2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering

Optimizing User Experience in Choosing Android Applications

Rubén Saborido[1], Giovanni Beltrame[2], Foutse Khomh[3], Enrique Alba[4], Giuliano Antoniol[1]
SOCCER[1]–MIST[2]–SWAT[3] Labs., DGIGL, Polytechnique Montréal, Canada
[4] Dep. of Computer Science, University of Málaga

***Abstract*—**
In this paper, we present a recommendation system aimed at helping users and developers alike. We help users to choose optimal sets of applications belonging to different categories (*e.g.,* browsers, e-mails, cameras) while minimizing energy consumption, transmitted data, and maximizing application rating. We also help developers by showing the relative placement of their application's efficiency with respect to selected others. When the optimal set of applications is computed, it is leveraged to position a given application with respect to the optimal, median and worst application in its category (*e.g.,* browsers).

Out of eight categories we selected 144 applications, manually defined typical execution scenarios, collected the relevant data, and computed the Pareto optimal front solving a multi-objective optimization problem.

We report evidence that, on the one hand, ratings do not correlate with energy efficiency and data frugality. On the other hand, we show that it is possible to help developers understanding how far is a new Android application power consumption and network usage with respect to optimal applications in the same category.

From the user perspective, we show that choosing optimal sets of applications, power consumption and network usage can be reduced by 16.61% and 40.17%, respectively, in comparison to choosing the set of applications that maximizes only the rating.

***Keywords*-Software Energy Consumption; Performance; Android; Multi-objective Optimization**

I. Introduction

The Android market is a place where Android users and developers meet to find and provide applications. They have different concerns as a developer may wish to maximize the number of downloads and the application ratings, while a user may be more interested in finding the best and cheapest application to fulfill some tasks. However, they are also likely to share concerns about the power consumption and data frugality of the applications. Developers may wonder if their application consumes more energy or data with respect to the top ranked applications in the same category (*e.g.,* browsers). Users, on the other hand, may wonder why the cell phone's battery is already low or why they already used almost all the data of their monthly phone plan.

For the Android ecosystem, a given application category (e.g., weather forecast, business, communication, etc.), contains many applications, often implementing very similar features. Within this broad range of applications, users find very limited support to help them determine which application is more energy hungry, uses more data or matches his preferences or needs. For example, a popular application can be greedy in energy and/or network bandwidth without users noticing it. This is the case for a weather forecasting application, which, considering our experiments in this work, transmits more than 14MB over the network in a typical usage scenario to get the weather forecast.

Power consumption and data traffic of mobile applications are nowadays a hot topic given the popular use of mobile devices. These issues have been addressed in many papers, see for example [1]–[4], just to name a few contributions. Most users data plans have a fixed monthly data transmission budget and they have to pay more money or the connection speed is slowed down if additional data are used. For this reason, network usage is important and, in this sense, we know that free applications may have hidden costs [5].

Comparing applications is not an easy task. From the user perspective, the best applications combination depends on the user preferences, the frequency of usage of different features, the performance of the features, the application's energy consumption, and the applications (hidden) costs such as the data transmission overhead [5]. In a similar way, the developer needs to access details of competing applications. Overall, users and developers need some means to compare, given a set of categories, similar applications and select the best possible combination of applications to install (user point of view) or to compare against (developer point of view).

In this paper we present ADAGO, an Android Application DAta enerGy Overhead advisor, to support both users and developers of Android applications. ADAGO models the application selection problem as a multi-objective optimization problem. ADAGO's goal is to select the most energy and data frugal set of applications, maximizing the user rating for a given set of categories. More precisely, ADAGO is based on multi-objective optimization and it aims at finding Pareto optimal solutions, non dominated solutions representing combinations of applications, minimizing energy consumption and network usage, and maximizing the rating (a number between one and five associated to each application in *Google Play*). To obtain a realistic evaluation of the approach's feasibility and usefulness, we selected a set of eight categories and, for each category, we downloaded the 100 most popular applications considering the number of downloads. Then, for the top 20, we manually defined per category of applications, realistic execution scenarios, which we played several times recording both power consumption and network traffic. We use the collected data to answer the

978-1-5090-1855-0/16 $31.00 © 2016 IEEE
DOI 10.1109/SANER.2016.64

following high level research question:

Is ADAGO effective in finding a set of optimal applications and quantifying the energy/data overhead gap between similar applications?

Our results show that considering the application's rating is not an effective strategy to select applications to install or compare against. In addition, we show that ADAGO is able to generate optimal sets of applications reducing the power consumption and network usage, and how we can help developers to measure how far is a new application to the optimal applications in the same category.

The remainder of the paper is organized as follows. In Section II we present basic concepts related to multi-objective optimization. In Section III, we define our approach to define a recommendation system to choose optimized sets of Android applications. In Section IV we present the case study used to evaluate the proposed approach. In Section V, we report the results of our measurements and we analyze the optimal sets of applications generated by the proposed recommendation system. In Section VI we discuss the threats to the validity of our experiment. Finally, in Section VII, we describe the related work and we conclude the paper in Section VIII.

II. Background in Multi-objective Optimization

Multi-objective optimization problems are mathematical programming problems with a vector-valued objective function, which is usually denoted by $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_n(\mathbf{x}))$, where $f_j(\mathbf{x})$, for $j = 1, \ldots, n$, is a real-valued function defined on the feasible region $F \subseteq \Re^N$. Consequently, the decision space belongs to $\Re^N$ while the criterion space belongs to $\Re^n$, and the multi-objective optimization problem can be stated as follows:

$$\begin{array}{ll} \text{optimize} & [f_1(\mathbf{x}), ..., f_n(\mathbf{x})] \\ \text{s.t.} & \mathbf{x} \in F \end{array}$$

In the functional space of criterion, some objective functions should be maximized ($j \in J_{max}$) while others should be minimized ($j \in J_{min}$), these subsets of indices verifying that $J_{max} \cup J_{min} = \{1, ..., n\}$. In this context, optimality is defined on the basis of the concept of *dominance*, in such a way that solving the above problem implies finding the subset of *non-dominated solutions*, that is those feasible solutions which are not dominated by any other feasible one. A feasible solution $\mathbf{x}^0$ *dominates* another solution $\mathbf{x} \in F$ if and only if $f_j(\mathbf{x}^0) \geq f_j(\mathbf{x})$, for every $j \in J_{max}$ and $f_j(\mathbf{x}^0) \leq f_j(\mathbf{x})$, for every $j \in J_{min}$, with at least one strict inequality. The set of non-dominated solutions will also be referred by *Pareto optimal solutions* and define the efficient frontier or *Pareto optimal front* of the multi-objective optimization problem (see, for instance, [6]). In general, in multi-objective optimization, there is not one single optimal solution but a set of "good" or *non-dominated* solutions.

Metaheuristics [7], as *Evolutionary Algorithms* (EA) and, specially, *Evolutionary Multi-objective Optimization* (EMO) algorithms [8] work well handling multi-objective optimization problems.

III. The ADAGO approach

ADAGO is an advisor, a recommendation system mimicking what an user/developer will likely do to select one or more applications in one or more application categories. Application selection will likely be based on application ratings and, if available, on energy and network data overhead footprints. The lower the energy and data overhead footprints, the longer the battery will last and the less costly data transmission will be.

The ADAGO process can be divided in different steps, as it is shown in Fig. 1. Each step will be detailed in the following sections. Here, in this section, we focus on the core ADAGO part, the generation of a set of optimal solutions (Step four). The user can manually specify different categories and select for each category a set of applications. Power consumption and data overhead can be measured in different ways. ADAGO is transparent to the data collection process and its inputs are a set of categories and application measures. Its output is a set of Pareto optimal solutions.

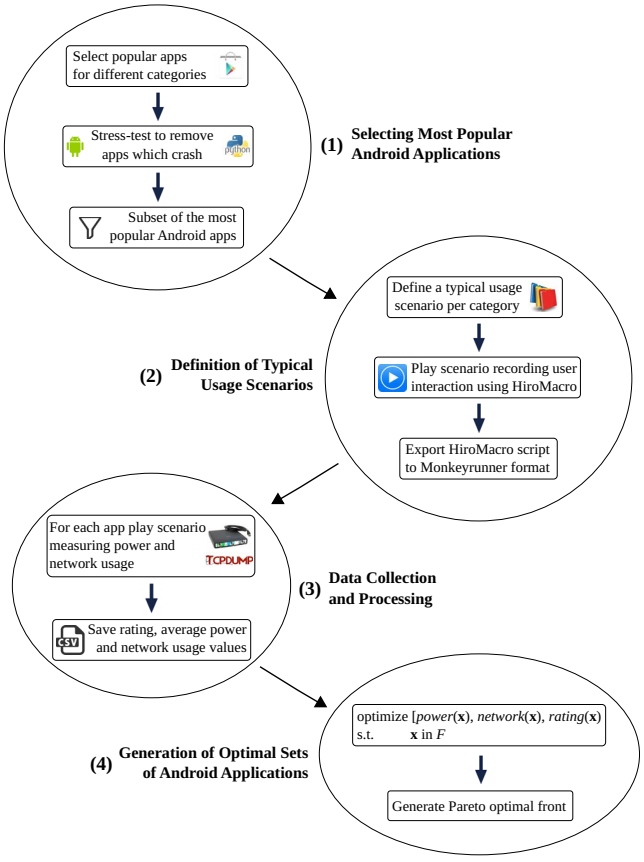

Fig. 1. ADAGO conceptual sequence of steps.

Let $\mathcal{C} = \{C_1, ..., C_N\}$ be a set of N categories. Further assume that, for each category C_i, a set A_i of applications has been selected. In other words, $\mathcal{A} = \{A_1, ..., A_N\}$ is the

application set of sets. An element $\mathbf{x}$ of the search space F, $\mathbf{x} = (x_1, \ldots, x_N)$, is a set of applications where x_j is an application selected from A_j. A solution $\mathbf{x}$ contains one application from each category in $\mathcal{C}$.

ADAGO goal is to analyze the trade-off between power consumption, network usage and global rating, and it models the following combinatorial multi-objective optimization problem:

$$\begin{array}{ll} \text{optimize} & [power(\mathbf{x}), network(\mathbf{x}), rating(\mathbf{x})] \\ \text{s.t.} & \mathbf{x} \in F \end{array} \quad (1)$$

Given a solution $\mathbf{x}$, the objective functions used in (1) are calculated as follow:

$$power(\mathbf{x}) = \frac{\sum_{i=1}^{N} power(x_i)}{N} \quad (2)$$

$$network(\mathbf{x}) = \frac{\sum_{i=1}^{N} network(x_i)}{N} \quad (3)$$

$$rating(\mathbf{x}) = \frac{\sum_{i=1}^{N} rating(x_i)}{N} \quad (4)$$

In (2) and (3), $power(x_i)$ and $network(x_i)$ are the average values of power (in Watts) and network usage (in megabytes) for application x_i in a certain number of runs and for a given number of exercised application functionalities. In (4), $rating(x_i)$ is the rating of the application x_i in *Google Play* (it is a number, between one and five, calculated as the weighted average of user ratings). Notice that the constant N is just a rescaling factor and thus, in this case, optimizing $\frac{\sum_{i=1}^{N} power(x_i)}{N}$ is the same as optimizing $\sum_{i=1}^{N} power(x_i)$. The same holds for network usage and application ratings.

IV. Case Study Design

The *goal* of this study is to assess ADAGO capabilities with the *purpose* of understanding ADAGO applicability, to help users and developers to select and compare an optimal set of applications. The *quality focus* is, on the one hand, the ability to select an optimal set of applications and, on the other hand, reducing users' effort during the selection of an application. The *perspective* is the point of view of users who wish to select and install a set of applications (selected from a set of categories) and the point of view of developers who need to benchmark their applications against the best/worst applications in the same category.

The *context* consists of eight application categories and 144 applications. The eight application categories are: *Browsers*, *Cameras*, *Email*, *Flash Lights*, *Music Players*, *News*, *Video Players*, and *Weather*. The categories have been selected based on the categories used in [9]. Within the categories, the applications have been selected based on the number of downloads in the *Play Store*.

The high level research question stated in Section I has been refined into the following three detailed research questions:

- **RQ$_1$:** *To what extent is rating a good indication of energy and data overhead frugality?*
  The rationale of this research question is to analyze whether ADAGO has a reason to exist. If, indeed, ratings are a good proxy for energy usage and data overhead then there is no need for ADAGO. Here we mimic a user selecting one application at a time, per category, solely based on ratings. This user solution is compared to (1) ADAGO applied to each single category and (2) ADAGO selecting simultaneously the N applications. Notice that this RQ is relevant to both an application developer and an application user.

- **RQ$_2$:** *Given ADAGO optimal solutions, what is the trade-off between power consumption, network usage and application rating?*
  This research question is a follow-up of the previous one and aims to quantify the potential gain offered by ADAGO in terms of saved energy and data overhead. However, here the focus is on the application user.

- **RQ$_3$:** *For a given category, can we assess how much better is a new application with respect to the others?*
  This research question looks at how different applications in the same category compare to each other. This allows us to achieve a deeper understanding of how spread power consumption and data overhead are. Using this information, developers will be able to compare their applications against similar applications from its competitors.

A. Experiments

In our experiments we define the user solution as the combination of applications per category which maximizes the rating in each category.

To address **RQ$_1$**, we perform two analyses. First, we inspected the values of power consumption and data overhead in the user solution and in the solutions generated by ADAGO. We plotted the user solution versus the optimal using bar charts. Then, we considered the power and data measures for each category as extracted by different populations. A population selected by the user and two populations selected by ADAGO. We compared the different solutions using *Wilcoxon rank sum* test with p-value *Holm* correction. We also computed the *Cliff-Delta* effect size.

To address **RQ$_2$**, we proceed in a way similar to **RQ$_1$**. We verified if marginally sacrificing the rating of ADAGO solutions could lead to a trade-off in power consumption, data, and rating. To measure this trade-off we selected the two best solutions found by ADAGO which minimize the power consumption and the network usage, and they were compared to the user solution, which maximizes the rating.

To address **RQ$_3$**, we perform two different kind of analyses. First we fixed one category, then we computed per category the optimal, median and worst power consumption and data overhead. Finally, we inspected the histogram of power consumption and network usage (separately) to show

how the new application is positioned with respect to the optimal applications in the same category. This mimics a developer who needs to understand if her/his new application is competitive and how far it is from the bests in the category.

B. Data Extraction

In this subsection we describe how we extract data with the aim of addressing the research questions formulated above. In our experiments, we use a *LG Nexus 4* Android phone, equipped with Android Lollipop operating system (version 5.1.1, Build number LMY47V). In the following, we detail the steps from one to four that are shown in Fig. 1.

1) Selecting Most Popular Android Applications: We select a set of free Android applications belonging to eight different categories. These applications are chosen considering the number of downloads in *Play Store*, because, somehow, this fact describes the tendency of Android users. Based on the categories used in [9], we define a subset of these categories considering the most common applications used nowadays by smart-phone's users: *Browsers*, *Cameras*, *Email* managers, *Flash Lights*, *Music Players*, *News* viewers, *Video Players*, and *Weather* forecast. For each category, 100 applications are selected and their descriptions, statistics (including the rating), and *apk* files are downloaded automatically using a *Perl* script developed by us and the tool *Play Store Crawler*[1]. In addition, we have developed a stress-test *Python* script which uses the *adb*[2] and *Monkey*[3] Android tools to remove the applications that crashed during their execution in the real phone used in our experiments. This stress-test is similar to the approach proposed in [9], configuring *Monkey* to generate 180 random events during 60 seconds (three events per second). Around 2% of the applications crashed during this test and, therefore, were removed. Considering the rest of applications, the subset of most downloaded 20 applications are finally selected for each category, except for email managers where the number of applications is four. In total, we analyze 144 applications for all of the eight selected categories. The final list of selected applications is available in our website[4].

2) Definition of Typical Usage Scenarios: For each application in a category, we propose a typical usage scenario and we play it automatically, taking care of measuring the power consumption and the network usage associated. To collect real scenarios, we interact with each application under study using the Android application *HiroMacro*[5]. This software allows us to generate scripts containing the *touch* and *move* events while a real user interacts with each application directly on the phone. The resulting script can be played automatically using the same application but we convert the *HiroMacro* script to a *Monkeyrunner* script format. Thus, the interaction to collect the scenario is done using the phone and the actions can be played automatically from our code using the *Monkeyrunner*[6] Android tool. For all of the applications, we simulate a usual scenario for users (for example, navigating using the browser, taking some pictures, or playing a song). The scenarios defined for each category and collected for each application are described in Table I. It is important to mention that we assign a fixed time duration in seconds for waiting between different actions. This time is set to ensure that previous actions have been finished completely.

TABLE I
TYPICAL USAGE SCENARIOS DEFINED FOR EACH APPLICATION CATEGORY.

Category	Scenario description
Browsers	- Go to wikipedia website. - Search the word "software" and select first article. - Wait 10 seconds to simulate user reading. - Scroll down. - Wait 10 seconds to simulate user reading. - Scroll down. - Wait 10 seconds to simulate user reading. - Go back and close the application.
Cameras	- Enable flash. - Focus and take a picture. - Wait five seconds. - Focus and take a second picture. - Wait five seconds. - Focus and take a new picture. - Wait five seconds. - Close the application.
Emails	- Go to inbox. - Select first message. - Download attached PDF file (12.2 MB). - Wait 40 seconds to download the file. - Go back and close the application.
Flash Lights	- Turn on the torch. - Wait five seconds. - Turn off the torch. - Turn on the torch. - Wait five seconds. - Turn off the torch. - Close the application.
Music Players	- Go to media library. - Select and play first song. - Wait 20 seconds. - Pause and go back. - Select and play second song. - Wait 20 seconds. - Pause, go back and close the application.
News	- Select first news. - Wait seven seconds to simulate user reading. - Scroll down. - Wait seven seconds to simulate user reading. - Scroll down. - Wait seven seconds to simulate user reading. - Go back. - Select second news. - Wait seven seconds to simulate user reading. - Scroll down. - Wait seven seconds to simulate user reading. - Scroll down. - Wait seven seconds to simulate user reading. - Go back and close application.
Video Players	- Go to media library. - Select and play movie for 30 seconds. - Pause, go back and close application.
Weather	- Add city Montréal (Canada). - Wait four seconds. - Add city Madrid (Spain). - Wait four seconds. - Select city Montréal (Canada). - Wait four seconds. - Select city Madrid (Spain). - Wait four seconds. - Close application.

[1]https://github.com/Akdeniz/google-play-crawler
[2]http://developer.android.com/tools/help/adb.html
[3]http://developer.android.com/tools/help/monkey.html
[4]http://ser.soccerlab.polymtl.ca/ser-repos/public/tr-data/adago.tar.gz
[5]https://play.google.com/store/apps/details?id=com.prohiro.macro
[6]http://developer.android.com/tools/help/monkeyrunner_concepts.html

3) Data Collection and Processing: For the experiments, each application is run and the collected scenario for it is played in an automatic way. Each application is run 20 times to get averaged results and, for each run, the application is uninstalled after its usage. A complete description of the followed steps is given in Algorithm 1, which has been implemented as a *python* script. As it is described, all the applications for a category are executed before a new run is started. Thus, we aim to avoid that cache memory on the phone stores information related to the application run. In addition, before the experiments, the screen brightness is set to the minimum value and the phone is set to keep the screen on.

```
forall categories do
    forall runs do
        forall applications do
            Install application (using adb).
            Run application (using adb).
            Wait 10 seconds (to load the app fully).
            if application requires initialization then
                Play set-up (using Monkeyrunner).
            end
            Start tcpdump (using adb).
            Start oscilloscope to measure energy.
            Play scenario (using Monkeyrunner).
            Stop oscilloscope.
            Stop tcpdump (using adb).
            Download the tcpdump file (using adb).
            Stop application (using adb).
            Clean application files (using adb).
            Uninstall application (using adb).
        end
    end
end
```

Algorithm 1: Steps in our script to do the experiments.

Because some applications need to be set up before they can be used, in Algorithm 1 there is a step to initialize the application when it is required. In these cases, the initialization process uses *Monkeyrunner* to run a sequence of events to set up the application. These events were collected previously for each application using *HiroMacro* as was explained before in this section.

Finally, when the data are collected, they are processed to calculate and save in a *Comma Separated Values* (CSV) file, the associated rating existing in *Google Play*, the average power consumption in Watts (W), and the network usage in megabytes (MB), for each application over all the runs.

Power Consumption Measurement: The power consumption is measured using a precise digital oscilloscope *TiePie Handyscope HS5*. We use this device because it allows to measure using high frequencies and because it offers the *LibTiePie SDK*, which is a cross platform library for using TiePie engineering USB oscilloscopes through third party software. Between the power supply and the phone we connect, in series, a *uCurrent*[7] device, which is a precision current adapter for multimeters converting the input current in a proportional output voltage (V_{out}). The input current (I) is calculated by the *uCurrent* device and, therefore, $I = V_{out}$. Knowing I and the voltage supplied by the power supply (V_{sup}), we use the *Ohm's Law* to calculate the power consumption (P) as $P = V_{sup} * I$.

[7]http://www.eevblog.com/projects/ucurrent/

The resolution is set up to 16 bits and the frequency to $125\,kHz$, therefore a measure is taken each eight microseconds. The *LG Nexus 4* phone is connected to an external power supplier which is connected to the phone's motherboard, thus we avoid any kind of interferences with the phone battery in our measurements. The diagram of the connection is shown in Fig. 2.

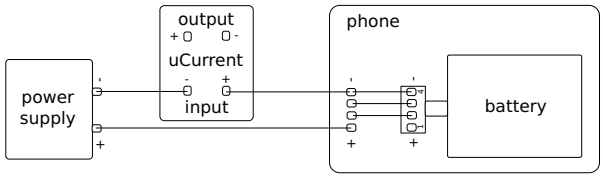

Fig. 2. Connection used to measure the power consumption on Nexus 4.

Because the phone is connected via USB to the PC to send and receive data, the USB charging was disabled on the device to avoid any kind of interferences in our measurements. A simple Android application was developed which allows to enable or disable the USB charging in *Nexus 4* phones. This application is free and it is available for download in the *Play Store*[8].

Network Usage Measurement: The number of transmitted bytes is collected using the tool *Android tcpdump*[9] on the phone, which has been used in recent works as [5]. It is a command line packet capture utility, useful for capturing packets from the Wifi and cellular connections and it has the same switches and options as its linux's counterpart. We use this tool via *adb* to capture the number of bytes transmitted over the network connection while an application is running.

4) Generation of Optimal Set of Android applications: In order to generate different optimal sets of applications taking into account power consumption, network usage, and global rating, the multi-objective optimization problem (1) can be solved using metaheuristics. This kind of techniques are useful when the search space is large and it is not possible to explore it exhaustively. Given that we select eight categories and 20 applications for each of them, except for *Emails* where we have four applications, there are $20^7 \cdot 4 = 5120000000$ possible combinations of applications. Because in each category there exist applications that have a lower power, lower network usage, and a higher rating than others applications in the same category, the search space can be reduced removing the dominated applications in each category regarding the Pareto dominance relation. Considering this fact, the total number of Pareto optimal applications per category after the reduction is shown in Table II.

Taking into account this reduction, there are 138240 possible combinations of applications, which can be generated exhaustively. After that, the Pareto dominance relation is applied over all the possible combinations and the final sets of applications are obtained. The resulting Pareto optimal

[8]https://play.google.com/store/apps/details?id=ruben.nexus4usbcharging
[9]http://www.androidtcpdump.com/

TABLE II
APPLICATIONS PER CATEGORY AFTER SEARCH SPACE REDUCTION

Category	Applications
Browsers	4
Cameras	10
Emails	2
FlashLights	9
MusicPlayers	4
News	2
VideoPlayers	4
Weather	6

front is shown in Fig. 3, which contains 896 optimal sets of applications. The time needed to compute the Pareto optimal front was 39.8 seconds. We have used a laptop equipped with a Intel Core i3-3217U 1.80 GHz processor and 6 GB of main memory running Debian Linux 8.

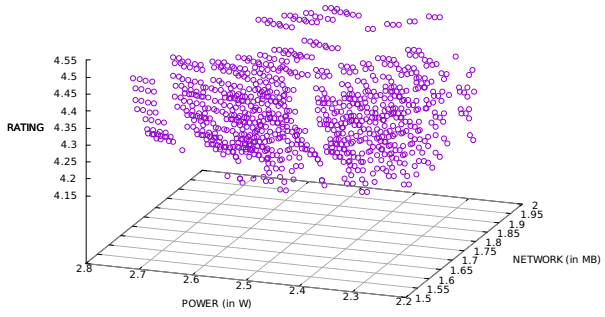

Fig. 3. Pareto optimal front generated after the search space reduction.

If the number of categories or the number of applications per category is greater, the search space could be too large to be explored exhaustively and metaheuristics would be needed. In these cases, the multi-objective optimization problem (1) can be solved using EMO algorithms, as *NSGA-II*. To check this fact, problem (1) is defined and included in *jMetal* [10], an object-oriented Java-based framework for multi-objective optimization with metaheuristics, and it is solved using *NSGA-II*. In our online technical report [11] we describe the algorithm configuration and the Pareto optimal front obtained, which is very similar to the one exposed here. In this case, the time needed to compute the Pareto optimal front was 59.5 seconds.

V. RESULTS

A global portrait of the average power consumption usage for selected applications in each category is shown in Fig. 4. *Cameras* and *Video Players* are the most expensive categories and, on the other hand, *Emails* and *Flash Lights* applications consume lower power than applications in other categories. The same information related to network usage is shown in Fig. 5. Surprisingly, there are cameras and flashlights applications transmitting data over the network; some of these data are likely due to ads but it is hard to know exactly what kind of information is sent/received. Similar charts for applications in each category are available in [11], showing the average power consumption and data usage for selected applications.

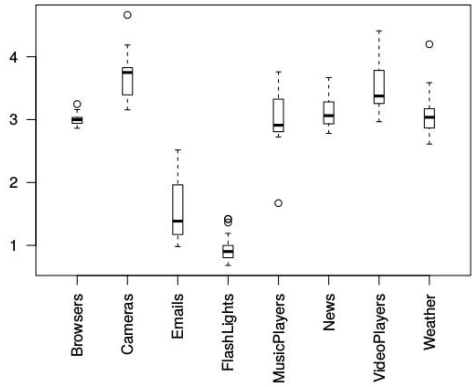

Fig. 4. Average power consumption (in W) for applications in each category.

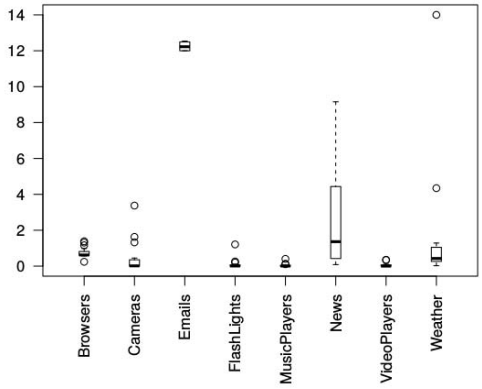

Fig. 5. Average network usage (in MB) for applications in each category.

In previous section we defined the user solution as the combination of applications per category which maximizes the rating in each category. If there are more than one application in a category with the maximum rating, one of them is selected randomly. Let us define $\mathbf{x}_{user}$ as the user solution. Considering the applications selected and our measurements, $power(\mathbf{x}_{user}) = 2.87$, $network(\mathbf{x}_{user}) = 2.58$, and $rating(\mathbf{x}_{user}) = 4.56$, calculated using (2), (3), and (4).

RQ$_1$: *To what extent is rating a good indication of energy and data overhead frugality?*

To answer this research question, firstly, the applications per category selected by the user, which maximizes the rating, are compared to the Pareto optimal applications found by ADAGO in each category. We expose this comparison for applications in the *News* category, where ADAGO found two optimal applications. In Fig. 6, the comparison between the first application (*com.mobilesrepublic.appy*) and the application chosen by the user (*com.guardian*) is shown. The application found by ADAGO has the same rating (4.5) as the user application,

but it improves the power consumption and network usage by 5.40% and 95.04%, respectively.

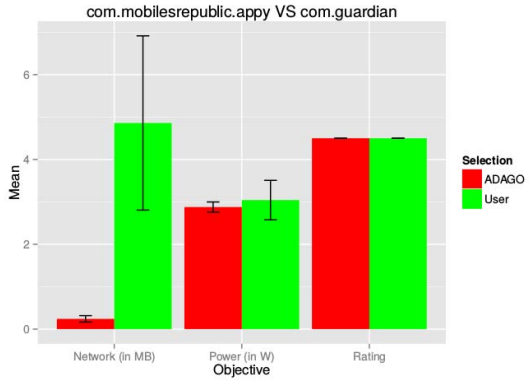

Fig. 6. Comparison of the first optimal application found by ADAGO and the application chosen by the user (in *News* category).

The comparison respect the second optimal application found by ADAGO (*net.aljazeera.english*) is shown in Fig. 7. In this case, the improvement in power consumption and network usage is 8.55% and 98.30%, but lower rating (4.1).

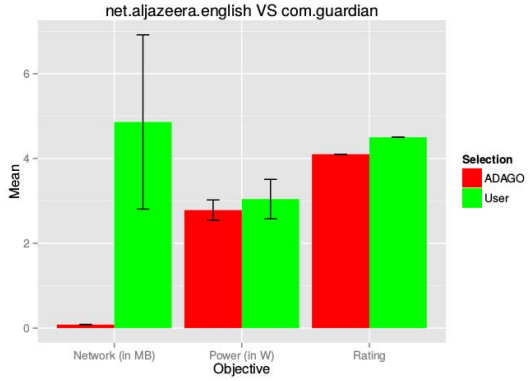

Fig. 7. Comparison of the second optimal application found by ADAGO and the application chosen by the user (in *News* category).

We conclude that, because in the same category several applications can have the same rating, the application chosen by the user cannot be optimal given that other applications with the same rating could have better power consumption and network usage. Figures for all the categories used in this work are available in our online technical report [11].

Table III reports, for two categories, the results of *Wilcoxon rank sum* and the *Cliff-Delta* effect size (this information for all categories is available in [11]). Bold values for p-value indicate that the comparison between an application selected by ADAGO and the application selected by the user is statistically significant (at 95%) after *Holm* correction. For each category (first table column), the maximum rating is reported which is the rating associated to the application chosen by the user. One can notice (second column) that the Pareto front contains applications with the user selected application. Actually (see the *Music Players* category) the user selection is sometimes an optimal application. In such cases the p-value is not significant and the effect size is zero. What we observe in Table III is that not always the ADAGO solution has the lower energy (or data overhead) consumption. Anyway, as we expected, either the user selection is optimal or ADAGO applications have either lower energy footprint or lower data footprint with significant p-values. Furthermore, the effect size is almost always very high. Cliff delta is considered to be negligible for values below 0.147, small below 0.33, medium below 0.475 and large otherwise. We thus conclude that when energy (or data overhead) of ADAGO's application is lower than the user selected, this is statistically significant and with a large effect size. That is true if the special case when the user selection is optimal is removed. Similar findings are reported for all categories in the online technical report [11].

In addition to the previous comparison, the user solution is compared to the combination of applications generated by ADAGO for all the categories simultaneously (the resulting Pareto optimal front). In this case, the user solution is not optimal because the Pareto optimal front generated by ADAGO contains 11 solutions with similar rating but better power and data. From these 11 solutions we select two, the best one considering the power consumption ($\mathbf{x}_{min_p}$) and the best one that minimizes the network usage ($\mathbf{x}_{min_n}$), and both are compared to the user solution in Fig. 8. In this chart, a bar is drawn for each objective and its height is defined by the minimum and maximum values of the corresponding objective in the Pareto optimal front obtained by ADAGO. A line is used to represent each solution specifying the value of each objective function. As it is shown in Fig. 8, ADAGO is able to generate better solutions. The improvement in power consumption and network usage is 5.42% and 25.67%, respectively, considering $\mathbf{x}_{min_p}$ versus $\mathbf{x}_{user}$. Regarding $\mathbf{x}_{min_n}$, the improvement with respects to $\mathbf{x}_{user}$ is 1.37% and 29.15%, respectively. In all the compared solutions the global rating is similar (4.52). In addition, as it is shown in the chart, the network usage of the user solution is worse than the data usage of any Pareto optimal solution found by ADAGO.

> *We conclude that a good rating associated to an application in the Play Store does not warrant an efficient use of power consumption or network.*

RQ$_2$: *Given ADAGO optimal solutions, what is the trade-off between power consumption, network usage and application rating?*

As we commented in RQ_1, the user solution is not optimal because ADAGO was able to generate several Pareto optimal solutions where 11 of these solutions had a similar rating but they improved the power consumption and data usage. If we consider that power consumption or network usage are more important for the final user than the rating, the two best Pareto optimal solutions considering the power consumption

TABLE III
STATISTICAL TESTS ON WHETHER THERE IS A DIFFERENCE BETWEEN APPLICATIONS SELECTED BY ADAGO AND THE USER SELECTION. THE RESULTS ARE SHOWN FOR POWER CONSUMPTION AND NETWORK USAGE IN TWO DIFFERENT CATEGORIES

Category (max. Rating)	Application	Rating	Power		Network	
			p-val	cliffd	p-val	cliffd
Browsers (4.60)	com.apusapps.browser	4.60	0.8299	0.205	**0.0000**	0.950
	com.ksmobile.cb	4.60	0.9042	0.025	**0.0000**	0.950
	com.opera.mini.native	4.40	**0.0448**	0.465	**0.0000**	0.950
	com.UCMobile.intl	4.50	0.8299	0.175	**0.0000**	0.950
Music Players (4.60)	cn.voilet.musicplaypro	4.40	**0.0000**	0.950	**0.0000**	-1.000
	com.aimp.player	4.50	**0.0000**	0.855	**0.0000**	0.950
	com.n7mobile.nplayer	4.50	**0.0064**	0.535	**0.0000**	0.950
	com.tbig.playerprotrial (*user selection*)	4.60	1.0000	0.000	1.0000	0.000

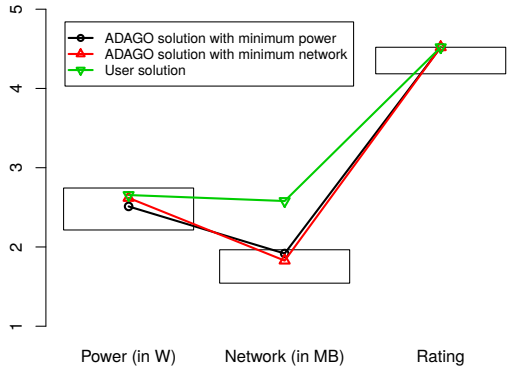

Fig. 8. Comparison of two optimal solutions found by ADAGO and the solution chosen by the user (all of them with a similar rating value).

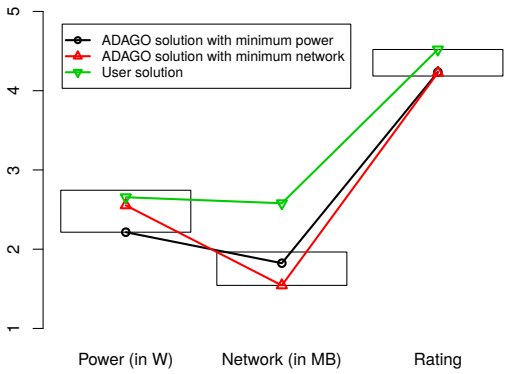

Fig. 9. Comparison of two optimal solutions found by ADAGO and the solution chosen by the user (they do not have a similar rating).

and data usage could be selected. Both solutions are compared versus the user solution in Fig. 9. Let us set $\mathbf{x'}_{min_p}$ as the best solution generated by ADAGO considering the power consumption. Regarding our data, when $\mathbf{x'}_{min_p}$ is compared to $\mathbf{x}_{user}$, the rating is sacrificed by 6.30%, but the improvement in power consumption and network usage is 16.61% and 29.32%, respectively. On the other hand, if the network usage is considered more important than the rating, the Pareto optimal solution with the minimum data usage could be selected. Let us set $\mathbf{x'}_{min_n}$ as the solution generated by ADAGO with the minimum network usage. Taking into account our data, when $\mathbf{x'}_{min_n}$ is compared to $\mathbf{x}_{user}$, the rating is sacrificed by 6.50%, but the improvement in power consumption and network usage is 3.84% and 40.17%, respectively.

We conclude that, sacrificing the rating, power consumption or network usage could be reduced by 16.61% and 40.17%, respectively, in comparison to the user solution based on rating only.

RQ$_3$: *For a given category, can we assess how much better is a new application with respect to the others?*

We consider that, given a new application (or a new version of an existing one), it is useful for developers to know how close or far is the new application to the others in the same category, considering the power consumption and network usage. For the new application, we define a similar typical usage scenario and we collect the power consumption and network usage while the application is running. After that, it is compared to the optimal applications in the same category using the reference values shown in Table IV. Here, for each category, the optimal, median, and worst values for power consumption, data usage, and rating are shown. In addition, using histograms, developers can visualize, in a graphical way, the distribution of power consumption and network usage of applications belonging to the same category. It allows developers to know how their new application is positioned with respect to the others.

Let us suppose that a new Android weather application has been developed. We play the scenario associated to the *Weather* category and we collect and calculate the average power consumption and network usage while the application is running. Let us consider that the associated power consumption and data usage are 2.50W and 0.05MB, respectively, for this new application. Using the reference values given in Table IV we can know that the power consumption associated to the application is better than the best existing value in this category. In addition, the network usage of the new application does not improve the best existing value, but it is better than the median. Fig. 10 and Fig. 11 show the histograms for power consumption and network usage of applications in the *Weather* category, respectively (where the red bar represents the new application). Analyzing the histograms, we confirm that the

TABLE IV
REFERENCE VALUES FOR DIFFERENT CATEGORIES OF ANDROID APPLICATIONS

Category	Power (in W)			Network (in MB)			Rating		
	Optimal	Median	Worst	Optimal	Median	Worst	Optimal	Median	Worst
Browsers	2.86	2.97	3.03	0.24	0.59	0.61	4.60	4.55	4.40
Cameras	3.16	3.56	4.66	0.00	0.00	1.63	4.50	4.30	4.10
Emails	0.98	1.19	1.41	12.00	12.22	12.45	4.30	4.30	4.30
Flash Lights	0.68	0.79	1.36	0.00	0.00	0.04	4.70	4.40	3.80
Music Players	1.67	2.82	3.06	0.00	0.01	0.40	4.56	4.50	4.38
News	2.78	2.83	2.88	0.08	0.16	0.24	4.50	4.30	4.10
Video Players	2.97	3.19	3.29	0.00	0.00	0.05	4.50	4.45	4.20
Weather	2.61	2.84	3.02	0.02	0.27	0.91	4.50	4.30	4.10

new application has a better power consumption because it is the only one with an average power lower than 2.6W. Considering the network usage, we conclude that the new application is close to the median but it is the second best application in the category with respect to data usage.

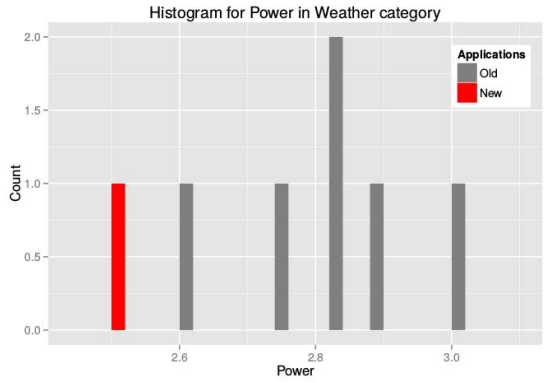

Fig. 10. Distribution of power consumption for optimal *Weather* applications.

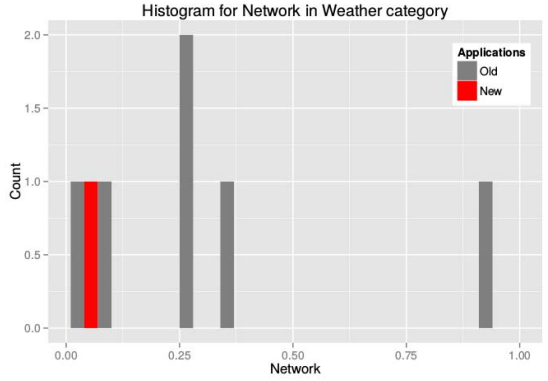

Fig. 11. Distribution of network usage for optimal *Weather* applications.

We conclude that, for a given category, it is possible to measure how far is a new application with respect to the optimal applications found by ADAGO in that category.

VI. THREATS TO VALIDITY

To reduce the threats to validity and allow replicate this study, all the content used is available in our replication package[10]. It is important to notice that the same model of phone and version of Android operating system should be used to replicate the study. In addition, considering the scenarios collected for each application, they are only valid for the *apk* versions used in this study, which are also available in our replication package. The reason is that the scenarios were collected considering approaches based on absolute coordinates and not on the identifier of components in the graphical user interface (GUI). Therefore, if another model of phone is used or the application was updated and the GUI changed, the scenarios will not be valid. In addition, we tried to wait the same quantity of time between different user actions when the scenarios were defined. Because they were collected manually, these intervals of time are not exactly the same for all the applications in the same category. Anyway, this fact only affects the application execution time and the influence over the average power consumption is negligible.

Threats to *construct validity* concern relationship between theory and observation, and to which extent what is measured is actually what it is claim to be measured. We used the same phone model used in other papers. Plus our measurement apparatus has a higher or the same number of sampling bits as previous studies and our sampling frequency is one order of magnitude higher than past studies. Overall, we believe our measurements are more precise or at least as precise as similar previous studies. As in most previous studies we cannot exclude the impact of the operating system. What is measured is a mix of Android and application actions. We mitigate this by running the application multiple times; resetting the environment before each run; and using the same setting and scenario for different applications in the same category. *Monkey* was running on the controller desktop sending events to the phone through the USB connection, and it introduces an extra energy consumption. This fact was checked in [9] and the authors reported that it was not noticeable in comparison to the total energy consumption of the application. We extend this to *tcpdump*, the tool which is used to collect information related to the network usage. In a similar way, data of network usages associated to each application may also be affected by the operating system, because *tcpdump* captures all the packets

[10]http://ser.soccerlab.polymtl.ca/ser-repos/public/tr-data/adago.tar.gz

transmitted over the network interface. Given that this tool is only used while the application is running, we suppose that all the network usages correspond to the application but Android could have used the network connection *e.g.,* to check for updates. To minimize this risk, before the experiments, all the processes that were not needed were killed and we performed several runs for each application.

Threats to *internal validity* concern factors, internal to our study, that could have influenced the results. We computed the energy using well know theory; scenarios were replicated several time to ensure statistical validity. As explained in the *construct validity* our measurement apparatus is at least as precise as previous measurement setup. Furthermore, we are not interested in the absolute values of energy or data overhead rather in the comparison of values obtained by runs of different applications.

Threats to *conclusion validity* concern the relationship between experimentation and outcome. While part of the analyses are supported by appropriate statistical procedures, other findings mainly have a qualitative nature (*e.g.,* $\mathbf{RQ}_3$), hence no statistical procedure is used. Clearly, as part of future works there is the need to perform a more extensive validation to really assess ADAGO usefulness. One major concern is whether or not the scenarios, and thus the measurement data, are representative of the overall application behavior. We are aware that different features have different energy and (possibly) data transfer characteristic. In a nutshell, we cannot claim the most frugal applications identified with our scenarios are the absolute best. However, our goal is to show ADAGO feasibility and support the evidence of its usefulness. Future work will be needed to extend the set of measured features and consider other environment setup. For example, the effect of Wifi connection versus cellular phone networks.

Threats to *external validity* concern the generalization of our findings. Admittedly, the study is limited to eight categories and 144 applications, which had the highest number of downloads in the *Play Store*. Although we are aware that further studies are needed to support our findings, our investigation was, intendedly, relatively limited in size to allow us to complement the quantitative analysis with a qualitative analysis of collected data and findings. External validity threats do not only apply only to the limited number of apps, but also to the way they have been selected (the top rated ones), their types (only free apps), and provenance (one app store). For this reason this work is susceptible to the App Sampling Problem [12], which exists when only a subset of apps are studied, resulting in potential sampling bias.

VII. Related Work

There has been a recent surge in the number of studies related to the Android market, energy consumption (of mobile applications), privacy concerns, application monitoring, and application analysis. The earliest works are focused on modeling, monitoring, and improving energy consumption at the hardware level (e.g., [13]). A detailed survey of energy measurement works is presented in [2]. Later, attention has shifted to techniques that model, monitor, and improve energy consumption at the level of the operating system and applications [14]. The energy consumption of Android applications is estimated at instruction level in [15], using a Low Power Energy Aware Processing measurement device (AtomLEAP). Regarding the same measurement device, the energy consumption at code line level is measured in [16] and the API level energy consumption patterns of 405 mobile applications is studied in [9]. Several previous works (see [9], [15], [16]) have adapted Ball-Larus path enumeration technique [17] to obtain statement level execution details. Researchers have also recognized the importance of threads, methods and procedures [3]. In this paper, as in many previous works such as [1], [5], we are not interested to the instruction or method energy level but the overall scenario of the energy consumption.

Early works were directed to improve code practices and provide users with guidelines to extend battery life, see for example [4]. In [18], authors mined Android energy-greedy API usage patterns. The idea is to make developers aware of which components and API is more energy efficient. Similarly, a set of recommendations to support developers in coding more energy aware applications is developed in [19]. A set of 21 Android applications is studied in [5], and results show that the use of ads in applications leads to an increase of energy consumption and network traffic. In [20] authors proposed an automated test generation frame- work that detects energy hotspots/bugs in Android applications. The effects of code obfuscations on the amount of energy consumed by executing different usage scenarios spread across 11 Android applications is analysed in [21]. Recently, compiler optimization techniques have been used (see [22]) to reduce design patterns energy consumption.

Nowadays, because the number of applications available increases, the users of mobile devices find it challenging to search new and relevant applications. A context-aware recommender system for mobile applications which produces recommendations from the first use is presented in [23].

We share with previous works the idea of supporting both users and developers. In this way we are closer to [1], [24], where screen colors are tuned to save energy. However, we have a different formalization and we tackle a totally different problem with different variables and objectives.

VIII. Conclusion

Different applications implementing similar or identical functionalities may have different power consumptions and network usages, but this kind of information is almost always unknown to users and developers. We proposed ADAGO, a recommendation system aimed at helping both users and developers. We help users to choose optimal sets of applications belonging to different categories and we also help developers to understand the relative placement of their application's efficiency with respect to others.

ADAGO models the application selection problem as a multi-objective optimization problem, and it aims at minimizing power consumption and data usage while maximizing the

application rating.

We investigated ADAGO applicability and usefulness and reported findings in the paper. Out of eight application categories and a set of 144 applications ADAGO was able to outperform in, most of the cases, the manual application selection solely based on application rating ($\mathbf{RQ}_1$). In other words, selecting an application on the ground of application rating does not guarantee to install the most frugal application. Nevertheless, in three cases out of eight selecting an application solely based on its rating gave an optimal application. However, that solution was one among many; other solutions could have preferred by the user due to either lower energy or lower data overhead consumption.

Indeed, in $\mathbf{RQ}_2$ we show that different compromises are feasible. Furthermore, if the user is willing to accept a lower rating, as lower as about 6%-7%, much bigger energy and data overhead savings are possible.

We conclude that, from the user perspective, choosing optimal sets of applications, power consumption and network usage can be reduced by 16.61% and 40.17%, respectively, in comparison to choosing the set of applications that maximizes only the rating. In addition, in $\mathbf{RQ}_3$, we show that it is possible to help developers understanding how far is a new Android application with respect to optimal applications in the same category, taking into account power consumption and network usage.

ACKNOWLEDGMENT

The authors would like to thank Francis Rivest for his valuable help. We also thank the Electrical Engineering department of Polytechnique Montreal for sharing their resources. This work has been partially funded by the Spanish MINECO and FEDER project TIN2014-57341-R (http://moveon.lcc.uma.es).

REFERENCES

[1] M. Linares-Vásquez, G. Bavota, C. E. B. Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Optimizing energy consumption of guis in android apps: A multi-objective approach," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: ACM, 2015, pp. 143–154. [Online]. Available: http://doi.acm.org/10.1145/2786805.2786847

[2] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 12–21.

[3] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168841

[4] D. Li and W. G. J. Halfond, "An investigation into energy-saving programming practices for android smartphone app development," in *Proceedings of the 3rd International Workshop on Green and Sustainable Software*, ser. GREENS 2014. New York, NY, USA: ACM, 2014, pp. 46–53.

[5] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond, "Truth in Advertising: The Hidden Cost of Mobile Ads for Software Developers," in *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, May 2015.

[6] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers, 1999.

[7] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[8] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[9] D. Li, S. Hao, J. Gui, and W. Halfond, "An empirical study of the energy consumption of android applications," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, Sept 2014, pp. 121–130.

[10] J. Durillo and A. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[11] "Online appendix for paper 'optimizing user experience in choosing android application'." [Online]. Available: http://ser.soccerlab.polymtl.ca/ser-repos/public/tr-data/adago-report.pdf

[12] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, ser. MSR '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 123–133. [Online]. Available: http://dl.acm.org/citation.cfm?id=2820518.2820535

[13] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, June 2000, pp. 83–94.

[14] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphones using kernel activity monitoring," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 36–36. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342821.2342857

[15] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 92–101.

[16] D. Li, S. Hao, W. G. J. Halfond, and R. Govindan, "Calculating source line level energy information for android applications," in *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, ser. ISSTA 2013. New York, NY, USA: ACM, 2013, pp. 78–89. [Online]. Available: http://doi.acm.org/10.1145/2483760.2483780

[17] T. Ball and J. R. Larus, "Efficient path profiling," in *Proceedings of the 29th Annual ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO 29. Washington, DC, USA: IEEE Computer Society, 1996, pp. 46–57. [Online]. Available: http://dl.acm.org/citation.cfm?id=243846.243857

[18] M. L. Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. D. Penta, and D. Poshyvanyk, "Mining energy-greedy api usage patterns in android apps: an empirical study," in *MSR*, 2014, pp. 2–11.

[19] I. L. M. Gutiérrez, L. L. Pollock, and J. Clause, "Seeds: a software engineer's energy-optimization decision support framework," in *ICSE*, 2014, pp. 503–514.

[20] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, "Detecting energy bugs and hotspots in mobile apps," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 588–598. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635871

[21] C. Sahin, P. Tornquist, R. Mckenna, Z. Pearson, and J. Clause, "How Does Code Obfuscation Impact Energy Usage?" in *ICSME'14*, 2014, pp. 131–140.

[22] A. Noureddine and A. Rajan, "Optimising energy consumption of design patterns," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, 2015, pp. 623–626. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2015.208

[23] C. Davidsson and S. Moritz, "Utilizing implicit feedback and context to recommend mobile applications from first use," in *Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation*, ser. CaRR '11. New York, NY, USA: ACM, 2011, pp. 19–22. [Online]. Available: http://doi.acm.org/10.1145/1961634.1961639

[24] D. Li, A. H. Tran, and W. G. J. Halfond, "Making web applications more energy efficient for oled smartphones," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 527–538. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568321