Genetic Algorithm for Pattern Detection in NIALM Systems

Michael Baranski
Faculty for Electrical Engineering
Sustainable Energy Concepts
University of Paderborn, Germany
baranski@nek.upb.de

Jürgen Voss
Faculty for Electrical Engineering
Sustainable Energy Concepts
University of Paderborn, Germany
voss@nek.upb.de

Abstract - *Nonintrusive Appliance Load Monitoring systems (NIALM) require sufficient accurate total load data to separate the load into its major appliances. The most available solutions separate the whole electric energy consumption based on the measurement of all three voltages and currents. Aside from the cost for special measuring devices, the intrusion into the local installation is the main problem for reaching a high market distribution. The use of standard digital electricity meters could avoid this problem with loss of information in the measured data. This paper presents a new NIALM approach to analyse data, collected form a standard digital electricity meter. To disaggregate the consumption of the entire active power into its major electrical end uses, an algorithm consisting of fuzzy clustering methods, a genetic algorithm and a dynamic programming approach is presented.*

Keywords: Clustering, genetic algorithm, dynamic programming, pattern detection

1. Introduction

The following paper presents a new method to detect patterns of electrical appliances from total load measurements at the main electric connection of a private household. The algorithm consists of fuzzy clustering methods (cf. [1]) combined with genetic algorithms (cf. [8]) and a dynamic programming approach, similar to the well known viterbi algorithm (cf. [3]). It is designed to detect frequently occurring patterns from the load trace of the active power consumption without any a priori knowledge concerning special appliances. The system tries to find repetitive appliance structures avoiding initial inputs of patterns or parameters with respect to operator convenience. For this purpose, it only detects recurring signatures of appliances switching behavior. The system is developed to work with rough data collected by ordinary digital meters in order to operate with future digital meters with optical interfaces (cf. [9]). Mature NIALM systems often need to measure the active and reactive power of all three electrical phases to analyse the data (cf. [2]).

The presented methods could also be used to analyse any other time series data according to automaticly detecting patterns of finite state machines (FSM), which behave similar to electric appliances e.g. gas, water or any other measurable consumption data. Further applications could be found in data mining or pattern detection and recognition systems like financial or speech analysis.

2. Overview NIALM

Non-intrusive appliance load monitoring (NIALM) is since the mid-nineties a mature technique for disaggregating the entire electrical load into the major end uses of domestic homes (cf. [4]- [7], [10] and [11]). The evaluation is often based on measuring the reactive and active power fluctuations on all three phases of the household connection, so that the information with regard to the energy and power consumption can be analysed with a resolution of one second. However, the measuring instrumentation for this costs approximately 1200 Euro per household (cf. [2]). This is too expensive for saturation monitoring in private households. Low cost systems are required for the data acquisition on the private customer sector, and it must be possible to integrate these systems in the existing household installations with minimised technical and financial effort. To accomplish area wide acceptance, low cost devices to be mounted on existing electricity meters or future digital electronic meters are needed.

3. New NIALM approach

The presented algorithm tries to find patterns of typical electrical appliances from total load measurements at the main electrictity connection of a domestic home. Only the load series of the active power consumption with timeresolution down to 1s is analysed. Let's consider $\{P_t\}$, the above mentioned load series a private home. With respect to the behavior of typical electric appliances in households, the active power value could be assumed to be greater or equal zero.

$$P_t \geq 0 \quad \forall t \tag{1}$$

This condition could be coppied to every single applinace, too.

$$P_{jt} \geq 0 \quad \forall j,t \tag{2}$$

The total load series $\{P_t\}$ could be described as a sum of series values $\{P_{j,t}\}$ of all detectable aplliances plus the resting power consumption assigned to the rest of unknown electrical devices to

$$P_t = \sum_{j=1}^{N_V} P_{j,t} + P_{rest} \quad \forall\, t. \tag{3}$$

State changes of the total load trace are detected and handled as switch events. These events contain the information about all involved electric appliances. For further explanations they will simply be named as events. To detect events, the difference series of the total load is computed by

$$\Delta P_t = P_t - P_{t-1} \tag{4}$$

clearing the offset value from $\{P_t\}$ additionally. The values of the difference series $\{\Delta P\}$ are distributed to events S_i according to

$$\Delta P_t \in \begin{cases} S_i & \text{if } |\Delta P_t| > \delta \wedge sgn(\Delta P_t) = sgn(\Delta P_{t-1}) \\ S_{i+1} & \text{if } |\Delta P_t| > \delta \wedge sgn(\Delta P_t) \neq sgn(\Delta P_{t-1}). \end{cases} \tag{5}$$

Following (5) not all important events are detectable. The switching behavior of typical AC-drives with short peaks will be swallowed. These peaks will result in two events with a very small distance Δt_{peak}. If we combine these events refering to

$$S_i = S_i + S_{i+1} \forall S_i | P(S_i) > 0;\ P(S_{i+1}) < 0;$$

$$t_{i+1} - t_i < \Delta t_{peak} \tag{6}$$

additional properties of AC-drive switch events could be handled more suitable. To reduce the number of events due to measurement noise, a suitable barrier δ to eliminate small events is used in (5). Events are handled as complex data types (i.e. struct value in C programming language), the relevant values are noted in an object vector

$$\underline{O_i} = [P(S_i), \hat{P}(S_i), T(S_i)] \tag{7}$$

where $P(S_i)$ denotes the total power step (sum of all assigned ΔP_t), $\hat{P}(S_i)$ the maximum power step value (boost value) according to a peak and $T(S_i)$ the time distance between the first and the last ΔP_t of S_i. After creating events switching patterns of electrical appliances have to be detected.

The problem of finding patterns of unknown aplliances could be formulated as an optimization problem. The discrete time series of each appliance V_j could be described as sum of its events. Let $\{P_{j,t}\}$ denote the discrete time series of V_j, then only a subset of all N_S events could be linked to this aplliance assuming at least two or more existing different appliances. Using a vector for each appliance, containing binary values, special events could be selected while masking all the others by setting the corresponding values $u_{j,i}$ to zero. The power series of appliance V_j at any time t could be written as

$$P_{j,t} = \sum_{i=1}^{N_S} u_{j,i}\sigma(t - t_i)P(S_i) \tag{8}$$

with

$$\sigma(t - t_i) = \begin{cases} 0 & \text{if } t < t_i \\ 1 & else \end{cases} \tag{9}$$

This could be done to all N_V appliances. Written in matrix notation, matrix $\underline{U}$ contains all information about the mapping between appliances and switch events.

$$\begin{bmatrix} P_{1,t} \\ \vdots \\ P_{N_V,t} \end{bmatrix} = \begin{bmatrix} u_{1,1} & .. & u_{1,N_S} \\ \vdots & \vdots & \vdots \\ u_{1,N_V} & .. & u_{N_V,N_S} \end{bmatrix} diag(\sigma(t - t_i)) \begin{bmatrix} S_1 \\ \vdots \\ S_{N_S} \end{bmatrix} \tag{10}$$

The time series of each appliance has to satisfy an additional constraint respectively

$$P_{j,t} \leq P_t \ \forall\, t \tag{11}$$

Each collumn of $\underline{U}$ in (10) denotes the membership of a single event S_i. diag means a $(N_S \times N_S)$-diagonal matrix containing step functions for every single event. If we assume to have the correct assignment for all events, only one position of each collum vector $\underline{u_i}$ equals one, all others have to be zero. Finding the best fitting configuration of $\underline{U}$ due to the properties of all concerned appliances is the goal of our optimization problem. This is not quite a simple problem with respect to the great number of events per day (about 16.000).

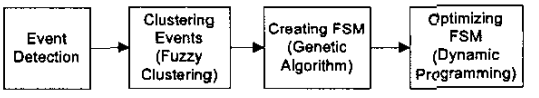

Figure 1. Structure of the main algorithm

We limit the solution space by reducing the number of detectable appliances. Only appliances with recurring patterns arouse our interest. Furthermore the number of different power steps of each appliance, handled as finite state machine, has to be limited with respect to execution time. Furthermore, the algortihm only concentrates on significantly strong or frequently occuring appliances. Rare and small events are neglected due to their irrelevant impact. Figure 1 gives a short overview of the main subroutines of the algorithm to detect appliance patterns. After clustering all detected events, the resulting clusters summarise events with similar structures. Combinations of different clusters represent potential candidates of appliances (finite state machines). The cluster algorithm computes conglomerations of similar switch events using fuzzy clustering methods referring to [1] and leads to N_C clusters C_r. The clustered objects are denoted by (7). Detectable appliances are handled as finite state machines with a fixed number of states. Furthermore is assumed that the state changes within the recurring sequences have all the same fixed order. $P(C_r)$ denotes the power value of the centre of cluster C_r. Binary combination values

help to select a limited number of cluster values P(C) to create a set of switch states describing a hypothetical appliance.

$$\underline{Z}_j = \left[c_{j1}P(C_1); c_{j2}P(C_2); \ldots; c_{jN_C}P(C_{N_C})\right] \quad (12)$$

Subsets of power values are joined to appliances V_j by altering binary values $c_{jr} \in \{0;1\}\ \forall j,r\,;\quad r = 1..N_C$ in (12) by a genetic algorithm with respect to execution time.

3.1. Combining clusters to finite state machines by a genetic algorithm

The genetic algorithm alters the binary values of a $(N_V \times N_C)$- matrix $\underline{X}$ containing the combinations due to (12) to create suitable finite state machines for different appliances. A limited number of rows is initialized representing the first generation of individuals. The number of rows has to exceed the number of clusters to enclose the solution space sufficiently. The quality of each row of $\underline{X}$ could be evaluated by a function like

$$Q_{V_j} = \gamma_1 Q_{V_j}^{(1)} + \gamma_2 Q_{V_j}^{(2)} + \gamma_3 Q_{V_j}^{(3)} \quad (13)$$

with

$$Q_{V_j}^{(1)} = \frac{\left|\sum_{r=1}^{N_C} c_{jr} P(C_r)\right|}{\max_{c_{jr}=1}\left(\left\{\left|c_{jr} P(C_r)\right|\right\}\right)} \quad (14)$$

a term to evaluate the congruence of all power values, relative to the greatest absolut value of the corresponding combination,

$$Q_{V_j}^{(2)} = \frac{\left|\sum_{r=1}^{N_C} h(C_r)\, c_{jr} P(C_r)\right|}{\max_{c_{jr}=1}\left(\left\{\left|c_{jr} P(C_r)\right|\right\}\right)} \quad (15)$$

where the relative frequency of each cluster is taken into account and

$$Q_{V_j}^{(3)} = \frac{1}{N_C}\left|\sum_{r=1}^{N_C} c_{jr}\right| \quad (16)$$

a term assessing small numbers of clusters combined to create an appliance better than greater ones. With

$$h(C_r) = \frac{H(C_r)}{\sum_{r=1}^{N_C} H(C_r)} \quad (17)$$

and $H(C_r)$ denoting the number of elements clustered in C_r. The weightings γ_1, γ_2 and γ_3 allow to optimise the quality function dependend on detected appliances. The developed genetic algorithm works similar to the pseudo code genetic algorithm presented in [8].

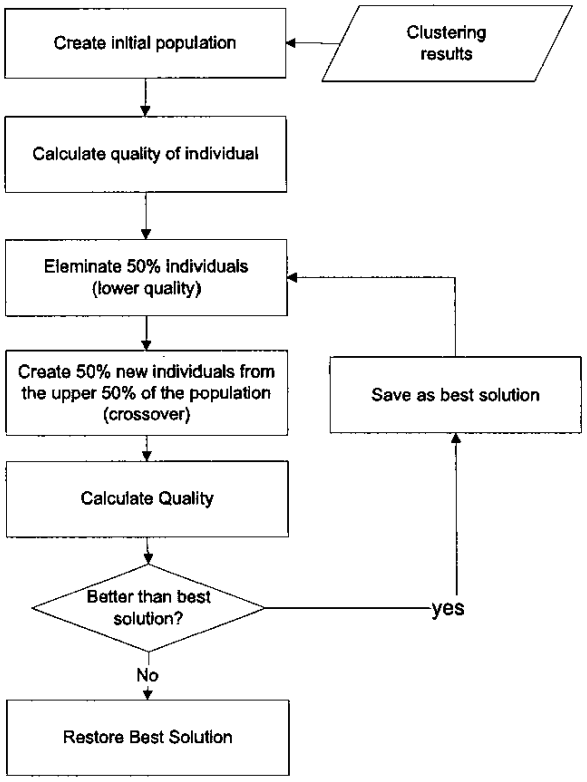

Figure 2. Block diagramm of the genetic algorithm to combinate cluster centres to create finite state machines

3.2. Optimizing sequences

Every row of $\underline{X}$ is used to create a finite state machine, containing a set of selected power values denoting the transitions between different power states of an appliance. After generating suitable combinations to create finite state machines, ν different variations of each combination have to be evaluated.

$$\underline{Z}_j^{(\nu)} = \left[Z_{j,1}, Z_{j,2}, .., Z_{j,k}\right] \quad (18)$$

Some values (cluster centres) are distributed to different appliances simultanously, i.e. when the number of ones in any collumn of $\underline{X}$ exceeds one. Those overlappings have to be solved after optimising the time series of each appliance. Each appliance is linked to a subset of events by the corresponding clusters. These events are ordered by time and used to generate a valid time series due to (8). However, very often the subset of events is not ordered in the right way to ensure the correct time series demanded by the current finite state machine. The problem of finding the best matching combination of switch events could be solved by dynamic programming, solving a shortest path problem. For this purpose, the optimal time series $\{P_{j,t}\}$ is subdivided into a finite number of sequences Γ_l stressing the following assumptions:

i) The load behavior (i.e. time series of active power) of any detectable appliances structure

should be considered as a recurring pattern, all parameters should range in a limited area around their median value (cf. [4]), which is different from zero

ii) Every state $Z_{k_{J_i}}$ from the set of states defining a finite state machine is activated exactly one time in each sequence

iii) Every sequence has the same fixed order of states to pass through

All events, assigned to the current appliance, are handled as instances of state values of the current finite state machine. Now, a series of sequences maximizing a quality criterion could be searched, assuring that no assigned event S_i referenced by $Z_{k_{J_i}}$ is used twice.

$$P_{j\mathcal{I}} = \sum_{l=1}^{N_\Gamma} \Gamma_l \text{ with } \Gamma_l = \left\{Z_{1_{J_1}}, Z_{2_{J_2}}, ..., Z_{k_{J_k}}\right\} \quad (19)$$

A quality criterion similar to Shannons entropy cf. [5] leads to suitable results evaluating the quality of each sequence by

$$Q_\Gamma = -\sum_i \Delta e_i \log|\Delta e_i| \quad (20)$$

with

$$\Delta e_i = \left|\frac{\mathcal{E}_i(\Gamma_l) - \mathcal{E}_i(\Gamma_c)}{\mathcal{E}_i(\Gamma_c)}\right| + e_0 \quad (21)$$

where $\mathcal{E}_i$ denotes any property like the time duration between state changes in a sequence or the deviation between the power value from S_i and the corresponding value of the state $Z_{k_{J_i}}$ built by the average value of the centre of the cluster C_r containing S_i. In (21) $\mathcal{E}_i(\Gamma_c)$ denotes the i-th property of the optimal sequence indexed by c, $\mathcal{E}_i(\Gamma_l)$ describes the properties of the sequence currently being evaluated.

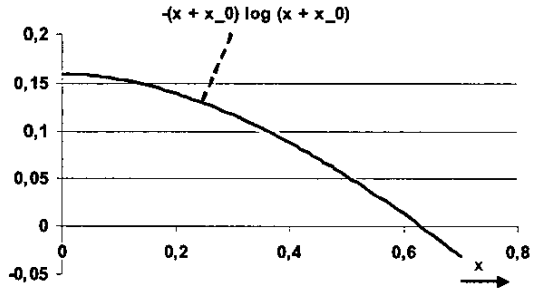

Figure 3. Quality function to evaluate a path

In Fig. 3, $f(x) = -x \log|x|$ is shifted by x_0= 0,3678, the optimal x value to maximize f(x) when a parameter of the corresponding sequence deviates minimum (x= 0) to it's optimum cf. (20).The shortest path bridging the tree of all valid sequences could be found by searching the maximum quality value computed by

$$Q_p = \sum_{l=1}^{N_{T_p}} Q_l \quad (22)$$

for each path. When the best path is found, the procedure will be repeated for all other valid variations of $\underline{Z}^{(v)}$. After choosing the best variation including the shortest path linked to this variation, the time series is created by altering the binary values $u_{j\mathcal{I}}$ in (8) according to the corresponding path. To calculate the quality of any sequence, default or target values have to be generated. This could be done by computing the median value of all linked events. The parameters are then optimised by an evolutionary algorithm, improooving the quality of the sequences by altering the parameters with respect to the range of their deviation.

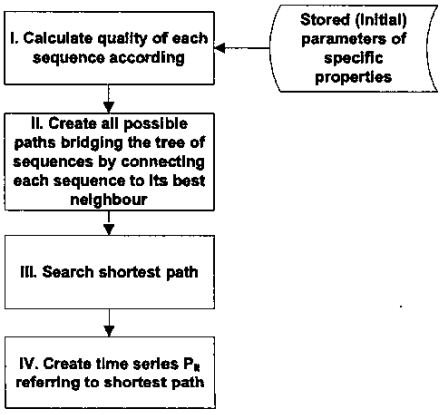

Figure 4. Optimizing a single finite state machine

Now lets consider a simple example for the optimization of sequences of an appliance V_1. We assume to have e.g. $P(C_1)$ = -150W, $P(C_3)$= 50W and $P(C_4)$ = 100W as a result of the genetic algorithm creating a set of power values from clusters C_1, C_3 and C_4. V_1 represents the first row of $\underline{X}$ with $c_{1,1} = c_{1,3} = c_{1,4} = 1$, all other $c_{1,r} = 0$. The state values of this finite state machine could be transformed referring to (12) to

$Z_1 = 100W$, $Z_2 = 50W$ and $Z_3 = -150W$.

With these three values, a set of 3!= 6 variations of sequence patterns could be generated as follows

1. $Z_1 \rightarrow Z_2 \rightarrow Z_3$ 2. $Z_1 \rightarrow Z_3 \rightarrow Z_2$

3. $Z_2 \rightarrow Z_1 \rightarrow Z_3$ 4. $Z_2 \rightarrow Z_3 \rightarrow Z_1$

5. $Z_3 \rightarrow Z_1 \rightarrow Z_2$ 6. $Z_3 \rightarrow Z_2 \rightarrow Z_1$

Only 1. and 3. satisfy (2), so all other variations are invalid and have to be discarded. Only these two variations $\underline{Z}_1^{(1)} = [Z_1, Z_2, Z_3]$ *and* $\underline{Z}_1^{(2)} = [Z_2, Z_1, Z_3]$ could be used to create valid sequences satisfying all other constraints For these variations the shortest path will be computed simultanously. The best variation will be handled as the finite state machine to model the appliance, the other one will be discarded. For further explanations we only analyze $\underline{Z}_1^{(1)}$. After creating a valid variation $\underline{Z}_1^{(1)}$, all events S_i, referenced

by the corresponding clusters C_1, C_3 and C_4 are assigned to this finite state machine and sorted by time in ascending manner. So we have a subset of switch events

$$\left\{S_{j,i}^{(V_1)}\right\} = \{S_i | S_i \in \{C_1 \vee C_3 \vee C_4\}\} (23)$$

Then subsequences of statestransitions $Z_1 \rightarrow Z_2$ and $Z_2 \rightarrow Z_3$ have to be created to generate all possible valid sequences and a sequence tree. Figure 6 shows an exapmle of a series of states is shown. Each of them is linked to an unique separate switch event, indexed by time. The arrows in Fig. 6 describe possible combinations to create subsequences.

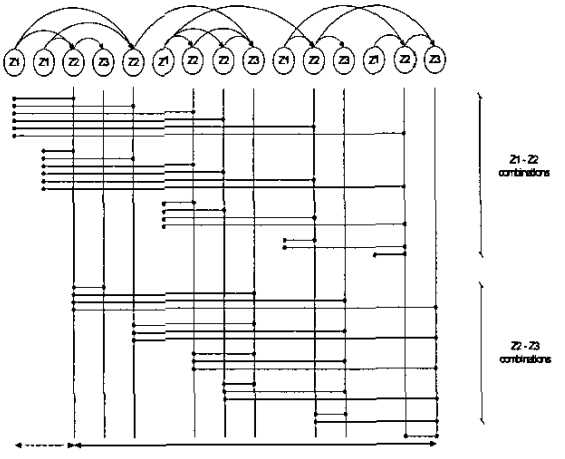

Figure 5. Creating subsequences

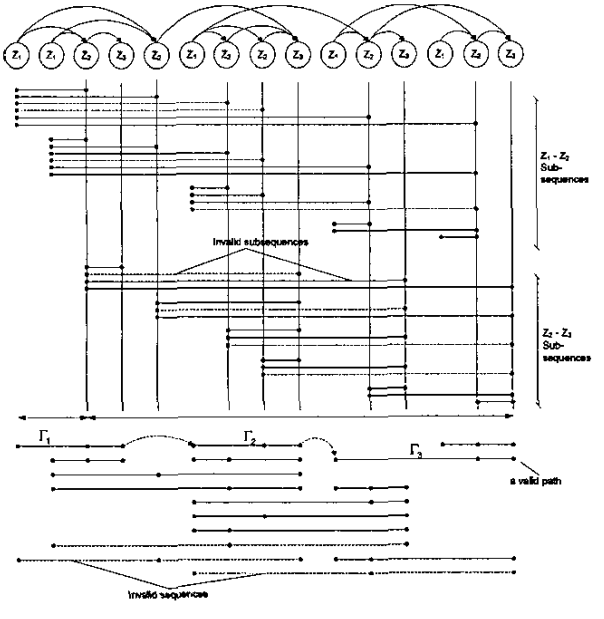

Figure 6. Connecting subsequences to sequences

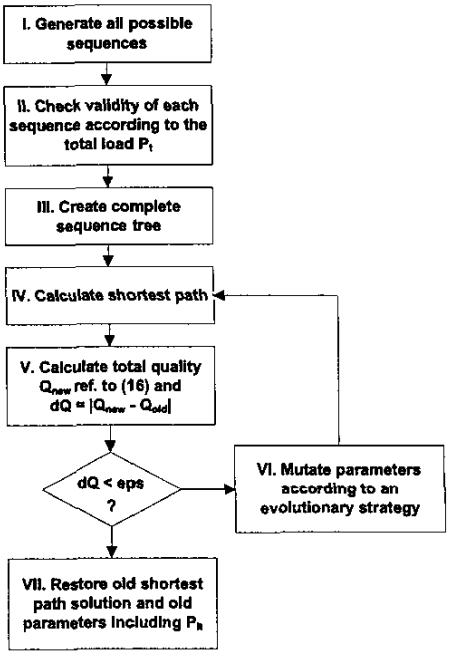

Figure 7. Algorithm to find the best fitting appliance parameter

The vertical lines in Fig. 5 symbolise branching points to connect different subsequneces $Z_1 \rightarrow Z_2$ to subsequences $Z_2 \rightarrow Z_3$ generating a sequence Γ_i, taking into account to avoid overlappings in time. The sequences are then checked to satisfy the total load trace constraint ref. to (1) and prooved according to additional constraints limitting the time periode of a complete sequence or the distance between subsequences. Invalid sequences are discarded before evaluating the quality and searching for the best matching path (cf. the dashed line in Fig. 6. After calculating the quality of each sequence, paths are created towards increasing time. Each sequence is only connected to a subgroup of next sequences due to the restriction, that non of the group could be connected to another member of the same group, to avoid skipping sequences. A sequence of a path is connected to its best neighbour, so after choosing the last best neighbour, the best way is found automaticly. A set of best matching finite state machines is analysed according to the corresponding time series optimising every finite state machine for itself. This is well done by a dynamic programming algorithm similar to the viterbi algoritm. Finally the sum of all simultaneously detected appliances has to be evaluated with fuzzy sets, to solve overlapping of events which are distributed to different finite state machines simultaneously.

4. Results

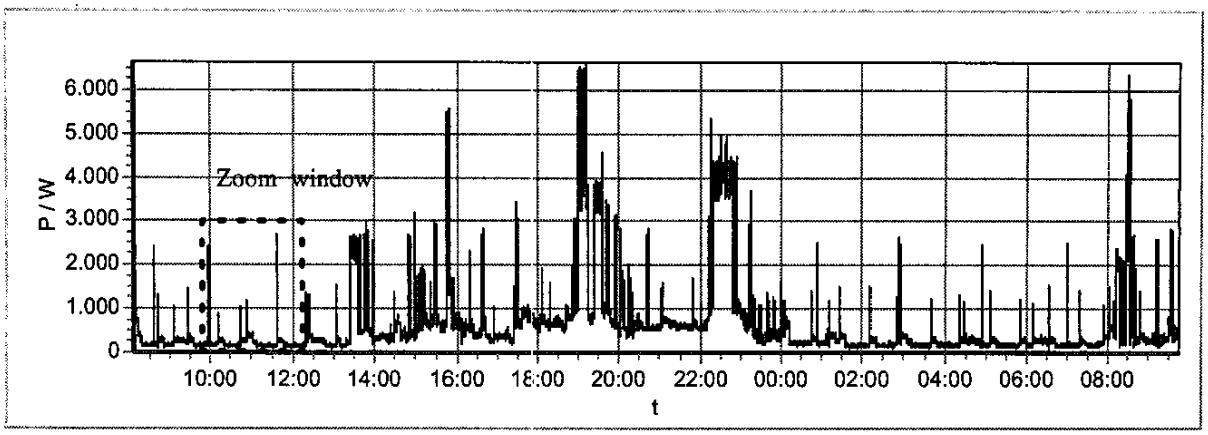

Figure 8. Total load trace of a single day, time resolution 1s (real data)

The algorithm has been tested with simulated and real data from different domestic homes with time resolution of one second. Figure 8 shows the load trace from a household with five persons.

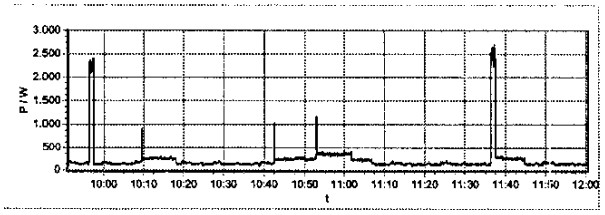

Figure 9. Zoom window from the Fig.8

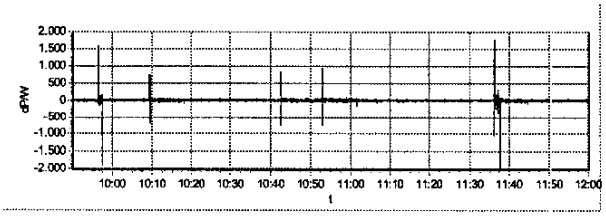

Figure 10. Difference series of Fig. 9, cf. (4)

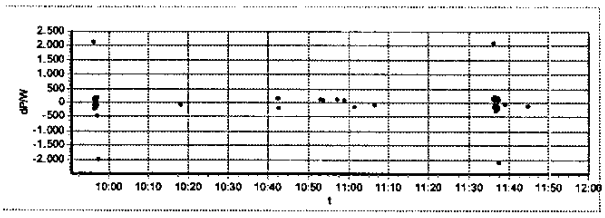

Figure 11. Detected events, cf. Fig. 9

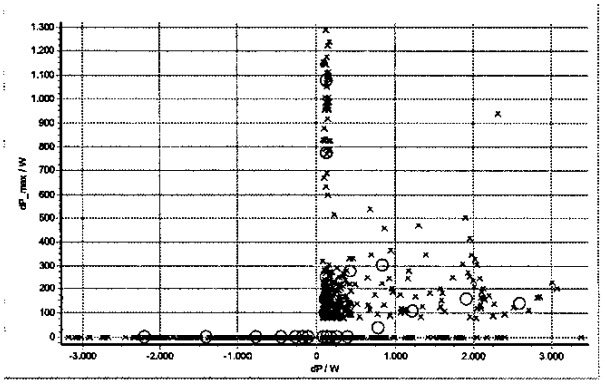

Figure 12. Cluster results from series data of a single day

The data from Fig. 8 to Fig. 12 show results from measurements using an optical sensor clamped on the already installed electricity meter (Ferraris meter) in a household. Fig. 9 shows the zoom area of Fig. 8, Fig. 10 and Fig. 11 are related to the Fig. 9. The detected events in Fig. 11 were generated using a barrier δ= 80W cf. (5). In Fig. 12 the corresponding clusters are symboliesed as circles. The most events range about -200W to +200W. Load peaks with significant power steps could be detected easily. A zoom window focused on the hours from 0:00 a.m. up to 06:00 a.m. is very suitable to find periodic loads as refridgerators and others. Results of clustered events from a single day are shown in Tab. 1, where the power step value of the cluster centre is denoted by dP. Count means the number of events assigned to the cluster, sigma denotes the standard deviation, dP_max means the boost value of a switch events and n_dP denotes the number of difference values combined to one event cf. (5). To find typical appliance patterns, data from about five to ten days has to be analysed. The results of these days are then corralated to detect the major appliances of this special household. The results show, that patterns of chief consumer load devices like refrigerators, electric flow heaters, stoves or cookers could be detected confidently. The total execution

time on a Pentium 4 personal computer (with clock frequency of 2.4 GHz) ranges between five to ten minutes.

Table 1:Clustering results

dP/W	count	sigma	dP_max	n_dP
-2057	53	419,9	0	3
-1250	45	197,0	0	3
-688	84	114,2	0	2
-328	195	66,8	0	2
-185	492	31,0	0	2
-105	731	19,1	0	2
92	723	9,8	0	2
98	75	19,8	100	4
114	51	30,5	161	4
131	458	12,4	0	2
135	43	164,0	985	5
149	29	100,7	248	4
156	48	30,3	103	4
180	239	17,8	0	2
256	106	27,3	0	2
261	38	50,4	133	3
369	22	43,1	118	4
383	12	74,3	251	6
389	42	40,6	0	2
618	11	105,1	90	4
823	18	97,8	21	3
870	8	162,1	308	4
1181	19	158,6	116	5
1630	20	147,7	66	4
2109	33	238,1	184	5
2962	6	230,3	151	5
Total	3601			

5. Summary and outlook

This special NIALM approach is designed to be integrated in modern smart home applications based on the use of standard inhouse communication techniques and a standard pc (686er or higher) to operate as a stand alone system. This automatic setup NIALM system analyses rough data form ordinary electricity meters in opposite to NIALM approaches from [4], [6], [7] and [10] using special expensive measuring equipment. With a low cost optical sensor, clamped on the existing meter, the active power consumption of a household could be measured without any intrusion into the main electrical installation of the household. The deficit of measuring information has to be compensated by more complex methods to extract patterns out of the total load data. Detecting switch events in consideration of load peaks is the only information source the system uses to find structures of unknown electrical appliances. Therefor the implemented clustering method finds typical switch states of frequently occuring appliances. A genetic algorithm combines different states to create appropriate finite state machines due to constraints of electrical end uses. The implemented genetic algorithm uses a variety of parameters to tune and adjust the detection of appliances or to adjust the algorithm to analyse different data sources. The time series data of each detected appliance is created by combining sequences containing discrete switching states. The number of valid sequences is heuristically limited with respect to execution time by limitting the maximum time duration of any sequence. Typical patterns of on-off appliances or finite state machines with less than about five different states could be detected without any apriori knowledge. After checking and validating the derived series data for each appliance simultanously, overlappings of multiple assigned switch events have to be solved with help of fuzzy sets. These methods have to be integrated next. However, additional tests are affordable to conduct a suitable and robust parameter set to detect appliances for a wide spectrum of electric appliances and create a knowledge base to store typical properties of different appliances. To detected patterns by name, a connection to a central controlled knowledge base will be necessary.

6. References

[1] Bezdek, James C.:"Fuzzy Models For Pattern Recognition, Methods That Search For Structre In Data",IEEE Press, 1991

[2] Enetics Inc, www.enetics.com

[3] Forney, G.D.: "The Viterbi Algorithm", Proceedings of the IEEE, Vol. 61, No.3, 1973, pp 268-278

[4] Hart, George W.:"Nonintrusive Appliance Load Monitoring", Proceedings of the IEEE, Vol. 80, No. 12, December 1992

[5] Marceau,M.L. ; Zmeureanu,R. "Nonintrusive Load Disaggregation Computer Program To Estimate The Energy Consumption Of Major End Uses In Residential Buildings", Energy Conversion & Management, Vol. 41, 2000, pp. 1389-1403

[6] [Margossian,B.: "Deriving End-Use Load Profiles Without End-Use Metering: Results Of Recent Validation Studies", Quantum Consult Inc.

[7] Norford, L.K.; Leeb,S.B.: "Non-Intrusive Electrical Load Monitoring In Commercial Buildings Based On Steady-State And Transient Detection Algorithms", Energy and Buidings, Vol. 24, 1996, pp. 51-64

[8] Schöneburg E., Heinzmann F., Feddersen S., "Genetische Algorithmen und Evolutionsstrategien", Addison Wesley, 1994

[9] VDN Lastenheft "Elektronischer Haushaltszähler", Draft Version 0.9 from 14.11.2003, VDN (network operator association) at The German Electricity Association VDEW, www.vdn-berlin.de

[10] Zmeureanu, Radu; Farinaccio, Linda:"Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses", Energy and Buildings 30 (1999), pp 245-259

[11] Zmeureanu, Radu; Farinaccio, Linda: "Use of soft computing techniques for the evaluation of energy performance of equipments in buildings", Department of Building, Civil and Enviromental Engineering Concordia University, Montreal, Quebec, CANADA