Dieses Dokument ist eine Zweitveröffentlichung (Postprint) /

This is a self-archiving document (accepted version):

Lars Dannecker, Robert Schulze, Matthias Böhm, Wolfgang Lehner, Gregor Hackenbroich

Context-Aware Parameter Estimation for Forecast Models in the Energy Domain

Erstveröffentlichung in / First published in:

Scientific and Statistical Database Management: 23rd International Conference. Portland, 20.-22.07.2011. Springer, S. 491–508. ISBN 978-3-642-22351-8.

DOI: http://dx.doi.org/10.1007/978-3-642-22351-8_33

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-830568

SLUB Wir führen Wissen.

TECHNISCHE UNIVERSITÄT DRESDEN

QUCOSA Quality Content of Saxony

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

Context-Aware Parameter Estimation for Forecast Models in the Energy Domain

Lars Dannecker[1], Robert Schulze[1], Matthias Böhm[2], Wolfgang Lehner[2], and Gregor Hackenbroich[1]

[1] SAP Research Dresden, Chemnitzer Str. 48, 01187 Dresden, Germany
{lars.dannecker,robert.schulze}@sap.com
[2] Technische Universität Dresden, Database Technology Group
Nöthnitzer Str. 46, 01187 Dresden, Germany
{matthias.boehm,wolfgang.lehner}@tu-dresden.de

Abstract. Continuous balancing of energy demand and supply is a fundamental prerequisite for the stability and efficiency of energy grids. This balancing task requires accurate forecasts of future electricity consumption and production at any point in time. For this purpose, database systems need to be able to rapidly process forecasting queries and to provide accurate results in short time frames. However, time series from the electricity domain pose the challenge that measurements are constantly appended to the time series. Using a naive maintenance approach for such evolving time series would mean a re-estimation of the employed mathematical forecast model from scratch for each new measurement, which is very time consuming. We speed-up the forecast model maintenance by exploiting the particularities of electricity time series to reuse previously employed forecast models and their parameter combinations. These parameter combinations and information about the context in which they were valid are stored in a repository. We compare the current context with contexts from the repository to retrieve parameter combinations that were valid in similar contexts as starting points for further optimization. An evaluation shows that our approach improves the maintenance process especially for complex models by providing more accurate forecasts in less time than comparable estimation methods.

Keywords: Forecasting, Energy, Maintenance, Parameter Estimation.

1 Introduction

In the energy domain, the balancing of energy demand and supply is of utmost importance. Especially, the integration of more renewable energy sources (RES) poses additional requirements to this balancing task. The reason is that RES highly depend on exogenous influences (e.g., weather) and thus, their energy output cannot be planned like traditional energy sources. In addition, RES cannot be stored efficiently and must be used when available.

Several research projects such as MIRACLE [1], and MeRegio [2] address the issues of real-time energy balancing and improved utilization of RES. Current

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

approaches in this area have in common that they require accurate predictions of energy demand and energy supply from RES at each point in time. For this purpose, mathematical models so called forecast models are used to model the behavior and characteristics of historic energy demand and supply time series. The most important classes of forecast models are: autoregressive models [3], exponential smoothing models [4] and models that apply machine learning [5]. Forecast models from all classes employ several parameters to express different aspects of the time series such as seasonal patterns or the current energy output. To exactly describe the past behavior of the time series, these parameters are estimated on a training data set by minimizing the forecast error (i.e., the difference between actual and predicted values) that is measured in terms of an error metric like the Mean Square Error (MSE) [3] or the (Symmetric) Mean Average Percentage Error ((S)MAPE) [6]. The so created forecast model instances are used to predict future values up to a defined horizon (e.g., one day). To allow efficient forecast calculations as well as the transparent reuse of forecast models, forecasting increasingly gains direct support by database systems. Besides research prototypes such as the Fa system [7] or the Forecast Model Index [8], forecasting has also been integrated into commercial products like Oracle OLAP DML [9] or Microsoft SQL Server Data Mining Extension [10]. However, the forecasting process remains inherently expensive, due to a large number of simulations involved in the parameter estimation process that can increase exponentially with the number of parameters when using a naive estimation approach.

Todays optimization algorithms used for parameter estimation can be divided into two classes: (1) Algorithms that need derivable objective functions and (2) algorithms that can be used with arbitrary objective functions. We focus on algorithms of the second class, because they can be used with any forecast models and error metrics and are hence more general. We can further classify algorithms of this class into local and global optimization algorithms. Global optimization algorithms such as Simulated Annealing [11] consider the whole solution space to find the globally optimal solution at the cost of slow convergence speed. In contrast, local optimization algorithms such as Nelder-Mead simplex search [12] follow a directed approach and converge faster at the risk of starving into local optima. These algorithms also highly depend on the provision of good starting points. Due to the limitations of local and global optimization algorithms, we enhance the parameter estimation by an approach that exploits knowledge about the time series context to store and reuse prior parameter combinations as starting points. This paper makes the following contributions:

First, we outline our general forecast model maintenance approach in Section 2.
Second, we introduce our Context-Aware Forecast Model Repository (CFMR) and define basic operations to preserve old parameter combinations in Section 3.
Third, we describe how to retrieve parameter combinations efficiently from the CFMR and how to revise them to yield the final parameters in Section 4.
Fourth, we evaluate our approach and demonstrate its advantages over comparable parameter estimation strategies in Section 5.
Finally, we present related work in Section 6 and conclude the paper in Section 7.

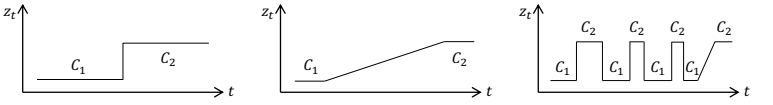
Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

2 Context-Aware Forecast Model Maintenance

The core idea underlying our approach is to store previously used forecast models and in particular their parameter combinations in conjunction with information about the time series context during which they were valid (i.e., produced accurate forecasts) into a *Context-Aware Forecast Model Repository* (CFMR). Assuming that similar contexts lead to similar model parameters, we retrieve beneficial starting values for further optimization from the repository by comparing the current time series context with previous contexts stored in the CFMR.

The term *context* has been coined in machine learning to describe the conglomerate of background processes and influence factors, which drives the temporal development of data streams [13]. Regarding electricity demand and supply time series, we identify influences from meteorological (e.g., weather), calendar (e.g., seasonal cycles, public holidays) and economic (e.g., local law) factors. While each of these factors takes an individual state at each point in time, they form in their entirety a specific time series context. Since electricity demand and supply time series are regularly updated with new measurements, the state of the influence factors changes over time, and hence the entire context. Such *context drifts* can modify the behavior and characteristics of the subjacent time series in unanticipated ways. We adopt the classification of Zliobaite [14] and distinguish three types of context drift based on their duration and number of re-occurrences. Figure 1 illustrates the different types of context drifts:



(a) Abrupt Drift. (b) Persistent Drift. (c) Cyclic Drift.

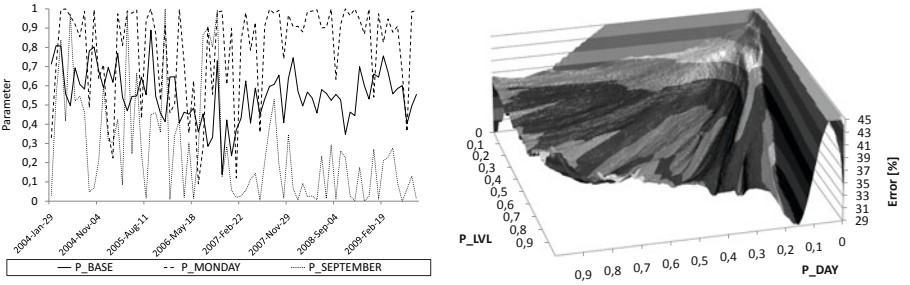
Fig. 1. Different Types of Context Drift

Abrupt Drift. A new context abruptly replaces the current context and causes a disruptive change within a short time frame. Example: power outages.
Persistent Drift. The old context slowly transforms into a new context that establishes permanently. Examples: gradual production changes, wind turbine aging.
Cyclic Drift. Old contexts re-appear and alternate. Examples: seasonal patterns (daily, weekly, yearly season), public holidays.

Context drifts can decrease the forecast accuracy, if the forecast model is not able to reflect the new situation and adapt to the changed time series characteristics. The reason is that changing contexts often lead to changing optimal forecast model parameters. Figure 2(a) illustrates the development of such optimal values for three example parameters of the electricity-tailored forecast model *EGRV* [15]. It can be seen that the optimal parameters greatly fluctuate over time, which we ascribe to context drift. In addition, the stationary parameter search space for a single parameter often exhibits multiple local minima for some

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33



(a) Parameter Changes Over Time (EGRV, sMAPE, Data Set D1)

(b) Stationary Search Space (TSESM, sMAPE, Data Set S1)

Fig. 2. Illustration: Temporal and Stationary Parameter Spaces

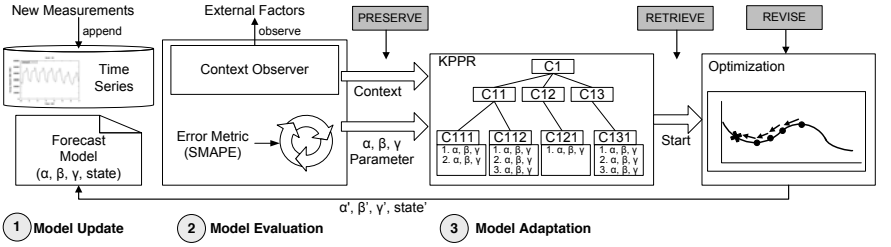


Fig. 3. Context-Aware Forecast Model Maintenance Process

error metrics as shown in Figure 2(b), which greatly decreases the probability of finding global optima using pure local or global searching. This clearly motivates a context-aware adaptation strategy of forecast model parameters to modified contexts from suitable starting points. In the following, we describe our general approach to maintain continuously accurate forecasts in the face of context drift.

Figure 3 illustrates the general forecasting process. First, new measurements are simply appended to the time series and the state of the forecast model is incrementally updated. This step is computational inexpensive and therefore not in the focus of the paper. Second, we continuously observe the forecast accuracy and the development of the current context using different model evaluation techniques. The recognition of context drifts requires enhanced model evaluation capabilities since changing contexts can increase the forecast error at arbitrary times. Evaluation techniques that regularly trigger model adaptations after fixed intervals turn out to be insufficient due to the difficulty of defining suitable adaptation intervals. If the interval is too long, context drifts and increasing errors occurring between adaptations are missed and lead to worse forecasts. Conversely, too short intervals may trigger unnecessary adaptations. We overcome these problems by a threshold-based model evaluation strategy that continuously checks the current forecast error against a defined threshold to ensure a maximal forecast error. The forecast model is adapted as soon as the forecast error

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

surpasses the threshold. While this strategy guarantees that a maximal forecast error is not exceeded, it shares the major drawback with fixed interval model adaptation as it also depends on the definition of suitable thresholds. To this end, we propose an ensemble strategy that combines several individual evaluation strategies. Compared to using only one evaluation technique, such combinations of multiple maintenance strategies reduce the dependence on single adaptation criteria and make it easier to determine suitable values for them.

Third, our model adaptation approach is inspired by an artificial intelligence technique called case-based reasoning (CBR) [16]. The idea of CBR is to solve new problems from solutions of previously encountered problems similarly to the way humans reason by preserving, retrieving and revising previous experiences in the face of new problems. We apply the CBR paradigm to forecast model adaptation by *preserving* old parameter combinations and information about their context (i.e., when they produced accurate forecasts) in a Context-Aware Forecast Model Repository (CFMR) to solve later adaptation problems (Figure 3: context=$C1$,$C12$,...; parameters=α, β, γ). Upon triggering model adaptations, we first *retrieve* promising parameter combinations from the CFMR that were valid in a context similar to the new context. These parameter combinations are *revised* by using them as input for optimization algorithms. This approach is not limited to the energy domain, CBR-based techniques can be applied to arbitrary forecasting applications where similar models are periodically reused.

3 Preserving Forecast Models Using Time Series Context

The *Context-Aware Forecast Model Repository* (CFMR) allows to store and retrieve previous parameter combinations based on context information. When a forecast model is invalidated, the CFMR is searched for parameter combinations that produced accurate forecasts in similar past contexts. The retrieved parameters then serve as starting points for subsequent local optimization. Consider for example a cloudy and rainy day. The meteorological state influences the context and hence, the shape of the time series. Provided the weather conditions change to a similar state on a later occasion, we can search the CFMR for parameter combinations that were valid during such weather conditions.

3.1 Model History Tree

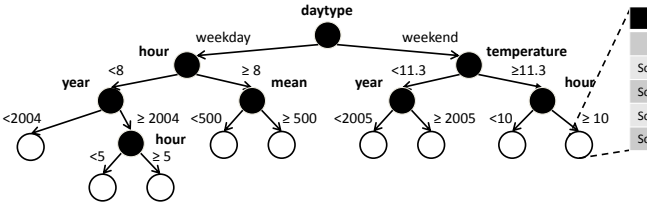
The CFMR is organized as a binary search tree named *Model History Tree*:

Definition 1 (Model History Tree). *A **model history tree** mht, defined over the similarity attributes $a_1, \ldots, a_n$, a maximum leaf node capacity c_{max} and a parameter vector size $V \in \mathbb{N}$, is a decision tree whose nodes are either decision nodes or leaf nodes.*

- Decision nodes *contain a splitting attribute $\dot{a}_i \in \{a_1, \ldots, a_n\}$, a splitting value $\dot{s} \in \dot{a}_i$ and references to the left and right successor nodes. Splitting*

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33



	Parameters P		End Index K
	p_1	p_2	K
Solution1	0.38	0.52	16
Solution2	0.57	0.13	104
Solution3	0.3	0.81	573
Solution4	0.29	0.77	692

Fig. 4. Example Model History Tree

attributes are different influence factors (compare Section 2) of the time series and divide the stored parameter combinations into several classes.

– Leaf nodes *contain a list* $[a_i]$ *of similarity attribute values, at most* $c \leq c_{max}$ *parameter vectors* $p = [p_i|i = 1, \ldots, P]$*, and for each vector an end index* K *representing the last time the parameters were used.*

Figure 4 shows an exemplary model history tree, built over the similarity attributes *daytype*, *temperature*, *hour*, *year* and *mean*. The highlighted leaf node stores four parameter vectors, each of which contains two parameter combinations. The tree essentially forms a recursive partitioning of the parameter space into a set of disjoint subspaces whereas splitting attributes can be thought of as $(n-1)$-dimensional, axially parallel hyperplanes. At each decision node, the tree branches the parameter space into parameter combinations with attribute values smaller than the splitting attribute and those with attribute values greater or equal than the splitting attribute. Leaf nodes store the actual parameter combinations along with the corresponding end indices.

We generally distinguish numerical, nominal and cyclic similarity attributes a_i, which can take values within a domain $[a_i^{min}, a_i^{max}]$. Cyclic attributes are numerical or nominal attributes, whose instance values repeat every c indexes, i.e., $a_i = a_{i+c}$. Accordingly, they are typically connected with seasonal cycles. Table 1 presents an example selection of possible splitting attributes.

The similarity attributes guide the search in the CFMR for promising parameter combinations. That way, the parameter space is initially restricted and the majority of old parameter vectors can quickly be excluded from further

Table 1. Similarity Attributes for Electricity Demand and Supply

		Numerical	Nominal	Cyclic	a_i^{min}	a_i^{max}	Example
Temporal	Year		✓		2000	2020	2005
	Month		✓	✓	1	12	Apr (4)
	Day		✓	✓	1	7	Tue (2)
	Special Day		✓		0	1	False (0)
Exogenous	Temperature	✓			-30	40	27.4
	Wind Speed	✓	✓		0	30	15 m/s
	Electricity Price	✓			0	100	70.38 €/MWh
Statistical	Mean $\bar{x}$	✓			0	40000	12435.5 MW
	Variance σ^2	✓			0	10000	1719.6 MW^2

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

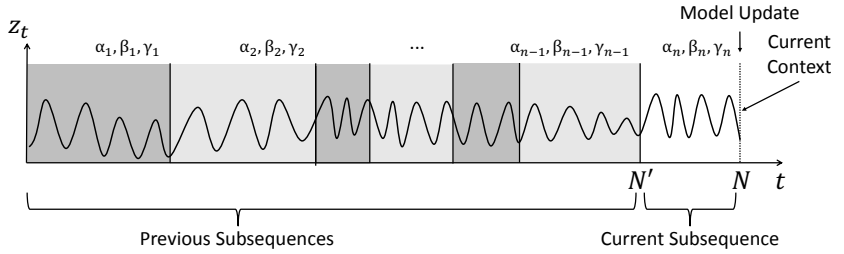


Fig. 5. Time Series Subsequences

processing. However, searching the tree may still result in a large number of results. To this end, we further restrict the solutions by comparing the shapes of corresponding past time series subsequences with the most current subsequence. For each parameter combination we save the time series index K the parameters were used the last time and compare the similarity between the most recent and past subsequences. The parameter combinations with the highest similarity score are finally chosen as starting values for subsequent optimization.

Figure 5 illustrates the subsequences of the time series. The parameter combinations $(\alpha_i, \beta_i, \gamma_i)$ were used to forecast during the corresponding subsequence. The time frame of the current subsequence is marked with indices from N' to N.

3.2 Inserting Models into the Model History Tree

Algorithm 1 defines how parameter combinations are inserted into the Model History Tree. The algorithm first traverses from the root node to the leaf node

Algorithm 1. mhtInsert().

```
input : currContext, paramComb, endIndex
if treeSize() = 0 then makeLeafNode()    // Create tree if necessary
curr_node ⟵ getRoot()    // Traverse to leaf node;
repeat
    // i is index of splitting attribute*/;
    if curr_node.ṡ_i < currContext[i] then curr_node ⟵ curr_node.left();
    else curr_node ⟵ curr_node.right();
until isLeafNode(curr_node);
curr_node.add(currContext, paramComb, endIndex);
if nodeSize(curr_node) > c_max then
    (ȧ_i, ṡ_i) ⟵ computeSplittingAttributeAndValue();
    curr_node.setSplittingAttributeAndValue(ȧ_i, ṡ_i);
    (left, right) ⟵ makeLeafNodes();
    curr_node.setSuccessors(left, right);
    foreach context, paramComb, endIndex in curr_node do
        if context[i] < ṡ_i then left.add(context, paramComb, endIndex);
        else right.add(context, paramComb, endIndex);
    makeDecisionNode(curr_node);
```


Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

that represents the most similar context by comparing the similarity attributes. Afterwards, the new vector is added to the node. If the number of stored vectors c exceeds the maximum node capacity c_{max}, the leaf node is split into two leaf nodes and subsequently converted into a decision node with references to the new successors (divide-and-conquer strategy). The new splitting attribute $\dot{a}_i$ and split value (cut-point) $\dot{s}_i \in \dot{a}_i$ is determined from the models stored in the node. We could potentially rotate the splitting attribute and use the average over all values in the node. This strategy however suffers from the fact that for some attributes we cannot assign unambiguous values (e.g., the attribute values 'hour' 4, 6, 7, 10 yield an average of 6.75, which has no corresponding hour attribute). Thus, using the simple average of all values, especially of nominal or cyclic splitting attributes, is not possible in all cases. For this reason, we use the median which works on numerical as well as nominal and cyclic attributes. The basic idea behind the median is to choose a central value that partitions the possible values $[a_i]$ for attribute a with $i = N', \ldots, N$ in even halves. A prerequisite for the median is to first sort the attribute values in ascending order, leading to a list $[a_j]$ with $j \in \{N', \ldots, N\} \wedge a_j \leq a_{j+1}$. For attributes that do not have a natural order, we apply an artificial order. However, such attributes are very seldom in the energy domain.

Definition 2 (Median). *Provided a_{min} and a_{max} are the minimum and maximum attribute values in the ordered list $[a_j]$, the* ***median*** *$\tilde{a}$ over $[a_j]$ is defined as follows:*

$$\tilde{a} = \begin{cases} a_{min+\frac{max-min}{2}} & , \textit{ if } N' - N \textit{ even} \\ \frac{1}{2}\left(a_{min+\frac{max-min-1}{2}} + a_{min+\frac{max-min-1}{2}+1}\right) & , \textit{ else.} \end{cases} \quad (1)$$

Example 1. Consider the numerical attribute *temperature* with $a_{i=7}$ = '13.5', $a_{i=8}$ = '12.3', $a_{i=9}$ = '15.6' and $a_{i=10}$ = '14.2'. Sorting in ascending ordering gives $[a_{i=8/j=7}, a_{i=7/j=8}, a_{i=10/j=9}, a_{i=9/j=10}]$. Because $N' - N = 10 - 7 = 3$ is uneven, we apply the second formula on the ordered list $[a_j]$ and obtain:

$$\tilde{a} = \frac{1}{2}(a_{j=7+\frac{10-7-1}{2}} + a_{j=7+\frac{10-7-1}{2}+1}) = \frac{1}{2}(a_{j=8} + a_{j=9}) = \frac{1}{2}(a_{i=7} + a_{i=10}) = \text{'13.25'}$$

Using the median as defined above to partition the parameter combinations ensures that selecting any of the attributes as splitting attribute results in the same number of models in both successors. However, the median does not distinguish homogeneously and heterogeneously spread attributes (compare Figure 6) and thus, cannot be used to choose an appropriate splitting attribute. We can assume that attributes with higher density towards the ends constitute better spliting attributes, as the median value separates both halves more clearly. For this reason, we additionally apply the *(Percental) Inter-quartile Range* ((P)IQR) as a measure of dispersion within attributes values and therefore as a measure for the suitability of the attribute as splitting attribute.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

Definition 3 (Inter-quartile Range / Percental Inter-quartile Range). *The **inter-quartile range** (IQR), defined over the list of attribute instances* $[a_i|i = 1, \ldots, N]$*, denotes the average of the first and third quartiles:*

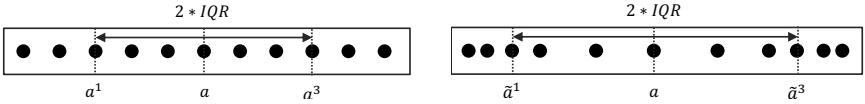
$$IQR = \frac{\tilde{a}^3 - \tilde{a}^1}{2}$$

$$\text{with } \tilde{a}^1 = \tilde{l} \text{ and } l = \{a_i \leq \tilde{a}\} \qquad \text{1st quartile (median of left half)}$$

$$\text{with } \tilde{a}^3 = \tilde{r} \text{ and } r = \{a_i \geq \tilde{a}\} \qquad \text{3rd quartile (median of right half)}$$

To ensures that attributes with a homogenous distribution, but large total range and thus large IQR, are not preferred over those with a heterogeneous distribution and a small range the IQR is normalized by the total range of attribute values leading to: $PIQR = \frac{IQR}{2(a_N - a_1)}$.

The attribute with the highest PIQR-value, i.e., the one with the lowest dispersion, is chosen as splitting attribute.



(a) Homogeneously Spread Attribute. (b) Heterogeneously Spread Attribute.

Fig. 6. Attribute Spread

Balancing of the Model History Tree

Although the median and PIQR guarantee an equal distribution of parameter combinations for single nodes, model history trees can still degenerate to imbalanced, list-like trees when new parameter combinations are always added to one side of the splitting value. The reason is that the employed median heuristics makes *local* decisions only, i.e., the models in the node considered for splitting represent only a small subregion of the whole tree. In order to keep the tree globally balanced, we supplement it with a *global* balancing strategy which is based on measuring the heights of subtrees:

Definition 4 (Node Height / Balanced Node)
*The **height** $h(n)$ of a node n is defined as*

$$h(n) = \begin{cases} 0 & \text{, if } n \text{ is a leaf node} \\ 1 + \max(h(n.left, n.right))) & \text{, if } n \text{ is a decision node.} \end{cases}$$

*If a node n is **B(max)-balanced** if its balance factor $B(n)$ meets the following criteria:* $B(n) = |h(n.left) - h(n.right)| \leq B_{max}$.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

$B(max)$-balanced nodes possess the property that the heights of their left and right subtrees differ at most by the predefined maximal balance factor B_{max} (e.g., $B_{max} = 3$). Global balance of the tree can then be assured by regularly checking the balancing condition. In case of imbalance the model history tree is regenerated, which means to re-build the tree from scratch, although other balancing strategies such as AVL-tree-like rotation [17] are conceivable in future work. When the tree is regenerated, the splitting decisions made at upper intermediate nodes is based on all models below that node. As we employ the median as splitting rule, it is guaranteed that the resulting tree is balanced again.

4 Retrieving and Revising Parameters from the CFMR

After introducing and defining the model history tree of the CFMR to preserve forecast model parameter combinations, we now show how to use the CFMR to quickly retrieve promising parameter combinations and conduct further revision.

4.1 Retrieving Forecast Models

Algorithm 2 defines how forecast model parameter combinations are retrieved from the model history tree. We provide the current context *currContext* (i.e., the state of the similarity attributes at time N) and an auxiliary variable *best*. The algorithm is an adapted *k-nearest neighbor search* and based on the principle of backtracking, which means that it first descents to a leaf node and gradually adds solutions on its way back to the root. The k most similar models with respect to the current situation are obtained using the following process:

Algorithm 2. mhtRetrieve().

```
input : currContext, currNode, best
if isLeafNode(currNode) then
    foreach (context, paramComb, endIndex) in currNode do
        dist ⟵ getEuclidDist(currContext, context)
        if dist < getMaximumDist(best) then
            best.update( context, paramComb, endIndex, dist )
            distanceComputation(paramComb, endIndex)
else
    if currNode.ṡ_i < currContext[i] then
        best, maxDist ← mhtRetrieve(currContext, currNode.left, best)
        if bobTest(currContext, currNode, maxDist) then
            best, maxDist ← mhtRetrieve(currContext, currNode.right, best)
    else
        best, maxDist ← mhtRetrieve(currContext, currNode.right, best)
        if bobTest(currContext, currNode, maxDist) then
            best, maxDist ← mhtRetrieve(currContext, currNode.left, best)
return best
```


Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

1. Traverse from the root to the leaf node that corresponds best to the provided situation (repeatly execute second if-branch).
2. Compute the Euclidian distance at the leaf node between the provided context $currContext(v)$ and any old context(w) stored in the leaf node. The Euclidean distance yields small distances for models which agree in important attributes (v_i/w_i represent the single attributes of context1 and context2).

$$getEuclidDist(context_1(v), context_2(w)) = \sqrt{\sum_{i=l}^{n}(v_i - w_i)^2}$$

Save results in *best*, calculate subsequence similarity for each intermediate result.
3. Ascend to the root node. Perform a *ball-overlap-bounds* (bob) test at each intermediate node to evaluate the existence of additional solutions in opposite branches by checking for points closer than the worst point in *best*. Test by intersecting a n-dimensional ball with radius $getMaximumDist(best)$ and the splitting hyperplane: $getMaximumDist(best) \geq |currContext[i] - node.\dot{s}_i|$. The bob-test evaluates whether the ball around the worst intermediate result overlaps the hyperplane. If true, descend into other branch to search for better solutions.

Example 2. Figure 7 illustrates the execution of Algorithm 2. Attribute a_1 is non-cyclic and attribute a_2 is cyclic.

1. Descent to leaf node corresponding best to provided context and compute the Euclidean distance to all models in the node(O and P). Save results to *best*. Start subsequence similarity calculation.
2. Ascent and perform bob-test at predecessor. Negative — continue ascent.
3. Perform bob-test at next node. Positive bob-test. Find R as nearest neighbor. Update *best*. Start subsequence similarity calculation.
4. Perform bob-test at root. Negative — algorithm finishes. Result: R.

In the worst case, the k-nn-search evaluates all nodes in the tree. While this misbehavior is very unlikely, we avoid long runtimes by further processing the intermediate results in parallel (subsequence similarity, optimization) and using them as temporary parameter combinations for forecasting. This procedure is feasible because even the first intermediate results were found in a node that at least is a good approximation of the current situation. Later results are only accepted if they improve upon the currently known worst intermediate result.

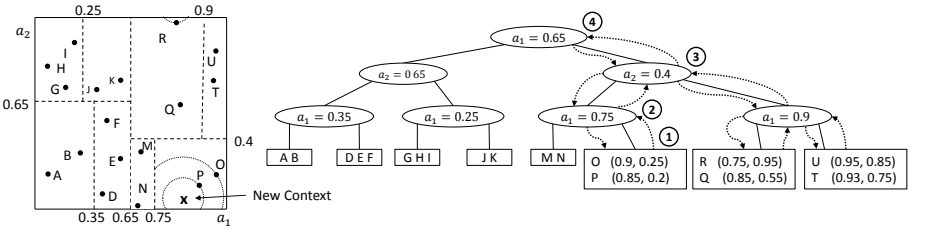


Fig. 7. Example: Model Retrieval in a 2-Dimensional Model History Tree

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

4.2 Comparing Time Series Similarity

In addition to comparing the contexts, we also compare the shapes of the most current time series values and past subsequences corresponding to found parameter combination candidates to identify the most promising parameter combinations. A high degree of coincidence between current and past load curves is regarded as sign for the similarity of the corresponding contexts. The distance between recent and past subsequences is expressed in terms of the Pearson cross-correlation coefficient $R_{zz'}(\tau)$:

$$R_{zz'}(\tau) = \frac{\sum_{i=1}^{N-\tau} (z_i - \overline{z})(z'_{i+\tau} - \overline{z'})}{\sqrt{\sigma_z^2 \sigma_{z'}^2}}.$$

High cross-correlation values entail strong similarity of the involved subsequences. Other distance measures such as Dynamic Time Warping [18] can be used as well. The main side condition imposed by the aforementioned definition, is that both subsequences have to be equally long. To ensure equally long time series, we chose a fixed length of the subsequence from a corresponding point of the time series and hence obtain as *past* and *current* subsequences. However, the cross-correlation will be low for phase-shifted subsequences (i.e., different start/end indices), even though their shapes are similar. We overcome this difficulty by shifting one sequence past the other through the specification of an lag τ. The lag τ is denoted with respect to the ending indexes K (end of former situation) and N (end of current situation) and the (known) period s of a seasonal cycle: $\tau = |K \bmod s - N \bmod s|$. This lag specification aligns the subsequences by cutting the outer values of both sequences. Altogether, the cross-correlation can be applied to time series which changes at most its amplitude, offset and level over time, but not its periods. This makes it a good choice for electricity demand time series, which possess comparatively stable seasonal cycles. As a result the similarity search provides the parameter combinations that yield the most similar subsequences.

4.3 Revising Forecast Models

A re-estimation of forecast model parameters serves as further refinement of the retrieved parameter combinations. We concurrently perform a local (e.g., Nelder-Mead) and a global optimization (e.g., Simulated Annealing). The starting points for the local search are the parameter combination candidates provided by the CFMR. Due to the continuous adaptation of the forecast model to drifting context, we assume that the parameters changed gradually only with respect to the old and current situation. For this reason, we find the global optimal parameter combination with high probability close to the provided starting point. However, it is still advisable to check for regions not covered by local search. The employed global search runs in parallel to the local search, because it is independent of starting values. We continue the global search even after the local search found its optimal solution. Thus, we consider all areas of the solution space. Due to the

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

long run time of the global search, the search runs asynchronously in the background, while the solution found by the local search is used as an intermediate parameter combination. If the global search finds a better solution, we use this solution as an additional starting point for the global optimization.

5 Evaluation

In this evaluation we proof the claims of our approach and show that with the help of the CFMR we can increase the parameter estimation efficiency by means of delivering more accurate forecasts in a shorter time frame compared to other approaches. Our evaluation compares the accuracy and the time necessary to gain the accuracy and is based on two forecast models and three electricity data sets from different parts of the European electricity market.
Data Set D1: National Grid Electricity Demand from National Grid (publicly available [19]). Electricity demand of the United Kingdom. Measures used: INDO, January 1st 2002 to December 31st 2009, 30min resolution.
Data Set D2: EnBW MeRegio Household Energy Demand from MIRACLE partner EnBW. Energy demand from 86 anonymized customers — We used customers 7 (D2a, more predictable behavior) and 40 (D2b, hardly predictable behavior). Measures used: November 1st 2009 to June 30th 2010, 1h resolution.
Data Set S1: CRES Photo-Voltaic Energy Supply from MIRACLE partner CRES. Supply of a 22kW photovoltaic panel. Measures used: January 11th 2008 to December 16th 2008, 1min resolution aggregated to 30min resolution.

The evaluation was conducted on a AMD Athlon 4850e with 4 GB RAM, Microsoft Windows 7 64bit and Microsoft Visual Studio C++ 2010.

For our evaluation we employed two forecast models tailor-made for the forecasting of energy demand and supply. The first model is Triple Seasonal Exponential Smoothing (TSESM) [20]. TSESM involves five forecast model parameters with values from 0 to 1, which lead to a five-dimensional solution space for the parameter estimation. The second model is named EGRV model and defines a separate model for each hour of the day to avoid the explicit modeling of the complex daily season [15]. In addition, different influences such as the current day and month are included as separate variables. Thus, one hourly model involves around 31 parameters, depending on the incorporated influences.

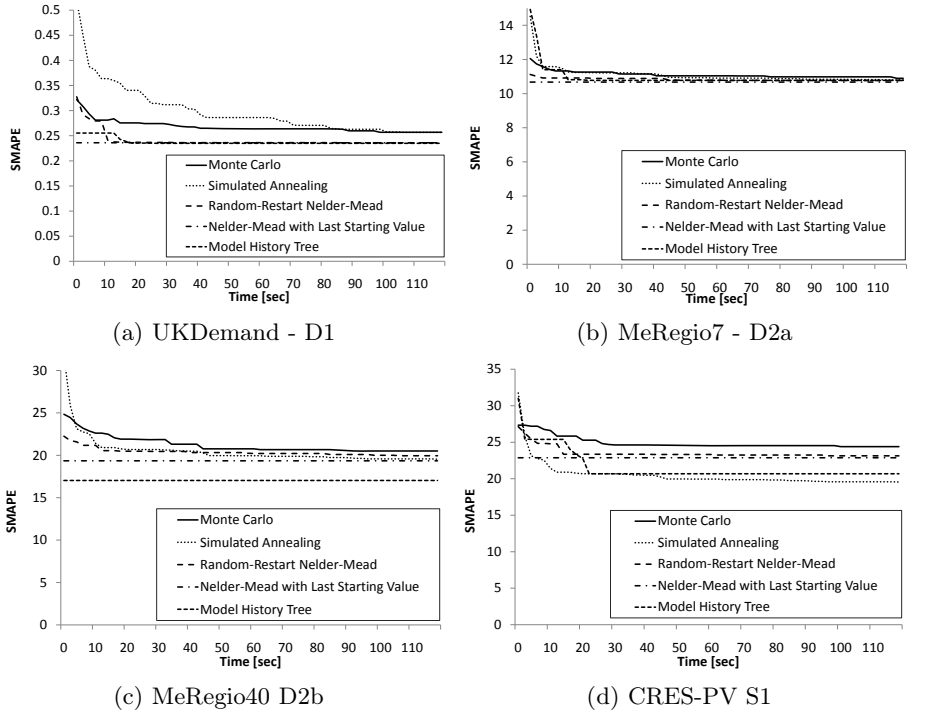
To contrast our approach, we used the following four common local and global parameter estimation approaches:

- *Monte-Carlo*: Iteratively evaluate random solutions.
- *Simulated Annealing*: Global search without starting values.
- *Random-Restart Nelder-Mead*: Iterated local searching with starting values obtained by random search.
- *Single Nelder-Mead*: Local search with current parameters as starting values

The forecast models were optimized for one-step ahead forecasts and using the sMAPE error metric [6]. We traced the lowest sMAPE obtained during optimization every two seconds and averaged over four runs per approach. For EGRV,

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33



(a) UKDemand - D1

(b) MeRegio7 - D2a

(c) MeRegio40 D2b

(d) CRES-PV S1

Fig. 8. Time vs. Accuracy - Triple Seasonal Exponential Smoothing

the random-restart Nelder-Mead strategy was budgeted at five minutes, because unsuitable starting points led to a very slow convergence. The presented results illustrate a single point in time only and are based on threshold-based model adaptations. In detail, we used data-set specific error thresholds/sliding windows sizes of 6%/12 D1,20%(30%)/3 D2a(D2b) and 30%/2 S1, which lead to about 100 models stored into the tree at the time the model was re-estimated. We repeated our evaluation at other points in time and achieved similar results.

Figure 8 illustrates the results for the Triple Seasonal Exponential Smoothing model. We observe that our approach in general quickly reaches good accuracies on all data sets. However, the subsequent global search does not find better parameter combinations. The reason could be that the parameters found by local searching are very good and with a high probability already the global optimal solution. For the datasets D1 and D2b our approach achieved the best results regarding accuracy and time for the entire test period. There, the single simulated annealing approach achieved the worst results for data set D1 and the Monte-Carlo approach performed worse for data set D2b. For data set D2a and S1 our approach also achieved good results but performed not as well as other approaches. Regarding data set D2a our approach had a good start, but both local search approaches that involve the Nelder-Mead algorithm achieved better results at the start. Simulated annealing and the Monte-Carlo approach

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

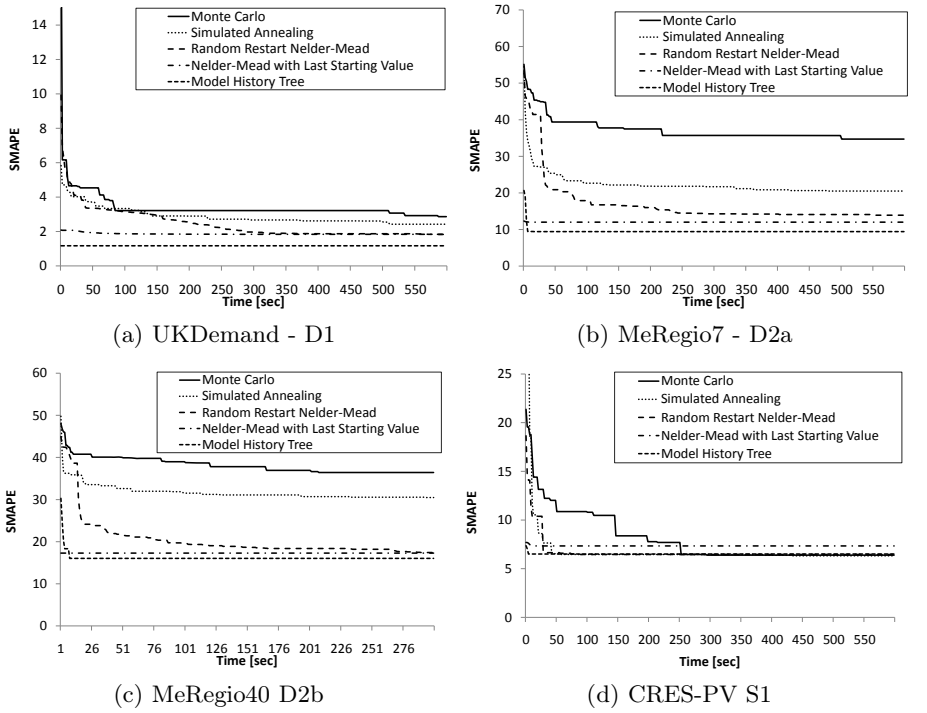
Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

performed worse at the start. With further progression all approaches achieved similar results and differ by less than 0.5% SMAPE. However, our approach performed worst on this data set. For dataset S1, our approach converged slower than all approaches except the Monte-Carlo sampling, but at the end it achieved the second best result. Only simulated annealing performed better by less than a half percent. We blame the results to the sequential execution of the local searches which are occasionally supplied unfavorable starting values in the beginning and the best starting values in the end. All other approaches differ by a more significant amount of two or more percent. Overall we can state that for the triple seasonal exponential smoothing our approach achieved good results on all data sets. In two cases it performed worse than other approaches, but however the other approaches have a larger divergence concerning their results, e.g. simulated annealing performed worse for data set D1 and best for data set S1. In contrast, our approach constantly achieved very good results, which leads to the educated guess that it is useable for all data sets from the energy domain without prior evaluation. We furthermore observed only small overhead from using the tree. Depending on the data set, the context computation, model insertion and model retrieval took together always less than 4 msec. We also tested scalability of these operations for trees with up to 20,000 models and obtained a joint worst case insertion and access time of less than 0.6 sec, which is negligible in comparison to the overall re-estimation time.

Figure 9 illustrates the results for the EGRV forecast model. In general, due to a much higher number of parameters, these models are more challenging to update than the previously discussed seasonal exponential smoothing models. In addition, we can observe that the differences in the reached maximal accuracy between the best and worst strategy are larger than for smoothing models which means that the choice of a good re-estimation strategy is hence more critical. Our approach achieved the best results for all evaluated data sets by means of both accuracy and time. All strategies obtained improvements particularly quickly within the first minute of execution, but with further progression they were not able to reach the accuracy of our approach. For data set D1 and D2a the accuracy gap between our approach and its competitors is comparatively large. For data set D2b the local search strategies at the end achieved similar but slightly worse results. In contrast, the produced accuracies of the global search strategies were far off. Regarding the supply data set S1 four out of five approaches converged to a similar result and except for Nelder-Mead with Last Starting Value the difference in accuracy is rather small. The results produced by Simulated Annealing are also at least as good or better than the results found by Monte-Carlo, which confirms our choice of simulated annealing as our global coverage strategy. In addition the random-restart Nelder-Mead strategy shows only slow convergence in average, but it often finds good results after some minutes. This demonstrates the need to start optimization from suitable starting parameters. Again, the run-time overhead for inserting and retrieving models from the CFMR depends on the data set, but was less than 5 msec in the worst case. Further experiments with 20,000 EGRV models in the tree still show access

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33



(a) UKDemand - D1

(b) MeRegio7 - D2a

(c) MeRegio40 D2b

(d) CRES-PV S1

Fig. 9. Time vs. Accuracy - EGRV

times of less than 1.1 sec which suggests that the tree scales also for large history bases. Overall, our approach achieved better results when used in conjunction with a more complex forecast model. For all data sets the CFMR outperformed all other estimation approaches, which makes it the most suitable approach for models that involve a large number of parameters. An interesting effect we observed during evaluation was the steadily improving capability of the tree to provide parameters that were already optimal without further optimization. For later stages of the demand data sets, optimization was fully redundant and improved the result only on special days such as Christmas. However, to ensure the optimality of the result, the parallel optimization should still be conducted.

6 Related Work

Approaches that use aspects similar to our approach exist in various domains: Kohara et al. described a system that uses prior knowledge and information about events manually extracted from newspapers in conjunction with neural networks to improve stock market predictions [21]. In the domain of conceptual modeling Becker et al. proposed the reuse of knowledge from other modelers or former projects by using context-based modeling that combines reuse

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

mechanisms like aggregation, restriction and specialization of previous models. Context-base modeling exploits the context of a conceptual model to for example restrict available constructs and their relations [22]. Breitman et al. proposed a similar solution that stores conceptual models created by expert designers in a repository that is used by less experienced designers to later create new conceptual models [23]. Luan Ou et al. introduced an approach for process models in the business intelligence domain. They store process models in a model base and use CBR and rule-based reasoning techniques to quickly find the right process for a given task. For new data mining tasks they retrieve the most similar model from the model base and present it to the user for confirmation. The similarity measures base on domain knowledge about the process models [24]. Compared to our approach, all presented solutions utilize similar basic ideas, especially, (1) the creation of a case base for later reuse and (2) the usage of context or domain knowledge to efficiently find suitable solutions. However, they are applicable in their specific domain only and cannot directly be applied to the forecasting domain. Our approach is the first adaptation of CBR that exploits the context of time series, for which reason it involves specific aspects like the utilization of a decision tree, a similarity measurement and a subsequent optimization.

7 Conclusion

In this paper, we presented a novel parameter estimation approach that exploits the context of a time series to quickly find starting parameters for further optimization. There, we used our Context-Aware Forecast Model Repository (CFMR) to store the parameter combinations in conjunction to their associated context. Besides basic definitions, we described the preservation, retrieval and revision of forecast models within our repository. Our evaluation on four datasets showed that our solution provides an efficient way to estimate parameters, especially when dealing with complex forecast models. In most cases we were superior to all evaluated competitors in providing more accurate forecasts in less time. There is plenty of future work, which includes the context-aware estimation of parameters, an experimental investigation of influences between different context components and a even better parallelization of our approach.

Acknowledgment

The work presented in this paper has been carried out in the MIRACLE project funded by the EU under the grant agreement number 248195.

References

1. MIRACLE Project (2011), `http://www.miracle-project.eu`
2. MeRegio Project (2011), `http://www.meregio.de/en/`
3. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control. John Wiley & Sons Inc., Chichester (2008)

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Scientific and Statistical Database Management: 23rd International Conference. Portland 2011", S. 491–508, ISBN 978-3-642-22351-8
http://dx.doi.org/10.1007/978-3-642-22351-8_33

4. Winters, P.R.: Forecasting sales by exponentially weighted moving averages. Management Science, 324–342 (April 1960)
5. Bunnoon, P., Chalermyanont, K., Limsakul, C.: A computing model of artificial intelligent approaches to mid-term load forecasting: a state-of-the-art- survey for the researcher. Int. Journal of Engineering and Technology 2(1), 94–100 (2010)
6. Hyndman, R.J.: Another look at forecast-accuracy metrics for intermittent demand. Foresight: The International Journal of Applied Forecasting 4, 43–46 (2006)
7. Duan, S., Babu, S.: Processing forecasting queries. In: VLDB (2007)
8. Fischer, U., Rosenthal, F., Boehm, M., Lehner, W.: Indexing forecast models for matching and maintenance. In: IDEAS. Dresden University of Technology (2010)
9. Oracle test: Oracle - Driving Strategic Planning with Predictive Modeling (2008)
10. Microsoft: SQL Server 2008 - Predictive Analysis with SQL Server (2008)
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science. New Series 220(4598), 671–680 (1983)
12. Nelder, J., Mead, R.: A simplex method for function minimization. The Computer Journal 7(4), 308–313 (1965)
13. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(69) (1996)
14. Zliobaite, I.: Learning under concept drift: An overview. Technical report, Vilnius University (2009)
15. Ramanathan, R., Engle, R., Granger, C.W., Vahid-Araghi, F., Brace, C.: Short-run forecasts of electricity loads and peaks. International Journal of Forecasting 13(2), 161–174 (1997)
16. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
17. Sedgewick, R.: Algorithms. Addison-Wesley, Reading (1988)
18. Berndt, J., Donald, J.C.: Using Dynamic Time Warping to Find Patterns in Time Series. Technical report, Stern School of Business, New York University (1994)
19. Nationalgrid UK.: Metered half-hourly electricity demands (2010), `http://www.nationalgrid.com/uk/Electricity/Data/Demand+Data/`
20. Taylor, J.W.: Triple seasonal methods for short-term electricity demand forecasting. European Journal of Operational Research 204, 139–152 (2009)
21. Kohara, K., Ishikawa, T., Fukuhara, Y., Nakamura, Y.: Stock price prediction using prior knowledge and neural networks. Intelligent Systems in Accounting, Finance & Management 6(1), 11–22 (1998)
22. Becker, J., Janiesch, C., Pfeiffer, D.: Towards more Reuse in Conceptual Modeling - A Combined Approach using Contexts. In: CAiSE Forum (2007)
23. Breitman, K.K., Barbosa, S.D.J., Casanova, M.A., Furtado, A.L., Hinchey, M.G.: Using analogy to promote conceptual modeling reuse. In: ISoLA, pp. 111–122 (2007)
24. Ou, L., Peng, H.: XML and knowledge based process model reuse and management in business intelligence system. In: Shen, H.T., Li, J., Li, M., Ni, J., Wang, W. (eds.) APWeb Workshops 2006. LNCS, vol. 3842, pp. 117–121. Springer, Heidelberg (2006)

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden