Non-Intrusive Appliance Load Monitoring using Genetic Algorithms

Hock, D.; Kappes, M.; Ghita, B.

Published in:
IOP Conference Series: Materials Science and Engineering

DOI:
10.1088/1757-899x/366/1/012003

Publication date:
2018

Non-Intrusive Appliance Load Monitoring using Genetic Algorithms

D Hock[1,a], M Kappes[1,b]and B Ghita[2,c]

[1] Frankfurt University of Applied Sciences, Nibelungenplatz 1,
D-60318 Frankfurt am Main, Germany
[2]Plymouth University, Drake Circus, Plymouth, Devon PL48AA, UK

[a]dehock@fb2.fra-uas.de; [b]kappes@fb2.fra-uas.de; [c]bogdan.ghita@plymouth.ac.uk

Abstract. Smart Meters provide detailed energy consumption data and rich contextual information which can be utilized to assist energy providers and consumers in understanding and managing energy use. Here, we present a novel approach using genetic algorithms to infer appliance level data from aggregate load curves without a-priori information. We introduce a theoretical framework to encode load data in a chromosomal representation, to reconstruct individual appliance loads and propose several fitness functions for the evaluation. Our results, using artificial and real world data, confirm the practical relevance and feasibility of our approach.

1. Introduction

Energy providers need to react fast and efficient on variable energy demand while maintaining grid stability – a problem which arises partially due to the large-scale use of decentralized volatile and renewable energy, such as wind and photovoltaic energy. Many energy providers aim to face this challenge by utilizing the fine-grained demand measurements of smart meters to control and monitor devices on residential level or foresee and respond to otherwise unexpected energy demand. The wide availability of load data and rapid advancements in statistical methods, akin to machine learning techniques, resulted in many concepts to extract information from aggregated load data as provided by a smart meter. A particular approach to generate sophisticated load profiles from smart meter readings, is Non-Intrusive Appliance Load Monitoring (NIALM) [1], which aims to estimate appliance level data without the intrusive deployment of sensors on individual end-uses. The idea to disaggregate the data from a single point of measurement, is cost efficient and convenient while the estimated load profiles can still provide rich information on residential households. We propose a genetic algorithm to infer appliance level load curves without a-priori information or relying on historic information. Inferring information about the energy consumption and presence of individual devices from energy demand is referred to as energy disaggregation. The usage of genetic algorithms in combination with machine learning algorithms is not new. However, while the machine learning methods are well described, many aspects of the genetic algorithm have not been studied yet. Our contribution is the chromosomal representation to solve NIALM as combinatorial optimization problem. Moreover, we present and evaluate different fitness functions and introduce a feature extraction method to reduce the number of genes of our problem set. We introduce a straightforward

approach to in-cooperate the likelihood of appliances state changes (on/off). The remainder of the paper is organised as follows. In the next section, we introduce our NIALM algorithm and identify a set of features, extracted from the aggregate load curve, to model the problem in a chromosome representation. Subsequently, we present our genetic algorithm as well as a fitness function to assess the similarity of the approximated individual loads to the original load. Furthermore, we estimate the most probable state for each individual appliance. We conclude this paper by evaluating the results using real world data.

2. Related Work

G.W. Hart [1] initiated the research on NIALM to distinguish appliances from aggregated load. The idea of NIALM as an optimization problem, which was taken up by Liang et al. [2] and Suzuki et al. [3], with soft computing methods, such as integer programming, to identify appliances. Other machine learning approaches were recently summarized by Zoha et al. [4]. Several articles introduce genetic algorithms to compute smart grid related problems: Baranski et al. [5] used genetic algorithms to identify certain appliance types by modelling the power steps of the appliance. A comprehensive overview of genetic approaches for demand side management was presented by Barbato et al. [6].

3. Energy Disaggregation

Here, we provide a brief overview of energy disaggregation and the reasoning behind our feature extraction algorithm as well as our fitness function to evaluate the feasibility of each generated solution. Colloquially speaking, a load curve consists of the sum of the loads of all active individual appliances and measurement errors. Hence, if a set of appliances and their load curves are given, the assignment to find the correct state (active/inactive) at each time can be formulated as a combinatorial optimization problem – choose the set of active appliances for all times to minimize the error – which is an NP-complete weighted set problem, similar to the knapsack problem [1]. Hart et al. [1] was the first to describe Load Disaggregation as combinatorial optimization problem as follows:

Algorithm 1. Load Disaggregation as Combinatorial Optimization Problem

```
for each t in time do
    activeAppliances ← loads[active_t == true]
    error ← error + (totalLoad[t] – sum(activeAppliances))²
end for
fitness ← error
```

Algorithm (1), adapted from Hart's [1] publication, evaluates the fitness of estimated loads using the deviation between the sum of individual loads and the total load. Under the assumption that we find the correct amplitude of each appliance, the load curve of each individual appliance can be reconstructed. We can further minimize this combinatorial problem by considering only turn on and off events instead of all measurements. *Note, that we use the term 'event' synonymous for a transition between the on and off state of an appliance.* Our reasoning here is, that all measurements without a state change do not contain any additional information since everything besides the sum of active appliance amplitudes is filtered by the algorithm. Hence, modelling the change-points is sufficient. However, when extracting the amplitude by heuristic means, we need to address another problem: Our set of amplitudes can contain measurement errors or detect several individual devices as one. Hence, there may be several cases of invalid amplitudes which should be terminated. Hart [1] introduced a problem related to this case as follows: Suppose a load curve contains the unique amplitudes 100W, 200W, 300W and 401W. If the measured total load at time t_i is 500W, the best estimate is t_i = *(200W,300W)*. If a moment later, at time t_{i+1} the measured load increases slightly to 501W, the best estimate would be t_{i+1} = *(100W,401W)*, which implies that every appliance changed state between t_i and t_{i+1}. While the problem was originally due to measurement errors, we think that addressing a transition likelihood may also help to minimize the number of invalid appliances.

3.1. Chromosome Representation

Genetic algorithms are based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent implementation of a random search to solve a given problem. The problem represented by a genetic algorithm is encoded as binary 'genes'. Whereas a string of *n* genes represents a so called chromosome, which is one solution candidate. The genetic algorithm maintains a population of *m* chromosomes, with associated fitness values. Parents are selected for reproduction on the basis of their fitness, which represents the abilities of an individual to solve the given problem. By inheriting the properties of a chromosome from parents to children, the reproduction improves the solution candidates in a population from iteration to iteration. The operations of genetic algorithms are *(i)* selection, *(ii)* crossover and *(iii)* mutation. Where *(i)* selects the parents based on the fitness, *(ii)* represents the mating of two individuals and *(iii)* introduces random modifications. In our case, the search space are all possible combinations of active appliance for each event. Hence, the chromosome is constructed as a bit string, which encodes the state of each appliance for each event. While the initial population of chromosomes is usually randomly generated, the genetic algorithms applies mutation and crossover on these chromosomes to create a population of different solutions. Since the length of the chromosome is directly related to the complexity of the genetic algorithm, we aim to limit the chromosome length by providing a set of amplitudes, which appeared in the original load curve. In order to extract these amplitudes we utilize the edge detection algorithm as introduced by Hart [1], which aims to finds step like changes. The algorithm categorizes the load curve in phases of steady and changing demand. A steady phase is defined as a specified number of measurements without change (in a given tolerance). The remaining periods, between steady phases, are defined transition phases. Each steady phase is averaged to minimize noise. The differences between two averaged steady phases result in the amplitude of one or several appliances switched at this moment.

3.2. Fitness Function

In each generation, the fitness of all individual chromosomes is evaluated and represents the ability to solve the problem, whereas a small values represents good fitness. The best candidates get selected for the reproduction process and, due to the crossover operation, inherit their genes to the next generation. Our fitness function models the likelihood for a state change, because we argue that it is more likely for an appliance to stay in the same state than to change its state. For illustration, assume a load curve with the unique amplitudes of 20W, 30W and 50W – hence, a transition from 20W to 50W, can have two possible solution sets:

a) t_i *= (20W),* t_{i+1} *= (50W)*

b) t_i *= (20W),* t_{i+1} *= (20W,30W)*

While the amplitude at t_{i+1} is equal, the transition from t_i to t_{i+1} in *a)*, can be seen as two events (a turn-off event + a turn-on event), whereas *b)* is only one event. Following this reasoning, a load curve with a total load 20W, 50W and 30W, can could be disaggregated as follows:

a) t_i *= (20W),* t_{i+1} *= (50W),* t_{i+2} *= (30W)*

b) t_i *= (20W),* t_{i+1} *= (20W,30W),* t_{i+2} *= (30W)*

Where *b)* is consists of 4 events and two appliances and *a)* of 6 events and three appliances. We assume that the sequence featuring the smallest effort is most likely to appear and utilize this fact to create a fitness function which considers this smallest effort. We model the above, using Algorithm (1) with the extension of an additional weight for the cost of an event. This weight is given by the number of state changes required for a given event and is defined by the difference of active appliances between the current and previous event. Assuming, the state t_i and t_{i+1} is given as binary string (which is the case in our genetic algorithm), the weighting can be computed using the sum of the XOR operation. The actual weight should be in range of [1, n+1], where n is the number of appliances – otherwise a constant demand would result in a fitness value of zero. Following this reasoning, the fitness functions derived from the previously mentioned combinatorial optimization problem can be defined as:

Algorithm 2. Fitness Function

for each t **in** events **do**
  activeAppliances ← loads[$gene_t$ == true]
  error ← error + (1+ω[$gene_{t-1}$, $gene_t$])*(totalLoad[t] – *sum(*activeAppliances*)*)²
end for
fitness ← error

Note, that there are still limitations in this approach: We use the unique amplitudes to build the chromosome. Hence, we cannot distinguish appliances that have the same amplitude.

4. Results

In this section, we evaluate the feasibility of our approach to reconstruct individual appliance load curves. We first use an artificially generated load curve to evaluate our proposed fitness function in optimal conditions without measurement errors. In the latter section, we repeat this process on a real world smart meter dataset and present the accuracy of our results.

Next, we evaluated a load curve with five artificial appliances, each with a unique amplitude – two appliances with permanent rectangular pattern and three single On/Off signals. We compare our fitness function, as described in the previous section to Hart's initial algorithm and the Euclidean function.

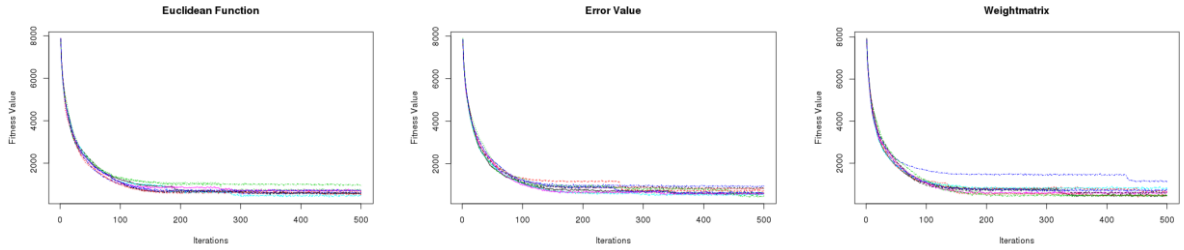

Figure 1. Comparison of Fitness Functions

Figure 1 shows the improvement of the fitness over time for each function (left to right, Euclidean distance, Algorithm (1), Algorithm (2)). The genetic algorithm was executed for 500 iterations and the best 10 chromosomes were used to evaluate the similarity to the ground truth chromosome, which we computed from the individual signals. The quality of each function has been evaluated using the Area Under Curve (AUC), which is a metric based on the true and false positive rate showing the statistical accuracy. Repeating each experiment 10 times, we received for each function a mean AUC greater 0.90. The equally good performance, even without the weighting function, is due to the fact that there is no event included were several appliances are switched at the exact same time. Hence, we were able to extract the exact values of each individual appliance with amplitudes 50W, 120W, 500W, 300W and 20W. However, this is unlikely to happen in real world data.

For our next experiment, we used the Electricity Consumption & Occupancy (ECO) data set, published by Beckel et al. [7], which provides - apart from declared exceptions - measurements in Watt with 4 decimal places and offers individual appliance readings every second (Hz). The measurements used here are real power (W) values of smart meters, showing the mean-cycle-power of a period. Figure 2 depicts one day of the original energy demand (dotted) and the expected result (line) of the genetic algorithm.

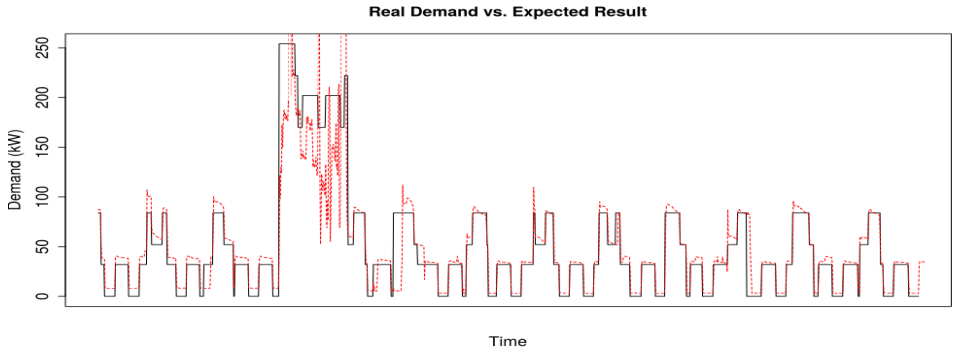

Figure 2. Original Energy Demand and Expected Result of the Genetic Algorithm

In contrast to the artificial energy demand, the real energy demand includes significant noise due to appliances in stand-by mode and characteristic saw-tooth pattern. In our experiment, with a total of four appliances (fridge, kettle, dyer and coffee machine) the noise resulted in a significant decrease of the performance for two reasons: First, the genetic algorithm considered the aggregated amplitude as possible appliance due to the simultaneous switch events and, furthermore, the saw-tooth pattern resulted resulted in several similar amplitudes; for example, while the real amplitudes of the appliances were 30W and 50W, Hart's edge detection algorithm extracted the values 80W, 50W, 30W, 40W, 20W and 10W. Using our weight as a penalty for the number of switch operations, some combinations should be excluded in favour of values the occur more often. However, since the extracted values define the gene length and larger error scope, the real energy demand still requires more computation time and, hence, it is reasonable to exclude as many of them in the pre-processing phase as possible. Our results, using measurements over 1 month (September) as provided by one of the six houses (House 1), indicate that our weighted fitness function performs significantly better with realistic demand then the previous error function. Using the same number of iterations (500), we obtained an AUC of 0.7 while the unweighed function resulted in little more than random results (mean AUC ~0.5). A closer analysis revealed that the performance of our algorithm shows huge diversity throughout the day. In the following experiment, we divided the energy demand in equal sized time windows of 180 minutes and tried to reveal the individual appliances using only 500 iterations – the experiment was carried out for 10 days and 5 repetitions for each day using the fitness function as described in Algorithms (2).

Table 1. Resulting AUC for 500 Iterations

	0:00	**3:00**	**6:00**	**9:00**	**12:00**	**15:00**	**18:00**	**21:00**
Day 1	1.00	0.70	0.55	0.60	0.63	0.56	0.54	0.50
Day 2	0.88	0.81	0.70	0.53	0.73	0.61	0.65	0.65
Day 3	1.00	0.71	0.57	0.67	1.00	0.60	0.57	0.66
Day 4	0.90	0.64	0.59	0.70	0.62	0.79	0.77	0.58
Day 5	0.99	0.50	0.53	0.59	0.92	0.54	0.55	0.68
Day 6	0.70	0.57	0.58	0.53	0.60	0.58	0.5	0.59
Day 7	0.88	0.54	0.57	0.65	0.52	0.68	0.61	0.53
Day 8	0.68	0.65	0.56	0.58	0.53	0.55	0.57	0.88
Day 9	0.68	0.54	0.54	0.69	0.81	0.60	0.61	0.57
Day 10	0.94	0.66	0.64	0.62	0.70	0.58	0.66	0.63

Table 1 shows the resulting AUC for each time window. The results indicate that the genetic algorithm has problems to disaggregate sections with many appliances stacked or with appliances that are rarely used, which need more iterations. This problem could be encountered with a dynamic number of iterations per time window, but is out of the scope of this paper.

5. Conclusion

In this paper, we provided a comprehensive overview of non-intrusive appliance disaggregation as combinatorial optimization method. We first presented a method to encode appliances in a chromosome encoding and described algorithms to evaluate the fitness of individual appliance loads generated by a genetic algorithm. The resulting best fitting chromosome can be used to read the amplitude as well as turn on and off timings of each appliance. The proposed approach does have a number of inherent limitations, such as the inability to model simultaneously running appliances with equal amplitude or to implement so called state-machines with several amplitudes, but represents also a robust alternative, given its simplicity and the fact it does not require historical information. The presented experiments aimed to evaluate the feasibility of genetic algorithms for NIALM in general and point at several challenges – in future work, we aim to test several parameters, such as customized mutation and crossover operations, which can heavily influence the performance of genetic algorithms, but were out of scope this work.

References

[1] Hart G W (1992). Nonintrusive appliance load monitoring *Proc. of the IEEE* **80** 1870-91

[2] Liang J, Ng K and Cheng J 2007 Identifying appliances using load signatures and genetic algorithms *ICEE*

[3] Suzuki K, Inagaki S, Suzuki T, Nakamura H and Ito K 2008 Nonintrusive appliance load monitoring based on integer programming *IEEE SICE* **2008** 2742-47

[4] Zoha A, Gluhak A, Imran M and Rajasegarar S 2012 Non-intrusive load monitoring approaches for disaggregated energy sensing: a survey *Sensors* **12** 16838-66

[5] Baranski M and Voss J 2004 Genetic algorithm for pattern detection in nialm systems *IEEE SMC* **4** 3462-68

[6] Barbato A and Capone A 2014 Optimization models and methods for demand-side management of residential users: A survey *Energies* **7** 5787-824

[7] Beckel C, Kleiminger W, Cicchetti, Staake T and Santini S 2014 The ECO data set and the performance of non-intrusive load monitoring algorithms *ACM BuildSys* **2014**