Adaptive context-aware energy optimization for services on mobile devices with use of machine learning considering security aspects

1st Piotr Nawrocki
Department of Computer Science
AGH University of Science and Technology
Krakow, Poland
piotr.nawrocki@agh.edu.pl

2nd Bartlomiej Sniezynski
Department of Computer Science
AGH University of Science and Technology
Krakow, Poland
bartlomiej.sniezynski@agh.edu.pl

3rd Joanna Kolodziej
Department of Computer Science
Cracow University of Technology
Krakow, Poland
jokolodziej@pk.edu.pl

4th Pawel Szynkiewicz
Research and Academic Computer Network
Warsaw, Poland
pawel.szynkiewicz@nask.pl

Abstract—In this paper we present an original adaptive task scheduling system, which optimizes the energy consumption of mobile devices using machine learning mechanisms and context information. The system learns how to allocate resources appropriately: how to schedule services/tasks optimally between the device and the cloud, which is especially important in mobile systems. Decisions are made taking the context into account (e.g. network connection type, location, potential time and cost of executing the application or service). In this study, a supervised learning agent architecture and service selection algorithm are proposed to solve this problem. Adaptation is performed online, on a mobile device. Information about the context, task description, the decision made and its results such as power consumption are stored and constitute training data for a supervised learning algorithm, which updates the knowledge used to determine the optimal location for the execution of a given type of task. To verify the solution proposed, appropriate software has been developed and a series of experiments have been conducted. Results show that due to the experience gathered and the learning process performed, the decision module has consequently become more efficient in assigning the task to either the mobile device or cloud resources. In face of presented improvements, the security issues inherent within the context of mobile application and cloud computing are further discussed. As threats associated with mobile data offloading are a serious concern, often preventing the utilization of cloud services, we propose a more security focused approach for our solution, preferably without hindering the performance.

Index Terms—adaptation, context-aware system, energy optimization, security, machine learning, Mobile Cloud Computing

I. INTRODUCTION

The rapid development of mobile devices and the growing importance of applications and services that run on these devices has resulted in a need to pay more attention to the quality parameters associated with the use of such solutions. So far, in the context of the operating quality of applications and services of this type [1], primarily QoS (Quality of Service) parameters were considered that cover the network connection status (including delay, jitter and bandwidth) and QoE (Quality of Experience) parameters that show the level of user satisfaction with a given application or service. It should be noted that in the context of mobile devices using battery power, a prerequisite for maintaining the highest level of both QoS and QoE parameters is ensuring the longest possible uptime of the device. It can even be claimed that ensuring the longest possible uptime of the mobile device (alongside the applications and services running on it) from the point of view of conserving power is in fact a QoE parameter because it increases the level of user satisfaction with mobile applications and services.

Increasing mobile device uptime is possible through optimizing power consumption while preserving the best possible quality parameters of mobile services and applications. Such optimization can be implemented at the software or hardware levels and should take into account the context in which the mobile device operates, including network connection quality, location and potential time and cost of executing the application or service. The use of context information may allow the adaptation of mobile services and applications to prevailing conditions in order to improve the quality parameters (including execution time) and to optimize power consumption. The adaptation process in the context of optimizing power consumption can be implemented with respect to the mobile device itself and also to the mobile services and applications running on the device. Methods allowing for adaptations of this type often use remote resources such as cloud computing (the Mobile Cloud Computing concept [2]) or another mobile device with appropriate resources (the cloudlets concept), to which applications/services or their components are offloaded in order to optimize the operation of the mobile device [3]. The choice of when and what to offload from the mobile device can be made offline during the software development process or dynamically when the device is working. Adaptation makes

it possible to reduce the time and costs of executing applications/services on mobile devices and to optimize their power consumption online.

In this paper we present our original concept for an adaptive system for optimizing the power consumption of mobile devices using context-based and machine learning mechanisms. Having analyzed existing solutions, we found out that there are no systems that would enable the dynamic, online adaptation of mobile applications/services by using machine learning algorithms while simultaneously accounting for the context in which the mobile device is located. A few articles describe solutions that use machine learning algorithms but these have some limitations such as requiring the use of models developed previously in offline mode or not taking into account specific applications/services when optimizing the power consumption of a mobile device. Our innovative solution works fully online on the mobile device being optimized and enables dynamic adaptation using the context in which applications/services are executed on the mobile device. It enables the costs and time required to execute mobile applications/services to be reduced and helps to optimize the power consumption of the mobile device.

The classic approach to online learning is based on reinforcement learning [4]. Our solution is based on supervised learning, like in [5], in which it is executed offline. We show that it is possible to apply it online. Instead of preparing the training data set once at the beginning, the training data can be automatically extended and the learned knowledge updated while it is in use, just like in reinforcement learning. It is a novel approach to online learning. Our results from other domains demonstrate that fewer trials are required to find a good policy than in the case of reinforcement learning, especially in a complex environment (see e.g. [6]). This research shows that this type of learning is applicable to practical problems that are encountered in mobile computing. As a result, knowledge specific to every device may be efficiently learned locally in this device even while it is in use. There is no need to prepare a separate strategy for every device type offline. This also resolves scalability and confidentiality issues.

As stated previously, our solution bases firmly on the parading of Mobile Cloud Computing *(abbrev. MCC)*, a novel design combining cloud computing and mobile devices. MCC extends the capabilities of mobile devices, allowing them to run rich, mobile applications by providing a scalable storage, computational power and resource availability. Despite its numerous advantages and growing popularity, mobile cloud still has its limitations and problems, the most significant being security-related issues. Mobile clouds are vulnerable to various attacks such as Distributed Denial of Service (DDoS) or Man-in-the-middle (MITM), possible ramifications of which include system failure, infection of malicious codes or private data leakage or loss. Those are the challenges the designers of mobile cloud services have to face. From our point of view, mobile cloud is a third party resource which security levels are independent of us. We recognize that there is no possibility to ensure that a given service adheres to the required standards of

security measures. In this work we did not focus on developing and presenting a security architecture for mobile clouds. To address the security issue we extended our original solution to take into account additional parameters, i.e. resources trust levels and tasks security requirement levels. The aim is to minimize the energy consumption without compromising security requirements for task execution.

The structure of the article is as follows: Section II presents the analysis of research in the field of power optimization in the context of mobile devices, Section III presents solutions in the field of machine learning on mobile devices, Section IV describes the adaptive power optimization system developed for mobile devices using context information, Section VI introduces the results of the experiments conducted and Section VII contains conclusions.

II. RELATED WORK

Issues related to power management are becoming increasingly relevant, *inter alia* in the context of modern distributed systems, including those using virtualization and cloud computing In [7], the authors present an analysis of power-saving techniques and examine the capabilities of machine learning in automatic power management systems. However, the article lacks a broader discussion of machine learning algorithms and aspects of possible adaptation. The analysis only considers desktop and similar systems and does not cover mobile devices, which have become an important part of modern distributed systems.

The power consumption aspect has been very important since mobile devices, including mobile phones, first came into existence. A lot of papers have been published on this matter including [8] [9] and some recent studies where authors present a general analysis of power management in the context of mobile devices [10] and energy saving strategies in the context of mobile device applications [11]. In [9], the authors present different methods described in literature that allow for increasing the energy efficiency of mobile devices at the software and hardware levels, including power management at the level of operating system, the management of sensors and communication interfaces and the use of cloud computing. [11] presents the strategies that can be implemented by the mobile application developer: Mobile Computation Offloading, sequential programs and GUI design.

Some publications such as [12] analyze in more detail the possibilities of saving mobile device power using cloud computing. The authors present an analysis of power consumption required when offloading calculations to the cloud using the network interfaces of mobile devices. They also analyze situations where using MCC concept may not lead to power savings. This may be related to confidentiality and data security considerations that necessitate more CPU usage (e.g. in the case of data encryption processes) and thus cause increased power consumption. Ensuring the reliable execution of certain services in the absence of proper communication with the cloud may also lead to increased mobile device power consumption. However, the authors only discuss theoretical

considerations related to power saving in MCC environment. Their research does not include practical tests and discussions of possible adaptation (e.g. with the use of machine learning algorithms) to prevailing conditions in which services are executed in the context of their execution time and power consumption.

The use of mobile device communication modules has a significant impact on power consumption. [13] introduces the concept of minimizing data transmission costs in MCC environment. The authors propose a solution that uses and at the same time extends the CloneCloud environment [14], which makes it possible to analyze the code and select only the most important elements that need to be offloaded to the cloud. The tests conducted, *inter alia* with the use of face recognition applications, showed a reduction in the time required for offloading the required data to the cloud and a reduction in service execution time as well as lower power usage by mobile devices. It is an interesting solution for reducing mobile device load; however, it does not take into account such factors as the context in which the service is executed. At the same time, some elements are always offloaded to the cloud, although in some cases this may not be efficient due to the service execution time and power consumption considerations. Therefore, adding to the concept developed the possibility of using machine learning algorithms that could learn which elements of the application and when should be offloaded to the cloud would in our opinion enable the required amount of data transferred to be reduced even further and mobile device power consumption to be optimized to a greater degree.

Machine learning is a popular technique used to optimize energy consumption. It may be applied in large scale systems, e.g. for data center scheduling [5]. Berral et al. apply supervised learning to create models that predict important system parameters (power consumption levels, CPU loads etc.). The models are learned from previous system behaviors and they are used to optimize scheduling decisions. It is a similar approach to ours, but we target mobile devices and the models learned are simpler because they only predict a single value for the computational task performed. Moreover in [5], the learning process occurs offline.

So far, few studies have been conducted that concern the analysis of power consumption and possibilities for reducing it and that would at the same time use the context in which the mobile device is operating, and machine learning algorithms that allow for power consumption optimization. One of the most important articles covering these topics is [15]. In that article, the authors analyze machine learning algorithms in the context of power saving in mobile systems. They present a concept where power saving is possible thanks to the dynamic adaptation of data transfer and network interface parameters, which is conducted automatically without user intervention. It involves the appropriate startup and shutdown of mobile device network and localization interfaces. The authors also conducted actual tests for five different user profiles and five different smartphones using the Android operating system. Each device was running the Context Logger application,

which logged the context of individual mobile device users to an external server where the data were analyzed offline using an algorithm developed by the authors. Those data, combined with the context in which the device and user were operating, included details such as the day of the week, location, Wi-Fi signal strength, 3G signal strength, battery status, CPU utilization and device motion. The analysis of those data from a week-long test allowed the evaluation of user behavior, including the correlation of the need for data offloading with location. On that basis, certain user behavior patterns were determined that were associated with the context of using mobile devices and five models were proposed that defined the users' behavior. At the same time, the authors analyzed five different machine learning algorithms: Linear Discriminant Analysis, Linear Logistic Regression, Non-Linear Logistic Regression with Neural Networks, K-Nearest Neighbor and Support Vector Machines, which were used in the tests. However, that analysis only concerned general use and did not take into account the character of mobile systems and solutions such as Mobile Cloud Computing. The authors also proposed a model for mobile device power consumption for devices running the Android system, using in laboratory the Monsoon Solutions power monitor device. However, power measurements and determining the model for a particular device, which is treated as a whole, exclusively under laboratory conditions does not answer the question concerning the practical characteristics of power consumption by communication interfaces of a mobile device. In this context, our research on power consumption (e.g. communication interfaces) included not only the mobile device itself but also individual services executed on it, which allowed for a more accurate analysis of energy aspects. The studies that we carried out used the PowerTutor software [16], which allows for a continuous analysis of power consumption. Using online analysis instead of predetermining a model under laboratory conditions (as the authors did [15]) allows for a better inclusion of the actual context in which the mobile device is located. The authors [15] carried out a series of tests that demonstrated the effectiveness of using different machine learning algorithms to dynamically predict the energy efficiency of different device interface configurations. Among other things, they were successful in 90% of cases when predicting power consumption using support vector machines, neural networks and k-nearest neighbor algorithms. However, it appears that such research requires tests on a larger scale than involving just a few users. For such a small number of devices, the patterns discovered may not be fully useful and universal. We believe that when the operation of the entire device is considered in isolation, without analyzing individual applications/services and without including the actual context and, moreover, it is limited to a few predefined user models, this does not enable power consumption optimization in the case of real-world mobile applications and services. Studies conducted by the authors mainly concern the prediction of energy efficiency of individual interface configurations; we think that subsequently, this knowledge is not adequately leveraged. In this respect our studies use power analysis,

710

including the predicted power consumption, to optimize (using machine learning algorithms) the execution of individual services, which allows for the optimization of the entire mobile device. Conclusions from the analysis of the article described here were among our motivations to carry out our research and to develop a completely different, and innovative, approach to the topic of mobile device energy consumption optimization with the use of machine learning algorithms.

The issues addressed in wide literature are techniques for protecting MCC against the cyber attacks. An overview of types of attacks on MCC is presented in [17], [18]. A comprehensive survey and discussion of strengths and weaknesses of various approaches and corresponding techniques for detecting attacks and vulnerabilities in MCC are presented in [19]. The surveys describe the problem of security in MCC as highly challenging. Mostly due to mobility of various and numerous network nodes. Especially when dealing with a mobile clouds consisting of heterogeneous client networks, such as Wireless Sensor Networks and Vehicular Networks, the security problem becomes more complicated, when the client networks have different security requirements in terms of computational complexity, power consumption, and security levels, which is often the case. Novel security schemes are being devised as traditional, legacy intrusion detection systems are not up to the task. In [20] a framework leveraging a deep learning approach to detect cyberattacks in mobile cloud environment is proposed. The results are promising, framework is able to recognize diverse threats with a high accuracy. Other solution described in [21] involves a multi-layered intrusion detection system with both feature-based and profile-based traffic filtration mechanisms. Classification and clustering techniques are being employed. The security of MCC platforms is still an open problem. New solutions are being devised, many of which, similarly to our solution, adopt machine learning methods. However, the greatest attention is directed at the systems operating inside the infrastructure of the cloud, e.g. intrusion detection system, while our solution is focused on the mobile device.

This research is a continuation of the results published in our paper [22] where an innovative recommender system that allows the optimization of the selection of multimedia services (for converting photos and videos) is presented. The main goal was to optimize the service execution time. The system developed allows the selection of multimedia services offered by different providers locally on the mobile device or remotely in the cloud. In order to choose the service execution location, the concept of learning agents is used that utilizes various machine learning algorithms such as C4.5, Random Forest and Naïve Bayes. The tests conducted included the context associated with the type of network connection (LTE/HSPA/EDGE) and also various methods for converting photos and videos. The article focuses primarily on optimizing the service execution time; energy aspects have not been studied thoroughly. At the same time, the solution developed only covers a system for recommending choices with respect to a single type of service (multimedia conversion), which

limits its versatility to some extent.

With respect to using the Mobile Cloud Computing concept, an important aspect is that the optimization of resources concerns not only the mobile device but the cloud as well. In [23], the authors demonstrate that it is possible to optimize resource usage in MCC by applying common patterns used in traditional cloud computing.

III. Supervised Learning Techniques in Context of Mobile Devices

Generally, supervised learning allows us to generate an approximation of the function $f : X \rightarrow C$, which assigns labels from set C to objects from set X. To generate knowledge, a supervised learning algorithm requires labeled examples that consist of pairs of f arguments and values. Let us assume that elements of X are described by a set of attributes $A = (a_1, a_2, \ldots, a_n)$ where $a_i : X \rightarrow D_i$. Therefore $x^A = (a_1(x), a_2(x), \ldots, a_n(x))$ is used instead of x. If the size of C is small, like in this study, the learning is called classification, C is the set of classes, and h is called the classifier.

The supervised learning module obtains *Training data*, which is a set $\{(x^A, f(x))\}$, and generates the hypothesis h, which is stored in *Generated Knowledge*. The *Problem* is x^A, and the *Answer* is $h(x^A)$.

There are many supervised learning methods, which use various hypothesis representations and various methods of constructing hypotheses. Three of them are described below.

Naïve Bayes (NB) is a simple probabilistic classifier, which is a special case of a Bayesian network. Generally, a Bayesian network is a pair (G, P) where G is a structure graph and P is a set of local, conditional probability distributions between variables and their parents. In the case of NB, G is very simple: the class node c is the parent of every attribute node a_1, a_2, ..., a_n. Learning is a process of calculating a priori probabilities $P(c)$ and conditional probabilities $P(a_i|c)$. The probability distribution of a class variable, for example $(a_1(x), a_2(x), \ldots, a_n(x))$, is calculated using the following formula:

$$P(c(x) = c_j) = \frac{P(c_j) \prod_{i=1}^{n} P(a_i(x)|c_j)}{\prod_{i=1}^{n} P(a_i(x))} \qquad (1)$$

C4.5 is a decision tree learning algorithm developed by Ross Quinlan [24]. The basic idea of learning is as follows: the tree is learned from examples recursively. If (almost) all examples in the training data belong to one class, the tree consisting of the leaf labeled by this class is returned. In the other case, the best attribute for the test in the root is chosen (using an entropy measure), training examples are divided according to the selected attribute values, and the procedure is called recursively (for every attribute test result with the rest of attributes and appropriate examples as parameters.)

The random forest algorithm builds an ensemble of decision trees. To reduce overfitting, trees are trained on subsets of the training set (selected at random) and using subsets of the

711

Learning algorithm	Learning speed	Hypothesis complexity	Readability to humans
NB	Fast	Low	Poor
C4.5	Medium	High	Good
Random forest	Slow	Very high	Poor

attribute set (also selected at random) [25]. The decision is calculated by voting.

In the context of the application considered, these three algorithms may be compared as follows (see Table I): NB is the fastest, but it is able to account for dependencies between attributes. The knowledge represented by probabilities is difficult to analyze. C4.5 learns more slowly, but the decision tree may represent any hypothesis. It is also readable for human experts. Random forest is the slowest algorithm but exhibits robust prediction results. Trees may represent complex hypotheses and ensembles make it possible to reduce overfitting. However, because of a large number of trees, the knowledge generated is also difficult to analyze.

In reinforcement learning, various techniques are used to prevent reaching a local optimum. The idea is to explore the solution space more thoroughly by choosing suboptimal actions from time to time (e.g. random or not performed in a given state yet). We decided to use this technique for supervised learning too because experimental results show that exploration is useful.

One such method is ϵ-greedy where the agent selects the action that it believes has the best long-term effect with probability $1-\epsilon$, and it chooses an action uniformly at random otherwise. $\epsilon \in (0,1)$ is a tuning parameter. In our experiments it is constant, but in reinforcement learning it often decreases with time.

IV. ADAPTIVE CONTEXT-AWARE ENERGY OPTIMIZATION FOR SERVICES

Let us define the learning agent (Figure 1) as a tuple:

$$LA = (P, L, TD, K, T, C, R, A, D), \quad (2)$$

where P is the processing module, L is the learning module, TD is training data, K is knowledge learned, T is a set of computational tasks, C is a set of possible contexts (battery state, connection, date, etc.), R represents possible task execution results, A is a set of attributes, which are used to describe tasks, results and the context, and D is a set of decisions. The aim of the agent is to return decision $d \in D = \{d_1, d_2, \ldots d_{ns}\}$, which corresponds to engaging one of ns services.

Input data for processing module P is a pair $x = (t, c) \in T \times C$. This pair describes it with attributes from $O \subset A$, which yields $x^O = (o_1(x), o_2(x), \ldots o_n(x))$. Next, using the knowledge stored in K it selects $d \in D$, which has the minimum predicted cost. If K is empty, d is randomized.

The decision d is then applied and the task is run using the corresponding service (e.g. locally or in the mobile cloud).

After the execution, the P module obtains execution results $r \in R$, which are described by $Res = \{r_1, r_2, \ldots r_m\} \subset A$ attributes (e.g. battery consumption $b(x, d)$, calculation time $ct(x, d)$).

The P module stores those results together with x^O and decision d in TD. Therefore the complete example stored in TD has the form

$$x^{A \cup D} = (o_1(x), o_2(x), \ldots o_n(x), r_1(x, d), r_2(x, d), \ldots$$
$$r_m(x, d), d). \quad (3)$$

The knowledge used to predict R values is trained by the learning module L using supervised learning algorithms and TD, and stored in K. The form of the knowledge depends on the learning algorithm applied. For example, if C4.5 is used, the knowledge has the form of a decision tree with leaves representing values of predicted resource usage (time or battery) and other nodes represent tests on O attributes. In cases like this, when the learning algorithm performs classification instead of regression, the predicted attribute must be discretized. The number of bins has a high impact on accuracy. However, five bins turned out to be the optimal value in all our cases.

Using value predictions $r_i \in Res$, the processing module P rates its decisions $d \in D$ by calculating predicted expenses $e(x, d)$:

$$e(x, d) = \sum_{i=1}^{m} w_i * r_i(x, d), \quad (4)$$

where w_i are weights of the result r_i. By choosing the weights, one sets priorities for the criteria. As a result, the system is flexible and universal, because it may be adjusted to user requirements.

The P module selects the decision for which execution is predicted to be successful and the expense is predicted to be the lowest. To avoid a local optimum, the ϵ-greedy strategy is applied and a suboptimal decision is executed from time to time.

V. ENERGY-AWARE AND SECURE MOBILE CLOUD COMPUTING

The main objective of this paper is to provide mechanisms for energy-aware workload allocation in MCC. However, as it has been already mentioned this should be accompanied by solutions that guarantee the maintenance of the expected level of security. Shared wireless medium, easy physical access to mobile devices, spontaneous nature of mobile clouds expose transmitted data for eavesdropping and unauthorized takeover. Using a computer with a wireless network adapter, anyone can gain access to an unprotected network. Hence, the outsider can monitor the network, participate in the communication and easily launch attacks or cause a malfunction in the proposed machine learning models for energy efficient workload allocation. Volumetric attacks consisting in generating massive, additional communication result in quick consumption of energy resources of a mobile device. On the other hand due to limitations of mobile devices in terms of computational
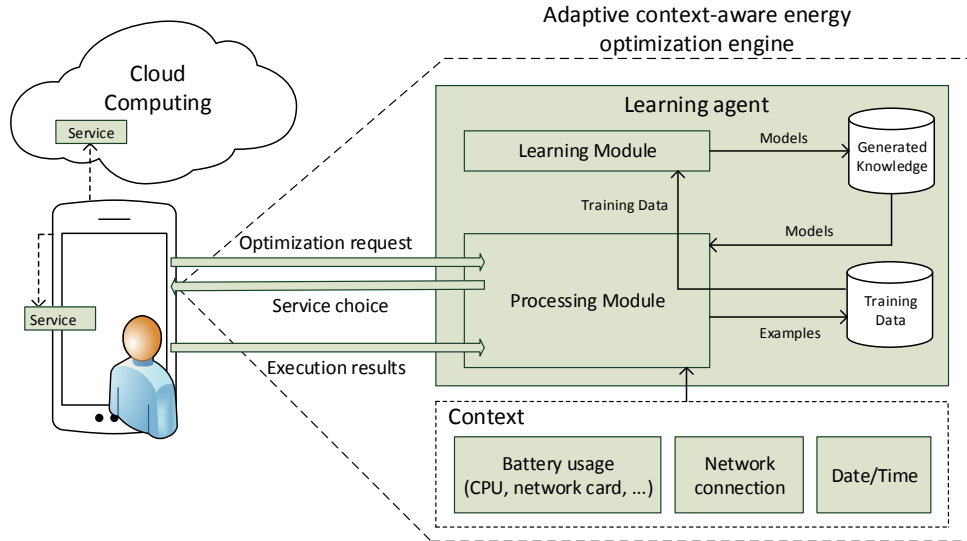
Fig. 1. Architecture of the adaptive service choice system

ability, storage and energy complex security algorithms cannot be applied to protect mobile MCC. Therefore, it is necessary to balance between the security level and the associated energy consumption used to mitigate the security risks. Being aware of those issues we propose to enhance our model to include the security aspect in decision making process. By extending the set C of possible contexts and the set of attributes A from the tuple 2 with additional parameters, we are able to solve the problem of energy efficient workload allocation without compromising on security. We consider three types of security level requirements referring to entities present in our environment: device, task and resource security level requirement.

- Device security level requirement refers to overall security requirements for a given device utilization. When confidentiality and data security considerations are a priority, this parameter should be set to a high value.
- Task security level requirement is attributed to a single task and its value depends on task's nature. This allows for a more fine-grained approach to secure workload allocation. If the task consists of computations performed on open data and the correct results are not crucial to proper operation of the device, the task can easily be allocated to execute on remote resource. On the other hand, tasks requiring access to sensitive data, like message encryption/decryption, should rarely be offloaded.
- Resource security level requirement corresponds to the amount of trust accredited to a remote resource. The aim is to enhance decision making process when more then one remote service for task execution is available.

Mentioned levels contribute to the set of attributes A. The equation 4 for calculating predicted expenses used in Processing Module stays valid. Additionally, the values of the security parameters need to be adjusted, some external

knowledge source maybe required to asses specific security level requirements for devices, task and resource types. It should be emphasized, that the evaluation and adjustment of aforementioned security levels is out of scope of this article. What we propose is a straightforward extension of our original solution to accommodate security requirements for task offloading scenarios.

VI. EVALUATION

In order to run experiments, software using the architecture developed was implemented, which makes it possible for a set of tests to be conducted and detailed results to be generated for a chosen configuration. The software developed can operate in two modes: for determining optimal classifier parameters (Hill Climbing) and for performing a set of tests. Both modes use the architecture developed, but in the first mode classifier parameters are modified on the go while the second mode uses predetermined values of classifier parameters.

All tests carried out consisted of multiple series and each series comprised a set of rounds. During each round, a set of tasks were performed (Face Recognition or OCR). After obtaining the result for the individual task, the time r_t and power consumption r_p were measured (i.e. costs of performing the task). After each round had been conducted, a classifier was built on the basis of the knowledge acquired from the previous and current rounds in the series in question. Those classifier parameters were established in the process of Hill Climbing. In the first round (reference round), the task execution location (cloud or mobile device) was selected randomly. The costs calculated on the basis of the results of this round were not used to build a classifier, but to calculate the penalty that was applied when the task ended in an error. Such a situation could occur e.g. when the network connection established to execute the cloud service was disrupted or terminated. Then, instead of the costs calculated, r_i values from the reference round

713

corresponding to this task were used, multiplied by a constant factor equal to 1.5. After all rounds in the series had been conducted, all knowledge gathered was deleted. During the tests carried out, multiple series were conducted in order to obtain average values for individual rounds.

Two services were used during tests: Face Recognition and OCR. For the Face Recognition (FR) service, five types of tests were developed for different input data (with different video stream parameters such as resolution and video frames). For the OCR service, five types of test were developed as well for different input data (with different image parameters such as resolution and format).

Three mobile devices were used during initial tests: the Lenovo Tab 2 A7-30D (1.3 GHz CPU, 1 GB RAM) tablet as well as the Samsung Galaxy Trend Plus (1.2 GHz CPU, 768 MB RAM) and HTC Desire 610 (1.2 GHz CPU, 1 GB RAM) mobile phones. All devices used the Android 4.4.2 operating system. The main experiment was conducted using only the Lenovo Tab 2 A7-30D device. In order to run remote tasks (Face Recognition and OCR) in the cloud, the AWS Lambda solution was used.

All experiments were executed using real-world Internet connections (Wi-Fi and HSDPA/HSUPA). Therefore we were not able to control connection quality and the results obtained suffer from relatively high variation. However, such conditions are similar to real-world applications in which connection quality may change.

A. Power measurement

The power consumption measurement (estimation) module was an important element used in power consumption tests in the process of learning and also an element allowing the classifier that predicted energy demand to learn. The choice of the method for measuring or estimating power was preceded by analysing and testing existing solutions. The tests were performed for Android 4.4.2, which was used by a large number of mobile devices at the time (in 2016). The method selected should work in real time (online), allow measurements for individual device components (CPU, wireless communication modules), and should have appropriate measurement resolution (less than 1% battery consumption). Owing to this resolution, it is possible to measure the power consumption of particular applications/services on mobile devices quite accurately.

The first study concerned power measurement methods on Android devices. The most commonly used measurement method is to use a public API (*BatteryManager*) that makes it possible to retrieve information about the current power status of the device. It uses a subscription mechanism, which prevents obtaining information about battery status regularly and continuous real-time measurements. At the same time, the maximum resolution of measurements is 1%, which may sometimes not be sufficient to measure the difference in power consumption between application/service launches in different contexts. It is possible to use an advanced API (via the *android.os.BatteryManager* class), which allows measurements

with a resolution better than 1%, but this is feasible only for a few devices with the Summit SMB347 and MAX17050 battery charger integrated circuits, which are present in Nexus series devices (such as Google Nexus 6 and 9), so this is not a solution which could be widely used on a variety of Android devices. There is also the non-public *BatteryInfo* Android API, which makes it possible to obtain low-level information about power consumption. However, it requires *android.permission.BATTERY_STATS* permissions, which are reserved for applications built into the system and cannot be easily used by user applications. Another option for obtaining data on power consumption is to use the *dumpsys batterystats* command and visualize the results in the Battery Stats and Battery Historian programs. This enables data to be obtained from system logs, including information on power consumption by the entire device as well as its individual components. The downside of this solution is that it does not work online and that it only shows the results on a PC. Another solution (Carat [26]) allows for the monitoring and analysis of data (on an external server) on power consumption from multiple mobile devices simultaneously and for detecting anomalies in the operation of individual applications. However, this solution does not allow local power measurement in real time (online). There are also closed applications available, such as Battery Doctor, Battery Saver 2017 and GSam Battery Monitor, which can be used for the monitoring and management of battery consumption on mobile devices. Still, due to the lack of their source codes or libraries, it was not possible to use them in the solution developed. An alternative to software solutions is the physical measurement of battery power consumption. This is the most accurate method, but it usually requires gaining access to mobile device internals and the use of additional measuring equipment. Moreover, such solutions only allow for the measurement of total power consumption of the device without obtaining any results for individual components such as CPU or wireless communication modules. Because of the nature of this measurement, this solution will never be widely used. Examples in this category are BattOr [27] (open-source), and the commercial Monsoon Mobile Device Power Monitor solution.

Our further research focused on the estimation of power consumption by mobile devices. Solutions using this method make it possible to obtain real-time results with a high measurement resolution. The most popular and widely used solution based on estimation is the PowerTutor program, which uses three basic energy characteristics of mobile device components. For a thorough analysis of the capabilities of this solution and its potential further use, we used the source code of this program to develop our own library for estimating power consumption. The solution developed allows for estimating power consumption of individual mobile device components (such as CPU and communication modules) with an error in the range of 1%–5% [16], but it was designed for older devices and does not support new LTE wireless communication modules. Finally, the latest method for estimating mobile device power consumption uses the power

profiles provided by device manufacturers. However, not all mobile devices have these profiles defined correctly which leads to problems with using this method. At the same time, no libraries have been developed that use this solution. In order to compare this method with PowerTutor, we conducted tests using the Lenovo Tab 2 A7-30D device. Preliminary results showed that for this device, there are no major differences between the two solutions when it comes to estimating the power consumption of the CPU and wireless communication modules.

The analysis of available solutions demonstrates that only estimation methods work on most devices and meet basic requirements, i.e. they work online, allow for measuring individual components of a mobile device and have the appropriate measurement resolution. While being aware of its limitations, we have decided to use our own library developed with the use of the PowerTutor source code. However, with the device used in tests, this solution allowed for a fairly accurate estimation of the power consumed by the CPU and wireless communication modules. In future research concerning new Android mobile devices, we are going to use a solution based on power profiles, which has been pre-tested by us. This will require developing a library for analysing the *power_profile.xml* system file and calculating the power consumption of individual components of a mobile device.

B. Power consumption during learning process

In the first stage of proper research, the energy cost of building classifiers was measured (including the operation of the Weka library [28], [29]), which made it possible to assess whether it was not excessive in relation to the potential savings resulting from the use of the solution developed. Tests were performed for the three classifiers used (J48 – Weka's C4.5 implementation, Random Forest, Naïve Bayes) using the Lenovo Tab 2 A7-30D tablet. For conducting tests, artificial training data were generated, containing 1,000 random examples, which corresponds to executing 1,000 tasks using the system developed. Every classifier was tested with different percentages (25%, 50%, 75% and 100%)[1] of the training data set used. Tests for specific percentage values were repeated 100 times and the final result was averaged. In Figure 2, battery consumption (as a percentage) during the process of building individual classifiers is shown. The results demonstrate that power consumption for all classifiers is low (up to 0.4% of battery charge) and does not significantly affect the ability to carry out the tests of the services developed. In addition, the data set used for these tests was very large and in practice when it comes Face Recognition and OCR service tests, the amount of data that had to be processed by the classifier was much smaller (usually from 100 to 200 examples). The lowest power consumption is associated with the Naïve Bayes classifier and this is due to the fact that Naïve Bayes exhibits linear time complexity, which results in

---

[1]For example, a 25% test meant that to build the classifier, 250 examples were used from the input data.
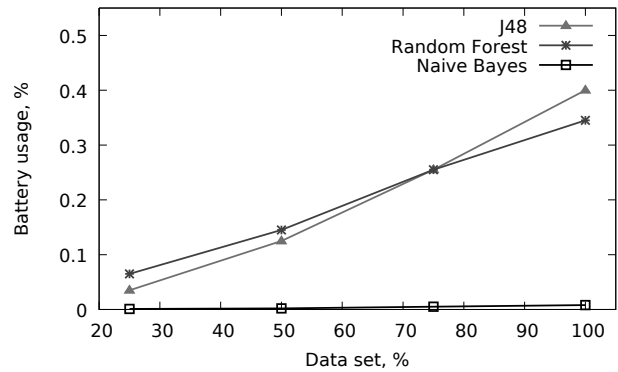

Fig. 2. Battery consumption while building a classifier

less stress on the device at the time of building the classifier (compared to the remaining classifiers used) and thus less energy expenditure.

C. Optimization

The final stage consisted of conducting the tests concerning the possibility of optimizing power consumption (and additionally execution time) using different classifiers. During the tests related to optimizing power consumption, the weights of the adaptive algorithm were set to $w_p = 90$ and $w_t = 10$. The Lenovo Tab 2 A7-30D mobile device was used during test. Each test carried out consisted of 20 series, each series of nine rounds, and each round comprised:

- seven tasks executed with the Wi-Fi connection available (9 Mb/s), including four tasks of the Face Recognition type and three tasks of the OCR type;
- seven tasks executed with the HSDPA/HSUPA connection available, including four tasks of the Face Recognition type and three tasks of the OCR type.

Figure 3 shows a comparison of power consumption optimization results for all three classifiers and for services performed without using machine learning methods (exclusively locally on the mobile device and exclusively in the cloud). It can be seen that for the J48 classifier power consumption decreases in subsequent rounds until it begins to oscillate around a single value of 17,500 mJ. The result of the t-Student test for this classifier (the p-value) equals 0.000018, which means that average values for the first and last rounds are statistically significantly different. For the Random Forest classifier power consumption decreases in subsequent rounds until it reaches a value of about 18,000 mJ. The result of the t-Student test for this classifier (the p-value) equals 0.00000017, which means that average values for the first and last rounds are statistically significantly different. For the Naïve Bayes classifier power consumption between the first and last rounds does decrease, but it is not a steady or large reduction. The result of the t-Student test for this classifier (the p-value) equals 0.1, which means that average values for the first and last rounds are not statistically significantly different.
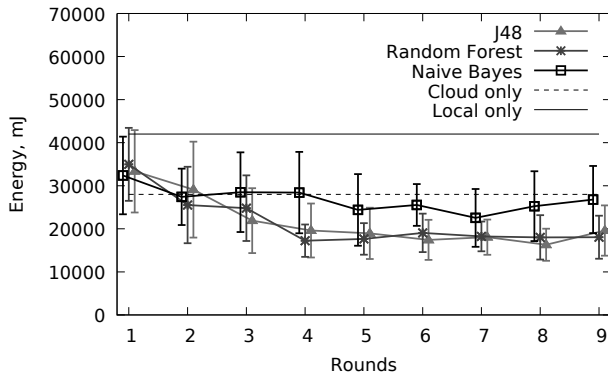
715

Fig. 3. Comparison of power consumption optimization for all three classifiers (J48, Random Forest and Naïve Bayes) and for services performed without using machine learning methods (exclusively locally on the mobile device and exclusively in the cloud)

It can be noticed that classifiers based on decision trees (J48 and Random Forest) perform much better than the Naïve Bayes classifier. They achieve almost the same optimization levels and the result of the t-Student test for both classifiers (the p-value) equals 0.3741, which means that average values for the last round for the J48 and Random Forest classifiers are not statistically significantly different. However, the Random Forest classifier achieves the final power consumption stage faster. The worst result was achieved by the Naïve Bayes classifier. The result of the t-Student test for the Naïve Bayes and J48 classifiers (the p-value) equals 0.0021, which means that average values for the last round for those classifiers are statistically significantly different. In cases where the service was executed in a single location (in the cloud or locally), the results were worse than in cases where classifiers and machine learning were used. However, the result for running the service exclusively in the cloud was only slightly worse than for the Naïve Bayes classifier.

In order to compare our software with already existing solutions, we analyzed various MCC solutions. Many of those (such as MALMOS [30], COMET [31] and COSMOS [32]) do not take into account energy aspects at all in their operation. Only a few (such as AIOLOS [33], CACTSE [34], Cuckoo [35], EMCO [36], IC-Cloud [37], MAUI [38] and ThinkAir [39]) account for energy aspects and only the IC-Cloud solution uses machine learning algorithms to optimize the operation of applications/services. However, almost all of the solutions analyzed (including IC-Cloud) are not being developed any further or there is no access to their source codes. It was only possible to find source codes for two solutions: AIOLOS and Cuckoo. Unfortunately, both of these solutions use old software development kit versions (such as Eclipse and the ADT plugin instead of Android Studio). In the case of AIOLOS, we were able to configure and build a sample project, but when running the sample (using the Androsgi plugin), the application closed and reported an error. The code was analyzed, but it was not possible to determine what caused the error. For the second solution – Cuckoo, it was

possible to run the sample application. However, comparing this solution with our system proved difficult due to the fact that Cuckoo lacked machine learning mechanisms and used completely different solutions in terms of the computing cloud – the server ran on an EC2 instance and it was not possible to use the AWS Lambda service.

VII. Conclusions

Multiple studies have been carried out recently concerning the possibilities of reducing the power consumption of mobile devices, primarily in the area of optimizing the functioning of their screens, processors or wireless communication modules. However, our analysis concerning existing solutions revealed that there are no systems that use machine learning for optimizing power consumption on mobile devices using MCC. In this article, we present an original concept for an adaptive system enabling the optimization of mobile device power consumption and at the same time taking into account the context of the device's operation. The use of machine learning algorithms in our solution allowed for the optimization of power consumption on the mobile device. Presented model can be easily extended to take security measures into consideration, making it an energy-aware and security-aware complex solution. The use of machine learning methods to securely allocate the tasks in mobile environment makes this a novel approach. Also, our concept distinguishes itself from other research by targeting endpoint mobile devices, rather then cloud infrastructure, as most present solution do.

Two services were used in the tests: Face Recognition and OCR. They were implemented on a mobile device with the Android operating system, with the ability to run them in the AWS Lambda compute cloud. The experiments carried out demonstrated that the adaptation algorithm developed made it possible to reduce power consumption for the J48 classifier by 41%, for the Random Forest classifier by 48% and for the Naïve Bayes classifier by 17%. Despite the relatively large standard deviation for the Naïve Bayes classifier, tests showed that the solutions developed can significantly reduce the power consumption of mobile devices. During the tests, we also found out that the power consumption of the system developed during the learning process itself is very low and does not significantly affect the operation of the mobile device.

Our research shows that it is possible to create a system using adaptive algorithms based on machine learning that enables efficient learning and making increasingly better decisions. This allows for power consumption optimization on mobile devices. In further tests, we would like to address, among other things, the possibility of using distributed learning. Local models would be learned on mobile devices like it is the case now; however, part of the data collected would also be sent to the cloud where global models could be learned from aggregated data provided by many users. Such global models could subsequently be sent to mobile devices and allow them to use global knowledge locally.

REFERENCES

[1] P. Nawrocki and A. Sliwa, "Quality of experience in the context of mobile applications," *Computer Science*, vol. 17, no. 3, p. 371, 2016.

[2] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng, "Mobile cloud computing: Challenges and future research directions," *Journal of Network and Computer Applications*, vol. 115, pp. 70 – 85, 2018.

[3] A. Malhotra, S. K. Dhurandher, M. Gupta, and B. Kumar, "Emcloud: A hierarchical volunteer cloud with explicit mobile devices," *International Journal of Communication Systems*, vol. 31, no. 17, 2018.

[4] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, March 1998.

[5] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proc. of the 1st Int. Conference on Energy-Efficient Computing and Networking*. ACM, 2010, pp. 215–224.

[6] B. Sniezynski, "A strategy learning model for autonomous agents based on classification," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 3, pp. 471–482, 2015.

[7] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. O. Fitó, J. Guitart, R. Gavaldà, and J. Torres, "Toward energy-aware scheduling using machine learning," in *Energy-efficient distributed computing systems*. Wiley, 2012, pp. 215–244.

[8] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," *SIGOPS Oper. Syst. Rev.*, vol. 33, no. 5, pp. 48–63, 1999.

[9] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets." *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 179–198, 2013.

[10] G. Bhatia, R. Mahajan, and S. K. Khatri, "A study for improving energy efficiency in mobile devices," in *International Conference on Infocom Technologies and Unmanned Systems (ICTUS)*, 2017, pp. 588–592.

[11] A. Meneses-Viveros, E. Hernández-Rubio, S. Mendoza, J. Rodríguez, and A. B. M. Quintos, "Energy saving strategies in the design of mobile device applications," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 86 – 95, 2018.

[12] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, pp. 51–56, 2010.

[13] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 11, pp. 2648–2660, Nov 2014.

[14] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.

[15] B. K. Donohoo, C. Ohlsen, S. Pasricha, Y. Xiang, and C. Anderson, "Context-aware energy enhancements for smart mobile devices," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1720–1732, 2014.

[16] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES/ISSS '10. New York, NY, USA: ACM, 2010, pp. 105–114.

[17] M. B. Mollah, M. Azad, and A. Vasilakos, "Security and privacy challenges in mobile cloud computing: Survey and way ahead," *Journal of Network and Computer Applications*, vol. 84, pp. 34–54, 04 2017.

[18] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680 – 698, 2018.

[19] A. N. Khan, M. L. Mat Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: A survey," *Future Gener. Comput. Syst.*, vol. 29, no. 5, p. 1278–1299, Jul. 2013.

[20] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.

[21] S. Dey, Q. Ye, and S. Sampalli, "A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks," *Information Fusion*, vol. 49, 09 2019.

[22] P. Nawrocki, B. Sniezynski, and J. Czyzewski, "Learning agent for a service-oriented context-aware recommender system in a heterogeneous environment," *Computing and Informatics*, vol. 35, no. 5, 2016.

[23] P. Nawrocki and W. Reszelewski, "Resource usage optimization in mobile cloud computing," *Computer Communications*, vol. 99, pp. 1 – 12, 2017.

[24] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[25] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[26] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, "Carat: Collaborative energy diagnosis for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '13. New York, NY, USA: ACM, 2013, pp. 10:1–10:14.

[27] A. Schulman, T. Schmid, P. Dutta, and N. Spring, "Demo: Phone power monitoring with battor," in *ACM Mobicom*, 2011.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[29] N. A. Azeez, O. J. Asuzu, S. Misra, A. Adewumi, R. Ahuja, and R. Maskeliunas, *Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection Using Weka*. Singapore: Springer Singapore, 2018, pp. 195–208.

[30] H. Eom, R. Figueiredo, H. Cai, Y. Zhang, and G. Huang, "Malmos: Machine learning-based mobile offloading scheduler with online training," in *Proceedings of the 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, ser. MOBILECLOUD '15. IEEE Computer Society, 2015, pp. 51–60.

[31] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "Comet: Code offload by migrating execution transparently," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 93–106.

[32] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "Cosmos: Computation offloading as a service for mobile devices," in *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '14. New York, NY, USA: ACM, 2014, pp. 287–296.

[33] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Aiolos: Middleware for improving mobile application performance through cyber foraging," *J. Syst. Softw.*, vol. 85, no. 11, pp. 2629–2639, Nov. 2012.

[34] W. Qing, H. Zheng, W. Ming, and L. Haifeng, "Cactse: Cloudlet aided cooperative terminals service environment for mobile proximity content delivery," *China Communications*, vol. 10, no. 6, pp. 47–59, June 2013.

[35] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, *Cuckoo: A Computation Offloading Framework for Smartphones*. Springer, 2012, pp. 59–79.

[36] H. Flores and S. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," in *Proceeding of the Fourth ACM Workshop on Mobile Cloud Computing and Services*, ser. MCS '13. ACM, 2013, pp. 9–16.

[37] C. Shi, P. Pandurangan, K. Ni, J. Yang, M. Ammar, M. Naik, and E. Zegura, "Ic-cloud: Computation offloading to an intermittently-connected cloud," Tech. Rep., 2013.

[38] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[39] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 945–953.