# Unsupervised Power Profiling for Mobile Devices

Mikkel Baun Kjærgaard and Henrik Blunck

Aarhus University, Denmark
{mikkelbk,blunck}@cs.au.dk

**Abstract.** Today, power consumption is a main limitation for mobile phones. To minimize the power consumption of popular and traditionally power-hungry location-based services requires knowledge of how individual phone features consume power, so that those features can be utilized intelligently for optimal power savings while at the same time maintaining good quality of service. This paper proposes an unsupervised API-level method for power profiling mobile phones based on genetic algorithms. The method enables accurate profiling of the power consumption of devices and thereby provides the information needed by methods that aim to minimize the power consumption of location-based and other services.

**Keywords:** power profiling, mobile devices, power consumption.

## 1 Introduction

Today, mobile phones include more and more positioning and sensor technologies such as GPS, cellular positioning, accelerometers and compasses. This development enables location-based services that aim to provide high-end user experiences, such as location-based search, games, sports trackers, social networking and activity recognition [12]. However, the power consumption of such location-based services is in many cases above 1 watt, which is twenty times higher than a mobile phone's standby consumption and thus significantly shortens the battery lifetime [9]. The high consumption requires frequent battery recharges which is often inconvinient, or, for instance in deserted areas, not possible. Therefore, power consumption inhibits the use of many promising always-on location-based services. The problem can only to a small extend be addressed by lowering hardware consumption because it is the total use of the hardware by software that is the problem in the same manner as faster CPUs alone do not lead to fastest answers if an inefficient algorithm invokes the instructions on the CPU. Our previous studies [8,10,11] have shown that power savings are possible if positioning and sensor technologies are managed in a smarter way: up to 73% for a continuous moving device and up to 95% for a periodically moving device.

To successfully minimize power consumption requires knowledge of how much power a specific phone feature consumes and how fast it powers on and off. To understand the power consumption of mobile phones one could as a first step consult their specifications (e.g. [13]). However, these will often not give the

A. Puiatti et al. (Eds.): MobiQuitous 2011, LNICST 104, pp. 138–149, 2012.

full picture, because values are missing (e.g. the power consumption for CPU operations) and dynamic aspects are not considered. The dynamic aspects are due to that features do not instantly power on or off, e.g., a 3G radio needs several seconds to power on before it is able to send or receive data. Similarly, also the radio powering off does not happen instantly, due to a prior initiation of a proper disconnection, maybe preceded by a short period of waiting for a potential next radio request. Therefore, the power consumed for sending or receiving data cannot be simply modeled as a static value, which is applied just for the span of the actual sending period. Similar holds for other phone features, e.g. for the in-phone GPS during the acquisition of a positioning fix. A solution for accurately modeling dynamic aspects of power consumption is via power profiling a device to learn the dynamic behavior of its features. However, a manual approach to power profiling has the drawback that it does not scale, considering that many of the different mobile platforms and models currently existing may soon be outdated and replaced by newer models. Even worse, a power model may even sooner be invalidated just by a software upgrade that alters the phone model's power profile. E.g., we observed for Symbian phones, that power consumption for location based services changes by more than a factor of two with the OS version, which is amongst others related to how often GPS assistance data is requested over the cellular network in the different OS versions.

Efforts to manually power profile recent smart phones can be found in Carroll et al. [5], Rice et al. [15] and Zhang et al. [16], all of which employed external hardware, and our earlier work [11], which utilized software APIs. To use a power model proactively it must be able to predict future power consumption; this is enabled by our model based on conditional functions presented in [11], whereas the linear model proposed by Dong and Zhong [6] and Zhang et al. [16] depends directly on system level metrics inhibiting prediction, since the model can not over time prescribe changes of the metrics and thereby of the modeled power consumption. Zhong et al. [6] recognize the need for unsupervised power profiling but also do not provide a model that enables prediction.

This paper proposes PowerProf, an unsupervised method for power profiling mobile phones based on genetic algorithms. Genetic algorithms, operating on profile measurements collected on the phone, are utilized to estimate a power model consisting of a set of conditional functions. This method enables accurate profiling of the power consumption on a wide selection of devices and thereby provides the information needed by methods that reduce the power consumption of location-based services. The proposed method has been evaluated on several generations of phones (N97, N8, C7) and has generated models that can predict the power consumption for the mentioned devices with a high accuracy.

## 2 Related Work

The power consumption of smart phones has been studied from several angles to provide measurement tools, analysis results and models of power consumption. Rice et al. [15] propopes a mobile measurement tool for decomposition of the power consumption of smart phones by designing an artificial measurement battery. Carroll et al. [5] provide an analysis based on external measurement equipment and Kjærgaard et al. [11] provides an analysis of the power consumption of Nokia phones using measurements from the internal battery interface.

Related work on modeling power consumption can be classified according to several dimensions. Firstly, according to how the measurements are collected for building the model: Measurements can either be collected using *external* equipment or using an *internal* smart battery interface, if existent, and the respective choice has a direct impact on the requirements for building new models. Secondly, according to what type of information the model is built from. A model for power consumption can be deduced from i) measurements of system *utilization*, e.g., disk and processor statistics available from the operating system, from ii) power consumption measurements per *system call* made to the operating system, or from iii) power consumption measurements per *API call* made in a specific programming language. Thirdly, according to what type of information the model can provide. The model can enable either solely the *estimation* of the current power consumption, or it can allow additionally for the *prediction* of the power consumption ahead of time. Fourthly, according to how the model is constructed: is it a human supervised process or an unsupervised process performed entirely by a software component. Table 2 provides a summary of related work according to these four dimensions.

| | Kjærgaard [11] | PowerTutor [16] | Sesame [6] | Pathak [14] | PowerProf |
|---|---|---|---|---|---|
| External | | x | | x | |
| Internal | x | x | x | | x |
| Utilization | | x | x | | |
| System-call | | | | x | |
| API-call | x | | | | x |
| Estimation | x | x | x | x | x |
| Prediction | x | | | | x |
| Supervised | x | | | | |
| Unsupervised | | x | x | x | x |

In this paper we propose PowerProf that utilizes measurements from the internal battery of a phone enabling models to be built on user demand, whenever a new model is needed, e.g. for a new phone type or after a software upgrade that might alter the power management and thus the power profile of the phone. Dong and Zhong [6] compared the accuracy of power models constructed based on measurements from internal battery interfaces versus external equipment and found that models constructed utilizing internal battery interfaces where only marginally less accurate than models built from external equipment. PowerProf builds models that captures the phone's power consumption per API call issued in the utilized programming language. This is enabled by utilizing a program language API to the battery interface. This API solution enables in-code placement of power consumption measurements with explicitly defined start and end times, which enables optimal synchronization between code timing and measurements. This in turn allows all code to be placed at the application layer, thus avoiding i) changes to the operating system such as those needed for detecting system calls [14], as well as avoiding ii) a strong dependence on the operating system to read out runtime statistics [6,16]. This also ensures that the resulting PowerProf-built models capture the logic of a program language interpreter or virtual machine that can have a large impact on the power consumption as they, e.g., define how resources are allocated and deallocated. Furthermore, the models built by PowerProf enable prediction which allows, e.g., sensor scheduling algorithms to make decisions comparing the predicted power consumption of different possible actions. The method of PowerProf aims –in agreement with related work– for unsupervised building of models, but does not require any predefined thresholds as earlier work uses to separate different power states [14]; e.g., PowerProf's modelling is able to identify new states without requiring from them, that they change the power consumption by more than a given threshold.

## 3 Automatic Power Profiling

In this section we will present the proposed method PowerProf, which is enabled by the increasing availability of smart battery interfaces on mobile devices. These interfaces include software APIs that provide access to measurements of voltage, current and other battery statistics [6]. The PowerProf method requires that a set of training measurements is collected to build models from. During collection, relevant phone features are exercised sequentially and independently of each other. Note, that the collection of these measurements does not require any user interaction and can thus be carried out when the device is not needed by the user for other tasks. Alternatively, the training can be performed as an installation step, thus making the process invisible to device users.[1] The training might then be reapplied if an application detects that an interpreter or operating systems has been updated to a new version or has been reconfigured, e.g., if the GPS is reconfigured to use (a different scope of) GPS assistance data.

*Overall System Structure.* The structure of and the individual steps when applying the PowerProf system are illustrated in Figure 1. First, a request is sent to a software API that interfaces with the smart battery to start providing time-stamped power measurements (1). Relevant phone features are independently exercised and time-stamps for the starts of features and for the return of results are logged (2). Measurements are processed and prepared, and additionally directly observable characteristics are determined, e.g., background power

[1] Also, accuracy in power modeling may be further increased through training in several environment types, e.g. indoors and outdoors, which differ in reception properties, and thus also in time and energy required for obtaining, e.g. GPS fixes or GSM communication. Detecting the type of environment, a phone is currently in, can be done in real-time on-device, e.g. via analysis of GPS reception quality data [4].

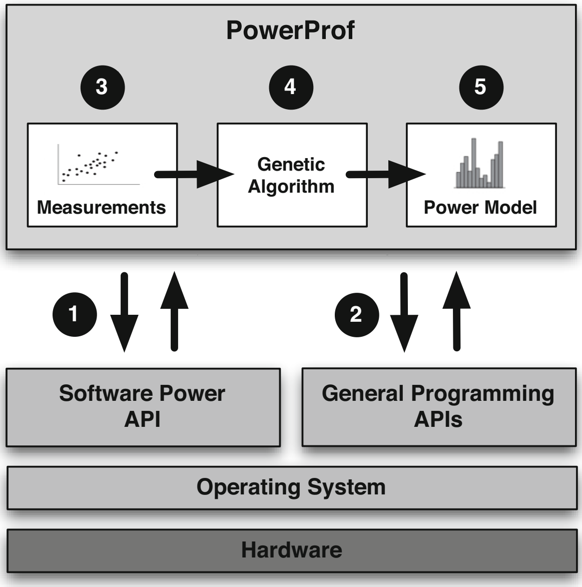

**Fig. 1.** Steps of the PowerProf system from measuring to generating power models

consumption (3). A genetic algorithm [7] is used to search for optimal parameter values for the power model (4). This step is delegated to a server as an extensive amount of processing is involved in running the genetic algorithm. The fitness function of the algorithm calculates the difference between the power consumption as predicted by the current model and as actually measured on the device. Finally, the parameter values resulting from minimizing the fitness function are used as input to the final power model (5).

*Power Model Design.* A power model, as built by PowerProf, is given by one conditional function per phone feature. Such a function models the feature's power consumption in dependence of the input parameter $t$, which resembles the time span, since the feature was last requested via a programming API call. Each conditional function is restricted to four power states, defined by four time parameters $t_1$, $t_2$, $t_{result}$ and $t_{end}$ where the first two parameters are free variables and the second two are the time that it takes the function to return, and the time when the power consumption finally drops to background consumption, respectively. The two free variables $t_1$ and $t_2$ defines the transition times between state one and two, and state three and four, respectively. The power consumption for each of the four states is defined by a parameter $p_1$, $p_2$, $p_3$ and $p_4$, respectively. The form of the resulting conditional function is given in Equation 1, and Figure 2 illustrates an example of the resulting power profile for such a conditional function. The restriction to four parameters is applied to decrease the search space when fitting the functions to actual measurements.

Furthermore, this restriction has been chosen based on the number of power states identified in our earlier work [11]. If the same function is exercised several times in the training phase, average values are used for $t_{result}$ and $t_{end}$. The drop to background consumption is detected, when the average power consumption over three consecutive measurements is below $\mu_{background} + \sigma_{background}$, the sum of the mean and the standard deviation of the background consumption, respectively. Note, that the this sum is well separated from the power consumption when using a phone feature, since the latter use considerably more power than experienced by fluctations in background consumption.

$$\mathcal{P}(t) = \begin{cases} p_1 & \text{if } t <= t_1 \\ p_2 & \text{if } t > t_1 \text{ and } t <= t_{result} \\ p_3 & \text{if } t > t_{result} \text{ and } t <= t_2 \\ p_4 & \text{if } t > t_2 \text{ and } t <= t_{end} \end{cases} \tag{1}$$

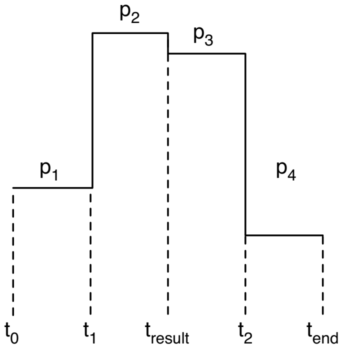

**Fig. 2.** Power Model for a single phone feature

*Power Model Construction.* The reason for choosing a genetic algorithm for finding model parameters, i.e. an algorithm which searches for an optimal solution based on the principles of natural evolution, is that it is a well known technique for finding approximate solutions to intractable parameter optimization problems [7] and especially to real world optimization problems on the basis of noisy data, e.g. noisy power consumption measurements.

To apply genetic algorithms, one needs, firstly, to define the format of the *chromosome* that has to be evolved towards a solution using the principles of natural evolution. Secondly, a fitness function is applied iteratively to evaluate the last generation's chromosomes, which are developed by the genetic algorithm, to only let the fittest survive. Our chromosomes consist, as shown in Figure 3, of the parameters of the power model that need to be found. Thus, a chromosome consists of six entries for each phone feature to represent the six unknowns in Equation 1: $t_1$, $t_2$, $p_1$, $p_2$, $p_3$ and $p_4$. Note, that $t_{result}$ is recorded by in-code measurement as the time span from API call till the return of the call, and $t_{end}$ is found as the time till the power drops down to background consumption.

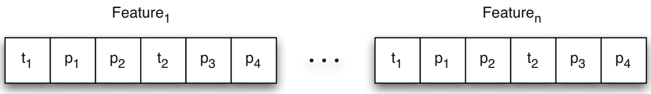

**Fig. 3.** Chromosome consisting of parameters for 1...$n$ features

To evaluate the fitness of a developed chromosome $\mathcal{C}$, we use the $n$ training measurements, which consist of the time-stamps $t_i, i = 1, \ldots n$ during which phone features were exercised and the corrsesponding measurements of power consumptions $m_i$ on the phone. From a chromosome $\mathcal{C}$ we parametrise a power model $\mathcal{P}_\mathcal{C}$ and use it together with the training time-stamps to predict the power consumption during each measurement phase $i$ for $i = 0 \ldots n$, so that each predicted power value is aligned in time with its actual measurement. The fitness of the chromosome is then found as the inverse sum of absolute differences as defined in Equation 2. Therefore, as the differences decrease, the fitness value will increase.[2]

$$fitness = \frac{1}{\sum_i^n |\mathcal{P}_\mathcal{C}(i) - m_i|} \tag{2}$$

## 4 Evaluation

To evaluate PowerProf we have implemented and tested the system on three different mobile device types. In our evaluation we consider the accuracy of the respective power models and the dependence of the accuracy on the number of the training iterations and on the number of generations that the genetic algorithm is allowed for evolving the chromosomes.

PowerProf has been implemented in python for S60 [2] to run on different mobile devices supporting the Symbian operating system. The implemented PowerProf system consists of the components shown in Figure 1. To interface with the smart battery interface of the devices we use a python software API that interfaces with the Nokia energy profiler [1]. The Nokia energy profiler provides power measurements at the same sample rate of 4 Hz as provided by the python software API. To implement the genetic algorithm component we utilize the Pyevolve software library for genetic algorithms [3], using the library's standard parameters, which specify a population size of 80 individuals, a mutation rate of 2%, a crossover rate of 80% and that a ranking selector is used. To enable our evaluation of the dependency between the accuracy and the number of training measurements and evolved generations, the genetic algorithm part has for the evaluation been implemented and run on a server.

To evaluate the PowerProf method, we have collected three datasets for each of the three Nokia Phones N97, N8 and C7. In each dataset we iterate three

[2] We decided against the alternative of using squared differences, since the latter strategy would result in power models that attempt to fit (potentially irregular and random) spikes in the power consumption, rather than more regular, and thus more likely reoccuring, patterns.

times over the functions for different phone features, where each iteration takes around five minutes. Table 1 lists the exercised phone features. Note, that for various applications, further phone features may be of interest. In such cases, the respective application developer should extend PowerProf's feature set and its set of training measurements accordingly. Besides the PowerProf program no other programs were running on the phone during data collection. The measurements were collected outdoors, and prior to the measurements a single GPS fix was obtained, and with it the current A-GPS data, so that we subsequently power profile only the GPS device, provided that the phone is configured not to obtain A-GPS data via the radio for every invocation. Note, that the latter A-GPS data, once obtained, is valid for several hours. The reason for choosing a measurement duration as long as five minutes is that we need to separate the usage of individual features by at least thirty seconds to be able to analyse the time it takes the individual features to power off. The thirty seconds as lower limit was selected based on prior work [11].

**Table 1.** Functions for features exercised

| Feature | Scenario |
|---|---|
| WiFi | WiFi Scanning at 1 Hz during ten seconds to provide measurements for WiFi positioning |
| Accelerometer | Acceleration measurements for ten seconds at the hardware-provided frequency |
| Compass | Compass measurements for ten seconds at the hardware-provided frequency |
| GSM | GSM scanning at 1 Hz during ten seconds to provide measurements for GSM positioning |
| CPU | Calculate $\pi$ with 1500 decimals |
| HTTP | Read 'www.google.com' using an HTTP connection |
| GPS | Receive one GPS position |

Before presenting our evaluation we provide with Figure 4 an exemplary comparison of the predictions of a model, generated by PowerProf, and real power measurements for a C7 phone. Generally, as indicated by the overlap of the two graphs, the predictions match the real measurements with a few exceptions. One such exception is, that the HTTP feature is predicted to power off too early, which is due to that both power on and off time spans vary for this feature to some extent between calls, due to changes in the network characteristics.

To analyse the performance of PowerProf we evaluate it based on the prediction error defined as the difference between the predicted power consumption and the measured one. We use the three datasets for each phone type, and to devide this data into training and test data we perform three-fold cross validation where each fold only contains data from one dataset. To evaluate specific aspects of our approach, we also vary i) the number of iterations included in the datasets used for training and ii) the number of generations that the chromosomes had to evolve. Furthermore, we have repeated each genetic evolution three times, so

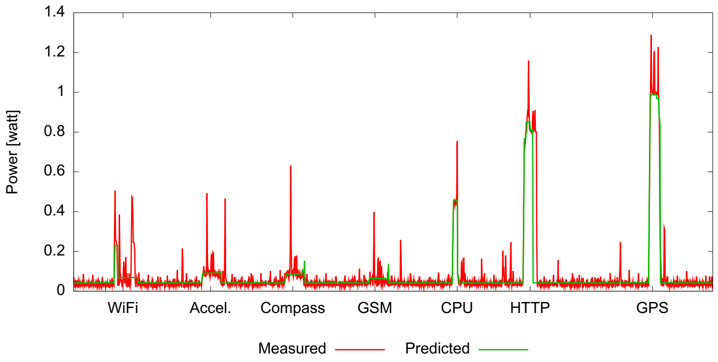

**Fig. 4.** Comparison for a C7 phone of model predictions and real measurements

that the results presented here are less dependent on the random elements of the evolution process.

Figure 5 shows a cumulative distribution over the prediction errors in each of the three datasets for each device type. The best performance is achieved for the C7 and N8 devices. For the N97 datasets the performance is a bit worse with a higher degree of large errors, the two reasons having been identified as the background consumption being less stable for the N97 phones, and the power consumption being often more spiky than for C7 and N8 devices.

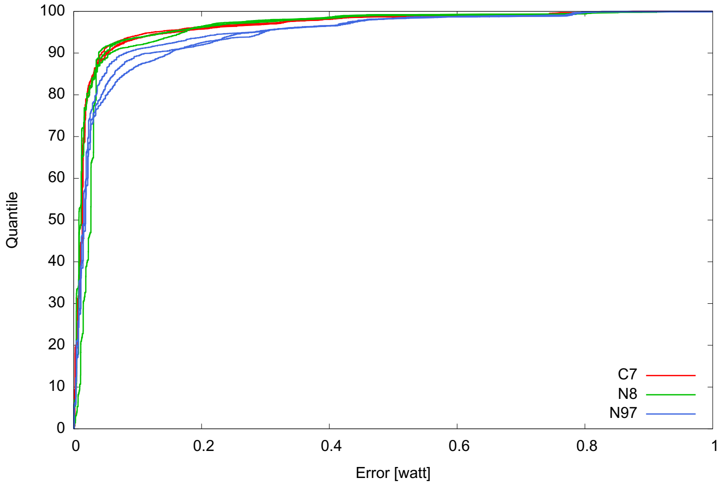

**Fig. 5.** Cummulative distributions for model accuracy after 200 generations

To evaluate how many generations are necessary for our genetic algorithm to evolve stable chromosomes, we list in Table 2 results in terms of prediction error of the modeled power consumption for chromosomes evolved for a varying

**Table 2.** Results with different number of generations

| | **25 gen.** [watt] | | **100 gen.** [watt] | | **200 gen.** [watt] | | **400 gen.** [watt] | |
|---|---|---|---|---|---|---|---|---|
| | *Median* | *95-quantile* | *Median* | *95-quantile* | *Median* | *95-quantile* | *Median* | *95-quantile* |
| C7 | 0.013 | 0.159 | 0.013 | 0.148 | 0.013 | 0.145 | 0.013 | 0.145 |
| N8 | 0.012 | 0.182 | 0.012 | 0.171 | 0.012 | 0.172 | 0.012 | 0.171 |
| N97 | 0.015 | 0.298 | 0.015 | 0.290 | 0.015 | 0.290 | 0.015 | 0.290 |

**Table 3.** Results with different number of iterations for 200 generations

| | **1 iteration** [watt] | | **2 iterations** [watt] | | **3 iterations** [watt] | |
|---|---|---|---|---|---|---|
| | *Median* | *95-quantile* | *Median* | *95-quantile* | *Median* | *95-quantile* |
| C7 | 0.013 | 0.155 | 0.013 | 0.154 | 0.013 | 0.145 |
| N8 | 0.012 | 0.177 | 0.012 | 0.166 | 0.012 | 0.172 |
| N97 | 0.015 | 0.318 | 0.015 | 0.294 | 0.015 | 0.290 |

number of generations.[3] Generally, the improvement of the models with additional generations is more significant for the 95-quantile than for the median prediction errors: To obtain an accurate model for the power consumption for the respective phone model, which can only be insignificantly improved in further generations, 25 (resp. 200) generations seem sufficient, when considering (the granularity of) the median (resp. the 95-quantile) error measurements.

We also considered the number of training iterations used. Each iteration contains one set of training measurements for each of the features, listed in Table 1. The results with one, two and three iterations, as listed in Table 3, indicate a significant gain in model accuracy by collecting more than one iteration for the 95-quantile, though not for the median error. The gain is caused by the extra training measurements increasing the model's generality and therefore increasing its ability to predict the test measurements.

Finally, we have also evaluated the model's performance in a real location-based-service use-case scenario, featuring more frequent and overlapping feature usage. For this test, we considered a deployment dataset of our system $EnTracked_T$ [10], that has been designed to provide energy-efficient user position and trajectory tracking. The deployment was conducted with a N97 phone configured to use A-GPS seldom, and during the experiment the phone logged both time-stamped function calls within $EnTracked_T$ as well as power consumption. In the following, we compare the power consumption predicted by a PowerProf-built model for the N97 phone with the measured power consumption. Figure 6 shows both the predicted and measured power consumption during an experiment where a person was walking in a city area while being position tracked by $EnTracked_T$. As we can observe from the figure, the model produced by PowerProf is able to capture the overall power consumption, whereas occasional spikes are not accurately captured. This is expected as the model will

[3] Note, that within PowerProf and given settings, computing one generation took on average 1.2 seconds on a PC, featuring a Intel X3430 processor and 8GB RAM.

be unable to capture non-repetitve phenomena. The overall prediction accuracy for both the previously described experiment and another one, where the target person was driving in a car, exhibited a median error of 0.143 watt and a 95th quantile error of 0.313 watt. This indicates that PowerProf models power consumption accurately and that the 95-quantile and worst-case error can be properly predicted from the PowerProf training measurements, whereas the average error increased with the more frequent and overlapping feature usage as it is to be expected when using location based services ’in the wild’.

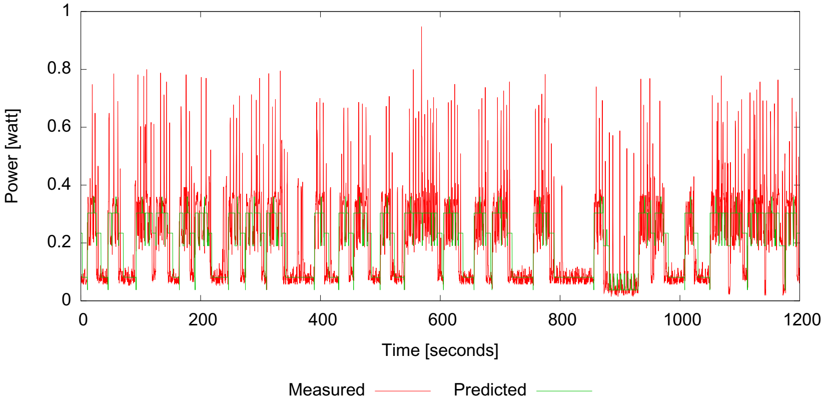

**Fig. 6.** Comparison of predicted and measured power consumption during a try out of the $EnTracked_T$ system

## 5 Conclusions

In this paper we presented PowerProf, an unsupervised API-level method for power profiling mobile phones based on genetic algorithms. The method has been evaluated for building models of common phone features utilized by location-based services. Evaluation results provide evidence that the system for the three tested phone types achieves predictions with high accuracy, with average errors of 0.012 watt for the median and 0.145 watt for the 95th quantile. Evaluation results for a real use-case scenario with the energy-efficient position tracking system $EnTracked_T$ showed an increase in the median error to on average 0.143 watt, but only a slight increase of the 95th quantile error.

Future work items are to evaluate whether the profiling accuracy of PowerProf can be further increased by extending the search space by adding more states to each conditional feature function and by experimenting with different strategies and parameters for the genetic algorithms PowerProf employs. Additionally, we would like to consider the impact on model accuracy of applying averaging filters to remove non-periodic power spikes.

**Acknowledgements.** The authors acknowledge the financial support granted by Nokia and the Danish National Advanced Technology Foundation for the project Galileo: A Platform for Pervasive Positioning under J.nr. 009-2007-2. We thank Niels Aleksander Mårup Christensen for helping with the implementation and Morten Videbæk Pedersen for providing access to a Python API for the Nokia Energy Profiler.

## References

1. Nokia - Energy Profiler (2008), `http://www.nokia.com`
2. Python for S60 (2008), `http://sourceforge.net/projects/pys60`
3. Pyevolve (2011), `http://pyevolve.sourceforge.net`
4. Blunck, H., Kjærgaard, M.B., Toftegaard, T.S.: Sensing and Classifying Impairments of GPS Reception on Mobile Devices. In: Lyons, K., Hightower, J., Huang, E.M. (eds.) Pervasive 2011. LNCS, vol. 6696, pp. 350–367. Springer, Heidelberg (2011)
5. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: Proc. 2010 USENIX Annual Technical Conference, pp. 1–12 (2010)
6. Dong, M., Zhong, L.: Self-constructive high-rate system energy modeling for battery-powered mobile systems. In: Proc. 2011 Intl. Conf. Mobile Systems, Applications, and Services, pp. 335–348 (2011)
7. Kantardzic, M.: Chapter 10 - Genetic Algorithms. Data Mining: Concepts, Models, Methods, and Algorithms. John Wiley & Sons (2003)
8. Kjærgaard, M.B.: On Improving the Energy Efficiency and Robustness of Position Tracking for Mobile Devices. In: Sénac, P., Ott, M., Seneviratne, A. (eds.) MobiQuitous 2010. LNICST, vol. 73, pp. 162–173. Springer, Heidelberg (2012)
9. Kjærgaard, M.B.: Minimizing the Power Consumption of Location-Based Services on Mobile Phones. IEEE Pervasive Computing 11(1), 67–73 (2012)
10. Kjærgaard, M.B., Bhattacharya, S., Blunck, H., Nurmi, P.: Energy-efficient trajectory tracking for mobile devices. In: Proc. 9th Intl. Conf. Mobile Systems, Applications, and Services, pp. 307–320. ACM (2011)
11. Kjærgaard, M.B., Langdal, J., Godsk, T., Toftkjær, T.: Entracked: energy-efficient robust position tracking for mobile devices. In: Proc. 7th Intl. Conf. Mobile Systems, Applications, and Services, pp. 221–234 (2009)
12. Küpper, A.: Location-Based Services: Fundamentals and Operation. Wiley (2005)
13. Nokia. S60 Platform: Effective Power and Resource Management. Nokia (2007)
14. Pathak, A., Hu, Y.C., Zhang, M., Bahl, P., Wang, Y.-M.: Fine-grained power modeling for smartphones using system call tracing. In: Proc. of the Sixth European Conference on Computer Systems, pp. 153–168 (2011)
15. Rice, A.C., Hay, S.: Decomposing power measurements for mobile devices. In: Proc. 8th Annual IEEE Intl. Conf. Pervasive Computing and Communications, pp. 70–78 (2010)
16. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Mao, Z.M., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proc. 8th Intl. Conf. Hardware/Software Codesign and System Synthesis, pp. 105–114 (2010)