ in the optimization problem the superordinate agent of λ has to solve. The domain D_A is given by the list of feasible production intervals for the agents in A, i.e., the collective represented by the intermediary. It is thus a subset of $\mathbb{R}$, and the codomain is given by $\mathbb{R}$ as well, representing costs or the maximal/minimal production in the next time step. Sampling these functions corresponds to solving optimization problems as generally there are several configurations of subordinates that achieve joint contribution x at different costs or maximal successor production.

We illustrate this algorithm detailed in [6] with a simplified exemplary intermediary v consisting of three agents $A_v = \{p, q, r\}$:

$$L_p = \langle [0,0], [50,100] \rangle, \qquad c_p = 13 \tag{4}$$
$$L_q = \langle [0,0], [15,35] \rangle, \qquad c_q = 70 \tag{5}$$
$$L_r = \langle [0,0], [200,400] \rangle, \qquad c_r = 5 \tag{6}$$

where c_a is the price per production unit such that the agents' cost functions are defined by: $\kappa_a(x) = c_a \cdot x$. We derive the feasible contribution ranges for v and get $L_v = \langle [0,0], [15,35], [50,135], [200,535] \rangle$. Inputs for the sampling points are selected from this contribution range which leads to a sequence of optimization problems to be solved in order to find the sampled outputs $\kappa_v(P)$ which denote the minimal costs to produce a given contribution P of the intermediary v:

$$\kappa_v(P) = \min_{S_p, S_q, S_r} \sum_{a \in A_v} \kappa_a(S_a) \text{ subject to } \sum_{a \in A_v} S_a = P \tag{7}$$

Concretely, we might consider a sequence $\langle \kappa_v(15), \kappa_v(25), \kappa_v(35), \kappa_v(50), \kappa_v(75), \ldots, \kappa_v(400) \rangle$ to collect sampling points that approximate the actual yet unknown minimal cost function of the collective A_v. More sampling points typically correlate with higher accuracy, as Fig. 2a shows. We instantiate $f : D_A \to C_A$ with $\kappa_v : L_v \to \mathbb{R}$, *cand* with a grid of user-defined accuracy, and f with CPLEX [16] solving the MILP corresponding to (7) in order to obtain a sampling problem specification. Similarly, we obtain specifications for the functions $\overrightarrow{P}_a^{\min}$ and $\overrightarrow{P}_a^{\max}$ by using a different objective. Other constraints, such as those restricting on/off settings and limited rates of change, are obtained by modifying the optimization problem shown in (3).

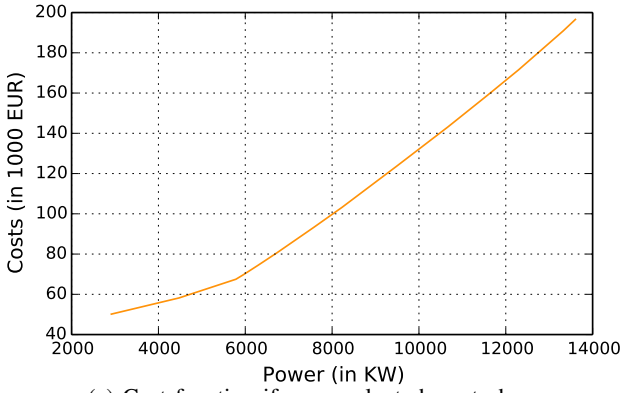
However, if these input-output pairs are selected in a weakly informative way, the resulting abstracted optimization problem introduces severe errors in quality. If, e.g., points are just sampled equidistantly without taking into account the actual shape of the function, many redundant points in linear segments are used despite the need for more information in other subsets of the domain. Figure 2b (Unguided) illustrates this situation for the previous example where 25 equidistantly selected sampling points lead to a worse mean squared error (MSE) (with respect to a validation set consisting of 50 actually sampled points) than about 17 sampling points at more informative positions.

The quality issue results from unfortunate interpolations since by equidistant probing the domain, "interesting" points that happen to be among, say, 25 equidistant points do not have to appear in the selected set for 40 points. In addition and as shown in Eq. (7), determining each sampling point comes at the expense of solving an optimization problem itself. Thus, it is desirable to avoid asking for uninformative points such as those between 200 and 400 in Fig. 2a.

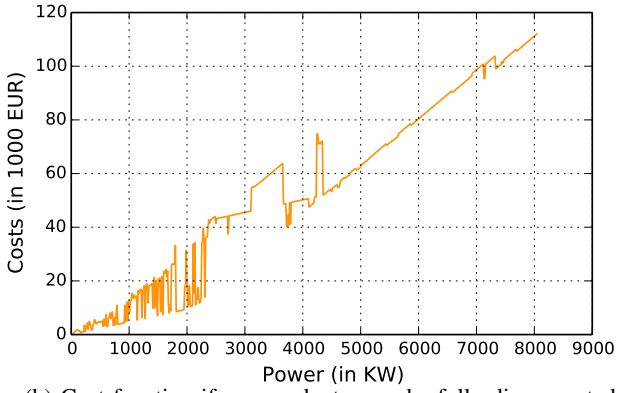
IV. Active Learning using Decision Forests

In the energy scenario, obtaining a single sampling point corresponds to solving a MILP. Having defined the sampling

For Review Only



(a) Cost function if power plants have to be on.



(b) Cost function if power plants may be fully disconnected.

Fig. 3: Influence of the ability to disconnect power plants on the resulting cost functions.

problem specification properly, we are left with the decision of what regression model and active learning technique to use for the concrete case. *Uncertainty sampling* is a straightforward active learning meta-algorithm, where one creates a predictive model with few initial sampling points and iteratively refines it with additional points in "interesting" areas (areas with high uncertainty, e.g., variance of the output). This technique can be applied with many regressors (e.g., Gaussian Processes (GPs) [17], [18], [19] or Decision Forests (DFs) [20]). A survey on active learning is presented in [4].

As mentioned in Sect. III-B, the ability to switch power plants on and off is paramount for the efficient operation of virtual power plants. It does, however, come at the expense of a highly nonlinear cost function in general, as Fig. 3 confirms with two cost functions of AVPPs based on our simulation presented in Sect. V. Additionally, Fig. 2a also showed possible discontinuities in the domain.

These characteristics revealed, in previous work [5], that training regression models based on DFs offer more effective active learning strategies in this scenario than those based on GPs (see Fig. 2b). The main reasons for this are twofold:

- GPs intuitively span a distribution over continuous functions with the kernel determining the shape and smoothness, making jumps and discontinuities hard to handle for them.
- Greedily selecting points with maximum variance corresponds to minimizing the entropy. This results in points being iteratively selected farthest away from already observed points, regardless of the true shape of the function [21].

As a consequence, DFs with linear models as leaf models [22] provide a better inductive bias for the given task: they quickly develop high confidence in areas of linear continuity and lower confidence further away from or with discontinuous of observed points. This closely corresponds to the aim of building a piecewise linear function from the sampled points. For a sampling problem specification $(f, cand, \mathrm{f})$, we obtain a probabilistic regression model and denote the probability distribution mapping D_A to C_A by $p(y \mid x)$. For an input $x \in D_A$, $p(y \mid x)$ returns the probability that $f(x) = y$, for instance mapping production to joint costs.

Each linear model at a tree leaf estimates $p(y \mid x)$ as

$$p(y \mid x) = \mathcal{N}\left(x^T b,\ s^2 x^T \left(X^T X\right)^{-1} x\right), \tag{8}$$

where b are the linear model parameters estimated by the samples reaching the leaf, s is their expected measurement error and X is their design matrix. The expected variance of this normal distribution is therefore sensitive to the distance to the nearest training data point as well as to the sample coherence at the leaf. Hence, it can provide good confidence estimates at points of discontinuity (c.t. [22, p. 58]). We noticed that averaging the estimated variances of the leafs of each tree of the forest produced the best confidence estimates for this setting.

We experimented with the GPy[3] implementation of GPs and the Fertilized Forest library of DFs [23]. The full code can be found online.[4] Each method started with 12 equally distributed sampling points to train an initial regressor. Then we incrementally added the point for which the learner's smoothed uncertainty estimate was highest and retrained the regressor. In Fig. 2b, we present the error traces of an unguided system using equidistant points and the system guided by active learning for different regressors. With only 20 sampling points, the system guided by DFs reaches the performance of the unguided system with 34 sampling points. A system based on active learning with GPs reaches the same performance with 25 points, but at more than double the runtime of the DF-guided system (approx. a factor of 2.24). The mean MSE (scaled with 10^{-9}) for the entire trace is 0.28 (unguided), 0.26 (GP-guided) and 0.18 (DF-guided).

Even in this preliminary, restricted setting, it was important not to start with "too few" points. Since the uncertainty quickly gets high close to nonlinear parts of the cost functions, uncertainty sampling will focus on these areas greedily, disregarding other areas and possibly ignoring other important nonlinear regions of the function. A "non-maximum suppression" strategy is required to address this issue, similar in nature to simulated annealing or tabu search in meta-heuristic optimization: For the first sampling points, selecting points in a large neighborhood of those already observed is prohibited. During later phases of the active learning process, points may

[3]Homepage: http://sheffieldml.github.io/GPy/.
[4]Experiment repository: http://git.io/buNW

For Review Only

be selected that are closer to already seen ones to find fine-grained non-linearities. This regularization is achieved by an adaptive window function, i.e., a factor that is multiplied with the average variance of the leaf nodes returned by the regressor.

We experimented with rectangular and Gaussian window functions, but found the rectangular windows to work significantly better. To determine its best shape, we parameterized the window function as

$$w(x, N, S, \alpha) = \begin{cases} 0 & \text{if} \min |x - s|_{s \in S} < d(N, S, \alpha), \\ 1 & \text{otherwise,} \end{cases} \quad (9)$$

where x is the evaluation point, N is the number of sampling points to be selected in total, S consists of the already chosen sampling points (cf. Alg. 1), and α is a vector of hyperparameters. The half width of the area to ignore is defined as

$$d(N, S, \alpha) = \max\left(1 - \frac{|S|}{N \cdot \alpha_1}, \alpha_2\right) \cdot \alpha_3. \quad (10)$$

This tolerance distance is by design decreasing with an increasing number of sample points, hence allowing for refinement of the sampling around discontinuities. At the same time, it has a minimum width α_2 and is linearly scaled with slope α_3. We optimized these hyperparameters by using a Bayesian optimization framework [24] on a separate, artificially generated training and validation dataset consisting of 1000 points (based on the same testbed described in Sect. V) where we did a sampling from 46 initial up to 200 (N) points in total. Even though the sampling point numbers are different from the values we use for our experiments in the following section (using more sampling points), they work reasonably well. This suggests that the model is sufficiently robust to the application scenario.

As evaluation measure for the search of optimal hyperparameters α with a fixed amount of pre-selected sampling points, we used the standard MSE (we build the MSE as the difference of the piecewise linear function built from k sampled points to a fine-grained function with the full 1000 points). However, the performance of the system for all k values from 46 to 200 must somehow be integrated. This is why we used the mean MSE (MMSE) to compare overall system performance.

V. Evaluation

To investigate our claims that active learning techniques enhance the accuracy and efficiency of existing abstraction algorithms empirically, we evaluated them on a simulated system for the case study presented in Sect. III, following-up our machine learning evaluation in the previous section. Power plant models were built from freely available data [25]. We obtained the optimal solutions as benchmarks by solving the instances centrally. These solutions were compared to the solutions found by a hierarchical approach in terms of costs as well as runtime.

To be precise, we consider biofuel, hydro, and gas plants in the region of Swabia in Bavaria. Nameplate capacities, i.e., maximal productions are drawn from a distribution according to this data. Minimal productions depend on the type of plant and are given as percentage of the nameplate capacity. Similarly, maximal rates of change per minute (also as percentages of the nameplate capacity) and costs per kWh are selected based on the type and taken from [26] and [27], respectively. We increased the cost heterogeneity present in the literature slightly in order to provide a scenario where active learning is actually able to provide quality benefits. To sum up, our data is based on the quantities in Table I.

TABLE I: Initially assumed distribution of input data including rates of change and costs ([26] and [27]). We list minimal and maximal bounds for the maximal production. Standard deviations for cost distribution are given in parentheses.

Type	Rel. Frequency	Max. Power	Min. Power	Rate of Change	Costs
	[%]	[KW]	[%]	[%/min]	[€ / KW]
Biofuel	54	[3.0, 17374.0]	35	6	2.3 (0.7)
Hydro	43	[2.0, 7800.0]	0	50	0.9 (0.2)
Gas	3	[1.0, 2070.0]	20	20	2.4 (1.0)

The available statistical data gives rise to a generative mixture model from which we can sample realistic power plants in our simulation. We first draw the type of plant and then the maximal production (given the type) within the bounds listed in Table I ($\mathcal{N}(368.72, 1324.62)$ for biofuel, $\mathcal{N}(264.63, 746.55)$ for hydro, and $\mathcal{N}(275.88, 494.80)$ for gas). Rates of change are added based on type and maximal production and costs are drawn from another normal random variable. The demand, i.e., the residual load, to be fulfilled by a set of power plants, is based on consumer data of [28] and scaled such that the peak loads map to 110% of the drawn plants' combined maximal productions – in order to have a representative load test for the system. Regarding Eq. (3), we fix α_Δ to a high value[5] to prioritize the goal of meeting the demand but still minimize costs among all balanced schedules.

Upon drawing a set of power plants, hierarchies are created similar to a B+ tree, i.e., only AVPPs at the lowest level control physical power plants. The hierarchies' shapes, i.e., depth and width, are controlled by restricting the maximal number of physical power plants per AVPP at the leaf level and the maximal number of directly subordinate AVPPs at the inner node levels. First, the "leaf" AVPPs are created by taking a random permutation of the physical plants and picking clusters of the maximal physical plant count. Then, new hierarchy levels in the form of intermediate AVPPs, i.e., inner nodes, are introduced when needed.

We export the optimization problems (both hierarchical ones at AVPP-level or central ones) as mixed integer linear programs that are solved by IBM ILOG CPLEX [16]. Given the same power plants and initial states, the problem is solved for a period of a quarter day (i.e., 24 time steps in 15 minutes intervals) both centrally and hierarchically — called a *run*. We obtain solutions for *identical* problems by taking care of random seeds and reproducibility. Consequently, our experiments follow the basic structure:

1) Draw n power plants
2) Load consumer data for a quarter day ($t = 24$ steps)
3) Perform abstraction steps – select equidistantly (*EQ*) or guided by active learning (*AL*)

[5]The value of α_Δ effectively acts as a "market price" since violations would have to be compensated by buying additional energy.

For Review Only

4) Solve the resource allocation t times hierarchically
5) Solve the resource allocation t times centrally as benchmark
6) Repeat k times ($k = 5$, resulting in 120 time steps in total)

Each experiment consists therefore of one *preprocessing* phase where AVPPs are formed and abstractions are calculated, and a *productive* phase where the demand is distributed using the abstracted models. In the productive phase, no structure changes are allowed to better isolate the effects of the chosen sampling point selection strategy (*EQ* or *AL*).

For our considerations, we fixed the system structure to consist of 300 power plants (with a control experiment of 400 plants), leaf AVPPs to control 20 power plants and inner AVPPs to be in charge of 8 sub-AVPPs. This results in 31 AVPPs in total. The experiment suite and full source code including an instruction on how to run the experiments and the python scripts used to generate the analyses and plots in this paper can be found online at (http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/staff/aschiendorfer/). The simulation and abstraction algorithms were developed in Java and the active learning algorithm in Python. Each presented experiment was run on a machine having 4 Intel Xeon CPU 3.20 GHz cores and 14.7 GB RAM on a 64 bit Windows 7 OS with 8 GB RAM offered to the Java 7 JVM running the abstractions as well as CPLEX.

All central models used for comparison were solved with a 30 minutes time limit per time step. When planning for 15 minute intervals, a solution must be available much earlier. We still wanted to collect optimal solutions as benchmarks and therefore allowed twice that period for the central solver. We now examine questions of interest and present the results of the experiment runs in the following sections.

A. Abstraction Quality

Does the selection strategy affect the quality of the obtained solutions in terms of costs for energy? We expected that guidance of the sampling point selection by means of active learning improves the solution quality, i.e., reduces the costs required to jointly provide the required energy. Indeed, in most cases, we can observe this fact, as Fig. 4 shows. Table II further presents the numerical values of the overhead costs achieved by both methods. Interestingly, the active-learning-based strategy needs a growing number of sampling points to strongly outperform the equidistant default selection due to the fact that a significant training set is required to fit the regressor. Hence, with 30 initial sampling points, statistically significant differences (with a student-t-test and significance level $\alpha = 0.05$) were measured. However, as shown for 30 initial and 30 additional sampling points, active learning can be slightly (albeit not significantly) worse than equidistant selection due to various random factors. For 20 initial sampling points, we achieve comparable average costs to 40 equidistantly additionally chosen points (887.44 €) already with 20 additional actively chosen points (898.96 €). With 30 initial and 40 additional sampling points, we can reduce the average overhead costs from 845.51 € to 560.06 € – yielding a 33.7% reduction in costs.

It is also noteworthy that the standard deviations are twice as large when using equidistant sampling, making the active learning approach much more robust to abstraction errors. We may conclude that, in terms of solution quality, informed selection is advantageous even with fewer sampling points. However, it is clear that we must pay a price in terms of runtime to achieve these more informative points. The next section aims to quantify these overhead runtimes.

B. Fixed Runtime Effort

Does active learning impede the abstraction process through time-consuming deliberation? We expected that the abstraction is not strongly affected by the active learning overhead. With respect to Fig. 5, we admittedly notice that active learning indeed requires us to pay a price upfront for the improved model quality. Even little sampling point numbers require about 60-70 seconds of abstraction runtime per AVPP, similar to runtimes achieved by sampling the most points using the equidistant strategy in Fig. 5a. For the highest number of sampling points with active learning (and thus the least overhead costs), we already have to invest about twice as much time (about 108 seconds). In the presented experiments, an offset in runtime of around 20 additional seconds per AVPP (for both 300 and 400 plants) can be observed. However, we again see a clear tendency of improved overall costs when using active learning. In particular, Fig. 5a illustrates that active learning performs significantly better than equidistant sampling when 60 to 80 seconds are allocated for abstraction – even if *fewer* sampling points are selected. While we parameterized our experiments in terms of the number of sampling points to allow for better comparison, a more realistic setting would be to limit the abstraction time. Then, active learning outperforms equidistant sampling even with fewer points.

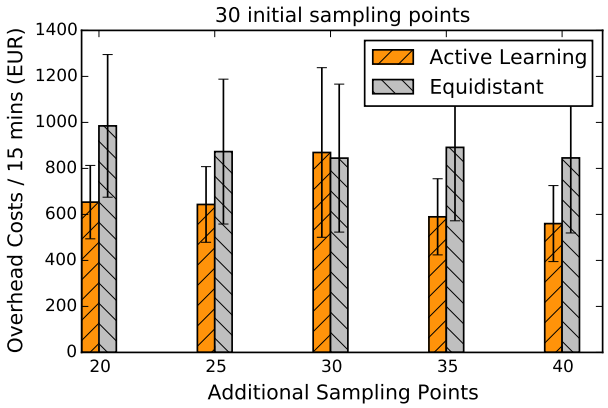
Table III contains the full list of average runtimes, both for the whole simulation run (a quarter day) and for the fixed abstraction times required upfront. Besides the rather obvious fact that runtimes increase with the number of sampling points, we can see that better sampling point selection improves the *overall* runtime (including abstraction efforts) at better costs: With 20 initial sampling points we need 2979 seconds on average with 25 actively selected sampling points to achieve 775.55 € overhead, whereas 40 equidistantly selected points take 3142 seconds for 887.44 € overhead (compare Tables II and III).

Concluding, we find that the training of Decision Forests embedded in the selection strategy does affect the runtimes noticeably. Nevertheless, the informative points pay off in terms of solution quality, as evidenced by the previous section. Moreover, when granting a fixed amount of abstraction time, one is better off using the actively selected points. Finally, upon calculating the abstractions using sampling, we may wonder if the runtime efforts incurred at each time step do increase due to perhaps higher model complexity.

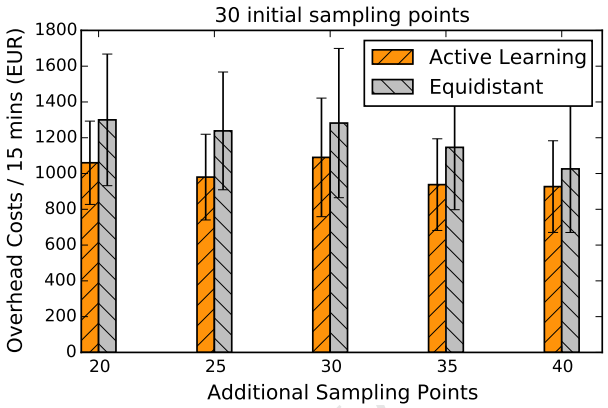
C. Variable Runtime Effort

Does the choice of sampling points affect the runtimes for each subsequent time step? We expected that, once the models are abstracted, no significant change in runtime per step emerges, or, perhaps worse, the increasingly complex

Not Reviewed Copy



(a) Difference in costs for 300 plants.

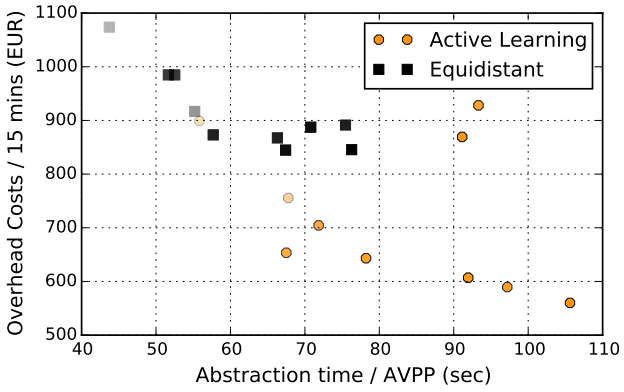


(b) Difference in costs for 400 plants.

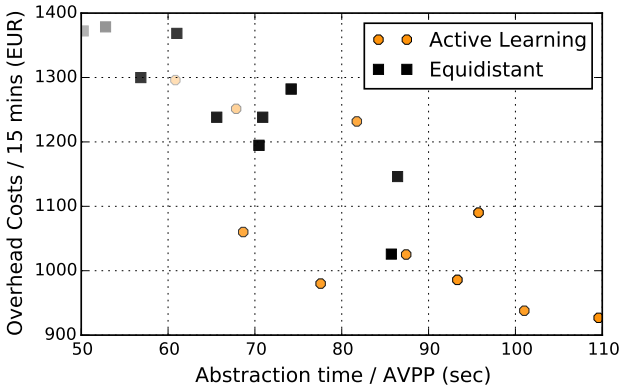
Fig. 4: Average overhead costs per time step w.r.t. the optimal solution. Error bars show $\frac{\sigma}{6}$ for visualization purposes since the standard deviation σ itself is very high due to the variations in demand directly influencing costs of the individual time steps.

TABLE II: Quality results for various configurations of initial and additional sampling points for 300 plants. Standard deviations are given in parentheses.

	Costs EQ	Costs AL	Diff	Significance
20 initial sampling points				
20 sps	1073.8 (1860.64)	898.96 (881.05)	−174.83 (1218.77)	No ($t = 1.56$; $p = 0.12$)
25 sps	916.83 (1882.01)	755.55 (989.84)	−161.28 (1203.79)	No ($t = 1.46$; $p = 0.15$)
30 sps	984.87 (1860.49)	704.46 (995.20)	−280.41 (1152.05)	Yes ($t = 2.66$; $p = 0.01$)
35 sps	867.49 (1918.55)	928.11 (2309.38)	60.62 (1163.94)	No ($t = -0.57$; $p = 0.57$)
40 sps	887.44 (1917.07)	606.91 (1024.06)	−280.53 (1146.74)	Yes ($t = 2.67$; $p = 0.01$)
30 initial sampling points				
20 sps	984.87 (1860.49)	653.42 (957.48)	−331.45 (1175.37)	Yes ($t = 3.08$; $p = 0.00$)
25 sps	873.08 (1889.63)	643.27 (986.49)	−229.81 (1143.48)	Yes ($t = 2.19$; $p = 0.03$)
30 sps	844.76 (1930.11)	869.25 (2212.01)	24.49 (1049.63)	No ($t = -0.25$; $p = 0.80$)
35 sps	891.42 (1915.39)	589.52 (992.98)	−301.90 (1135.65)	Yes ($t = 2.90$; $p = 0.00$)
40 sps	845.51 (1957.19)	560.06 (991.44)	−285.45 (1164.30)	Yes ($t = 2.67$; $p = 0.01$)



(a) Experiment with 300 plants.



(b) Experiment with 400 plants.

Fig. 5: Tradeoff between invested runtime and quality. Darker points indicate a higher number of sampling points (initial and additional combined). Corresponds to the abstraction time values in Table III but presents the times per AVPP.

For Review Only

TABLE III: Runtimes for simulating a quarter day and abstraction efforts. Standard deviations are given in parentheses.

	Total Time EQ	Total Time AL	Abstraction Time EQ	Abstraction Time AL
20 initial sampling points				
20 sps	2181.99 (165.79)	2460.78 (150.06)	1355.34 (28.45)	1730.41 (50.00)
25 sps	2483.51 (84.42)	2979.51 (426.69)	1711.35 (48.01)	2101.13 (67.87)
30 sps	2304.65 (98.41)	2947.73 (140.48)	1601.32 (42.88)	2227.27 (59.45)
35 sps	2819.65 (93.74)	3647.17 (86.88)	2055.70 (45.20)	2892.96 (54.41)
40 sps	3142.09 (321.27)	3598.95 (257.23)	2194.40 (238.75)	2850.09 (156.61)
30 initial sampling points				
20 sps	2342.89 (94.84)	2791.66 (120.83)	1628.04 (42.24)	2092.63 (78.02)
25 sps	2509.33 (75.46)	3144.84 (126.00)	1788.27 (62.44)	2424.68 (78.17)
30 sps	2903.71 (80.94)	3698.65 (172.04)	2089.65 (43.09)	2824.97 (81.40)
35 sps	3118.51 (57.10)	3774.79 (147.24)	2339.20 (71.21)	3013.06 (116.21)
40 sps	3198.61 (110.85)	4059.65 (115.01)	2364.93 (65.96)	3274.18 (121.69)

TABLE IV: Runtimes per step and the longest serial path (lsp) times for equidistant and AL-based sampling. The longest serial path is the lower bound on the runtime achievable by perfect parallelization. Standard deviations are given in parentheses.

	Time/Step EQ	Time/Step AL	LSP EQ	LSP AL	Significant (LSP)
20 initial sampling points					
20 sps	34.44 (11.90)	30.43 (8.02)	12.22 (10.82)	9.76 (6.15)	Yes ($t = 2.18$; $p = 0.03$)
25 sps	32.17 (4.12)	36.60 (21.78)	10.76 (3.85)	15.91 (18.81)	Yes ($t = -2.88$; $p = 0.00$)
30 sps	29.31 (6.65)	30.02 (5.14)	9.63 (5.91)	9.60 (2.65)	No ($t = 0.05$; $p = 0.96$)
35 sps	31.83 (4.37)	31.43 (3.03)	11.01 (3.27)	10.59 (2.36)	No ($t = 1.29$; $p = 0.20$)
40 sps	39.49 (12.46)	31.20 (5.05)	14.92 (10.9)	10.66 (2.79)	Yes ($t = 4.22$; $p = 0.00$)
30 initial sampling points					
20 sps	29.79 (6.68)	29.13 (3.67)	9.83 (5.99)	9.42 (2.66)	No ($t = 0.68$; $p = 0.50$)
25 sps	30.04 (6.13)	30.01 (3.89)	10.53 (5.86)	10.03 (2.46)	No ($t = 0.87$; $p = 0.38$)
30 sps	33.92 (8.57)	36.40 (11.39)	12.14 (7.70)	14.34 (10.75)	No ($t = -1.78$; $p = 0.08$)
35 sps	32.47 (8.65)	31.74 (6.66)	12.02 (7.94)	11.90 (6.03)	No ($t = 0.13$; $p = 0.89$)
40 sps	34.74 (9.49)	32.73 (6.18)	13.66 (8.88)	12.41 (5.55)	No ($t = 1.39$; $p = 0.17$)
Central solution	**214.89** (553.40)				

piecewise linear functions complicate the intermediaries' optimization problems (due to being less "abstract" and thus more complicated to solve). But to our surprise, this was not actually the case. Selecting sampling points actively tends to even *reduce* the runtimes per time step, as Table IV shows. With 30 initial sampling points, the differences in runtime become, however, not significant. Nonetheless, once we have selected the informative points facing higher initial runtime efforts (as evidenced in the previous section), we do not have to expect higher runtime costs per time steps while obtaining more accurate control models (as shown in Sect. V-A). Consequently, the longer the system runs with stable structures, the higher the benefit in quality at the same runtimes. Compared to the central solution that takes 215 seconds per step on average, even the sequential executions taking around 30 seconds show a tremendous speedup, even more so if the longest serial path times with about 13 seconds are considered. The solutions obtained by the hierarchical approach are nevertheless of high quality, ranging between 1.7% (equidistant, 20 initial and 20 additional) and 0.9% (active learning, 30 initial and 30 additional) above the optimum.

VI. Conclusion

We presented a meta-algorithm to create abstract input-output control models for collectives that relies on sampling using discrete optimization and active learning for the selection of informative sampled points instead of merely adding more points in equal steps. In the context of distributed energy management, we instantiated this framework with MILP-formulations of the sampling problems and Decision Forests with uncertainty sampling as active learning strategy – following previous evaluations [5].

For the problem of balancing supply and demand cost-effectively in a hierarchy of virtual power plants, we could show significantly improved abstracted models by using active learning based on Decision Forests. These findings confirm our "pure" machine learning evaluations on previously collected datasets [5]. With the same number of sampling points, in almost all considered cases, significant cost reductions could be achieved. We learned that higher time investment upfront was typically required to obtain the more informative sampling points but no higher time commitment resulted in the productive period of the system.

Our case study presents a first implementation of this meta-algorithm but our results may also stipulate future research in similar areas. This could (a) continue in improving the existing active learning approach with Decision Forests or (b) enhancing it with additional strategies. Analyzing the selection traces of the method, it becomes clear that the uncertainty sampling based on the probabilistic linear models is highly capable of improving the localization of nonlinearities in a sampled function. The uncertainty is (and remains) very high at these points, so they can be well identified, and the additional sampling improves the exact coordinates of nonlinear 'jumps'. However, this approach makes use of prior knowledge

For Review Only

regarding discontinuities or, generally, the kind of the sampled functions. That way, the method reduces to a way to improve the fine-grained localization of nonlinearities, but does not take into account the general shape of the function. Looking at the cost functions of the collectives, it becomes obvious that they belong to several 'classes' and do have regions in which they must be sampled more or less dense. This originates from the fact that the resulting combined functions strongly depend on the types and characteristics of subordinate agents. Even if this structure is not perfectly known, there might be a minimum number of sample points from which it could be possible to determine a sampling strategy suited for the class of function.

Moreover, we saw in Sect. V-B that the runtime for the fixed sampling abstraction increased due to the re-training of Decision Forests for every sampling point. Therefore, one could investigate *online learning* models, e.g., extensions of Decision Forests that can be trained incrementally at runtime. Other selection criteria such as maximizing expected mutual information could be explored. Depending on the concrete sampling problem specifications, one could exploit properties of the sampled functions such as *submodularity* to derive specialized active learning algorithms [21]. Finally, the effectiveness of the approach for other hierarchical control or planning systems based on a self-organizing structure such as, e.g., a swarm of robots has yet to be investigated.

REFERENCES

[1] S. Frey, A. Diaconescu, D. Menga, and I. Demeure, "A Generic Holonic Control Architecture for Heterogeneous Multi-Scale and Multi-Objective Smart Micro-Grids," *ACM Trans. Auton. Adapt. Syst.*, 2015.

[2] A. Girard and G. J. Pappas, "Hierarchical Control System Design using Approximate Simulation," *Automatica*, vol. 45, no. 2, pp. 566–571, 2009.

[3] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory & Practice*. Elsevier, 2004.

[4] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010. [Online]. Available: http://burrsettles.com/pub/settles.activelearning.pdf

[5] A. Schiendorfer, C. Lassner, G. Anders, R. Lienhart, and W. Reif, "Active Learning for Abstract Models of Collectives," in *Proc. 3rd Int. Wsh. "Self-optimisation in Organic and Autonomic Computing Systems" (SAOS15)*, March 2015.

[6] A. Schiendorfer, J.-P. Steghöfer, and W. Reif, "Synthesis and Abstraction of Constraint Models for Hierarchical Resource Allocation Problems," in *Proc. 6th Int. Conf. Agents and Artificial Intelligence (ICAART'14), Vol. 2*. SciTePress, 2014, pp. 15–27.

[7] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings, "Putting the 'Smarts' Into the Smart Grid: A Grand Challenge for Artificial Intelligence," *Commun. ACM*, vol. 55, no. 4, Apr. 2012.

[8] G. Anders, A. Schiendorfer, F. Siefert, J.-P. Steghöfer, and W. Reif, "Cooperative Resource Allocation in Open Systems of Systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 10, no. 2:11, May 2015.

[9] A. Nieße, S. Beer, J. Bremer, C. Hinrichs, O. Lunsdorf, and M. Sonnenschein, "Conjoint Dynamic Aggregation and Scheduling Methods for Dynamic Virtual Power Plants," in *Proc. 3rd Int. Wsh. Smart Energy Networks & Multi-Agent Systems (SEN-MAS'14)*, 2014, pp. 1505–1514.

[10] A. Schiendorfer, J.-P. Steghöfer, and W. Reif, "Synthesised Constraint Models for Distributed Energy Management," in *Proc. 3rd Int. Wsh. Smart Energy Networks & Multi-Agent Systems (SEN-MAS'14)*, 2014, pp. 1529 – 1538.

[11] J. Karl, *Dezentrale Energiesysteme: Neue Technologien im liberalisierten Energiemarkt*. Oldenbourg, 2012, in German.

[12] G. Anders, A. Schiendorfer, J.-P. Steghöfer, and W. Reif, "Robust Scheduling in a Self-Organizing Hierarchy of Autonomous Virtual Power Plants," in *Proc. 2nd Int. Wsh. Self-optimisation in Organic and Autonomic Computing Systems (SAOS'14)*, W. Stechele and T. Wild, Eds., 2014, pp. 1–8.

[13] F. Frantz, "A Taxonomy of Model Abstraction Techniques," in *Simulation Conference Proceedings, 1995. Winter*, 1995, pp. 1413–1420.

[14] P. Cousot, "Abstract Interpretation," *ACM Comput. Surv.*, vol. 28, no. 2, pp. 324–328, Jun. 1996.

[15] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Algorithmic Analysis of Nonlinear Hybrid Systems," *IEEE Trans. Automatic Control*, vol. 43, no. 4, pp. 540–554, 1998.

[16] CPLEX, "IBM ILOG CPLEX Optimizer," Dec. 2013, online Resource, last accessed December 2013: `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

[17] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian Process Regression: Active Data Selection and Test Point Rejection," in *Proc. Int. Jnt. Conf. Neural Networks (IJCNN'00)*, vol. 3, 2000, pp. 241–246.

[18] A. Krause and C. Guestrin, "Nonmyopic Active Learning of Gaussian Processes: An Exploration-Exploitation Approach," in *Proc. 24th Int. Conf. Machine Learning (ICML'07)*. ACM, 2007, pp. 449–456.

[19] M. Park, G. Horwitz, and J. W. Pillow, "Active learning of neural response functions with Gaussian processes," in *Advances in Neural Information Processing Systems (NIPS)*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2043–2051.

[20] J. E. Iglesias, E. Konukoglu, A. Montillo, Z. Tu, and A. Criminisi, "Combining Generative and Discriminative Models for Semantic Segmentation of CT Scans via Active Learning," in *Proc. 22nd Inf. Conf. Information Processing in Medical Imaging (IPMI)*, 2011.

[21] A. Krause, A. Singh, and C. Guestrin, "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, Jun. 2008.

[22] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," Microsoft Research, Tech. Rep. MSR-TR-2011-114, Oct 2011. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=155552

[23] C. Lassner and R. Lienhart, "Norm-induced entropies for decision forests," in *Proc. IEEE Winter Conference on Applications of Computer Vision 2015 (WACV'15)*, 2015. [Online]. Available: http://www.fertilized-forests.org

[24] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, 2012. [Online]. Available: https://github.com/JasperSnoek/spearmint

[25] Deutsche Gesellschaft für Sonnenenergie e.V., "Energymap," Dec. 2013, online Resource, last accessed December 2013: `http://www.energymap.info/`.

[26] M. Hundt, R. Barth, N. Sun, S. Wissel, and A. Voß, "Verträglichkeit von erneuerbaren Energien und Kernenergie im Erzeugungsportfolio," *Technisch-ökonomische Aspekte. Studie des Instituts für Energiewirtschaft und rationelle Energieanwendung (IER) im Auftrag der E. ON Energie AG. Stuttgart*, 2009, in German.

[27] C. Kost, J. N. Mayer, J. Thomsen, N. Hartmann, C. Senkpiel, S. Philipps, S. Nold, S. Lude, N. Saad, and T. Schlegl, "Levelized Cost of Electricity- Renewable Energy Technologies," *Techno-Economic Assessment of Energy Technologies, Fraunhofer ISE*, 2013.

[28] LEW Verteilnetz GmbH, "LEW Netzdaten," Dec. 2013, online Resource, last accessed December 2013: `http://www.lew-verteilnetz.de/`.

For Review Only