J Netw Syst Manage (2018) 26:1–22
https://doi.org/10.1007/s10922-017-9405-4

Adaptive Service Management in Mobile Cloud Computing by Means of Supervised and Reinforcement Learning

Piotr Nawrocki[1] · **Bartlomiej Sniezynski**[1]

Received: 19 February 2016 / Revised: 4 February 2017 / Accepted: 19 February 2017 /
Published online: 24 February 2017

Abstract Since the concept of merging the capabilities of mobile devices and cloud computing is becoming increasingly popular, an important question is how to optimally schedule services/tasks between the device and the cloud. The main objective of this article is to investigate the possibilities for using machine learning on mobile devices in order to manage the execution of services within the framework of Mobile Cloud Computing. In this study, an agent-based architecture with learning possibilities is proposed to solve this problem. Two learning strategies are considered: supervised and reinforcement learning. The solution proposed leverages, among other things, knowledge about mobile device resources, network connection possibilities and device power consumption, as a result of which a decision is made with regard to the place where the task in question is to be executed. By employing machine learning techniques, the agent working on a mobile device gains experience in determining the optimal place for the execution of a given type of task. The research conducted allowed for the verification of the solution proposed in the domain of multimedia file conversion and demonstrated its usefulness in reducing the time required for task execution. Using the experience

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 Michał Janiec for assistance with implementation and testing. Neither the entire paper nor any part of its content has been published or has been accepted for publication elsewhere. It has not been submitted to any other journal.

✉ Piotr Nawrocki
piotr.nawrocki@agh.edu.pl

Bartlomiej Sniezynski
bartlomiej.sniezynski@agh.edu.pl

[1] Department of Computer Science, Faculty of Computer Science, Electronics and Telecommunications, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Kraków, Poland

gathered as a result of subsequent series of tests, the agent became more efficient in assigning the task of multimedia file conversion to either the mobile device or cloud computing resources.

Keywords Machine learning · Optimization · Handheld device · Internet-based computing

1 Introduction and motivation

In recent years, an increase in the importance of mobile devices in computer systems has become apparent [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 Central Processing Unit (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.

The main objective of the studies described in this paper was to design an adaptive service management solution that would make it possible to optimize the cost of execution of services on mobile devices. The cost corresponds to the QoE (Quality of Experience) parameter and may take into account battery life, service execution time or user satisfaction. The adaptation (optimization) process should be performed online on the mobile device to take into account the heterogeneity of mobile systems and fluctuating conditions. We demonstrate that this can be done by applying agent-based solutions with learning capabilities in the Mobile Cloud Computing (MCC) environment. As a result of the research conducted, an agent-based model has been proposed that was used in the development of the case study environment related to multimedia file processing.

We have investigated the possibilities for optimizing the operation of services on mobile devices using the concept of cloud computing. We have noticed that it is possible to optimize the utilization of resources by moving selected tasks performed within the framework of individual services to the cloud. In order to accomplish this, we have decided to apply the MCC paradigm, which allows services to be performed in the cloud, and introduces agents that can decide under what circumstances, and which tasks should be executed in the cloud instead of on the mobile device. In decisions about migrating particular services (or parts thereof), the agents may use information about the condition of the device (memory, CPU, network interface card) and its sensors (light and location sensors, and the accelerometer) as well as the current time (hour, time of day, day of week, holidays) and potential costs (data transmission charges). In the meantime, they gather experience and autonomously learn how to make decisions on service migration

using machine learning. Thanks to the use of agents and the MCC concept, it is possible to optimize the performance of services on mobile devices, especially with respect to energy consumption, data transmission charges and execution time. The learning process is executed by the agent online, which makes it possible to use the agents' specific experience and adapt to changing conditions.

This paper is structured as follows: Section 2 contains a description of related work, Sect. 3 is concerned with employing the learning agent in the process of adapting the service selection strategy on mobile devices, Sect. 4 presents the case study, Sect. 5 describes the performance evaluation and Sect. 6 contains the conclusion.

2 Related Work

Along with the development of cloud computing [2] and mobile systems, research into the implementation of this concept in the mobile device environment began. In [3], the authors present an overview of the background, techniques and research areas for offloading computation to the cloud, which enables energy savings [4] and improves performance on mobile devices. In this way, the MCC paradigm [5], which enables the migration of individual services (tasks [6], data) from mobile devices to the cloud [7], emerged. On the basis of the general MCC concept, open (e.g. Open Mobster, Clonecloud [8]) and commercial (e.g. Perfecto Mobile) implementations of the solution have been developed. MCC is further developed [9] using two approaches—Mobile Computational Offloading (MCO) [10] and Cloud Assisted Mobile Augmentation (CMA) [11]. MCO defines the concept of dynamically offloading computation from mobile applications to the cloud infrastructure and CMA is a mobile augmentation model that employs the cloud to increase, enhance, and optimize the computing capabilities of mobile devices.

Many sample scenarios and applications using mobile devices and cloud computing have been described in literature [12]. In [13], the authors present typical MCC applications, such as:

- mobile commerce [14], comprising applications in the areas of finance, advertising, and shopping;
- mobile learning [15, 16], primarily concerning e-learning, accommodating mobile device hardware limitations;
- mobile healthcare, allowing easy and effective access to the patient's medical records [17] or enabling the monitoring of the patient's condition while at home [18];
- 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 [19].

Apart from the examples mentioned above, an increasing number of applications related to social networks [20] and using mobile device sensors [21] use the MCC paradigm.

The MCC paradigm yields many benefits [13], especially for mobile devices. One of the most vital advantages is enhancing battery life without the necessity of replacing 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 allows 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. Apart from possible problems related to wireless network communications (low bandwidth or data loss), other important challenges for MCC are notification methods [22] and security. Providing the proper level of data security [23] is a key aspect of MCC, especially in the context of e-health [24] or finance and banking applications.

The possibilities of employing the MCC paradigm in order to optimize services on mobile devices for numerous applications and scenarios [5, 25] have been discussed in literature. In [26, 27], the authors present the possibility of using this paradigm in image processing for an application that reads text (e.g. descriptions of exhibits in South Korean museums [27]) and translates it to 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 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 cloud and to gather comprehensive information on the missing person.

Other interesting approaches in the area of MCC are Mobile Assistance Using Infrastructure (MAUI) [28] and ThinkAir [29]. The MAUI system enables energy-aware offloading of mobile code. Developers can annotate which methods of the mobile application can be offloaded to the cloud. Once a method is called and a remote server is available, MAUI uses the optimization framework to decide whether the method should be offloaded depending on three factors: energy consumption, execution characteristics (time and resources) and network characteristics (including bandwidth, latency and packet loss). The ThinkAir is an universal Android framework for code offloading for mobile devices. This framework allows running some parts of the application's logic (methods) in a remote cloud-based service. It can be used by existing applications, with a minimal set of changes required from the developer and no modifications are needed in the Dalvik virtual machine. The decision whether to offload a method or not depends on profiling information and historical invocations of the method in question. The profiler (decision module) may be configured to minimize execution time, power consumption, cost or all variables at the same time.

A concept that extends today's cloud computing infrastructure is the cloudlet [30, 31]. It is a cloud-like structure located near to the edge devices it serves. The main purpose of the cloudlet is supporting resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency. The concept of cloudlet is also known as mobile edge computing, Follow-Me Cloud, and mobile micro-cloud [32].

A significant area of research is how efficiently the cloud can handle requests from a mobile application when cloud resources are limited. In [33], the authors propose a novel MCC adaptive resource allocation model using a semi-Markov decision process (SMDP) to manage resource requests. The model proposed achieves higher system performance and lower service blocking probability compared to classical solutions that are based on greedy resource allocation algorithms.

An important aspects of service adaptation are possibility of reconfiguration [34, 35] and providing an efficient mechanism—a repository—that allows searching for services in a distributed environment [36]. The services found should have a Service Level Agreement (SLA) parameter that specifies the conditions for their use [37].

As of yet, there have not been many studies of MCC and the use of agents that learn autonomously and 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 [38]. 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 computing complexity and requirements concerning the memory allocated to particular services. In another paper [39], the authors propose a learning agent for a service-oriented, context-aware recommender system using mobile devices.

There are several other studies of multi-agent system applications in mobile computing [40, 41]. A problem similar to the one considered here is presented in [42], where an agent-based system for MCC optimization is investigated. The main component of the system is the Execution Manager, which is a service on a mobile device that is responsible for deciding where to execute application components. In order to make this decision, a cost model is used, in which execution times are collected offline by the application profiler (in our solution we leverage online learning). Learning on a mobile device is used in [43], where an agent performing behavioral detection is discussed that samples selected system metrics (e.g. CPU usage, network communication, active processes, battery level) and detects anomalies using classification methods.

Autonomous reasoning about resources and tasks by agents is discussed in [44]. The domain knowledge is represented using an ontology but no learning is used. The application of machine learning algorithms in agent-based systems has been broadly discussed in literature. Valuable studies in this area are [45, 46]. In most cases, reinforcement learning or evolutionary computations are applied for the purpose of agent adaptation (see surveys [46, 47]).

In contrast to evolutionary computation where a population of agents is necessary, reinforcement learning [48] is particularly interesting because it enables

autonomous online learning. The learning agent model assumes that the agent interacts with the environment in discrete steps by observing the environment, choosing the appropriate action and executing it. Next, the agent receives a reward $r \in \mathbb{R}$. The reward is high if its actions are appropriate, and low if they are inappropriate. The agent has to learn which action should be executed in a given state. The formal model of learning is based on a Markov process. An interesting example of reinforcement learning application in mobile devices is [49] where media streaming is adapted.

There are also works where supervised learning has been applied to agent-based systems [50, 51] like in this paper. Using this method, a single agent can also learn a strategy autonomously and online [52]. This makes it possible to accelerate the learning process compared to reinforcement learning, especially if the state space is large [53, 54].

All the examples described assume that the user has decided to send the task or data from his or her mobile device to the 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 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 data transferred (and the related charges) and execution time. At the same time, the authors have conducted research into the use of agents and the supervised learning process in connection with deciding under what circumstances and which services (tasks, data) should be sent to, and executed in, the cloud.

3 Agent-Based Adaptive Service Management on a Mobile Device

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, e.g., 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 send 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. This allows the service management strategy to be adapted to the characteristics of a specific mobile device. Therefore a solution that uses agents that are 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.

3.1 Assumptions

An agent operating on a mobile device in the MCC paradigm monitors the environment and collects information on:

- the task to be executed in 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 the service;
- the location of the device (domestic/roaming) indicates whether the mobile device uses the data transmission service from local mobile operators (lower costs), or must use roaming service abroad (higher transmission costs).
- possible device connection modes (Wi-Fi, 2G/3G/4G) and connection quality affecting the 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 specifies 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, etc.) affect the ability to take advantage of better rates related to data transmission or to transmit/receive data during periods when the telecommunication operators' 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, etc.).

These data are called a Task Allocation Problem (*Problem* for short). The agent has to find a solution for the *Problems*: it makes decisions when and where to perform the service (locally or in the cloud). After completing the task, the agent assesses its decision, considering one or more criteria such as:

- mobile device power consumption;
- the time spent waiting for the result;
- the user's satisfaction (the user could override the agent'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).

The agent gathers experience and updates its strategy, applying some learning algorithm. Simultaneously, the agent may generate:

- 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.

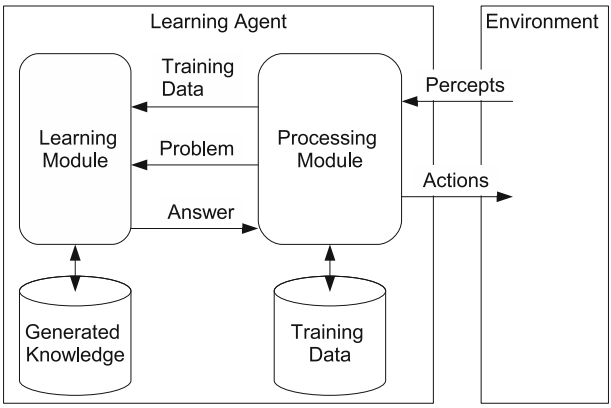

Fig. 1 The service management agent architecture reflecting its learning abilities

The internal structure of the agent should reflect its learning abilities. The architecture of the agent is presented in Fig. 1; the agent consists of four main modules:

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

These components interact in the following way: the *Processing Module* receives *Percepts* from the environment (parameters listed in the previous subsection), processes them and executes *Actions*. If the knowledge learned is needed during processing, it formulates a *Problem* representation (x^O) by describing observations with available attributes and sends it to the *Learning Module*, which generates an *Answer* for the *Problem* using *K*. The *Processing Module* also decides what data should be stored in *D* storage. When required (e.g. periodically or when *D* contains many new examples), it calls the *Learning Module* to execute the learning algorithm that generates new knowledge from *D*. The knowledge learned is stored in the *K* base.

Currently, all examples are stored in *D*. However, it is also possible to remove examples that are too old. Such an approach would resolve potential storage issues.

3.2 Task Allocation Q-Learning Agent

Let us define the Task Allocation Q-Learning Agent (Ag^Q) as a tuple:

$$Ag^Q = (T, S, A, Q), \tag{1}$$

where T is a set of attributes used to describe computational tasks that the agent is able to allocate; S is a set of attributes describing environment states representing a context in which the agent has to make the decision (e.g. current battery state, type of internet connection, position, time and date). A is a set of actions to represent local and cloud execution $A = \{\mathrm{l}, \mathrm{c}\}$. Q is a quality function that is used to select an action for a task in a given context.

There are many reinforcement learning algorithms. We have chosen Q-Learning [55] for our works but other algorithms could also be used. The algorithm of the agent is as follows:

1. The agent observes the task and the environment state and describes it with attributes $O = T \cup S = \{o_1, o_2, \ldots o_n\}$ which yields
$$x^O = (o_1(x), o_2(x), \ldots, o_n(x)) \in X^O. \tag{2}$$
2. To select an action to execute $a \in A$, the agent uses quality function $Q : X^O \times A \to \mathbb{R}$. Two strategies are tested in experiments:
   - highest – action $a \in A$ for which the quality $Q(x^O, a)$ has a maximum value is selected with $1 - \epsilon$ probability, and the other action with ϵ probability (it is ϵ-greedy exploration [48], which allows a local minimum to be avoided during learning);
   - proportional —the action is selected with a probability proportional to its Q value. This is not a typical strategy in reinforcement learning. It was used because a similar approach is applied in the case of the supervised learning agent (see below).
3. The agent observes the results of task execution: if it was successful (sometimes there may be e.g. a communication error), execution time, result quality, etc.
4. The agent calculates the reward $r \in \mathbb{R}$.
5. The agent observes and describes the next task and state x'^O.
6. It updates the Q function:
$$Q(x^O, a) := Q(x^O, a) + \alpha(\gamma \max_{act} Q(x'^O, act) + r - Q(x^O, a)), \tag{3}$$
   where $\alpha \in (0, 1)$ is the learning rate and $\gamma \in [0, 1]$ is a discount factor representing the importance of future rewards.
7. Goto 2.

In the experiments, we used γ=0. Therefore, step 5 may be omitted. Q may have a tabular representation or may be represented by some approximator. We have applied the latter approach with the use of neural networks (see [56]). There are two of them: Q_l, Q_c corresponding to both actions. As a result, network Q_a approximates $Q(x^O, a)$.

3.3 Task Allocation Supervised Learning Agent

This type of agent is described in a more detailed way because it is a novel approach to agent learning. Let us define the Task Allocation Supervised Learning Agent (Ag^{SL}) as a tuple:

$$Ag^{SL} = (T, S, R, K, D, A), \tag{4}$$

where T and S are sets of attributes defined above, R is a set of attributes describing the results of task execution (like in step 3 of the reinforcement learning agent). K is *generated knowledge*, D is *training data*, and A is a set of actions (defined above). K is knowledge generated by some supervised learning algorithm(s) from the agent's experience represented by D. This knowledge is used to choose the action.

The agent observes the task and the state and its processing module describes them with attributes $O = T \cup S$, which yields x^O, i.e. a description of the *Problem* as described in (2). The translation of observations from x to x^O depends on the exact domain in which the system is applied. In the experiments described below, we have only discretized numerical values using the equal frequency method, which splits the attribute's domain into intervals that contain equal numbers of examples. However, more advanced preprocessing (e.g. feature selection) may be required in more complex domains. In the next step, the agent using the knowledge stored in K solves the *Problem* by selecting $a \in A$, which has the minimum predicted cost. If K is empty, a is randomized.

For instance, K may be a decision tree used to predict the execution time of the task locally and in the cloud. If the Internet connection is slow and the task is executed in the cloud, the execution time will be long. If it is executed locally, it will be faster. K can be learned from D, which consists of examples of local and cloud task execution for various Internet connection types.

The agent's action a is then executed and the task is run locally or in the cloud. The agent observes execution results, which are described by $R = \{r_1, r_2, \ldots r_m\}$ attributes (e.g. whether execution was successful $es(x, a)$, battery consumption $b(x, a)$, calculation time $t(x, a)$, payments $p(x, a)$ representing costs of cloud services and user's dissatisfaction $d(x, a)$, which may be measured by observing if the user overrode the agent's decision). Therefore, the set of all attributes used to describe percepts is a sum of O and R:

$$Attr = O \cup R. \tag{5}$$

The agent stores these results together with x^O and action a in D. Therefore the complete example x description stored in D has the form

$$x^{Attr \cup A} = (o_1(x), o_2(x), \ldots o_n(x), r_1(x, a), r_2(x, a), \ldots r_m(x, a), a). \tag{6}$$

The models to predict R values are constructed using supervised machine learning algorithms and stored in K. These models influence the choice of the action taken.

Using predictions of $r_i \in R$, the agent may rate its decisions $a \in A$ for the observed task and context x^O by calculating predicted costs $e(x, a)$:

$$e(x,a) = \sum_{i=1}^{m} w_{r_i} r_i(x,d), \tag{7}$$

where w_{r_i} are weights of the result r_i. Currently the weights are set by hand. However, some adaptation algorithm can also be used to tune them. The agent should select the action for which the cost is predicted to be the lowest. If the weight of computation time is much higher than the other weights, the fastest execution location should be selected. If the weight of payments is much higher than the others, the cheapest location should be selected, even if it takes more time.

The weighted sum is only one of the methods for generating non-dominated solutions for Multiple-criteria decision-making. Other methods of choosing the action are also possible (from a simple ordering of criteria to complex non-dominated solution finders [57], Lexical Evaluation Function [58] or Analytic Hierarchy Process (AHP) [59]). However, the weighted sum affords the flexibility to simultaneously take several criteria into account, while still being easy to explain to the user. It may be implemented as a set of sliders corresponding to individual criteria.

As mentioned above, it may happen that for some types of tasks, execution on a mobile phone or in the cloud fails (e.g. because of insufficient memory). To deal with that, an additional classifier can be used: $K_e s : O \cup A \rightarrow \{\text{yes}, \text{no}\}$ predicting if the calculation will be successful. If for some action the prediction is *no*, the other action is selected.

The use of the "user satisfaction" criterion allows for the assessment of user experience and makes it possible to account for it in the learning algorithm. As it has been mentioned above, the user may set the weight of this criterion in the calculation. If $w_d = 0$, the decision of the agent does not take user experience into consideration and is based on other criteria related to computation efficiency aspects.

The agent's algorithm, which is executed in the *Processing Module*, is presented in Fig. 2. At the beginning, K and D are empty. Next, if there is no learned knowledge, the action is randomized. If there is some knowledge, the *Problem* variable is a description of the observations (x^O). Next, the *Learning Module* is used to select the best action for the *Problem* according to the current knowledge. Results are observed and example ($x^{Attr \cup A}$) is stored in D. After processing a given number of tasks, the *Learning module* is called to generate a new knowledge from D. This knowledge is stored in K.

The form of the knowledge stored in K 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 a lower-level form such as parameters representing a linear regression model, an action-value function or a neural network approximator of such a function if reinforcement learning is applied. It is also possible to store more than one model in K.

```
1  begin
2      Generated Knowledge:= ∅;
3      Training Data:= ∅;
4      while agent is alive do
5          if Generated Knowledge = ∅ then
6              a:= random action
7          end
8          else
9              Problem := description of the current (observed) state;
10             a:= action determined for Problem by model(s) stored in
               Generated Knowledge
11         end
12         execute a;
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
```

Fig. 2 Learning agent algorithm making it possible to generate the strategy for the agent using online supervised learning based on the agent's experience

4 Mobile Multimedia Processing System

In order to verify the possibility of using agents to optimize service management in MCC, the authors developed a Mobile Multimedia Processing System that enables the processing of multimedia files. Varying demands on computational power for different tasks and the necessity to transfer various amounts of data in case of processing in the cloud make this case a representative domain for testing the solution proposed.

The system developed consists of two components: the learning agent (which works within a mobile application) and the multimedia data conversion service (which operates on the mobile device and in the cloud). A detailed description of the learning agent was presented in the previous section. In the solution developed, the multimedia data conversion service runs locally on the mobile device or remotely in the cloud in exactly the same way (using the same libraries). This enables a comparison of the effectiveness of the conversion service in two environments. In further studies, it will be possible to diversify the conversion service in such a way that it utilizes different libraries optimized either for mobile devices or the cloud environment. In such a situation, it will be necessary to distinguish between different components: the conversion service on the mobile device and the conversion service in the cloud.

During tests, multimedia files were converted using the cloud or directly on the mobile device (Google Nexus 5 with Android 5.0). 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 that data, the mobile application (learning agent) selected the conversion

type and place that would offer better expected efficiency. Figure 3 shows the component diagram of the solution developed.

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. 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.

Two tasks were related to the conversion of multimedia files. For this purpose, the *jcodec* library was used. It can operate both on a mobile device with the Android operating system and in the Google App Engine cloud. This enabled the comparison of conversion times on the mobile device and in the cloud when the same mechanism was used for encoding multimedia files. The *jcodec* library was used to implement the following conversion types:

- from the H.264 Advanced Video Coding (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 Portable Network Graphics (PNG) images (where every 5th, 15th or 25th frame was encoded).

The third type of task was the conversion from a PNG image into a Portable Document Format (PDF) document. It was implemented using the *pdfJet* library on a mobile device and in the cloud.

The Android operating system version of the Weka library [60] (WekaforAndroid) was used to implement learning algorithms and the application of learned knowledge.

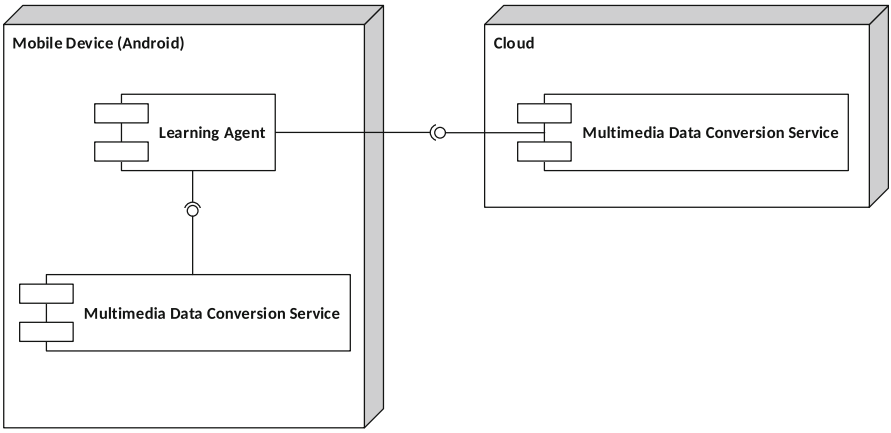

Fig. 3 Component diagram of the Mobile Multimedia Processing System

In the domain considered, the proposed solution may be specified in detail as follows. The task type is described by three attributes: $type_{\text{codec}}, type_{\text{pdf}}, type_{\text{frame}}$. The first two are binary (with domains of $\{0, 1\}$), the third domain is $\mathbb{N}$ and a value greater than zero represents the number of frames to skip during frame selection. Only one of these attributes may have a value greater than zero and this represents the corresponding type of task. The last attribute *length* represents file size and has a [0,1) domain. $length = \frac{fl}{fl+1}$, where *fl* is the file size in MB. State attributes S consist of four attributes: the binary attribute *charging* representing charger connection, *connection* representing internet connection type (a value of 0 represents no connection, 0.5 represents a GSM connection, and 1 represents Wi-Fi), *battery* representing battery level (with a [0,1] domain) and *time* representing the minute of the day ($time = \frac{min}{24*60}$, where *min* is the current minute of the day). As a result $O = \{type_{\text{codec}}, type_{\text{pdf}}, type_{\text{frame}}, length, connection, time, \quad battery\}$. Results are described by the *es* Boolean attribute showing whether the execution was successful or not, b representing battery usage and the numeric t attribute representing calculation time in seconds. Therefore $R = \{es, b, t\}$.

In the supervised learning approach, two models are built. The first one is K_{es} – a Naïve Bayes classifier, which allows to predict the *es* category from $O \cup A$ attributes. The second one is K_t – a linear regression model which is used to predict t from $O \cup A$. As a result, $K = (K_{es}, K_t)$.

Currently, the agent takes into account only $t(a)$ in Eq. (7): $w_b = w_d = 0, w_t = 1$. Hence, the action is selected in the following way: first, K_{es} is used to determine if for the task observed, *to* and state s calculations will be successful locally: $succ_l = K_{es}(to, s, l)$ and using the cloud: $succ_c = K_{es}(to, s, c)$. Next, computation time is predicted for both resources: $t_l = K_t(to, s, l)$, $t_c = K_t(to, s, c)$. If $succ_l$ is true and $succ_c$ is false, local calculation $a = \text{l}$ is selected. Conversely, if $succ_c$ is true and $succ_l$ is false, cloud execution $a = \text{c}$ is selected. If both predictions are false, the action is randomized with a uniform probability distribution. If both predictions are true, the action is randomized with the probability of action a inversely proportional to calculation time: $p(a) = 1 - \frac{t_a}{t_l + t_c}$. The reward r of the reinforcement learning agent corresponds to negative costs (see Eq. 7).

5 Performance Evaluation

Each experiment consists of a series of rounds. In each round, an identical task package is executed. Each task package is a combination of selected task parameters and system state values. This means that each test package involves measuring the codec conversion time for ten multimedia files (sizes from 60 kB to 650 kB) and two PNG to PDF conversions for two files (sizes 8 and 14 kB). These tasks are executed for two network connection types (Wi-Fi/3G). This yields 24 tasks per package.

During the series of measurements conducted, information was collected on conversion time and its result (task execution success or failure). The experiment was carried out using a mobile device—LG Nexus 5. This mobile device has the

following specifications: SoC—Qualcomm Snapdragon 800, CPU—2.26 GHz quad-core Krait 400, Graphics Processing Unit (GPU)—Adreno 330, 450 MHz, memory—2 GB of LPDDR3-1600 RAM and storage—16 GB.

In the case of reinforcement learning, the Q function is reinitialized before the first round and the function is updated after each task execution.

In the case of supervised learning, examples ($x^{O \cup R \cup A}$) are stored in D after each task execution. When the full task package has been executed, the agent initiates the learning process and builds the K_{es} classifier using Naïve Bayes and K_t using linear regression, which are used in the next round to process tasks. During round n, the agent uses for learning the examples collected in rounds $1...n-1$. The learning process is presented in Fig. 4. When the series is completed, D and K are cleared and the next series is executed to collect statistical data.

For each round, the task package execution time is measured. By execution time we mean elapsed (wallclock) time. If task execution results in failure, task execution time is set to the maximum successful execution time observed. In all experiments, ten rounds were executed and repeated ten times to collect statistical data.

In the first experiment, supervised learning was used. The results of those tests are shown in Table 1.

The result achieved demonstrated that the employment of supervised machine learning in multimedia and PNG to PDF file conversion tests caused a significant reduction in the time required for the execution of such tasks. The time required for the execution of the task in the tenth round of tests was significantly shorter than in the first one (when machine learning was not employed), however slightly longer than during the fourth round of tests. The difference between the first and tenth rounds is statistically significant using the Student's t-test (the p-value is less than 0.0001).

The second and third experiments were carried out under the same conditions but using a reinforcement learning algorithm. In the second experiment, the action is selected according to the highest profit strategy. The results of that experiment are shown in Table 2.

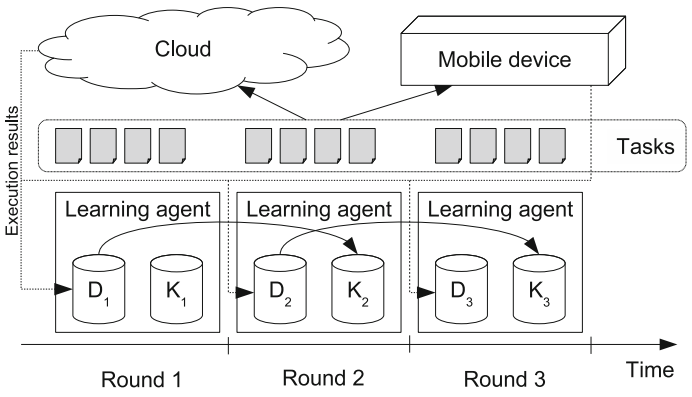

Fig. 4 Process of task allocation learning in the consecutive rounds

Table 1 Test results for supervised learning

Round	1	2	3	4	5	6	7	8	9	10
Minimum execution time (s)	472	203	149	134	184	219	181	211	186	206
Maximum execution time (s)	781	516	493	423	410	412	411	405	390	337
Average execution time (s)	**573**	392	326	298	305	323	273	314	306	**262**
Standard deviation (s)	100	87	95	89	77	59	68	74	70	57

The first and the last average results are given in bold to highlight performance improvement which is result of learning

Table 2 Test results for reinforcement learning and Q-learning algorithm (solution with the highest estimated profit)

Round	1	2	3	4	5	6	7	8	9	10
Minimum execution time (s)	410	361	152	121	118	135	159	139	124	144
Maximum execution time (s)	828	793	731	476	489	482	458	407	382	484
Average execution time (s)	**612**	548	370	281	267	290	299	275	273	**265**
Standard deviation (s)	158	124	182	108	106	105	106	104	102	104

The first and the last average results are given in bold to highlight performance improvement which is result of learning

In the third experiment, the action is selected according to the proportional strategy. The results of the third experiment are shown in Table 3.

As can be seen in Tables 1, 2, and 3, the employment of machine learning in making the decision as to the place where the task should be carried out (mobile device/cloud) significantly increases the execution speed of the tasks requested. The examined mean task execution time in subsequent rounds of tests using the knowledge gained as a result of machine learning shows a downward trend. The time decrease is statistically significant in both cases. The p value for the second experiment is less than 0.0001 and for the third experiment it equals 0.0009.

A comparison of the results of all three experiments is presented in Fig. 5. Supervised learning with the regression model was learning quickly, outperforming reinforcement learning in the second and third rounds. However, supervised learning finally yielded slightly better results. This difference is not statistically

Table 3 Test results for reinforcement learning and Q-learning algorithm (solution proportional to the estimated profit)

Round	1	2	3	4	5	6	7	8	9	10
Minimum execution time (s)	380	285	264	358	177	179	290	238	224	221
Maximum execution time (s)	795	692	632	620	586	523	588	536	567	512
Average execution time (s)	**603**	495	425	476	417	404	413	424	399	**394**
Standard deviation (s)	136	137	103	92	124	103	102	99	109	97

The first and the last average results are given in bold to highlight performance improvement which is result of learning

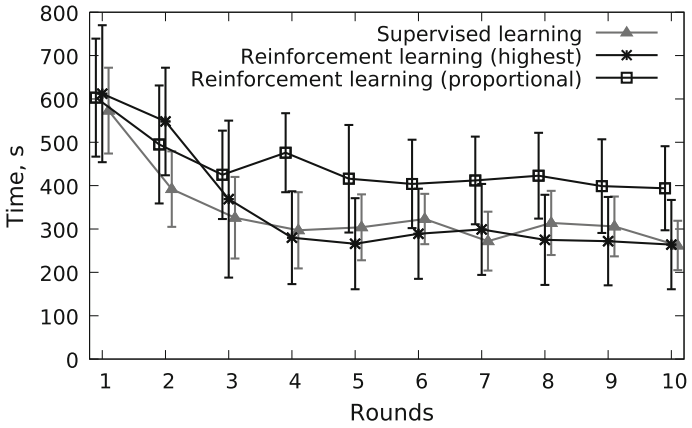

Fig. 5 Comparison of the results of various machine learning algorithms applied in adaptive service management

significant using the Student's t test (the p value equals 0.9371). Reinforcement learning using the proportional strategy yielded the worst results. The difference between both reinforcement learning approaches is statistically significant using the Student's t test (the p value equals 0.0102). The reason is that the exploration rate in the proportional case is too high.

One can observe that the standard deviation of execution times only decreases in the beginning. There are two reasons for that. The first is exploration, which results in non-optimal action execution. The second one are random delays observed when the task is executed in the cloud. We used standard free-of-charge Google cloud services. Google has cloud centers in USA, Asia and Europe (where the experiments were run). We had no influence on cloud load balancing or resource location. However, in our opinion, the adoption of such an approach results in a real-world scenario.

For a more valid comparison, we need at least 30 rounds in each group. We can create such groups using existing data. The first group consists of initial rounds taken from all experiments (10 from supervised and 10 from every reinforcement learning experiment), which gives 30 rounds in total. The second group consists of 30 last rounds (8–10) from every experiment. As a result, we can assume a normal distribution (because of the Central Limit Theorem) and apply the t-Student test. For each learning algorithm, the average from group 1 is significantly higher than average from group 2 (p value is less than 0.0001).

In addition, energy consumption measurements (using PowerTutor [61]) were conducted during supervised learning tests. Those measurements made it possible to validate the use of metrics other than execution time and demonstrated that the employment of supervised machine learning in multimedia and PNG to PDF file conversion tests also caused a significant reduction (of about 42%) in the battery usage required for the execution of such tasks (Table 4). The Student's t test confirmed that the difference is statistically significant (the p value equals 0.0042). We also measured energy consumption during the learning process and established

Table 4 Test results of energy consumption for supervised learning

Round	1	2	3	4	5	6	7
Energy (J)	**120**	82	89	96	54	78	**69**
Standard deviation (J)	44	22	21	22	10	20	22

The first and the last average results are given in bold to highlight performance improvement which is result of learning

that supervised machine learning algorithm consumed less than 0.1% of battery capacity for learning.

In order to compare the solution developed with others, we have analyzed most of the environments currently available such as Adaptive Code Offloading for Mobile Cloud Applications, AIOLOS, AlfredO, CACTSE, COMET, COSMOS, Cuckoo, Elijah, EMCO, IC-Cloud, MALMOS, MAUI, Mirroring Mobile Device, Mobile Cloud Execution Framework, Mobility Prediction based on Machine Learning, MOCHA, Replicated Application Framework, ThinkAir and VMCC. We have taken into account a variety of properties such as the optimization of execution times, the optimization of energy consumption, the use of virtual machines and the existence of a decision-making process. Most of these environments are not commercial solutions but rather ones developed at universities and tested with respect to the optimization of service execution times. Many of these solutions have now been abandoned and only a few (AIOLOS, CACTSE, Cuckoo, EMCO, IC-Cloud, MAUI and ThinkAir) take into account energy-saving aspects in addition to time optimization. In addition, only two of the environments analyzed (MALMOS and IC-Cloud) utilize machine learning algorithms in order to determine the optimal location for executing the service. We have tried to compare these two environments to our solution but both are no longer being developed and it was not possible to run them.

During our tests, we were able to compile examples related to two environments (AIOLOS and Cuckoo) but only managed to run the latter (Cuckoo) in an old version of Eclipse. In further work, we plan to add machine learning mechanisms to the decision module in Cuckoo and compare this environment with the solution developed.

6 Conclusions

In this paper, we investigated the possibilities for adaptive service management enabling the optimization of service execution in an MCC environment. We developed an agent-based architecture that uses supervised learning and is designed for MCC, which is a novel solution. 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 cost of service execution on mobile devices has been optimized. The learning agent solution proposed for selecting services in the

MCC environment has been able to optimize the location where tasks are to be executed.

The results demonstrate a significant decrease in the time required for the execution of conversion services owing to the automatic selection (using machine learning methods) of the location (mobile device/cloud) where the task is to be performed. The learning algorithm was executed by the learning agent autonomously and online on the mobile device using the agent's own experience from the past. Both supervised and reinforcement learning methods appear to be appropriate for this application. However, reinforcement learning is much simpler and therefore computationally less expensive. As a result, in conditions similar to the ones in experiments, reinforcement learning seems to be a better choice. Nevertheless, if the state space is larger, i.e. we have more attributes describing tasks or context (or they have larger domains), supervised learning yields improvements faster than reinforcement learning [62]. Knowledge learned may be also represented in a human readable form, if an appropriate learning algorithm is used (e.g. C4.5). It enables the verification of the knowledge learned by human experts, which can be important during the development of the system.

Further research in this area could involve similar studies for different types of services using machine learning techniques, taking into account CPU usage during task execution and a wide range of sensors installed in mobile devices such as Global Positioning System (GPS) modules, accelerometers and light sensors. In the future, we plan to perform tests of energy consumption for reinforcement learning and the Q-learning algorithm. We also plan to apply other learning algorithms (especially with symbolic knowledge representation), investigate whether the exchange of learned knowledge between agents will prove beneficial and compare the performance of the solution developed with others. We also plan to add learning rate adaptation the to reinforcement learning algorithm to deal with rapid changes of conditions in the cellular network.

References

1. Roberts, J., 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. Future Gener. Comput. Syst. **25**(6), 599–616 (2009)
3. 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)
4. Hlavacs, H., Hummel, K.A., Weidlich, R., Houyou, A.M., Meer, H.D.: Modelling energy efficiency in distributed home environments. Int. J. Commun. Netw. Distrib. Syst. **4**(2), 161–182 (2010)
5. Fernando, N., Loke, S.W., Rahayu, W.: Mobile cloud computing: a survey. Future Gener. Comput. Syst. **29**(1), 84–106 (2013)

6. Ma, R., Wang, C.L.: Lightweight application-level task migration for mobile cloud computing. In: Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on. (March 2012) pp. 550–557
7. Nawrocki, P., Sobon, M.: Public cloud computing for software as a service platforms. Comput. Sci. **15**(1), 89–103 (2014)
8. 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, New York, NY, USA, ACM (2011) pp. 301–314
9. Khanna, A.: Sarishma: Mobile Cloud Computing: Principles and Paradigms. I K International Publishing House, New Delhi (2015)
10. Juntunen, A., Kemppainen, M., Luukkainen, S.: Mobile Computation Offloading—Factors Affecting Technology Evolution. In: International Conference on Mobile Business, ICMB 2012, Delft, The Netherlands, June 21–22, 2012. (2012) pp. 9
11. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R.: Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges. CoRR **abs/1306.4956** (2013)
12. Satyanarayanan, M.: Mobile computing: The next decade. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond. MCS '10, New York, NY, USA, ACM (2010) 5:1–5:6
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: Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on. (July 2010) pp. 198–201
15. Chen, X., Liu, J., Han, J., Xu, H.: Primary Exploration of Mobile Learning Mode Under a Cloud Computing Environment. In: E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on. Vol 2. (April 2010) pp. 484–487
16. Li, J.: Study on the Development of Mobile Learning Promoted by Cloud Computing. In: Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on. (Dec 2010) pp. 1–4
17. 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. (Aug 2010) pp. 1037–1040
18. Tang, W.T., Hu, C.M., Hsu, C.Y.: A Mobile Phone Based Homecare Management System on the Cloud. In: Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on. Vol 6. (Oct 2010) pp. 2442–2445
19. Wang, S., Dey, S.: Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming. In: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE. (Dec 2010) pp. 1–6
20. Li, H., Hua, X.S.: Melog: Mobile Experience Sharing Through Automatic Multimedia Blogging. In: Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing. MCMC '10, New York, NY, USA, ACM (2010) pp. 19–24
21. 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, New York, NY, USA, ACM (2010) pp. 25–30
22. Nawrocki, P., Jakubowski, M., Godzik, T.: Analysis of Notification Methods with Respect to Mobile System Characteristics. In: 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015, Lódz, Poland, 13–16 Sept 2015. pp. 1183–1189
23. 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)
24. Huang, D., Zhou, Z., Xu, L., Xing, T., Zhong, Y.: Secure Data Processing Framework for Mobile Cloud Computing. In: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. (April 2011) pp. 614–618
25. Ahmed, E., Gani, A., Sookhak, M., Hamid, S.H.A., Xia, F.: Application optimization in mobile cloud computing: motivation, taxonomies, and open challenges. J. Netw. Comput. Appl. **52**, 52–68 (2015)
26. 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, New York, NY, USA, ACM (2010) 6:1–6:5

27. Cheng, J., Balan, R.K., Satyanarayanan, M.: Exploiting rich mobile environment. Technical Report Technical Report Carnegie Mellon University-CS-05-199, Carnegie Mellon University (2005)
28. Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: Making Smartphones Last Longer with Code Offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. MobiSys '10, New York, NY, USA, ACM (2010) pp. 49–62
29. Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X.: Thinkair: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. In: INFOCOM, 2012 Proceedings IEEE. (March 2012) pp. 945–953
30. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
31. Verbelen, T., Simoens, P., De Turck, F., Dhoedt, B.: Cloudlets: Bringing the Cloud to the Mobile User. In: Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services. MCS '12, New York, NY, USA, ACM (2012) pp. 29–36
32. Wang, S., Tu, G.H., Ganti, R., He, T., Leung, K., Tripp, H., Warr, K., Zafer, M.: Mobile Micro-Cloud: Application Classification, Mapping, and Deployment. In: Proc. of Annual Fall Meeting of ITA (AMITA). (2013)
33. Liang, H., Xing, T., Cai, L.X., Huang, D., Peng, D., Liu, Y.: Adaptive computing resource allocation for mobile cloud computing. Int. J. Distrib. Sens. N. **2013**, 181426 (2013). doi:10.1155/2013/181426
34. Brzoza-Woch, R., Nawrocki, P.: Fpga-based web services—infinite potential or a road to nowhere? IEEE Internet Comput. **20**(1), 44–51 (2016)
35. Bachara, P., Brzoza-Woch, R., Dlugopolski, J., Nawrocki, P., Ruta, A., Zaborowski, W., Zielinski, K.: Construction of hardware components for the internet of services. Comput. Inf. **34**(4), 911–940 (2015)
36. Nawrocki, P., Mamla, A.: Distributed web service repository. Comput. Sci. **16**(1), 55 (2015)
37. Kosinski, J., Nawrocki, P., Radziszowski, D., Zielinski, K., Zielinski, S., Przybylski, G., Wnek, P.: SLA Monitoring and Management Framework for Telecommunication Services. In Bi, J., Chin, K., Dini, C., Lehmann, L., Pheanis, D.C., eds.: Networking and Services, 2008. ICNS 2008. Fourth International Conference on, IEEE Computer Society (2008) pp. 170–175
38. Liu, Q., Jian, X., Hu, J., Zhao, H., Zhang, S.: An optimized solution for mobile environment using mobile cloud computing. In: Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on. (Sept 2009) pp. 1–5
39. Nawrocki, P., Sniezynski, B., Czyzewski, J.: Learning agent for a service-oriented context-aware recommender system in a heterogeneous environment. Comput. Inf. **35**(5), 1005–1026 (2016)
40. Abidar, R., Moummadi, K., Medromi, H.: Mobile Device and Multi Agent Systems: An Implemented Platform of Real Time Data Communication and Synchronization. In: Multimedia Computing and Systems (ICMCS), 2011 International Conference on. (April 2011) pp. 1–6
41. Sankaranarayanan, S., Cuffe, K.: Intelligent Agent Based Scheduling of Student Appointment-Android Environment. In: Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on. (Nov 2010) pp. 46–51
42. Angin, P., Bhargava, B.: An agent-based optimization framework for mobile-cloud computing. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. **4**(2), 1–17 (2013)
43. Shabtai, A., Elovici, Y.: Applying behavioral detection on android-based devices. In: Cai, Y., Magedanz, T., Li, M., Xia, J., Giannelli, C. (eds.) Mobile Wireless Middleware, Operating Systems, and Applications. Lecture Notes of the Institute for Computer Sciences, vol. 48, pp. 235–249. Social Informatics and Telecommunications Engineering. Springer, Berlin Heidelberg (2010)
44. Sensoy, M., Vasconcelos, W.W., Norman, T.J., Sycara, K.: Reasoning support for flexible task resourcing. Expert Syst. Appl. **39**(2), 1998–2010 (2012)
45. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agents Multi Agent Syst. **11**, 2005 (2005)
46. Tuyls, K., Weiss, G.: Multiagent learning: basics, challenges, and prospects. AI Mag. **33**(3), 41–52 (2012)
47. Sen, S., Weiss, G.: Learning in Multiagent Systems. MIT Press, Cambridge (1999)
48. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (1998)
49. Charvillat, V., Grigora, R.: Reinforcement learning for dynamic multimedia adaptation. J. Netw. Comput. Appl. **30**(3), 1034–1058 (2007)

50. Singh, D., Sardina, S., Padgham, L., Airiau, S.: Learning Context Conditions for bdi Plan Selection. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. AAMAS '10, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2010) pp. 325–332
51. Czarnowski, I., Jedrzejowicz, P.: Machine learning and multiagent systems as interrelated technologies. In: Czarnowski, I., Jedrzejowicz, P., Kacprzyk, J. (eds.) Agent-Based Optimization. Studies in Computational Intelligence, vol. 456, pp. 1–28. Springer, Berlin Heidelberg (2013)
52. Sniezynski, B.: Agent strategy generation by rule induction. Comput. Inf. **32**(5), 1055–1078 (2013)
53. Sniezynski, B., Dajda, J.: Comparison of strategy learning methods in farmer-pest problem for various complexity environments without delays. J. Comput. Sci. **4**(3), 144–151 (2013)
54. Sniezynski, B.: Comparison of reinforcement and supervised learning methods in farmer-pest problem with delayed rewards. In: Badica, C., Nguyen, N.T., Brezovan, M. (eds.) Computational Collective Intelligence. LNCS, vol. 8083, pp. 399–408. Springer, Berlin Heidelberg (2013)
55. Watkins, C.J.C.H.: Learning from Delayed Rewards. PhD thesis, King's College, Cambridge (1989)
56. Rummery, G.A., Niranjan, M.: On-line q-learning using connectionist systems. Technical report. Cambridge University Engineering Department, Cambridge (1994)
57. Bragge, J., Korhonen, P., Wallenius, H., Wallenius, J.: Bibliometric Analysis of Multiple Criteria Decision Making/Multiattribute Utility Theory. Springer, Berlin, Heidelberg (2010)
58. Michalski, R.S.: AQVAL/1—Computer Implementation of a Variable Valued Logic VL1 and Examples of its Application to Pattern Recognition. In: Proc. of the First International Joint Conference on Pattern Recognition. (1973)
59. Saaty, T.L.: The Analytic Hierarchy Process : Planning, Priority Setting, Resource Allocation. McGraw-Hill International Book Co., New York; London (1980)
60. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, Los Altos (1999)
61. 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, New York, NY, USA, ACM (2010) 105–114
62. Sniezynski, B.: A strategy learning model for autonomous agents based on classification. Int. J. Appl. Math. Comput. Sci. **25**(3), 471–482 (2015)

Piotr Nawrocki, Ph.D., is an Assistant Professor in the Department of Computer Science at the AGH University of Science and Technology, Krakow, Poland. His research interests include distributed systems, computer networks, mobile systems, mobile cloud computing, Internet of Things and service-oriented architectures. He has participated in several national and EU research projects including MECCANO, 6WINIT and UniversAAL. He is a member of the Polish Information Processing Society (PTI).

Bartlomiej Sniezynski received his Ph.D. degree in Computer Science in 2004 from AGH University of Science and Technology in Krakow, Poland. In 2004 he worked as a Postdoctoral Fellow under the supervision of Professor R. S. Michalski at the Machine Learning and Inference Laboratory, George Mason University, Fairfax, USA. Currently, he is an assistant professor in the Department of Computer Science at AGH. His research interests include machine learning, multi-agent systems, and knowledge engineering.