Selective Mobile Cloud Offloading to Augment Multi-Persona Performance and Viability

Hanine Tout, *Student Member, IEEE*, Chamseddine Talhi, Nadjia Kara, and Azzam Mourad, *Senior Member, IEEE*

Abstract—Fueled by changes in professional application models, personal interests and desires and technological advances in mobile devices, multi-persona has emerged recently to keep balance between different aspects, in our daily life, on a single mobile terminal. In this context, mobile virtualization technology has turned the corner and currently heading towards widespread adoption to realize multi-persona. Although recent lightweight virtualization techniques were able to maintain balance between security and scalability of personas, the limited CPU power and insufficient memory and battery capacities, still threaten personas performance and viability. Throughout the last few years, cloud computing has cultivated and refined the concept of outsourcing computing resources, and nowadays, in the coming age of smartphones and tablets, the prerequisites are met for importing cloud computing to support resource constrained mobiles. From these premises, we propose in this paper a novel offloading-based approach that based on global resource usage monitoring, generic and adaptable problem formulation and heuristic decision making, is capable of augmenting personas performance and viability on mobile terminals. The experiments show its capability of reducing the resource usage overhead and energy consumption of the applications running in each persona, accelerating their execution and improving their scalability, allowing better adoption of multi-persona solution.

Index Terms—Multi-persona, mobile device, mobile virtualization, mobile cloud computing, offloading, multi-objective optimization, heuristic algorithms

1 Introduction

In today's high-tension and fast-paced world, technological advances have changed the concept of mobile devices from primitive gadget to full computers that accommodate work, personal and mobility needs. Out of this wave, bring your own device (BYOD) revolution has emerged across a variety of industries, as a policy to allow end-users to use personally owned mobile devices for business tasks [1]. However, clearly having personally owned devices accessing corporate data and apps raise a number of risks and security concerns. One solution is corporate-owned, personally-enabled (COPE), a business model in which employees use corporate issued devices [2]. Yet, by granting manageability to enterprises, employees are sacrificing both usability and privacy. Another solution is to carry two mobile devices, but the natural tendency for most people is to combine professional and personal needs on the same physical device, hence again not a good solution. As an alternative winning technology, mobile devices with dual persona were released enabling two phones-in-a-phone, one for private personal use and another for business use [3].

However, nowadays, multi-persona has become the name of the game. Customizing isolated personas for banking services, e-commerce, corporate data, social networking and games, allow parents to efficiently manage financial transactions, prevent untrusted applications from accessing critical information and share the device with children without ending up with accidental phone calls, unintended in-app purchases or even access to restricted content [4]. Also, traveling for a conference or attending a trade show is a very common practice for businesspersons, who are not tied anymore with just corporate and personal personas. An additional persona customized by the event coordinator to push relevant apps, files, feeds, agenda, maps and tourism guides, offers better management and seamless access to event resources [5]. Further, multi-persona proves its utility in other areas where neither carrying multiple mobile devices nor complying with single policy is a choice. While working at their private clinic and at multiple hospitals, doctors are subject to different mobile policies, reflecting each of the different institutions. Carrying multiple devices to accommodate with different systems drains their productivity. Whereas with personal, clinic and hospitals personas, multi-persona allow doctors to comply with the policy of each and effectively treat their patients while maintaining their own unburdened personal use of the device [6]. Whether in these or any other example, the success of multi-persona lies in its capability of consolidating multiple mobile devices on a single terminal, while making the latter able to clearly distinguish between the different contexts in which it is used.

Mobile virtualization is one of the key technologies applied to realize multi-persona. Similar to virtualization

- *H. Tout, C. Talhi and N. Kara are with the Department of Software Engineering and Information Technologies, École de Technologie Superieure, Montreal, CA. E-mail: hanine.tout.1@ens.etsmtl.ca, {chamseddine.talhi, nadjia.kara}@etsmtl.ca.*
- *A. Mourad is with the Department of Computer Science and Mathematics, Lebanese American University, Beirut, LBN. E-mail: azzam.mourad@lau.edu.lb.*

Manuscript received 20 May 2015; revised 30 Nov. 2015; accepted 30 Jan. 2016. Date of publication 26 Feb. 2016; date of current version 5 June 2019. Recommended for acceptance by C.H. Foh, S.N. Srirama, E. Benkhelifa, B. Kantarci, P. Chatzimisios, and J. Wu. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCC.2016.2535223

on servers and desktop machines, mobile virtualization allows to create multiple virtual environments that live alongside on a single terminal, where in this case, the latter is a mobile device and the environments are called personas. Yet, to realize multi-persona, mobile virtualization is much more challenging since it requires a trade-off between secure isolation and scalability of personas on mobile devices with limited resources. Therefore, in the last few years, researchers have proposed lightweight virtualization techniques [4], [7], [8] towards mitigating the virtualization overhead on mobile terminals while keeping a certain level of isolation between the virtual environments. Nevertheless, even with these techniques, the limited CPU power and insufficient memory and battery, threaten personas performance and viability at any time being. Our experiments in Section 3 show drastic increase in the CPU usage, energy consumption and execution time of the running applications. Even worse, because of lack of memory, the personas are forced to shut down under certain circumstances. These severe problems call for the integration of new techniques capable of augmenting personas performance and viability.

A lot of attention has been given recently to mobile cloud computing, which imports new cloud computing services, applications and infrastructures to support mobile devices [9], [10], [11]. In order to address the resource limitations of mobile platforms, many researchers have proposed offloading techniques [12], [13], [14], [15], [16], [17] to migrate computation intensive components to be executed on resourceful infrastructure. While these approaches are proposed to optimize single application, running multiple apps in multi-persona implies resource profiling and offloading evaluation to be done solely for each app, which impose high overhead on the mobile terminal. Different offloading approach [18] has been proposed towards optimizing the execution of applications on multiple mobile devices. Yet the proposed technique is not able to identify components to be offloaded but rather generates only their fraction/percentage.

We propose in this article an offloading-based approach to augment multi-persona performance and viability on resource constrained mobile devices. Our proposition consists first of monitoring the components running in each persona using per persona profiler, and then determining their optimal execution environment based on a generic and adaptable decision model. Taking into account four conflicting objectives of minimizing CPU and memory usages, energy consumption and execution time, we formulate the decision model as multi-objective optimization problem, generic enough to be applied on any components unit (i.e., applications, services, methods and threads) and adaptable to different execution settings. Using heuristics to solve this latter, our approach dictates for each component whether it should be executed locally or offloaded for remote execution.

The main contributions of our approach are threefold:

- Proposing offloading-based architecture to augment multi-persona performance and viability on mobile devices.
- Providing generic and adaptable multi-objective optimization model to formulate multi-persona problems independently of the offloading granularity and adapt offloading evaluation to different execution contexts.
- Generating exact optimal distribution of multi-persona components through heuristics.

The roadmap of the paper is as follows. In Section 2, we present relevant background information and we study existing related works. In Section 3, we highlight the problems caused by running multiple personas on a single mobile device, while in Section 4, we illustrate our proposed solution to address these issues. In Section 5, we present our multi-objective optimization model for offloading evaluation and its complexity analysis whereas in Section 6, we describe the heuristic algorithm to solve it. Later, in Section 7, we provide details about the implementation as well as the experimental results that prove the efficiency of our proposition. Finally, in Section 8, we conclude the paper and draw our future research directions.

2 Background and Related Work

We present in this section some background information about mobile virtualization and offloading and the relevant state of the art review.

2.1 Mobile Virtualization

As smartphones and tablets are growing more and more sophisticated, researchers were able to bring virtualization to such mobile terminals. System-level virtualization [19], [20], [21] is a technique that offers the ability to run multiple virtual environments on one physical device using an additional software layer called a hypervisor (or microkernel) [7]. The virtual environments may run the same or even different operating systems. Even though this technique offers full isolation between the virtual environments, it suffers from significant overhead due to the complete software stack in each instance (i.e., Kernel, Middleware and Apps) [4]. Therefore, when it comes to more than just two personas on the device, this architecture will not be the right choice to go. Per contra, the user-level isolation technique [21] reaches separation by wrapping applications instead of creating virtual environments, which makes it very lightweight when applied on mobile devices. However, by keeping separation just at the applications level, critical, malicious, personal, business and any other type of applications will be running in the same environment. This makes user-level isolation a bandage more than a real solution that can realize multi-persona which requires much higher security [22].

As a trade-off between both above techniques, researchers have proposed recently OS-level virtualization, also called container-based virtualization, which is a technique that shares the kernel layer to run multiple virtual instances on a single operating system [4], [8]. Allocating a minimum set of resources for each instance, make the available OS resources enough for running more than just two personas on top of it. Also using isolation techniques that leverage namesapces at multiple levels of the mobile platform, this technique is able to ensure that the virtual environments are completely independent, secure from each other and any failure that might occur in one of them will not affect the others. These properties are what make such architecture the most adequate for building multi-persona [4], [8]. Wessel et al. present a lightweight isolation mechanism for Android with access control policies, to separate one or

TABLE 1
Taxonomy of Mobile Code Offloading Approaches

Technique \ Criteria	Target	Granularity	Decision Model	Cloud Features	Gain
[12]	One App on Single Device	Application	Does Not Apply	Virtual Phone	Faster Execution
[13]		Method	Energy	Server	Faster Execution and Energy Saving
[14]		Method	Energy and Time	Dynamic Allocation and Management of VMs	
[15]		Service	Energy and Time	Server	
[16]		Service	Time, Energy and Battery Level	Virtual Phone	
[17]		Thread	Energy and Time	Server	
[18]	Multiple Apps from Multiple Devices	Service	Energy and Time	Server	Distribution Fraction, Faster Execution and Energy Saving
Our Approach	Multiple Apps from Multi-Persona	Generic	Energy, Time, CPU and Memory	Server	Exact Distribution, Faster Execution, Energy Saving, Minimized CPU and Memory Usages

more Android userland instances from a trustworthy environment [7]. The proposed architecture provides userspace containers to isolate and control the resources of single application or groups of applications running on top of one kernel. Another approach is Cells [4], which enables multiple virtual phones (VPs) to run simultaneously on the same physical smartphone. It uses device namespaces to multiplex access among VPs to kernel interfaces and hardware resources such that VPs can run side-by-side in virtual OS sandboxes. Lately, Chen et al. have proposed Condroid [8], a lightweight virtualization architecture that allows creating multiple personas by virtualizing identifiers and hardware resources on a shared OS kernel. The proposed architecture leverages namespaces for resource isolation and cgroups feature for resource control. Together, they allow Condroid to run multiple independent and securely isolated virtual instances on the same physical device.

Despite the lightweight approaches, running multiple personas on the same physical mobile terminal remains challengeable. The limited CPU, memory capacity and battery power, all threaten the performance of the running personas and make the device unable to tolerate their survivability. While varying the number of personas, number and type of applications running in each, our experiments in Section 3 show drastic increase in the CPU and memory usages, energy consumption on the device as well as in the execution time of the running applications. They reveal also the inability of the device to run more than three personas. Even though we are able to start a fourth persona, they all shut down once a new application starts executing in this latter.

2.2 Offloading

In turn, mobile cloud computing has brought cloud computing capabilities to support mobile devices, ranging from outsourcing software and platforms all the way to infrastructure [10], [23], [24], [25]. Different offloading concepts exist in this context. The explosion of mobile internet applications, like multimedia newspapers, social networking services, audio and video streaming, is the main reason behind the significant overload on the cellular networks. In this context, traffic offloading [26], [27], [28] has been proposed, which is the use of complementary networks like Wi-Fi for data transmission in order to reduce the data carried on the cellular network. On the other hand, the limited resources of mobile devices have triggered another research domain of computation offloading, which has different concept and objective compared to traffic offloading. In computation offloading, resource-hungry applications, services, methods or threads are offloaded out of the device to be executed on resource-rich and more powerful infrastructure like remote servers. These components are profiled and an offloading decision is taken based on predefined optimization metrics that determine the cost-benefit of offloading. Our proposition is based on computation offloading, therefore in Table 1, we provide a classification of existing relevant techniques to better position our work. Target indicates what the offloading techniques aim to optimize, granularity identifies the type of components where offloading is applied and the decision model defines the metrics for offloading evaluation. Gain shows the benefits of each technique on the mobile terminal and cloud features are the assets used to attain these gains.

From Table 1, approaches aiming to optimize an application execution on single mobile device can be further distinguished based on their granularity:

Application-based offloading. Hung et al. [12] have proposed an approach to execute mobile applications in a cloud-based virtualized environment. The proposed architecture consists of a mobile device connected to virtual phone in the cloud, and an agent program installed on the device whose purpose is to allocate a delegate system on the cloud and communicate the application status. This work presents an application-level migration using the pause/resume concept in android. The application is copied in case it does not exist on the virtual phone. Otherwise, the agent triggers OnPause function of the application and sends its state to the remote agent where it get resumed using OnResume function. Their approach has been able to prove its ability to offload applications out of the mobile device, to be executed on virtual phone in the cloud. However, what is missing in their proposition is the criteria, objective, or circumstances under which a certain application should be considered for offloading.

Method-based offloading. MAUI [13] is an offloading framework that aims to reduce the energy consumption of mobile applications. The framework consists of a proxy server responsible of communicating the method state, a profiler that can monitor the device, program and network conditions, and a solver that can decide whether to run the method locally or remotely. MAUI uses its optimization framework to decide which method to send for remote execution based on the information gathered by the profiler. The results show the ability of MAUI to minimize the energy consumption of a running app. ThinkAir [14] aims to improve both computational performance and power efficiency of mobile devices by bridging smartphones to the cloud. The proposed architecture consists of a cloud infrastructure, an application server that communicates with applications and executes remote methods, a set of profilers to monitor the device, program, and network conditions, and an execution controller that decides about offloading. ThinkAir applies a method-level code offloading. It parallelizes method execution by invoking multiple virtual machines (VMs) to execute in the cloud in a seamless and on-demand manner to achieve greater reduction in execution time and energy consumption. ThinkAir was also able to demonstrate its capability in that regard.

Service-based offloading. Cuckoo [15] is another offloading framework that follows a different strategy for offloading computation-intensive tasks. As precondition, all compute intensive code should be implemented as an Android service. The framework includes sensors to decide, at runtime, whether or not to offload particular service since circumstances like network type and status and invocation parameters of the service call on mobile devices get changed continuously, making offloading sometimes beneficial but not always. Cuckoo framework has been able to reduce the energy consumption and increase the speed of computation intensive applications. Chen et al. [16] have proposed another framework that follows similar strategy to automatically offload heavy back-end services of a regular standalone Android application. Yet, based on a decision model, the services are offloaded to an Android virtual machine in the cloud. Their proposition has not been implemented and evaluated yet.

Thread-based offloading. CloneCloud [17] is a system that aims to minimize both execution time and energy consumption of a running application. It consists of a profiler, which collects the data about the threads running in this app and communicates the gathered data with an optimization solver. Based on cost metrics of execution time and energy, the solver decides about the best partitioning of these threads between local and remote execution. This approach does not require modification in the original application since it works at the binary level. The experiments of CloneCloud showed promising results in terms of minimizing both execution time and energy consumption of an application. However, only one thread at a time can be encapsulated in a VM and migrated for remote execution, which diminishes the concurrency of executing the components of an application.

On the other hand, Mazza et al. [18] proposes an approach that aims to optimize the execution of applications in a system that involves multiple mobile devices. In their work, a partial offloading technique has been proposed for heterogeneous networks infrastructure (HetNets). Depending on the number of devices connected in this network and constrained by both the energy consumption and execution time, the proposed approach is able to generate the percentage of tasks to be offloaded, aiming to optimize the entire system rather than just a single device.

2.3 Proposed Approach Positioning

Computation offloading requires device status, network conditions and applications to be monitored, in order to study the effectiveness of offloading when a decision should be made. The gathered information formulate the input of a solver that evaluates the decision model metrics to decide whether a component is to be offloaded or executed locally. Existing approaches [12], [13], [14], [15], [16], [17] are proposed for single application where profiling and offloading evaluation are done solely for each app. Therefore, with multiple applications on the mobile terminal, a profiler and a solver should be dedicated for each, which cause significant overhead in multi-persona where many independent components from different applications are running on the device. Multi-persona necessitates higher view of resource consumption and global formulation of the decision making metrics, therefore our work provides per persona profiler, and eventually global formulation of the optimization model. The proposed model is generic enough to be applied with any offloading component unit (i.e., at any granularity level) and adaptable to different execution settings like lack of memory and low battery. The proposed approach is capable of minimizing CPU and memory usages that affect the performance as well besides the energy and execution time of the applications.

On the other hand, different work [18] exists involving multiple mobile devices, where the aim is to optimize the entire system rather than a single terminal. The proposed approach generates the percentage/fraction of components to be offloaded from the system, yet identifying what components to be offloaded is needed in order to execute the offloading process. Therefore, more effectively, our approach can generate the exact distribution of all the components running in each persona at any time being with the intent of augmenting personas performance and viability. Seeing that such valuable decision can be costly, we discuss later in the paper some alternatives that can decrease its overhead.

3 Problem Illustration

No matter how sophisticated mobile devices are growing, they still have limited hardware in terms of computing power, memory capacity and battery lifetime. Running multiple personas on a single mobile device is yet impeded by these limitations, rendering personas performance and viability on the line [29]. To shed the light on these issues that we address in this paper, we vary the number of personas running diversity of lightweight, moderate and heavy applications, and we compare the resource consumption

TABLE 2
Applications in Each Persona

Weight	Application
Lightweight	Zip: This application creates archive folder from original files. To make it lightweight, we use files having total size of 3 MB. Unzip: The unzip application extracts the content of the archived folder created using the Zip app.
Moderate	Virus Scanning: This application scans the contents of some files on the phone against a library of 1000 virus signatures, one file at a time. To implement moderate application, we fix the size of the files to 100 KB.
Computation Intensive	NQueens Puzzle: This puzzle implements the algorithm to find all possible solutions of the typical NQueens problem, and return the number of solutions found. We consider N=13 to create computationally intensive problem.

that affect personas lifetime as well as the execution time that influence their performance. To model different usage scenarios of the device, we consider in each persona four applications of different weights as described in Table 2.

The results depicted in Fig. 1 show drastic increase in the CPU usage that was originally 49 percent with 1 Persona (1P) running the four apps, but reached 62 percent with 2 Personas (2Ps). Also the energy consumption, which is consumed by the applications usage on CPU as well as the one spent on the screen, has significantly increased from 330 J to 420 J. As for the execution time, which denotes the time taken till the end of execution of the last app in each persona, it took 780 s in one persona, yet up to 1,380 s with two personas. This long execution time is due to the NQueens puzzle, which we use to overload the device. Another interesting observation is in the third scenario (3Ps) where it was impossible to run the same apps in three personas, as the personas kept shutting down due to lack of memory on the mobile terminal as well as the high consumption of other resources. These results reveal the inability of the mobile device resources to afford high performing personas neither to tolerate their viability. In the light of these serious problems, it is indispensable to integrate new techniques capable of minimizing the resource consumption and execution time of different type of applications running in each persona. For details about the implementation and tools used, please refer to Section 7.1 which is devoted for that end.

4 Offloading Meets Multi-Persona

In Section 2, we distinguished our proposition from existing offloading approaches. In what follows we go deeper to

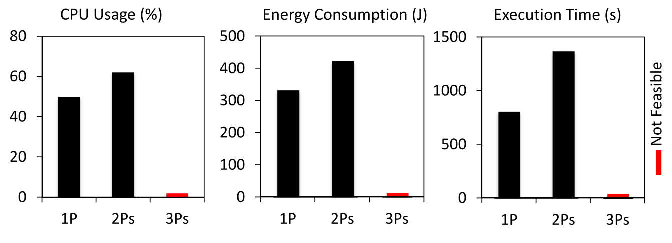


Fig. 1. Multi-persona efficiency and viability.

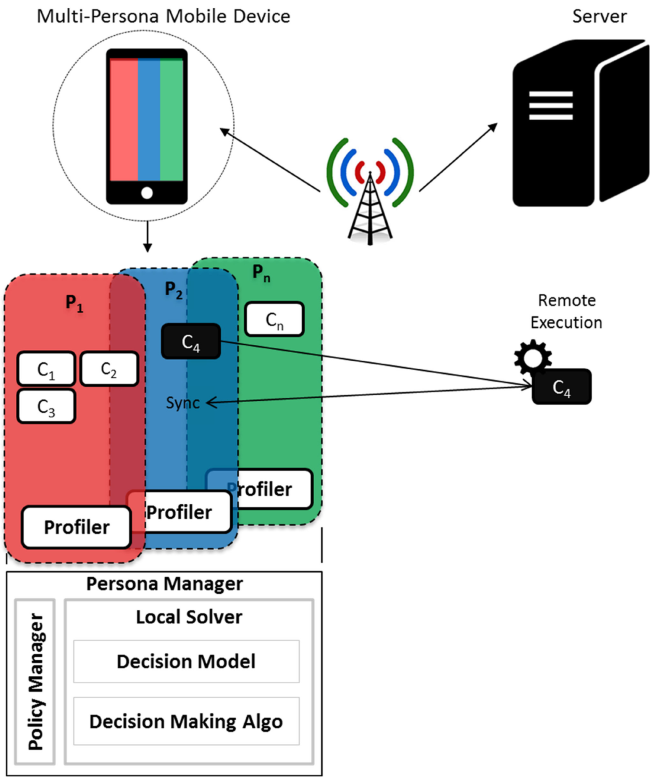


Fig. 2. Proposed architecture.

explain it in details and highlight its contributions. The architecture of our approach is depicted in Fig. 2 that essentially focuses on the mobile device structure, since the main dilemma lies there. We build our approach on top of OS-level virtualization, which is as we discussed in Section 2, the architecture that best fit for multi-persona solution. In each persona ($P_1 ... P_n$) we add a profiler to monitor the latter resources. Differently from the literature, this profiler is not dedicated to one application at a time neither to components belonging to one application, but rather it is devoted for the entire persona. Each profiler gathers information about CPU and memory usages, energy consumption, execution time and other relevant data for all the components running in the persona. These components ($C_1 ... C_n$) can be applications, services, methods or even threads. The profiler examines also the connectivity availability, bandwidth and latency in the persona where it runs.

Next comes the role of the solver which uses the gathered information to construct a global decision model involving all the components running in each persona, rather than just one application on the device like in existing approaches. The decision model is based on four metrics that affect personas performance and viability, which are minimizing CPU usage, memory and energy consumptions and execution time. With these four conflicting metrics, we formulate the decision model as multi-objective optimization problem. It is worth to mention that the latter is generic enough to be independent of the offloading component unit (i.e., application, services, methods, threads). Further, this model can be automatically adapted to different execution settings. Particularly, with low CPU, memory, battery level or the need for high performing applications, the model can be adapted giving more priority for the relevant metric(s), which is/are in critical situation. We believe that this will be a valuable track in future work. Other type of adaptations may also apply, like restricting the execution of particular component(s) to the mobile device (or to remote server) for security reasons. Furthermore, the model can exclude persona(s) and/or component(s) to try whether shutting them down can be more efficient for the personas performance, and hence notify the user accordingly (e.g., component is running but has not been used for a while). All these adaptation settings can be enforced by the policy manager.

Finally, after formulating the problem, a decision algorithm and part of the solver module, is responsible of generating the exact distribution of the components running in each persona on the device. The decision dictates for each component whether it should be executed on the device or offloaded for remote execution. This is also another added value to the existing approaches, which generate only the fraction or percentage of local and remote tasks. For the solver algorithm, we use heuristics and more specifically genetic algorithms (GAs), which are able to find the optimal distribution that complies with multi-persona problem's objectives. To decrease the overhead of the decision making process, which is needed for granular offloading, one solution provided in our approach is to generate good solution rather than an optimal one. This can offer a trade-off between components overhead and decision overhead. Whenever connectivity is not available, our approach do not call the solver but takes directly a decision to run the components locally in order to reduce the overhead caused by the solver. Also whenever certain scenario is repeated, yet with different resources availability or constraints, a delta value is to be computed in order to reduce the solver overhead. We also offer the ability to even offload the decision making process by implementing a counterpart of the solver on the server side, since based on the decision model complexity, the decision making process can be time consuming and might require additional resources.

5 Multi-Objective Optimization for Multi-Persona

In this section, we present a formal definition of the multi-persona problem, explain the computational analysis to prove its complexity and finally show its formulation as multi-objective optimization problem.

5.1 Problem Definition

Assumptions:

- Some components might not be offloadable
- Components are independent
- Network might not be stable in terms of availability, bandwidth and latency

We consider a mobile device of multiple personas $P = \{P_1, \dots P_m\}$ where each of them is running a set of components $C = \{C_1, \dots C_n\}$, which can be applications, services, methods or even threads that implement the applications functionalities. Each component has demands in terms of energy consumption, execution time, memory and CPU usages. Due to limited resources on the mobile device in terms of CPU, memory and battery, part/all of the components in the running personas should be offloaded for remote execution. Finding the best distribution of these components is a complex and challenging problem. The multi-persona problem can be formulated as follows:

Problem Definition 1. *Given a set of components running in each persona P_i, where each of these components C_j has energy consumption $E^l_{c_j,p_i}$, execution time $t^l_{c_j,p_i}$, memory usage $M^l_{c_j,p_i}$ and cpu usage $CPU^l_{c_j,p_i}$ for local execution and $E^r_{c_j,p_i}$, $t^r_{c_j,p_i}$, $M^r_{c_j,p_i}$, $CPU^r_{c_j,p_i}$ for remote execution, α_{c_j,p_i} an indicator whether they are offloadable or not, network bandwidth B and latency L, find the best distribution of components between local and remote execution in a way to minimize their energy consumption, execution time, memory and CPU usages. Minimizing the energy consumption, execution time and memory and CPU usages form the fundamental objectives that can augment performance and ensure viability of the personas running on the mobile device. Yet, this is a complex and challenging problem for the following reasons.*

- *First, minimizing energy consumption, execution time, memory and CPU usages are conflicting objectives, therefore finding the best tradeoff among them is not a simple task.*
- *Second, computing local and remote partitions of these components suffers from an exponential search space in the number of different possibilities in which these components can be distributed, which renders the problem heavy. This is similar to the various ways n distinct objects (components) can be distributed into m different bins with k_1 objects in the first bin, k_2 in the second, etc. and $k_1{+}k_2 + \dots k_m = n$. This indeed is obtained by applying the multinomial theorem where $\sum_{k_1+k_2+\dots k_m} = n\binom{n}{k_1,k_2,\dots k_m}$. In our case $m = 2$ bins, one is the mobile device and the second is the remote server thus, for n components, there are 2^n different distribution possibilities.*

To further emphasize the complexity of the problem, we consider the case of three personas with only four components in each. To compute the portion of the components in each persona that should be offloadded and the other that will run locally on the mobile device, there are $2^{12}(4{,}096)$ different mapping possibilities. When this number might not appear to be that big in case the components are applications or services, it will dramatically increase when the components are methods or threads. In such case, the number of components will reach hundreds or even thousands and it would be there 2^{100} or $2^{1{,}000}$ possible distributions! which makes it hard to find their exact distribution.

Theorem 1. *Multi-Persona Multi-Objective Optimization problem is NP-Hard.*

Proof. We reduce the multi-objective-m-dimensional Knapsack Problem [30] to our multi-persona problem (MPP). The idea is that if a case of the multi-persona multi-objective optimization problem can be solved, then it can be used to solve the multi-objective-m-dimensional Knapsack

TABLE 3
Formulas Notations

Variable	Description
m	Number of personas
p	Persona
n	Number of components in a persona
c	Component
P_{cpu}	Power consumed by the cpu when the component is executed locally
P_{sc}	Power consumed by the screen when the component is executed locally
$P_{cpu,idle}$	Power consumed by the cpu when it is idle waiting for remote results
P_{na}	Power consumed by the network when it is active
P_{tr}	Power consumed by the device during transmission
$t^l_{c,p}$	Execution time of the component c running locally
$t^r_{c,p}$	Execution time of the component c running remotely
$D_{c,p}$	Size of data exchanged between the device and the cloud for offloading c
$M^l_{c,p}$	Memory consumed by the device when c is executed locally
$M^r_{c,p}$	Memory consumed by the device when c is executed remotely
$C^l_{c,p}$	CPU usage on the device when c is executed locally
$C^r_{c,p}$	CPU usage on the device when c is executed remotely
L	Latency
B	Bandwidth
$x_{c,p}$	Decision variable that indicates whether c should be offloaded or not
$\alpha_{c,p}$	Indicates whether c is offloadable or not
a_p	Indicates whether persona p should be included in the decision process or not
$b_{c,p}$	Indicates whether c should be included in the decision process or not

Problem (MOMKP). Given the MOMKP - a collection of n items $a_1, \ldots, a_n$, where each item a_i has m weights $w_{ki} \in \mathbb{N}, k = 1, \ldots, m$ and t values $p_{ki} \in \mathbb{N}, k = 1, \ldots, t$ and a knapsack of m capacities $c_k \in \mathbb{N}$, $k = 1, \ldots, m$ - we construct the Multi-Persona problem as follows: Setup m personas $P = \{p_1, \ldots, p_m\}$ with n components in each, forming a set of $n * m$ denoted as x components $C = \{C_1, \ldots, C_x\}$, one corresponding to each item in MOMKP.

- For components C_i, set the resource demands in terms of memory and CPU of each component as the weights of the items in the sack. $M^l_{c_i}$, $M^r_{c_i}$, $CPU^l_{c_i}$, and $CPU^r_{c_i}$, are the memory and CPU usages when C_i is running locally and when executed remotely, respectively.
- For components C_i set $a_{c_i}.f_1$, $a_{c_i}.f_2$, $a_{c_i}.f_3$ and $a_{c_i}.f_4$ as the values of each item, where they form the cost of each component in terms of energy consumption, execution time memory usage and CPU usage respectively. So that $\sum_{i=1}^{x} a_{c_i}.f_j$ where $j = 1, \ldots, 4$ constitute each of our objective functions correspondingly.

With this reduction, the content of the knapsack is a portion of components, which is selected to run on the mobile device such that the total of each value (cost) is minimized (rather than maximized as in knapsack, but they are essentially the same), and vice versa, and a solution to our problem yields a solution to the MOMKP. Thus an algorithm for solving the multi-persona problem can be used to solve the multi-objective m-dimensional Knapsack problem. Hence it follows that our problem is NP-Hard. □

5.2 Problem Formulation

Table 3 describes the notations used in the problem formulation.

1) *Minimize energy consumption:* The total energy consumption is equal to the one consumed on local components plus the one for offloaded components. When running components locally, they consume energy on the CPU processing and screen brightness. As to execute components remotely, the energy is spent on the CPU being idle, screen brightness and network active while waiting for the remote execution. In addition it consists also of the energy consumed by the device for data transmission (i.e., upload and download)

$$F_1 = min\Big(\sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p}(1 - x_{c,p})(((P_{cpu} + P_{sc}) \times t^l_{c,p})) + \sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p} x_{c,p}((((P_{cpu,idle} + P_{sc} + P_{n,a}) \times t^r_{c,p}) + (P_{tr} \times (L + \frac{D_{c,p}}{B}))) \times \alpha_{c,p})\Big). \quad (1)$$

2) *Minimize execution time:* The overall execution time is equal to the time taken by the components running locally and those running remotely

$$F_2 = min\Big(\sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p}(1 - x_{c,p})((t^l_{c,p})) + \sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p} x_{c,p}\Big(\Big(t^r_{c,p} + L + \frac{D_{c,p}}{B}\Big) \times \alpha_{c,p}\Big)\Big). \quad (2)$$

3) *Minimize memory consumption:* The total memory consumption in the running personas, is equal to the memory consumed by the components running locally plus the one consumed while waiting the remote execution

$$F_3 = min\Big(\sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p}(1 - x_{c,p}) \times (M^l_{c,p}) + \sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p} x_{c,p}(M^r_{c,p} \times \alpha_{c,p})\Big). \quad (3)$$

4) *Minimize CPU usage:* The total CPU usage in the running personas, is equal to the CPU usage of the components running locally plus the one used while waiting for remote execution

$$F_4 = min\Big(\sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p}(1 - x_{c,p}) \times (C^l_{c,p}) + \sum_{p=1}^{m} a_p \sum_{c=1}^{n} b_{c,p} x_{c,p}(C^r_{c,p} \times \alpha_{c,p})\Big). \quad (4)$$

So our multi-objective optimization problem is:

$$F = min\{F_1, F_2, F_3, F_4\}$$

S.t

$$
\begin{array}{ll}
n \in \mathbb{N} & \text{(c1)} \\
m \in \mathbb{N} & \text{(c2)} \\
x_{c,p} = \{0, 1\} & \text{(c3)} \\
a_p, b_c, \alpha_{c,p} = \{0, 1\} & \text{(c4)} \\
0 \leq M^l_{c,p} \leq 1 & \text{(c5)} \\
0 \leq M^r_{c,p} \leq 1 & \text{(c6)} \\
0 \leq C^l_{c,p} \leq 1 & \text{(c7)} \\
0 \leq C^r_{c,p} \leq 1 & \text{(c8)} \\
F3 \leq t_m\% & \text{(c9)} \\
F4 \leq t_c\% & \text{(c10)}
\end{array}
$$

Constraints c1 and c2 ensure that the number of personas and their components belong to the set of natural numbers. Constraints c3 and c4 define the binary variables. Constraints c5-c8 ensure that the memory consumption and CPU usage on the mobile device vary between 0 and 1 since they are represented as percentages in our model. Finally, constraints c9 and c10 ensure that the amount of the memory and CPU usages do not exceed certain thresholds based on the capacity of the mobile device. Solving this model will generate the best distribution of the components running in each persona that complies with the formulated objectives aiming to augment personas performance and viability. This distribution is represented by $x_{c,p}$, which represents whether a component c should be offloaded or not. If $x_{c,p} = 0$, then c should run on the mobile device, while it should be offloaded otherwise i.e., for $x_{c,p} = 1$.

6 Heuristic Algorithms for Optimal Distribution of Multi-Persona Components

Genetic algorithms [31] are heuristic methods that mimic the natural evolution process to solve a problem. Using the concepts of natural selection, GAs simulate the propagation of the fittest individuals over consecutive generations to determine the best solution. Particularly, GAs start by initializing random set of solutions represented by chromosomes/individuals, forming together what is called a population. According to their fitness, solutions from one population are selected to form new candidate solutions called offspring. The fitness of a solution is determined based on a function that aims to minimize or maximize particular objective. The more suitable the solutions are, the more chances they can have to reproduce. To generate offspring, GAs apply crossover and mutation operators to evolve the solutions trying to find better ones. After evaluation, the process is then terminated if stopping criteria is met. Over time, this process will result in increasingly favourable individuals for solving the problem. As such, GAs have been able to prove, through their method of evolution-inspired search, their capability to solve complex optimization problems in many areas [32], [33], [34]. In this paper, we exploit the intelligent evolution of solutions in GAs to solve the multi-objective optimization problem of multi-persona. In what follows we show how the main elements and operators of GAs are mapped to solve the problem.

6.1 Representation of Individuals

Each individual is a candidate solution represented as a set of bits having a length of L. Each bit represents a component running in particular persona, and hence the size of an individual is determined based on the number of components. A bit has two possible values 0 and 1. For instance having three components, a randomly generated individual can be represented by 000, 001, 011, 111, 110, 100, 101 or 111. For any component, a bit of 0 is for local execution while a bit of 1 is for remote execution. With this representation, we are able to decode the distribution (local/remote execution) of components running in each persona.

6.2 Fitness Evaluation

Genetic algorithms require a fitness function that assigns a score (fitness) to each individual in the current population. The fitness depends on how efficiently that individual can solve the problem at hand. In our multi-persona problem, the fitness of a solution is calculated by evaluating the four objective functions F_1, F_2, F_3, and F_4 that we defined in the previous section. The solutions are ranked based on their ability to minimize these functions.

6.3 Operators

6.3.1 Selection

This operator selects individuals in the population for reproduction. It selects random individuals and picks the x best of them able to minimize mostly the objective functions.

6.3.2 Crossover

This operator applies modification on individuals selected based on particular rate μ_c to generate offspring. It randomly chooses a bit and exchanges the subsequence before and after that bit between two individuals to create two offspring. For example, the individuals 110 and 101 could be crossed over after the second bit in each to produce the two offspring 111 and 100.

6.3.3 Mutation

This operator randomly flips some of the bits in individuals selected based on mutation rate μ_m. For example, an individual 010 might be mutated in its second position to yield 000.

6.4 Algorithm and Time Complexity Analysis

Based on these definitions, the algorithm of the solver works as described in Algorithm 1. It starts by creating a population of N randomly generated individuals for the running personas. Each individual is a point in the search space that represents a possible distribution solution. The fitness of an individual is calculated by the computation of the four objective functions F_1, F_2, F_3 and F_4. The fittest individuals in the population are then selected to go through a process of evolution based on crossover and mutation rates μ_c and μ_m respectively. In this process, crossover and mutation operators are applied on the selected individuals to create next generation of individuals for new possible distribution solutions of components. The fitness of these generated individuals is also calculated. This process continues over and over until a stopping criterion is met. This criterion can be number of iterations, time or other relevant condition.

Finally, the solver returns the fittest distribution(s) of components to solve the multi-persona problem.

The time complexity of this algorithm depends on many factors, the fitness function evaluation, the population size, the individual length, variation and selection operators and the number of iterations or generations. Initializing and generating the population (Lines 3 and 4 respectively) have time complexity $\mathcal{O}(1)$. The evaluation of the fitness function (Line 5 till Line 7) has time complexity of $\mathcal{O}(N)$ where N is the population size. The tournament selection, single point crossover and bit flip mutation (Line 8 till Line 18), have time complexity of $\mathcal{O}(INL)$ where I is the number of iterations (i.e., generations) and L is the length (i.e., number of bits) of an individual. Finally, the return statement (Line 19) has $\mathcal{O}(1)$. Subsequently, the time complexity of the algorithm is $\mathcal{O}(1) + \mathcal{O}(N) + \mathcal{O}(INL) + \mathcal{O}(1)$ which is equivalent to $\mathcal{O}(INL)$.

Algorithm 1. MultiPersonaSolver(N, L, μ_m, μ_c)

```
1: Input: $N$ := Population size, $L$ := Individual length, $\mu_m$ := mutation rate and $\mu_c$ := crossover rate
2: Output: $S$ := Set of fittest individual(s)
3: Initialize populations index $k := 0$
4: Generate random population $P_k := GenerateRandomPop(N, L)$
5: for each individual $i \in P_k$ do
6:     Evaluate objective functions $F_1(i), F_2(i), F_3(i), F_4(i)$
7: end for
8: do
9:     {
10:    Select $x$ best distribution possibilities and insert them into $P_{k+1}$
11:    Crossover $\mu_c \times n$ individuals to produce new offspring distributions and insert offspring into $P_{k+1}$
12:    Mutate $\mu_m \times n$ individuals by inverting a randomly-selected bit in each to generate new possible distribution solutions
13:    for each $i \in P_{k+1}$ do
14:        Evaluate objective functions $F_1(i), F_2(i), F_3(i), F_4(i)$
15:    end for
16:    Increment $k := k + 1$
17:    }
18: while stopping criterion is not met
19: return $S$ the fittest distribution(s) from $P_k$
```

7 Implementation and Experiments

We dedicate this section to describe the implemented components and discuss our experiments finding.

7.1 Implementation

To create personas, we use Cells [4], since by the time this work is done Cells was the first and only open-source virtualization architecture that enables multiple virtual smartphones and tablets to run simultaneously on the same physical device [35]. We set up the environments on Asus Nexus 7 tablet as Cells open source project has been ported for this device only. The tablet runs Android operating system, has quad-core processor and 1 GB of RAM.

On the other hand, and as we discussed throughout the paper, the offloading unit can be an application, service, method or thread. Offloading the entire image of a running application, involves the encapsulation of the latter in a VM instance, which imposes high overhead for creating, cloning, migrating and configuring the VM on the remote server [36], [37]. More recent offloading approaches are based on service-level offloading, which distinctly do not have such high overhead and can even reduce the overhead caused by finer granularity components [37]. In addition, Android platform architecture supports and encourages the implementation of applications using activity/service model in android. In this model, the logic code of the computation-intensive tasks is implemented as services through an interface defined using interface definition language (AIDL) [38] and the user interface as activities. In case the applications do not meet with this requirement, an interface can be easily extracted from the original code as stated by kemp [15]. Following these facts, a small statistic that we did, in which we downloaded 70 applications from Google play store from different categories (e.g., games, social media, video conferencing, notebook, EMR), showed that 51 percent of them contain services varying from 1 to 20 services per app. Therefore, for the sake of the implementation in this paper, we decided to take the services to be the offloadable components using specific libraries [15].

For the profiler, we implemented one that exploits Linux-based commands to monitor CPU and memory usages. To get the power consumed on idle CPU and screen, active Wi-Fi and during transmission, we use the power profile of android [39], whereas, PowerTutor tool [40] is used to get the power consumed on the CPU, screen and network, during the execution of local components. As for the execution time, we implemented and embedded a timer to monitor the execution of the relevant components. The connection between the mobile terminal and the server is done through Wi-Fi network in infrastructure mode and not in an ad-hoc fashion. So the communication is done indirectly through an access point and not in a peer to peer mode and it is characterized by the IEEE 802.11n standard. Enhancing the implementation of profilers will be part of future work. Finally in the solver, we implemented the decision model based on the metrics that we described in Section 5. For the decision maker, we implemented different genetic algorithms in order to compare them and check which one is more adequate in terms of execution time and resource consumption. The first algorithm is NSGA-II [41], a multi-objective genetic algorithm which uses pareto ranking mechanism for classification of solutions and crowding distance to define proximity between them. SPEA2 [42], is another multi-objective evolutionary algorithm based on pareto dominance, yet characterized by its strength scheme that not only takes into account the number of solutions that dominate particular solution, but also the number of solutions by which it is dominated. The third algorithm is SMSEMOA [43], which is a steady state algorithm, in which a random selection of individuals is done for the mating process and the offspring replaces the individuals of the parent population. Next, IBEA [44] is an algorithm that employs a quality indicator in the selection process. Finally, MOCell algorithm [45] which is characterized by both decentralized population and archive to store non-dominated solutions. We implemented these algorithms as introduced by their authors, yet based on the mapping that we

TABLE 4
Distribution of Services in Different Scenarios

Weight\Scenario	S1	S2	S3	S4	S5	S6	S7	S8	S9
Lightweight	3	0	0	1	3	4	6	8	16
Moderate	0	3	0	1	2	3	3	4	0
Heavy	0	0	3	1	1	2	3	3	2
Total	3	3	3	3	6	9	12	15	18

described in Section 6. As for the formulated problem, it can be adapted at any time by modifying the input parameters.

7.2 Experiments

The first experiment aims to compare the algorithms described above in order to determine the most efficient one to solve the proposed multi-objective optimization model of multi-persona. While adopting the most performing algorithm, we study in the second experiment the efficiency of our approach compared to two different strategies. In the first strategy, all services in all personas are executed locally on the mobile device, while in the second one, the execution of these services is always offloaded to remote server.

7.2.1 Testbed Setup

Based on our results in Section 3, running three personas on the mobile device terminal was the most problematic scenario. This makes it the best environment to perform our experiments for both comparing the overhead of the algorithms as well as studying the efficiency of our approach. For fair comparison, we use the same applications that we presented in Section 3. Yet as shown in Table 4, we vary the number of applications and their distribution in the running personas to reflect different possible usage scenarios of the multi-persona mobile device. To demonstrate the efficiency of our proposition, Table 4 includes also S7, which is the scenario that the mobile device was not capable to run, as presented in Section 3. S8 is also another scenario that cannot run on the device.

In each scenario, we profiled beforehand all the applications in the running personas as well as the network characteristics in order to generate the input parameters that we presented in Table 3. The generated data set forms the input for the decision making algorithm in order to solve the formulated multi-persona problem according to each scenario. The connection is characterized by the IEEE 802.11n wireless networking standard with data rate varying between 54 and 600 Mbps and an average latency of 16 s. The server side is running Ubuntu 12.04 with 7.3 GB of memory and quad core AMD Phenom(tm) II X4 B95 processor.

TABLE 5
Parameters

Parameter	Value
$PopulationSize$	100 individuals
$ArchiveSize$	100 individuals
$Selection$	Binary Tournament
$CrossoverProbability$	0.9
$MutationProbability$	1/n (n=number of decision variables)
$CrossoverOperator$	Single Point
$MutationOperator$	Bit Flip
$CrossoverDistributionIndex$	20
$MutationDistributionIndex$	20
$Offset$	10
$Feedback$	20 individuals

TABLE 6
Number of Iterations

Algo\Scenario	S1	S2	S3	S4	S5	S6	S7	S8	S9
NSGA-ii	8	8	8	10	30	60	100	105	135
SPEA2	13	13	13	15	50	100	100	110	135
SMSEMOA	8	8	8	10	50	90	170	170	175
IBEA	14	14	14	15	60	110	-[1]	120	165
MOCell	13	13	13	15	40	60	100	110	135

[1] *optimal solution is not found, and therefore IBEA is excluded from being compared with the rest of the algorithms in scenario S7.*

Concerning the algorithms configuration, we used the following values presented in Table 5. For NSGA-ii, the population size is 100 individuals and the selection is based on binary tournament. The operators for crossover and mutation are single point crossover and bit flip mutation with distribution indexes of $\eta_c = 20$ and $\eta_m = 20$ respectively. A crossover probability of $p_c = 0.9$, and a mutation probability of $p_m = 1/n$, where n is the number of decision variables. For SPEA2, both the population and the archive sizes are 100 individuals, and all selection, crossover and mutation operators are the same used in NSGA-ii, with the same values of probabilities and distribution indexes. Also SMSEMOA has the same parameters settings as NSGA-ii, with an offset of 10. Same applies on IBEA and MOCell with feedback of 20 individuals for the latter.

The last parameter is the stopping criterion. In these experiments we set it to be the number of iterations needed to find the optimal solution. We set the value of this criterion for each algorithm as illustrated in Table 6. The values of this criterion are selected based on multiple executions of these algorithms while trying to find the optimal distribution of services in each scenario. Yet as discussed in Section 6, this criterion can be time threshold or any other relevant parameter.

7.2.2 Assumptions

- The four objective functions have the same priority level; hence an optimal solution is defined as the best trade-off among them: We experimented other models where we prioritized certain objective functions yet they added no value neither to the algorithms comparison nor to the approach efficiency experiments, therefore we do not present them here.
- Connectivity is always available: Whenever connectivity is not available, our approach does not have to run the solver but rather takes directly the decision of running all the services locally on the device. Therefore, we assume in these experiments that the network is always available, in order to be able to compare the efficiency of the proposed approach whenever offloading the execution of the services is a possible choice.

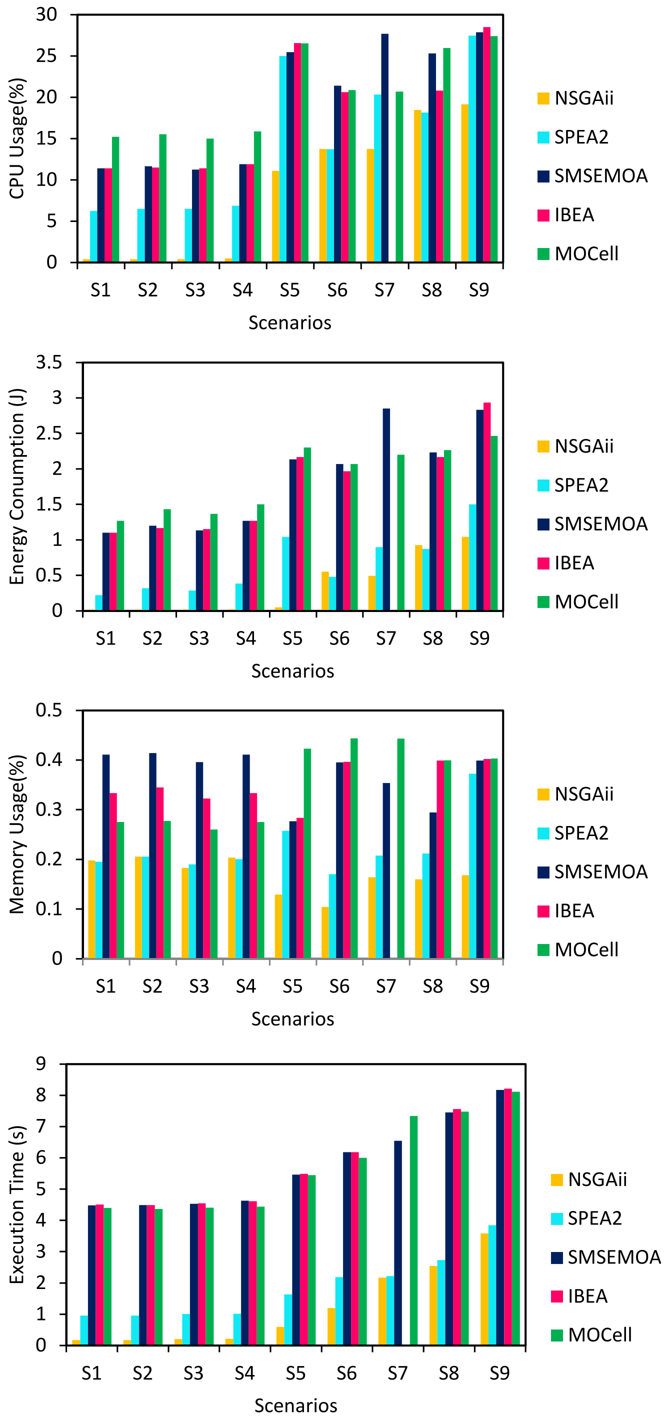


Fig. 3. Algorithms overhead on the mobile device.

7.2.3 Results and Analysis

In what follows, we present the evaluation of the algorithms to select the best performing one, and the efficiency of our approach compared to other strategies.

7.2.4 Algorithms Overhead

Fig. 3 shows the overhead of each algorithm in terms of CPU usage, energy consumption, memory usage and execution time.

The results show that in most of the scenarios, NSGA-ii algorithm consumed the least CPU compared to the other algorithms. Yet even in scenarios S6 and S8, where SPEA2 algorithm was the best, NSGA-ii had very comparable results. In terms of energy consumption, the results are proportional to those of CPU usage, which can be expected since the energy consumption for the algorithms is the one consumed on their usage of the CPU. Thus, the same analysis applies on the energy consumption results. In terms of memory, NSAG-ii has proved again its efficiency over the other algorithms. The results show that it consumed the least memory in all the scenarios except S6, where yet it had comparable result to SPEA2, which was the best in this case. Finally, in terms of execution time, NSGA-ii was the fastest to find the optimal solution in all the scenarios. Based on these results, we opted to use NSGA-ii as the decision making algorithm for the solver component of our model.

TABLE 7
Distribution of Services Based on the Decision Making Algorithm

Scenario	Decision
S1	000
S2	111
S3	111
S4	101
S5	110111
S6	100110011
S7	100110011001
S8	100110011001100
S9	000000000000000011

7.2.5 Approach Efficiency

Table 7 shows the optimal distribution of the services running in each scenario based on our solver module that implements NSAG-ii algorithm. The algorithm returns more than just one possible solution in each scenario. Yet, since all solutions are non-dominated, they are considered equally good. Therefore, we can randomly select any of them to be applied. In Table 7, we show one of these solutions in each scenario. To recall what we explained in Section 7.1, the distribution is represented in a binary set. Each bit corresponds to particular service running in particular persona. A bit having a value of zero means that the decision algorithm recommends to run this service locally, while a bit of one, means that it would be more efficient to offload the service out of the mobile device. For instance in Scenario S4, the virus scanning service was running in the first persona, the unzip service in the second persona, and the NQueens service in the third one. Based on the solver, the optimal distribution that has the best trade-off between the four objective functions in this scenario, is to run the unzip service locally while to offload the two others to be executed on a remote server.

Based on the decision of the solver in each scenario, we study in Fig. 4 the overhead of our proposition compared to two other different approaches. Current usage of mobile devices involves running all the services in all personas locally on the mobile terminal. Hence, we opted to compare our proposition to such strategy that we call NDAL. In addition, since we are proposing an offloading-based approach to address personas problems, it is indispensable to study

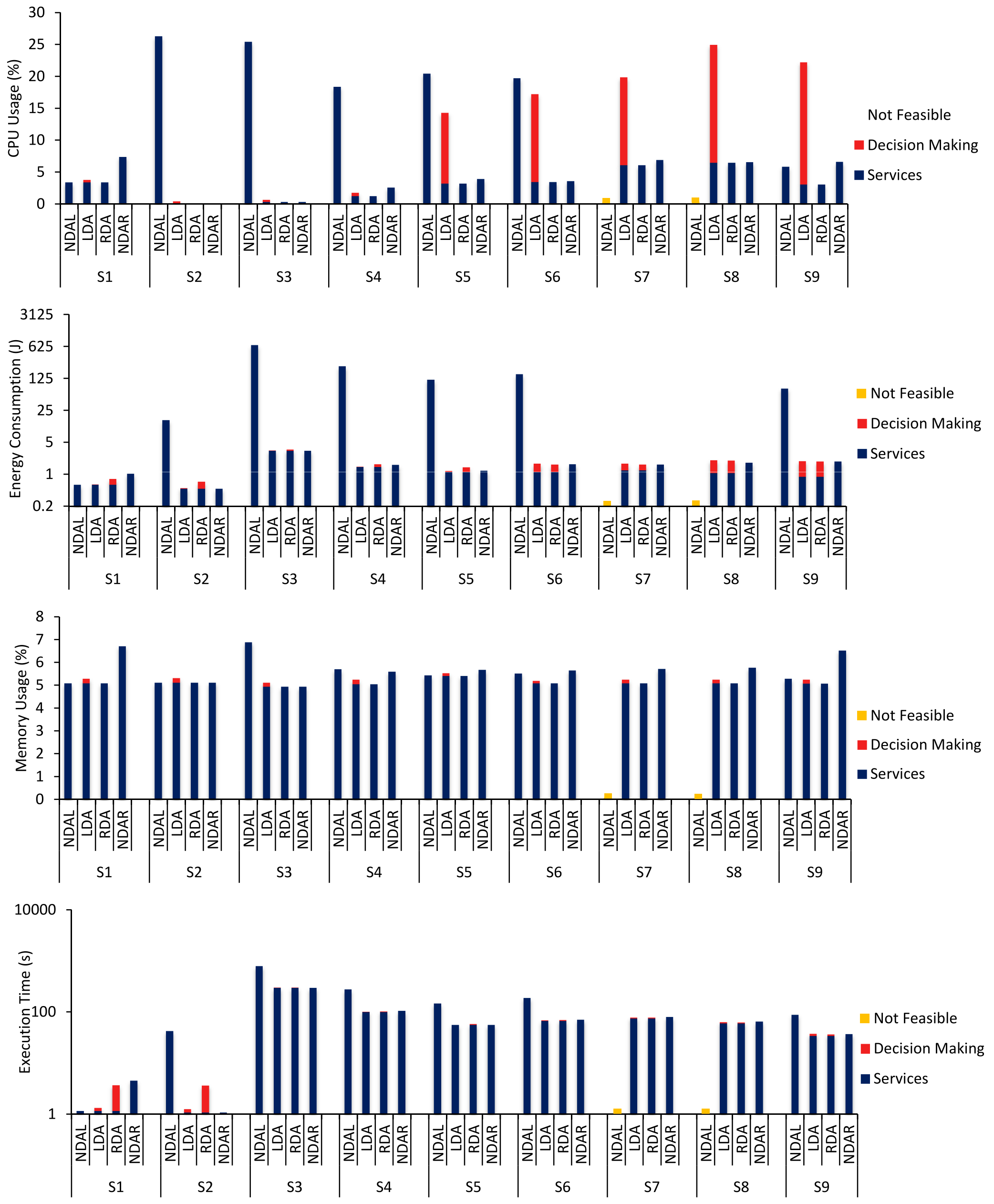


Fig. 4. Approach evaluation. LDA and RDA represent our approach. In LDA, the decision making process is done on the mobile terminal, while in RDA, it is conducted on the server side. NDAL is the case when all components in all personas are running locally on the device, and NDAR is when these components are always offloaded.

whether always running services out of the mobile device can be more efficient. Therefore we compare our proposition to NDAR, where all services are always offloaded to be executed on the remote server. The aim of these experiments is to demonstrate that neither running all the services locally on the mobile terminal nor always offloading their execution to remote server, can offer an efficient multi-persona solution, but rather other appropriate distribution can do. The results of our approach are depicted by LDA and RDA in Fig. 4. LDA is the case when the solver algorithm is running on the mobile device, while in RDA, it is running on the remote server. Both NDAL and NDAR do not include any decision making process, but rather statically consider local device and remote server, respectively, for the execution of the services. Therefore, the only overhead caused by these approaches is the one of the services, whereas in our approach, we add also the overhead of the solver in the results for reasonable comparison (red bars).

The results in Fig. 4 show how our approach (LDA and RDA) can remarkably minimize CPU and memory usages, energy consumption and execution time of the services (comparing the blue bars). Particularly, our finding show that, in scenarios S4 throughout S9, our approach was able to achieve way better results, for the four metrics, compared to NDAL and NDAR.

Our approach LDA in scenario S9 is generating higher CPU usage overhead than NDAL due to the solver overload, however we were able to overcome this issue by running the decision making process remotely (RDA) achieving better results than both approaches (NDAL and NDAR) with up to 93 percent reduction in the CPU usage compared to NDAL in scenario S4. For scenario S1, our approach had results similar

TABLE 8
Required Number of Iterations

Algo\Scenario	S1	S2	S3	S4	S5	S6	S7	S8	S9
NSGA-ii	3	3	3	4	24	30	45	54	67

to those of NDAL. This is due the optimal distribution found by the solver was to run all the services locally on the device (first row in Table 7), which is the same case as of NDAL. Same analysis applies on Scenarios S2 and S3, where the optimal distribution found by the solver was to offload all the services (second and third row in Table 7), which is the same case as of NDAR. Our approach (RDA) was also able to reduce the memory consumption by 22 percent compared to NDAR (scenario S9), reach 97 percent less energy consumption compared to NDAL and accelerate twice the execution compared to NDAL (scenario S9).

Another interesting observation is in scenarios S7 and S8, where it was impossible to run these scenarios on the device (NDAL yellow bars), but now it is possible using our approach (LDA and RDA) and even with better results than NDAR. The results also show that even after adding the overhead of the decision making process (red bars), either LDA or RDA is still giving better results than those of the other approaches. So for instance, if at any time being the CPU and/or the memory on the multi-persona device goes low, we opt to run the solver on the server side, so it doesn't consume from local CPU and memory. Whereas, in case more priority is given to the energy and/or execution time, the solver will be executed locally since for these metrics LDA had better results than RDA.

Finally, in scenarios S1 and S2, the overhead of the decision making process was remarkable. Therefore in what follows, we discuss some alternatives to decrease this overhead. For instance, without running the solver, a decision can be taken based on historical profiled behavior of the same situation. Another interesting idea is to find and generate 'good' solution rather than 'optimal' one, where in this case the algorithm runs for less period and hence decreasing its overhead. Even though on the other hand such proposition increases the overhead of the running services, yet as overall cost it might be beneficial in some scenarios. To investigate this option, we reduced the number of iterations of NSGA-ii in each scenario as depicted in Table 8. The results depicted in Fig. 5 show indeed that in some cases, even though the services have higher overhead but the reduced decision making cost is able to reduce the overall approach overhead. For instances, scenarios S5, S6, S7, S8 and S9 for CPU usage, S5, S6, S8 and S9 in terms of energy consumption, S2, S5 and S7 in terms of memory usage while in S5 and S8 for execution time. Yet deeper investigation is still needed and even other alternatives are to be investigated in future work.

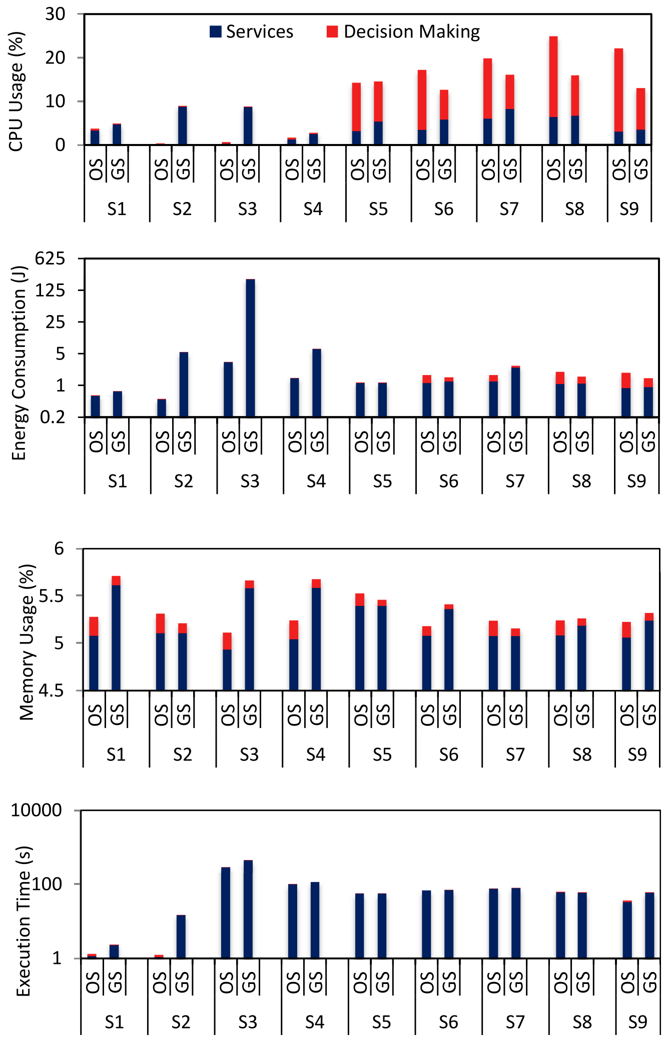


Fig. 5. Optimal(GS) and good(GS) solutions overheads.

8 Conclusion and Future Directions

Multi-persona solution is still impeded by the limited resources of mobile devices. These impediments and their implications of putting personas performance and viability on the line have been studied throughout the paper. To address these problems, we presented a novel offloading approach to be integrated with multi-persona on the mobile terminal. Through profiling, multi-objective optimization and heuristics, our proposition is capable of minimizing CPU and memory usages, energy consumption and execution time of the components running in each persona and hence augmenting the latter performance and viability. Most significantly, it was able to realise scenarios that were not previously feasible to run on the mobile device with multi-persona. Experiments demonstrated the efficiency and qualification of our proposition. Our approach was able in some scenarios to reduce the CPU usage by 93 percent, the memory usage by 22 percent and the energy consumption by 97 percent proved its capability to accelerate the execution of the applications with more than twice faster runtime, compared to existing approaches. This work opens the door for valuable future research directions. Investigating the frequency of calling the profiler on the device is an interesting track, while another valuable direction is to analyze the trade-off achieved between the proposed conflicting optimization objectives [46], [47], [48], [49].

Acknowledgments

The work has been supported by École de Technologie Supérieure (ETS), NSERC Canada, the Associated Research Unit of the National Council for Scientific Research CNRS Lebanon and the Lebanese American University (LAU).

REFERENCES

[1] M. Rouse (2012). BYOD (bring your own device) [Online]. Available: http://whatis.techtarget.com/definition/BYOD-bring-your-own-device.

[2] M. Rouse. (2014). COPE (corporate-owned, personally-enabled) [Online]. Available: http://searchconsumerization.techtarget.com/definition/COPE-corporate-owned-personally-enabled.

[3] M. Rouse. (2012). Dual persona (mobile device management)," [Online]. Available: http://searchconsumerization.techtarget.com/definition/Dual-persona.

[4] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, "Cells: A virtual mobile smartphone architecture," in *Proc. 23rd ACM Symp. Operating Syst. Principles*, 2011, pp. 173–187.

[5] (2014). How do spaces work?. [Online]. Available: http://securespaces.com/.

[6] O. Eiferman. (2014). How to balance security and freedom in medical BYOD [Online]. Available: http://health-information.advanceweb.com/Features/Articles/Physicians-Mobile-Devices.aspx.

[7] S. Wessel, F. Stumpf, I. Herdt, and C. Eckert, "Improving mobile device security with operating system-level virtualization," in *Proc. Security Privacy Protection Inform. Process. Syst.*, 2013, pp. 148–161.

[8] W. Chen, L. Xu, G. Li, and Y. Xiang, "A lightweight virtualization solution for android devices," *IEEE Trans. Comput.*, vol. 64, no. 10, pp. 2741–2751, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1109/TC.2015.2389791

[9] A. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 1, pp. 393–413, Jan.–Mar. 2014.

[10] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts*, vol. 16, no. 1, pp. 337–368, Jan. –Mar. 2014.

[11] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 869–876.

[12] S.-H. Hung, J.-P. Shieh, and C.-P. Lee, "Virtualizing smartphone applications to the cloud," *Comput. Inform.*, vol. 30, no. 6, pp. 1083–1097, 2012.

[13] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62.

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

[15] R. Kemp, "Programming frameworks for distributed smartphone computing," Ph.D. dissertation, VRIJE UNIVERSITEIT, Amsterdam, the Netherlands, 2014.

[16] E. Chen, S. Ogata, and K. Horikawa, "Offloading android applications to the cloud without customizing android," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2012, pp. 788–793.

[17] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[18] D. Mazza, D. Tarchi, and G. E. Corazza, "A partial offloading technique for wireless mobile cloud computing in smart cities," in *Proc. Eur. Conf. Netw. Commun.*, 2014, pp. 1–5.

[19] K. Barr, P. Bungale, S. Deasy, V. Gyuris, P. Hung, C. Newell, H. Tuch, and B. Zoppis, "The VMware mobile virtualization platform: Is that a hypervisor in your pocket?" *ACM SIGOPS Operating Syst. Rev.*, vol. 44, no. 4, pp. 124–135, 2010.

[20] C. Dall and J. Nieh, "Kvm/arm: Experiences building the linux arm hypervisor," Columbia Univ., New York, NY, Tech. Rep. CUCS-010-13, April 2013. [Online]. Available: http://academiccommons.columbia.edu/item/ac:162668.

[21] (2014). E. Inc. BYOD: What containers and wrappers don't tell you [Online]. Available: http://www.enterproid.com.

[22] O. Eiferman. (2013). BYOD: What containers and wrappers don't tell you [Online]. Available: http://www.cellrox.com/blog/byod-what-containers-and-wrappers-dont-tell-you.

[23] E. Ahmed, A. Gani, M. K. Khan, R. Buyya, and S. U. Khan, "Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 154–172, 2015.

[24] E. Ahmed, A. Gani, M. Sookhak, S. H. Ab Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 52–68, 2015.

[25] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya, "Network-centric performance analysis of runtime application migration in mobile cloud computing," *Simul. Model. Practice Theory*, vol. 50, pp. 42–56, 2015.

[26] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Y. Zomaya, "Network-assisted offloading for mobile cloud applications," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 5833–5838.

[27] B. Han, P. Hui, V. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: A case study," in *Proc. 5th ACM Workshop Challenged Netw.*, 2010, pp. 31–38.

[28] S. Andreev, A. Pyattaev, K. Johnsson, O. Galinina, and Y. Koucheryavy, "Cellular traffic offloading onto network-assisted device-to-device connections," *IEEE Commun. Mag.*, vol. 52, no. 4, pp. 20–31, Apr. 2014.

[29] H. Tout, C. Talhi, N. Kara, and A. Mourad, "Towards an offloading approach that augments multi-persona performance and viability," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf.*, 2015, pp. 455–460.

[30] T. Lust and J. Teghem, "The multiobjective multidimensional knapsack problem: A survey and a new approach," *Int. Trans. Oper. Res.*, vol. 19, no. 4, pp. 495–520, 2012.

[31] K. Deb, "An introduction to genetic algorithms," *Sadhana*, vol. 24, no. 4-5, pp. 293–315, 1999.

[32] J. J. Grefenstette, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hove, U.K., Psychology Press, 2013.

[33] C.-W. Wu, T.-C. Chiang, and L.-C. Fu, "An ant colony optimization algorithm for multi-objective clustering in mobile ad hoc networks," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2963–2968.

[34] Z. Cai and C. Chen, "Demand-driven task scheduling using 2d chromosome genetic algorithm in mobile cloud," in *Proc. Int. Conf. Progress Informat. Comput.*, 2014, pp. 539–545.

[35] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh. (2012). Cells: Lightweight virtual smartphones [Online]. Available: http://systems.cs.columbia.edu/projects/cells/.

[36] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Commun. Surveys Tuts*, vol. 15, no. 3, pp. 1294–1313, Jul.–Sep. 2013.

[37] M. Shiraz and A. Gani, "A lightweight active service migration framework for computational offloading in mobile cloud computing," *J. Supercomput.*, vol. 68, no. 2, pp. 978–995, 2014.

[38] (2012). "Android Interface Definition Language (AIDL). [Online]. Available: http://developer.android.com/guide/components/aidl.html.

[39] (2013). Power profiles for android. [Online]. Available: https://source.android.com/devices/tech/power/index.html.

[40] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardware/software Codes. Syst. Synthesis*, 2010, pp. 105–114.

[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[42] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," presented at the *Evolutionary Methods Design, Optimisation Control Application Industrial Problems*, K. Giannakoglou, et al., Eds. International Center for Numerical Methods in Engineering (CIMNE), Athens. Greece, 2002, pp. 95–100.

[43] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Evol. Multi-Criterion Optimization*, 2005, pp. 62–76.

[44] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. Parallel Problem Solving Nature-PPSN VIII*, 2004, pp. 832–842.

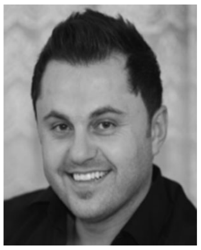
[45] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "Mocell: A cellular genetic algorithm for multiobjective optimization," *Int. J. Intell. Syst.*, vol. 24, no. 7, pp. 726–746, 2009.

[46] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2013, pp. 728–732.
[47] J. Liu, K. Kumar, and Y.-H. Lu, "Tradeoff between energy savings and privacy protection in computation offloading," in *Proc. 16th ACM/IEEE Int. Symp. Low Power Electron. Des.*, 2010, pp. 213–218.
[48] H. Wu and K. Wolter, "Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric," in *Proc. 8th Int. Conf. Perform. Eval. Methodologies Tools*, 2014, pp. 90–97.
[49] J. Song, Y. Cui, M. Li, J. Qiu, and R. Buyya, "Energy-traffic tradeoff cooperative offloading for mobile cloud computing," in *Proc. IEEE 22nd Int. Sym. Quality Service*, 2014, pp. 284–289.

Hanine Tout received the MSc degree in computer science from the Lebanese American University, Beirut, Lebanon. She is currently working toward the PhD degree in software engineering at ETS, University of Quebec, Montreal, Canada. Her research interests include mobile cloud computing, mobile virtualization, optimization, Web services, security and formal verification. She is serving as TPC member for NTMS 2016 and a reviewer in IEEE Communications Letters, Computers & Security journal and several international conferences. She is a student member of the IEEE.

Chamseddine Talhi received the PhD degree in computer science from Laval University, Quebec, Canada. He is an associate professor in the department of software engineering and IT at ETS, University of Quebec, Montreal, Canada. He is leading a research group that investigates smartphone and embedded systems security. His research interests include cloud security and secure sharing of embedded systems.

Nadjia Kara received the PhD degree in electrical engineering from the Ecole Polytechnique of Montreal, Canada. She is an associate professor in software engineering and IT at ETS, University of Quebec and an affiliate professor at Concordia University, Montreal, Canada. She worked in the industry as researcher and system architect for more than 10 years. Her research interests include network and service architectures, next generation networks, distributed systems, traffic engineering, and quality of service.



Azzam Mourad received the PhD degree in electrical and computer engineering from Concordia University, Montreal, Canada. He is an associate professor of computer science at the Lebanese American University. His research interests include information security, Web services, mobile cloud computing, Big Data analysis, vehicular networks, and formal semantics. He is coordinator of the associated research unit on intelligent transport & vehicular technologies. He is serving as associate editor for IEEE Communications Letters, general co-chair of WiMob2016 and Track Chair, TPC member and reviewer of several prestigious conferences and journals. He is a senior member of the IEEE.