J Grid Computing (2017) 15:343–356
DOI 10.1007/s10723-017-9406-2

Autonomous Context-Based Service Optimization in Mobile Cloud Computing

Piotr Nawrocki · Bartlomiej Sniezynski

Received: 9 February 2016 / Accepted: 4 July 2017 / Published online: 20 July 2017
© The Author(s) 2017. This article is an open access publication

Abstract As the concept of merging the capabilities of mobile devices and cloud computing is becoming increasingly popular, an important question arises: how to optimally schedule services/tasks between the device and the cloud. The main objective of this paper is to investigate the possibilities for using a decision module on mobile devices in order to autonomously optimize the execution of services within the framework of Mobile Cloud Computing while taking context into account. A novel model of the decision module with learning capabilities, service-oriented architecture, and service selection optimization algorithm are proposed to solve this problem. To achieve autonomous, online learning on mobile devices, we apply supervised learning. Information about the context, task description, the decision made and its results such as calculation time or power consumption are stored and form training data for a supervised learning algorithm, which updates the knowledge used by the decision module to determine the optimal place for the execution of a given type of task. To verify the solution proposed, service-oriented mobile processing systems for multimedia file conversion have been developed and series of experiments have been executed. Results show that the decision module has become more efficient in assigning the task to either the mobile device or cloud resources.

Keywords Service optimization · Mobile device · Cloud · Context-aware · Supervised learning

P. Nawrocki (✉) · B. Sniezynski
Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: piotr.nawrocki@agh.edu.pl

B. Sniezynski
e-mail: bartlomiej.sniezynski@agh.edu.pl

1 Introduction and Motivation

In recent years, the importance of mobile devices in computer systems has increased [1]. The development of mobile phones, which at first were only meant for voice communication, has taken a turn in the direction of multi-function devices known as smartphones, which combine the functionalities of both phones and computers and additionally incorporate various sensors. Alongside hardware development, software began to develop as well; more and more programs started to leverage the resources of mobile devices, and especially the CPU, memory and battery. When it comes to CPU and memory capabilities, mobile devices have become more like computers; however, there still are some characteristics that differentiate them, e.g. relatively small screens (even in tablets), battery power supply and communication modules that rely solely on the wireless technology.

Currently, one of the main areas in distributed systems is cloud computing [2]. However, other directions

are also being explored, like visual tools for managing computations [3] or using novel middleware for distributed computing [4].

Along with the development of cloud computing and mobile systems, which are very specific middleware, research into the implementation of this concept in the mobile device environment began.

One of the most important publications in this area is the work of Kumar and Lu [5] who research the potential energy savings on mobile devices through offloading computation to the cloud. In another paper [6], they extend their research on the possibility of improving performance through offloading computation from the mobile device to the cloud. However, in offloading frameworks there is an additional overhead related to component migration at runtime. In paper [7], a novel distributed Energy Efficient Computational Offloading Framework (EECOF) is proposed for the processing of intensive mobile applications in MCC. The solution focuses on leveraging application processing services in cloud computing with minimal instances of computationally intensive component migration at runtime. Through the use of EECOF, the amounts of data transmitted and energy consumed are reduced in computational offloading for MCC.

On the basis, *inter alia*, of the research described above, the Mobile Cloud Computing (MCC) paradigm [8], which enables the migration of individual services (tasks [9], data) from mobile devices to the cloud [10], emerged. On the basis of the general MCC concept, open (e.g. Open Mobster, Clonecloud [11]) and commercial (e.g. Perfecto Mobile) implementations of the solution have been developed. Many sample scenarios and applications using mobile devices and cloud computing have been described in literature [12]. Authors Dinh et al., in [13], have conducted a survey of typical MCC applications, such as mobile commerce [14], mobile learning [15], mobile healthcare, allowing easy and effective access to the patients' medical records [16] or enabling the monitoring of the patients' condition while at home [17], and also mobile gaming, enabling the user to play while simultaneously sending certain game tasks that require the most computing resources (such as the graphic rendering process) to the cloud [18]. In [13], the authors also describe many examples of other practical applications using MCC, including a mobile localization service and a voice-based or tag-based searching service, among others. However, in that paper multimedia aspects are missing (e.g. image, face and text recognition using a mobile device and MCC). These applications contain calculations that require a fairly large amount of computational power and therefore can be advantageously offloaded to the cloud. Apart from the examples mentioned above, an increasing number of applications related to social networks [19] and using mobile device sensors (such as GPS) [20] use the MCC paradigm.

The main objective of the studies described in this paper was to optimize the expenses of execution of services in the MCC environment [21]. The optimization process is performed autonomously on the mobile device to take into account the heterogeneity of mobile systems, environmental context and fluctuating conditions. Optimization criteria represented by the expenses correspond to the Quality of Experience (QoE) parameter [22] and may involve, *inter alia*, execution time, energy consumption by the mobile device and user satisfaction. We demonstrate that this optimization can be ensured by using a novel decision module with learning capabilities in the Mobile Cloud Computing environment.

Importantly, we show that to provide autonomous online learning on a mobile device, supervised learning may be applied. It is not a traditional approach because autonomous online learning, usually considered in the context of agent-based systems, is typically realized using reinforcement learning (see surveys [23, 24]). However, our research demonstrates that where the parameter space is larger, supervised learning is faster than reinforcement learning [25].

The solution proposed was applied in the development of the case study environment related to the service-oriented mobile processing system for multimedia file conversion. Experimental results show that it is possible to optimize the utilization of resources (execution time) by moving selected tasks performed within the framework of individual services to the cloud.

Our approach applies new technologies and is a general solution at the same time. It is possible to use it to optimize any service-oriented system in which there is a choice with respect to the place of execution of a given service, and execution characteristics (such as calculation time or energy consumption) are predictable.

This paper is structured as follows: Section 2 contains a description of related work, Section 3 is

concerned with employing the learning decision module in the process of optimizing the service selection strategy on mobile devices, Section 4 describes in detail the decision module, Section 5 presents the case study, Section 6 describes performance evaluation and Section 7 contains the conclusion.

2 Related Work

The MCC paradigm yields many benefits [13], especially for mobile devices. One of the most vital advantages is enhancing battery life without the necessity to replace mobile device hardware and software. The possibility of migrating complex services/tasks to the cloud results in decreased energy consumption by the CPU and, as a result, increased battery use efficiency. Another important advantage is improved reliability. Saving data in the cloud reduces the risk of data loss and enables the introduction of additional functionalities, such as copyrighted digital content and virus scanning.

Nonetheless, all MCC scenarios and applications must accommodate the limitations resulting from the nature of mobile systems, which necessarily rely on the wireless transmission technology [26] and take into account the optimal deployment of software in the MCC environment [27]. Apart from possible problems related to wireless network communications (low bandwidth or data loss), another important challenge for MCC is security. Providing the proper level of data security [28] and privacy-aware communication mechanism [29] are key aspects of MCC, especially in the context of e-health [30] or finance and banking applications. In [31], the authors deal with the issue of ensuring confidentiality of data in MCC through the use of encryption and appropriate cryptographic methods. The choice of encryption methods should take into account the limited resources of mobile devices and the peculiar characteristics of the technology used for wireless communication. This article describes a Cloud-Manager-based Re-encryption Scheme (CMReS), which combines the characteristics of manager-based re-encryption and cloud-based re-encryption for providing better security services with a minimum of processing burden on mobile devices. Experiment results demonstrate that the solution proposed results in a significant improvement in turnaround time, energy consumption and resource utilization on the mobile device as compared to existing re-encryption schemes.

The possibilities of employing the MCC paradigm in order to optimize services on mobile devices for numerous applications and scenarios [8] have been discussed in literature. In [32, 33] the authors present the possibility of using this paradigm in image processing for an application that reads text (e.g. descriptions of exhibits at South Korean museums [33]) and translates it into the language of choice using optical character recognition (OCR). If it is not possible to translate the text on the mobile device, the task is shifted to the mobile cloud where the text is translated and the result is sent back to the user. Another example described in [12] is a service for managing multimedia files, which can be collected from numerous mobile devices and combined into a single file presenting the image from different angles and perspectives. In the "lost child" scenario, the author describes a situation where a child is missing and it is possible to collect and send records (or photos) from different users' mobile devices to the mobile cloud and to gather comprehensive information on the missing person. An example of another system using MCC, which provides multimedia services, is [34]. The developed mobility-aware framework for mobile cloud streaming services makes dynamic, optimized server selection functions available in order to support user mobility.

In the context of the use of Mobile Cloud Computing, the management of cloud-based resources is an important issue. One aspect of resource management is optimization that should concern not only the mobile device but the cloud as well. In [35], the authors demonstrate that it is possible to optimize resource usage in MCC by applying common patterns used in traditional cloud computing.

An important issue in service-oriented systems is accounting for the context in which the service in question is executed. An example of such a solution is a context-aware service discovery framework based on virtual personal space (VPS), which is described in [36]. In this framework, the middleware embedded in the device provides personalized responses. Another solution is the MobiByte [37] context-aware application model, which uses multiple data offloading techniques in order to support a wide range of applications. In this model, many important aspects such as energy efficiency, performance, generality, context awareness and privacy are taken into account.

The increase in user requirements related to mobile access to a variety of resources has resulted in the growing popularity of the Vehicular Mobile Cloud. In [38], adaptive Learning Automata based Contention Aware Data Forwarding (LACADF) is proposed. This solution is evaluated in different network scenarios with respect to multiple parameters (such as throughput and delay) by varying the density and mobility of vehicles.

As of yet, there have not been many studies of MCC and the use of autonomous machine learning algorithms executed online in the process of optimizing the service selection strategy on mobile devices. A similar topic concerning the optimization of the mobile environment using MCC is investigated in [39]. In the paper, the authors employ genetic algorithms in the optimization process; however, they do not e.g. consider energy aspects (mobile device battery life), focusing solely on computational complexity and requirements concerning the memory allocated to particular services. Optimization of SOA systems with machine learning is discussed in [40]. However, this paper does not consider the mobile environment and another learning strategy is used (reinforcement learning). Agent-based approach to SOA optimization is also discussed in [41]. Supervised learning is applied to generate a strategy for choosing a service to execute tasks. In this research, mobile devices were not taken into account. There are several other solutions that use MCC and machine learning algorithms but in those solutions, some elements related to learning are always analyzed offline in contrast of our solution where the learning process only takes place online on a mobile device. In [42], mobile device location is predicted using supervised learning with a sliding window. Three machine learning algorithms are applied: 1NN, C4.5, and voting (based on 1NN and C4.5). In that paper, the problem of task allocation (which we are working on) is not studied extensively and all experiments are performed offline on the data generated previously. In another paper [43], machine learning algorithms are used to predict task execution time in MCC. That solution consists of two components: an offline one (developing a task performance model for each device) and an online one (applying the model learned to the current case). In our solution, no task performance model needs to be developed offline for each mobile device. One of the most important paper covering MCC solutions in the context of machine learning algorithms is [44]. In that paper, the authors 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, using machine learning algorithms. In experiments, the authors used 5 mobile devices (with the Android system) to define user models. The analysis of data from those devices was performed offline. Owing to the lack of online analysis of the individual services and the context of their execution, that solution, which was developed using a few predefined models, does not enable power consumption optimization in the case of real-world mobile applications and services. Another concept that allows power consumption to be reduced is presented in [45]. However, that solution does not use MCC but rather cloudlets that allow the offloading of services to nearby mobile resource-rich devices.

One of the few solutions that enables online analysis is the MALMOS framework [46], which makes it possible to decide, with the application of machine learning technologies, when to offload applications from the mobile device to the cloud. For the offloading itself, the solution uses the DPartner environment (Java-based on-demand offloading framework). In order to properly teach the system where it should run applications, the solution uses an online training mechanism. The system works in two modes: online training phase and runtime scheduling phase. In the former, training information is collected by executing the same task locally and remotely. As a result, a classifier is learned that determines where the task should be executed. In the latter phase, the task is scheduled on the basis of the learned model. In our solution, machine learning is applied to learn models for tasks that are used to predict calculation time and energy consumption in a given context. The predictions are then used to estimate costs of local or cloud execution and the less expensive decision is chosen. As a result, there is no special learning mode, because training data can be collected during normal runtime. To estimate training phase duration, an adaptive online training mechanism is proposed in [46]. In our solution, such a strategy is not required and the system is simpler. In tests, three machine learning algorithms were used: instance-based learning, perceptron and naïve Bayes. The results demonstrated an improvement in the process of selecting the location

for executing applications as compared to solutions that did not utilize machine learning mechanisms. However, those tests were performed using laptop and desktop computers with simulated workloads and network bandwidths, while in our experiments we used public clouds and real-life mobile/WiFi network connections. Furthermore, the solution mentioned completely fails to address the important aspect of energy consumption during the launching of the application and transferring the data to the cloud and also the amount of energy used by the online learning mechanism.

Another solution that enables online analysis has been developed by the authors of a service-oriented context-aware recommender system [47]; it utilizes learning agents in an MCC environment. However, the system has been developed for service recommendations only and is not a universal solution that would allow for autonomous context-based service optimization in MCC.

All the examples described assume that the user has decided to send the task or data from his or her mobile device to the mobile cloud in order to perform a particular operation. However, we assume that, in contrast to the aforementioned examples, the mobile device also has the option to perform the service in question but it might prove more cost-effective to send the task/data to the mobile cloud, perform the operations required and return the results to the device. This assumption makes it possible to optimize the operation of mobile devices on the basis of various criteria such as energy consumption, the amount of data transferred (and the related charges) and execution time. At the same time, we have conducted research into the use of the decision module and the supervised learning process in connection with making decisions on the relevant circumstances and the services (tasks, data) that should be sent to, and executed in, the mobile cloud.

3 Optimization of Service Selection: Context, Execution Results and Models

An important question related to MCC is the optimization of the service selection strategy on mobile devices. Increasingly often, it is possible to perform complex services (tasks) on mobile devices (thanks to, *inter alia*, their greater processing power and memory size), but this increases energy consumption at the same time (shortening battery life). This is why, apart from cases where mobile devices have to delegate services (tasks) to the cloud, the MCC paradigm should take into consideration situations in which services (tasks) may either be migrated to the cloud to preserve the resources of the mobile device or may be executed locally. The decision on this migration may be made by the user, e.g. where he or she is not satisfied with the performance of a service or the outcome. However, the process may also be automated and online adaptation by applying machine learning is possible. Therefore a solution that uses a decision module that is able to monitor the environment and, on that basis, to make decisions about the place where a service (task) is to be executed, has been proposed.

The decision module operating on a mobile device in the MCC paradigm monitors the environment and collects information on:

- the task to be executed by the service in question, including its type, key arguments, estimated data input/output size, estimated execution time, the cost of performing the computation and the time when the result is needed;
- the cost of performing the service in the cloud, affecting the assessment of the cost-effectiveness of service;
- the location of the device (domestic/roaming), indicating whether the mobile device uses the data transmission service offered by local mobile carriers (lower costs), or must use roaming services abroad (higher transmission costs);
- possible device connection modes (Wi-Fi, 2G/3G/4G) and connection quality, affecting network throughput between the mobile device and the cloud. The type of connection can also affect the power consumption of the mobile device;
- battery status, determining how long the mobile device can operate and how long the service can be performed on this device;
- the current time and date (including day of week, holidays), affecting the ability to take advantage of better data transmission rates or to transmit/receive data during periods when the telecommunication operator's infrastructure is less busy;
- readings of sensors such as the accelerometer, light sensor, etc. for determining the status of the mobile device (device movement, ambient lighting).

This information forms a context, on the basis of which the module makes decisions when and where to perform the service (locally or in the mobile cloud). After completing the task, the module assesses its decision, considering one or more execution results such as:

- mobile device power consumption;
- the time spent waiting for the result;
- the user's satisfaction (the user could override the module's decision, which means that he or she does not agree with it);
- costs (e.g. charges related to data transfer or using cloud resources).

These results are used to formulate optimization criteria (see (4)).

The module gathers experience and updates its strategy, applying a supervised learning algorithm. The generated knowledge stored in the module may include:

- models of the user's behavior that enable it to assess the impact of factors such as his or her location/connection accessibility/ability to charge the device;
- models of estimated outcomes of performing a service locally/in the cloud (energy consumption, time).

These models may be used to improve the estimates of decision consequences.

4 Decision Module: Model, Architecture and Algorithm

Based on the assumptions mentioned in the previous section, we would like to propose a solution for service selection adaptation which applies supervised learning, executed on-line, on a mobile device, in contrast to [36, 38] cited above, in which other optimization techniques are used or [41] in which MCC is not considered. The decision module, located on a mobile device, works as follows: it gets task and context as an input and applies knowledge to make a decision where the task should be executed in the given context. These data together with execution results are stored as training data. These data are, in turn, an input for a supervised learning algorithm, which returns the knowledge used to make the decision about tasks.

Let us define the Decision Module as a tuple:

$$DM = (A, GK, TD, Dec), \tag{1}$$

where A is a set of attributes, which are used to describe tasks and the context, GK is *generated knowledge*, and TD is *training data*, which is a set of *examples*. The aim of the module is to return a decision $d \in Dec = \{d_1, d_2, \ldots d_{ns}\}$, which corresponds to engaging one of ns services.

Input data for the module is a pair $x = (t, c)$ representing the task t which should be executed in the context c. The processing module describes it with observation attributes $O = \{o_1, o_2, \ldots o_n\} \subset A$, which yields $x^O = (o_1(x), o_2(x), \ldots o_n(x))$, i.e. a description of the *Problem*. Next, using knowledge stored in GK it solves the *Problem* by selecting $d \in Dec$, which has the minimum predicted cost. If GK 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 module obtains execution results r, which are described by $Res = \{r_1, r_2, \ldots r_m\} \subset A$ attributes (e.g. whether execution was successful $es(x, d)$, battery consumption $b(x, d)$, calculation time $ct(x, d)$ and user's dissatisfaction $dis(x, d)$, which may be measured by observing if the user overrode the module's decision). Therefore, the set of all attributes used to describe the input data is a sum of O and Res

$$A = O \cup Res. \tag{2}$$

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

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

The models to predict R values are constructed using supervised learning algorithms and stored in GK. These models influence the choice of the decision. There may be several models stored in GK.

Using value predictions $r_i \in R$, the module may rate its decisions $d \in Dec$ by calculating predicted expenses $e(x, d)$:

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

where w_i are weights of the result r_i. By choosing the weights one sets the priorities of the criteria. As

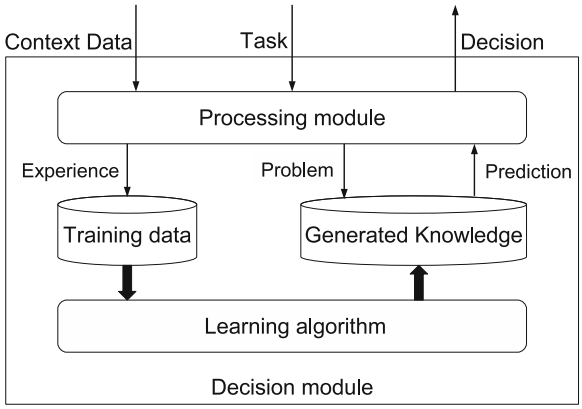

Fig. 1 The decision module architecture reflecting its learning capabilities

a result, the system is flexible and universal, since it may be adjusted to the user's requirements.

The module should select the decision for which execution is predicted to be successful and the expense is predicted to be the lowest.

The internal structure of the module should reflect its learning capabilities. The architecture is presented in Fig. 1; three main components may be distinguished:

- *Processing Component*, which is responsible for basic activities such as the processing of input data, storing training data, executing the learning process and leveraging the knowledge learned;
- *Learning Component*, which is responsible for executing the learning algorithms and providing answers to problems using the knowledge learned;
- *Training Data* (TD), which provides storage for the examples (experience) used in learning;
- *Generated Knowledge* (GK), which provides storage for the knowledge learned (models).

These components interact in the following way: the *Processing Component* receives *Context data* and description of the *Task* (parameters listed in the previous subsection), formulates a *Problem* by describing input data with available attributes O and applies *Generated Knowledge* to get the *Prediction* for the *Problem*, which is used to choose the *Decision*. After executing the task, *Results* are observed. A description of the *Problem*, *Decision* and *Results* is an example, see (3). The *Processing Module* decides what examples should be stored in TD storage. Currently all examples are stored, but it is also possible to store outliers (e.g. with extremely high computation times) to save space. When required (e.g. periodically or when TD contains many new examples), it calls the *Learning Component* to execute the learning algorithm that generates new knowledge from TD. The knowledge learned is stored in the GK base.

The algorithm, which is executed in the *Processing Component*, is presented in Fig. 2. At the beginning, GK and TD are set to empty (lines 2–3). Next, if there

Fig. 2 Algorithm of the *Processing Component* allowing the optimization of the decision strategy using online supervised learning

```
1  begin
2      Generated Knowledge:= ∅;
3      Training Data:= ∅;
4      while Module is working do
5          if Generated Knowledge = ∅ then
6              d:= random decision
7          end
8          else
9              Problem := description of the current state (the task and the
                context);
10             d:= decision determined for Problem by model(s) stored in
                Generated Knowledge
11         end
12         execute task at the service determined by d;
13         observe execution results;
14         store example in the Training Data;
15         if it is learning time (e.g. every 100 steps) then
16             learn from Training Data;
17             store knowledge in Generated Knowledge;
18         end
19     end
20 end
```

is no learned knowledge, the decision is randomized (lines 5–6). Else there is some knowledge, therefore *Problem* is set to a description of the observed context and task, x^O (line 9). Next, GK is used to select the best decision for the *Problem* (line 10). The task is executed in the selected service (line 12). Results of the execution are observed (line 13) and example ($x^{A \cup Dec}$) is stored in the TD (line 14). After processing a given number of tasks (line 15), *Learning Component* is called to generate new knowledge from TD and the learned knowledge is stored in GK (lines 16–17).

The form of knowledge stored in GK depends on the learning algorithm utilized. It may have an explicit form, e.g. rules, a decision tree or a Bayesian model in the case of supervised learning. It may also be stored in lower-level form such as parameters representing a linear regression model, a decision value function or a neural network approximator of such a function if reinforcement learning is applied.

5 Service-Oriented Mobile Processing System

In order to verify the possibility of using a decision module with learning capabilities to optimize the practical execution of services in the MCC environment, the authors developed a Service-oriented Mobile Processing System (SMPS) that enables the processing of multimedia files. Varying demands for computational power for different tasks and the necessity to transfer varying amounts of data in the case of cloud processing make this case a representative domain for testing the solution proposed.

In the SMPS, multimedia files may be converted using a multimedia data conversion service in the cloud environment or directly on the mobile device. At application runtime, data concerning conversion efficiency were collected with regard to different conditions: the size of the file, the codec used, task type and where the conversion took place. Based on those context data, the application (Decision Module) selected the conversion type and place that would offer better expected efficiency. Execution results are submitted to the Decision Module to provide data for optimization. The architecture of the solution developed is shown in Fig. 3.

The mobile application was developed for the Android operating system. In order to implement the functionality used in the tests conducted the authors picked the Google Cloud Endpoints solution. This is a technology enabling easy communication between mobile and web apps with a backend operating in the Google App Engine cloud [48]. The solution involves both client libraries and server ones, which are available for the Java, Python, PHP and Go languages. Since the test mobile application was developed for the Android operating system, the authors decided to use Java both for the client on the mobile device and for the backend operating in the cloud.

In order to convert multimedia files, the jcodec library was used, which can operate both on a mobile device with the Android operating system and in the Google App Engine cloud. This made it possible to compare conversion times on the mobile device and in the cloud where the same mechanism was used for encoding multimedia files.

The jcodec library was used to implement the following conversion types:

- from the H.264 AVC format (MP4 container) to the Apple ProRes 422 format (Proxy) (MP4 container);
- from the H.264 AVC format (MP4 container) to a series of PNG images (where every 5th, 15th or 25th frame was encoded).

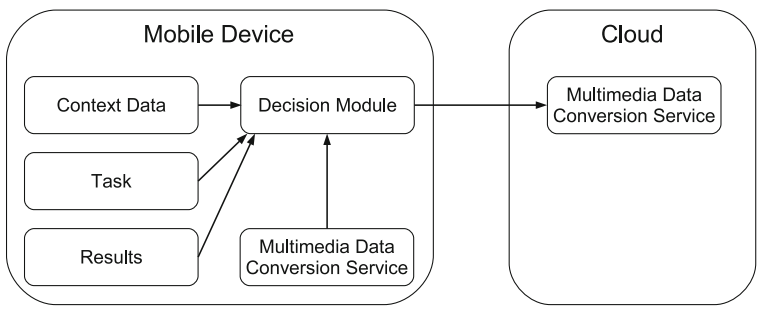

Fig. 3 Architecture of the Service-oriented Mobile Processing System

Table 1 Experiment results for a naïve Bayes classifier

Round	1	2	3	4	5	6	7	8
Minimum execution time, s	3589	3402	2827	3013	3415	1940	2683	2861
Maximum execution time, s	4705	5227	4074	4477	4760	2967	4690	3743
Average execution time, s	3676	4297	3503	3378	4221	2538	3684	3185
Standard deviation	996	780	591	763	571	432	841	387

The Android operating system version of the Weka library [49], Weka for Android [50] was used to implement learning algorithms and to apply the knowledge learned.

In the domain considered, the solution proposed may be specified in detail as follows. The task t is described by three attributes: *type* representing task type (with two values: codec conversion or frame selection), $framegap \in \{5, 15, 25\}$ representing the number of frames to be skipped during frame selection, and $length \in \{1, 2, \ldots, 9\}$ representing the duration of the movie processed (measured in seconds). The context c is described by the *time* numeric attribute representing the minute of the day and the nominal *connection* attribute showing a type of connection (LTE, HSPA+, HSPA, EDGE), hence $O = \{type, framegap, length, time, connection\}$. Results are described by the *successful* Boolean attribute showing whether the execution was successful or not, *battery* representing battery usage and the numeric *calctime* attribute representing calculation time in milliseconds. Therefore $Res = \{successful, battery, calctime\}$. $Dec = \{d_1 = l, d_2 = cl\}$, which corresponds to engaging local or cloud resources.

During learning, a classifier GK_{ct} is learned. It is used to predict $calctime$ from $O \cup Dec$ attributes. The *calctime* is discretized into 7 equal ranges. As a result, $GK = (GK_{ct})$.

6 Performance Evaluation

In this section we describe experiments whose goal was to check how well our solution performed in a real-life environment. Every experiment consists of a series of task packages being executed. Every task package is a combination of selected task parameters and system state values. This means that each test package involves measuring the conversion time and battery usage for the multimedia file in question on the mobile device and using cloud for various file sizes (with the maximum file size being 650 kB), various conversion types and network connection statuses.

During the series of experiments, examples ($x^{A \cup Dec}$) are stored in TD. They contain information about the context, task, results (computation time and battery) and decision. After every task package, the decision module initiates the learning process and builds the GK_{ct} classifier (using supervised learning), which is used in the next round to process the task package. During the round n, the learning module uses the examples collected in rounds $1...n-1$ as training data. When the series is completed, TD and GK are cleared and the next series is executed to collect statistical data. If task execution results in failure, task execution time and battery usage are set to the maximum successful value observed.

The experiment consisted of executing tests in series of eight rounds, which were repeated ten times. Before the experiments presented here, preliminary tests with higher numbers of rounds were performed. The results demonstrated that increasing the number of rounds does not yield a significant improvement in learning results, while it makes experiments much more difficult (because of the cloud transfer quota) and more time consuming as well.

We have executed our experiments with default settings for all learning algorithms. Therefore, parameters

Table 2 Experiment results for a random forest classifier

Round	1	2	3	4	5	6	7	8
Minimum execution time, s	2790	3671	1556	1763	1617	2915	2064	1733
Maximum execution time, s	3882	4897	2823	2071	4404	3588	3316	2246
Average execution time, s	3496	4218	2342	1933	3030	3268	2822	1933
Standard deviation	612	623	686	156	1394	337	667	274

Table 3 Experiment results for a neural network

Round	1	2	3	4	5	6	7	8
Minimum execution time, s	3141	4349	2497	2110	1850	1639	1812	1718
Maximum execution time, s	3705	5753	3575	2617	2400	2625	3324	2117
Average execution time, s	3434	5024	2902	2307	2138	2058	2462	1951
Standard deviation	283	703	587	271	276	510	778	208

have the following values. For naïve Bayes, the *useKernelEstimator* and *useSupervisedDiscretization* parameters are set to false. For the random forest, *bagSizePercent* is 100, *maximum tree depth* is unlimited, *breakTiesRandomly* and *calcOutOfBag* are set to false, the *number of iterations* (trees) is 100, and *numFeatures* is 0 (which means that the number of randomly selected attributes is $log_2(\#predictors) + 1$). For the neural network, the *hiddenLayers* parameter is set to the "a" value, meaning that there is a single hidden layer with four neurons, *training time* is equal to 500 epochs, *learning rate* is 0.3 and *momentum* is 0.2. The following three parameters are set to true: *nominalToBinaryFilter*, *normalizeAttributes* and *normalizeNumericClass*.

In each test, EDGE/WiFi network communication, the Google App Engine cloud and the supervised learning method were used. The experiment was carried out using the Samsung GT-i9505 mobile device. This mobile device has the following specifications: SoC – Qualcomm APQ8064T Snapdragon 600, CPU – quad-core 1.9 GHz Krait 300, GPU – Adreno 320, RAM – 2 GB and storage – 16 GB. For experiments related to the execution time for the multimedia file, the decision module takes into account calculation time ct in cost calculations (weights $w_b = w_d = 0$, $w_{ct} = 1$). In the first experiment, a naïve Bayes learning algorithm is used to learn the GK_{ct} classifier. The results of those tests are shown in Table 1. Application of machine learning using the naïve Bayes classifier significantly increases execution speed for the tasks requested. The mean task execution time in subsequent rounds of tests using the knowledge gained as a result of machine learning shows a downward trend. The difference between mean values from the first and the last round is about 500 seconds (about 13%). However, according to the t-Student test, this difference is not statistically significant (p-value is equal to 0.0831).

In the second experiment, a random forest classifier is applied to generate GK_{ct}. The results of those tests are shown in Table 2. In this case the difference between mean values from the first and the last rounds is larger – over 1,500 seconds (45%). The t-Student test demonstrates that the difference is statistically significant (the p-value is less than 0.0001).

In the third experiment, a back-propagation neural network was used to model GK_{ct}. The results of those tests are shown in Table 3. The difference between mean values from the first and the last rounds is slightly smaller than for the random forest. It is almost 1,500 seconds (43%). The t-Student test demonstrates

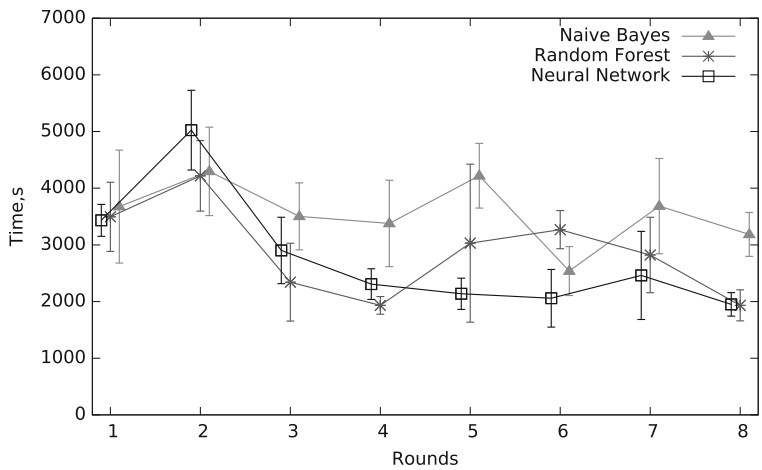

Fig. 4 Comparison of the average execution times in subsequent rounds for three machine learning algorithms applied to select the appropriate location for running a multimedia service

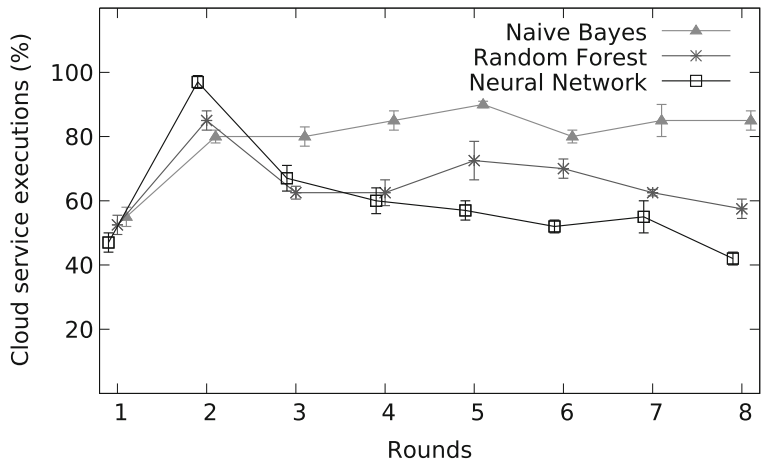

Fig. 5 Comparison of the average number of service executions in the cloud in subsequent rounds for three machine learning algorithms (as percentages)

that the difference is statistically significant (the p-value is less than 0.0001).

A comparison of the results of all three experiments is presented in Fig. 4. As we can see, the random forest and neural network have similar final performance. However, the neural network is more stable. Naïve Bayes is the worst. The reason is that it is unable to represent the dependencies between input attributes.

To explain the varying execution times, we checked how the location for executing the service changes. Figure 5 presents the average numbers of multimedia data conversion service start-ups within the cloud (as a percentage) in subsequent rounds for all three learning algorithms (random forest, naïve Bayes and neural network). The number of total service calls in the round was 40 (100%). It can be noticed that, based on Figs. 4 and 5, the shortest service execution times occur when about half of calls are executed on the mobile device (round 8). However, it should be remembered that executing the service locally, on a mobile device, can cause higher load on the device, with the resulting increase in battery usage and reduction in the operation time of the mobile device. What is more, executing all tasks from the package on the mobile device results in a much longer average execution time (5,563 seconds). This means that decisions about task allocation are not trivial.

We have also performed experiments measuring mobile device energy consumption (using the PowerTutor software [51]) during the execution of tasks. Two classifiers are learned: $GK = \{GK_{ct}, GK_b\}$, with the former discussed above and the latter being used to predict battery usage from $O \cup Dec$ attributes. The $battery$ attribute is also discretized into 7 equal ranges. During these tests, the decision module takes into account calculation time ct and battery usage b in cost calculations (weights $w_b = 0.5$, $w_{ct} = 0.5$, $w_d = 0$). The results of those tests are shown in Fig. 6. As we

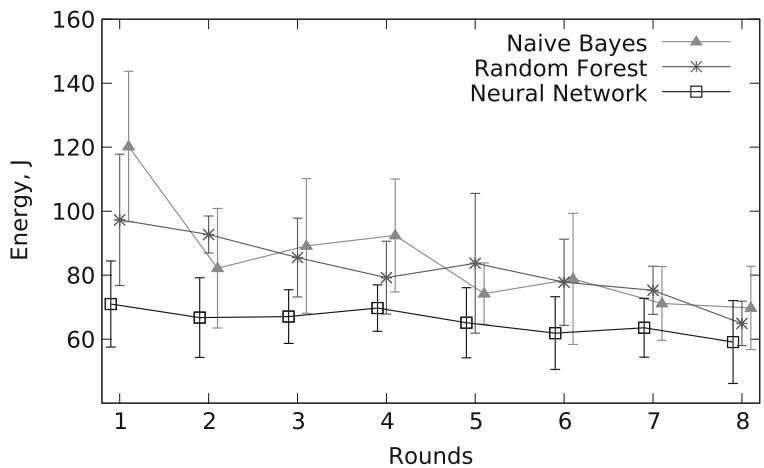

Fig. 6 Comparison of average battery usage in subsequent rounds for the three machine learning algorithms used to select the appropriate location for running a multimedia service

can see, the application of the machine learning algorithm reduces battery usage for the tasks requested. The average reduction in battery usage between the first and last rounds is equal to 41% for naïve Bayes, 33% for random forest, and 16% for neural network. According to the t-Student test, these differences are statistically significant for naïve Bayes (p-value is equal to 0.0001) and random forest (p-value is equal to 0.0002). For neural network, the difference is not statistically significant (p-value is equal to 0.0594), which may result from the default settings for this classifier in the Weka library.

Experimental results show that the execution time of multimedia data conversion services and battery usage were significantly decreased. It suggests that the autonomous context-based service optimisation method with the use of learning algorithms, which we have proposed, may be applied to improve Quality of Experience in real-life scenarios.

7 Conclusions

In this paper, we have proposed a novel solution for autonomous context-based service optimization in the MCC environment which includes a formal model of the learning decision module, service-oriented architecture, and service selection optimization algorithm. The solution is based on an analysis of possible approaches. After an investigation we have selected appropriate optimization methods in the MCC environment. The experiments related to the optimization of video file processing services have demonstrated that the main objective of our studies has been achieved and the expenses of the execution of services in the MCC environment have been optimized. The learning decision module proposed for selecting services in the MCC environment has been able to optimize the location where tasks are to be executed.

The supervised learning approach works very well and allows mobile devices to learn the models online and in an autonomous manner. The results demonstrate a statistically significant decrease in the time required for the execution of conversion services owing to the automatic selection (using learning methods) of the location (mobile device/cloud) where the conversion of multimedia data is to be performed. The decision module was able to autonomously collect training data, which was the input for the learning algorithm executed online on the mobile device.

Summing up, by applying new technologies, such as supervised, on-line learning performed in the MCC environment we have introduced a novel approach to optimizing the execution of services and the operating time of mobile devices in this environment. The solution is a general one and allows us to optimize various service-oriented systems where the location for executing a given task is flexible.

Further research in this area will include similar studies for different types of services with learning techniques, taking into account context data from the wide range of sensors installed in mobile devices such as GPS modules, accelerometers and light sensors. Research on discovering changes in conditions will be carried out. We also plan to apply other learning algorithms and investigate whether the exchange of learned knowledge between decision modules will prove beneficial.

Acknowledgements The research presented in this paper was supported by the Polish Ministry of Science and Higher Education under AGH University of Science and Technology Grant 11.11.230.124. We thank Małgorzata Płażek, Jakub Czyżewski and Daniel Olszowski for assistance with implementation and testing.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Roberts, J.R., Incorporated, M.: Mobile Tech Report 2014: Technology news from 2013 and predictions and insights about 2014. Mindwarm Incorporated (2014)
2. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur. Gener. Comput. Syst. **25**(6), 599–616 (2009)
3. Cushing, R., Koulouzis, S., Belloum, A., Bubak, M.: Applying workflow as a service paradigm to application farming. Concurrency and Computation: Practice and Experience **26**(6), 1297–1312 (2014)
4. Cushing, R., Putra, G.H.H., Koulouzis, S., Belloum, A., Bubak, M., De Laat, C.: Distributed computing on an ensemble of browsers. IEEE Internet Comput. **17**(5), 54–61 (2013)

5. Kumar, K., Lu, Y.-H.: Cloud computing for mobile users: can offloading computation save energy? Computer **43**(4), 51–56 (2010)
6. Kumar, K., Liu, J., Lu, Y.-H., Bhargava, B.: A survey of computation offloading for mobile systems. Mob. Netw. Appl. **18**(1), 129–140 (2013)
7. Shiraz, M., Gani, A., Shamim, A., Khan, S., Ahmad, R.W.: Energy efficient computational offloading framework for mobile cloud computing. Journal of Grid Computing **13**(1), 1–18 (2015)
8. Fernando, N., Loke, S.W., Rahayu, W.: Mobile cloud computing: a survey. Future Gener. Comput. Syst. **29**(1), 84–106 (2013)
9. Ma, R.K.K., Wang, C.-L.: Lightweight application-level task migration for mobile cloud computing. In: IEEE 26th International Conference on Advanced Information Networking and Applications (AINA), 2012, pp. 550–557 (2012)
10. Nawrocki, P., Sobon, M.: Public cloud computing for software as a service platforms. Comput. Sci. **15**(1), 89–103 (2014)
11. Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, pp. 301–314. ACM, New York, NY, USA (2011)
12. Satyanarayanan, M.: Mobile computing: the next decade, pp. 5:1–5:6. ACM, New York, NY, USA (2010)
13. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput. **13**(18), 1587–1611 (2013)
14. Yang, X., Pan, T., Shen, J.: On 3g mobile e-commerce platform based on cloud computing. In: 3rd IEEE International Conference on Ubi-Media Computing (U-Media), 2010, pp. 198–201 (2010)
15. Li, J.: Study on the development of mobile learning promoted by cloud computing. In: 2nd International Conference on Information Engineering and Computer Science (ICIECS), 2010, pp. 1–4 (2010)
16. Doukas, C., Pliakas, T., Maglogiannis, I.: Mobile healthcare information management utilizing cloud computing and android os. In: Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, pp. 1037–1040 (2010)
17. Tang, W.-T., Hu, C.-M., Hsu, C.-Y.: A mobile phone based homecare management system on the cloud. In: 3rd International Conference on Biomedical Engineering and Informatics (BMEI), 2010, vol. 6, pp. 2442–2445 (2010)
18. Wang, S., Dey, S.: Rendering adaptation to address communication and computation constraints in cloud mobile gaming. In: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pp. 1–6 (2010)
19. Karadimce, A., Davcev, D.: Building context-rich mobile cloud services for mobile cloud applications. In: Mesquita, A., Peres, P. (eds.) Proceedings of the 2nd European Conference on Social Media 2015: ECSM 2015, pp. 505–513 (2015)
20. Ye, Z., Chen, X., Li, Z.: Video based mobile location search with large set of sift points in cloud. In: Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing, MCMC '10, pp. 25–30. ACM, New York, NY, USA (2010)
21. Ahmed, E., Gani, A., Sookhak, M., Hamid, S.H.A., Xia, F.: Application optimization in mobile cloud computing. J. Netw. Comput. Appl. **52**(C), 52–68 (2015)
22. Ivesic, K., Skorin-Kapov, L., Matijasevic, M.: Cross-layer QoE-driven admission control and resource allocation for adaptive multimedia services in LTE. J. Netw. Comput. Appl. **46**(0), 336–351 (2014)
23. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agent. Multi-Agent Syst. **11**, 2005 (2005)
24. Stone, P., Veloso, M.: Multiagent systems: a survey from the machine learning perspective. Auton. Robot. **8**, 345–383 (2000)
25. Sniezynski, B.: A strategy learning model for autonomous agents based on classification. Int. J. Appl. Math. Comput. Sci. **25**(3), 471–482 (2015)
26. Bagchi, S.: The software architecture for efficient distributed interprocess communication in mobile distributed systems. Journal of Grid Computing **12**(4), 615–635 (2014)
27. Verbelen, T., Stevens, T., De Turck, F., Dhoedt, B.: Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. Futur. Gener. Comput. Syst. **29**(2), 451–459 (2013). Special section: Recent advances in e-Science
28. Khan, A.N., Mat Kiah, M.L., Khan, S.U., Madani, S.A.: Towards secure mobile cloud computing: a survey. Future Gener. Comput. Syst. **29**(5), 1278–1299 (2013)
29. Lin, H., Xu, L., Mu, Y., Wu, W.: A reliable recommendation and privacy-preserving based cross-layer reputation mechanism for mobile cloud computing. Future Gener. Comput. Syst. (2014)
30. Huang, D., Zhou, Z., Xu, L., Xing, T., Zhong, Y.: Secure data processing framework for mobile cloud computing. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2011, pp. 614–618 (2011)
31. Khan, A.N., Mat Kiah, M.L., Ali, M., Shamshirband, S., Khan, A.U.R.: A cloud-manager-based re-encryption scheme for mobile users in cloud environment: a hybrid approach. J. Grid Comput. **13**(4), 651–675 (2015)
32. Huerta-Canepa, G., Lee, D.: A virtual cloud computing provider for mobile devices. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS '10, pp. 6:1–6:5. ACM, New York, NY, USA (2010)
33. Cheng, R.K.B.alan.J., Satyanarayanan, M.: Exploiting Rich Mobile Environment. Technical Report Technical Report Carnegie Mellon university-CS-05-199. Carnegie Mellon University (2005)
34. Liu, R., Yuan, X., Xu, J., Chen, J., Zeng, Y., Cao, M., Liu, J., Xu, L., Fang, Q.: A novel server selection approach for mobile cloud streaming service. Simul. Model. Pract. Theory **50**, 72–82 (2015). Special Issue on Resource Management in Mobile Clouds
35. Nawrocki, P., Reszelewski, W.: Resource usage optimization in mobile cloud computing. Comput. Commun. **99**(C), 1–12 (2017)

36. Park, K.-L., Yoon, U.H., Kim, S.-D.: Personalized service discovery in ubiquitous computing environments. IEEE Pervasive Comput. **8**(1), 58–65 (2009)
37. Khan, A.U.R., Othman, M., Khan, A.N., Abid, S.A., Madani, S.A.: Mobibyte: an application development model for mobile cloud computing. J. Grid Comput. **13**(4), 605–628 (2015)
38. Kumar, N., Iqbal, R., Misra, S., Rodrigues, J.J.P.C.: Bayesian coalition game for contention-aware reliable data forwarding in vehicular mobile cloud. Futur. Gener. Comput. Syst. **48**, 60–72 (2015). Special Section: Business and Industry Specific Cloud
39. Liu, Q., Jian, X., Hu, J., Zhao, H., Zhang, S.: An optimized solution for mobile environment using mobile cloud computing. In: 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009. Wicom '09, pp. 1–5 (2009)
40. Skalkowski, K., Zielinski, K.: Automatic adaptation of soa systems supported by machine learning. In: Camarinha-Matos, L.M., Tomic, S., Graça, P. (eds.) Technological Innovation for the Internet of Things, vol. 394 of IFIP Advances in Information and Communication Technology, pp. 61–68. Springer, Berlin (2013)
41. Sniezynski, B.: Agent-based adaptation system for service-oriented architectures using supervised learning. Procedia Computer Science **29**, 1057–1067 (2014)
42. Anagnostopoulos, T., Anagnostopoulos, C., Hadjiefthymiades, S.: Mobility prediction based on machine learning. In: 2011 IEEE 12th International Conference on Mobile Data Management, vol. 2, pp. 27–30 (2011)
43. Shi, C., Pandurangan, P., Ni, K., Yang, J., Ammar, M., Naik, M., Zegura, E.: Ic-cloud: Computation Offloading to an Intermittently-Connected Cloud. Technical report (2013)
44. Donohoo, B.K., Ohlsen, C., Pasricha, S., Xiang, Y., Anderson, C.: Context-aware energy enhancements for smart mobile devices. IEEE Trans. Mob. Comput. **13**(8), 1720–1732 (2014)
45. Zhang, Y., Niyato, D., Wang, P.: Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Trans. Mob. Comput. **14**(12), 2516–2529 (2015)
46. Eom, H., Figueiredo, R., Cai, H., Zhang, Y., Huang, G.: Malmos: machine learning-based mobile offloading scheduler with online training. In: 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 51–60 (2015)
47. Nawrocki, P., Sniezynski, B., Czyzewski, J.: Learning agent for a service-oriented context-aware recommender system in heterogeneous environment. Computing and Informatics **35**(5), 1005–1026 (2016)
48. Malawski, M., Kuźniar, M., Wójcik, P., Bubak, M.: How to use google app engine for free computing. IEEE Internet Comput. **17**(1), 50–59 (2013)
49. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques with java implementations. Morgan Kaufmann (1999)
50. Marsan, R.J.: Weka for android. https://github.com/rjmarsan/weka-for-android
51. 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: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES/ISSS '10, pp. 105–114. ACM, New York, NY, USA (2010)