Optimizing Energy Consumption of GUIs in Android Apps: A Multi-objective Approach

Mario Linares-Vásquez[1], Gabriele Bavota[2], Carlos Bernal-Cárdenas[1]
Rocco Oliveto[3], Massimiliano Di Penta[4], Denys Poshyvanyk[1]
[1]The College of William and Mary, Williamsburg, VA, USA — [2]Free University of Bozen, Bolzano, Italy
[3]University of Molise, Pesche (IS), Italy — [4]University of Sannio, Benevento, Italy

ABSTRACT

The wide diffusion of mobile devices has motivated research towards optimizing energy consumption of software systems—including apps—targeting such devices. Besides efforts aimed at dealing with various kinds of energy bugs, the adoption of Organic Light-Emitting Diode (OLED) screens has motivated research towards reducing energy consumption by choosing an appropriate color palette. Whilst past research in this area aimed at optimizing energy while keeping an acceptable level of contrast, this paper proposes an approach, named GEMMA (Gui Energy Multi-objective optiMization for Android apps), for generating color palettes using a multi-objective optimization technique, which produces color solutions optimizing energy consumption and contrast while using consistent colors with respect to the original color palette. An empirical evaluation that we performed on 25 Android apps demonstrates not only significant improvements in terms of the three different objectives, but also confirmed that in most cases users still perceived the choices of colors as attractive. Finally, for several apps we interviewed the original developers, who in some cases expressed the intent to adopt the proposed choice of color palette, whereas in other cases pointed out directions for future improvements.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

User interfaces

Keywords

Energy consumption, Mobile applications, Empirical Study

1. INTRODUCTION

The impressive adoption and inclusion of mobile devices and apps in daily activities has motivated the need for reducing their energy consumption. While significant efforts have been aimed at engineering energy-friendly hardware components in mobile devices, some recent research has also focused on energy-aware development practices for reducing the energy consumption in mobile apps. For instance, common energy bugs in mobile apps have been identified and catalogued [27, 28, 32, 34, 42], as well as typical hot spots [39] together with energy greedy APIs [26,33]. In addition, several infrastructures and methods have been proposed to measure and estimate the energy consumption of mobile devices and apps [16, 18, 23, 29].

Some practices for avoiding and fixing energy hotspots (bugs) in mobile apps focus on how the apps should use energy-greedy hardware components in the device, such as GPS, Wi-Fi, or the screen. In the case of LCD displays, the energy drawn from the battery is constant regardless of the colors displayed on the screen. However, this is not the case for OLED displays, for which the energy consumption depends on the combinations of colors at the sub-pixel level. This property of OLED displays motivated the adoption of power models for estimating the energy drawn by the graphical user interfaces (GUI) displayed on the screen. In fact, previous work have used power models to estimate and improve the energy consumption of web browsers [11], mobile web apps [24], and mobile apps in general [12, 39]. All these prior techniques on improving energy consumption of GUIs running on OLED devices have been mostly driven by a single objective to reduce energy consumption at all costs. Also, these approaches generally exploit only a small subset of possible color schemas one can define to optimize the GUI. For example, using predefined themes or using color transformations starting only from the original scheme can drastically reduce the color palettes and compositions that can be explored as a potential solution [12]. This is also the case when the proposed solution uses by default dark colors in the background and generates the composition using the background as a reference [24, 39]. Another issue with previous approaches is that the solutions are generated individually for each GUI in the app, which can leed to inconsistencies in the whole design concept [24, 39].

In this paper we propose a multi-objective approach, namely GEMMA[1], for generating color compositions that reduce the energy consumption of GUIs in Android apps and are visually attractive at the same time. GEMMA combines power models, pixel-based engineering, color theory, dynamic analysis, and a multi-objective optimization technique—namely Non-dominated Sorting Genetic Algorithm (NSGA)-II [10]—to produce a Pareto-optimal set of design solutions (*i.e.,* GUI

[1]**G**ui **E**nergy **M**ulti-objective opti**M**ization for **A**ndroid apps

color compositions) across three different objectives: (i) reducing energy consumption, (ii) increasing contrast, and (iii) improving the attractiveness of the chosen colors by keeping the palette close to the original one.

To the best of our knowledge GEMMA introduces the following unique advantages:

- *Is multi-objective.* This is the first approach adopting multi-objective optimization to choose colors for mobile apps with the main goal of reducing energy consumption. Multi-objective optimization avoids a direct aggregation of different, potentially conflicting objectives, and, as it will be shown in our empirical evaluation, it allows developers to evaluate different possible solutions that optimize specific objectives or achieve a compromise between two or three of them;
- *Accounts for multiple screens and the duration of time they are displayed.* Since a GUI consists of multiple screens, it is particularly important to optimize colors for screens that are used longer in typical usage scenarios, while screens that just appear for few moments may have lesser contribution to the overall energy consumption. This is accounted for in the GEMMA's energy objective function;
- *Produces pleasant and consistent color combinations.* In addition to energy consumption and contrast, this is the first approach taking GUI design principles—at least in terms of choosing colors—into account. First, for the color palette generation we combine the original color scheme in an app with three models based on color theory harmonies. Then, GEMMA generates compositions such that if two or more components have a given color (say yellow) in the original palette, they will appear with the same color (say blue) in the proposed solutions, so to avoid producing inconsistent GUIs. Finally, as mentioned before, the multi-objective optimization has a third, "design-related" objective aimed at choosing colors that are not too distant from the original ones.

To validate the quality of the color schemas recommended by GEMMA, we used it to generate optimized GUIs for 20 free apps available in Google Play and for five commercial apps developed by Italian companies. The colorfulness of the GUIs derived for the 25 apps was evaluated by 85 mobile apps users in an online survey; also, we interviewed developers and managers from Italian companies to gather their feedback concerning the GUIs generated by GEMMA for their apps. Overall, GEMMA has been evaluated across three dimensions: (i) the ability to optimize the three objectives, (ii) the extent to which potential users consider the choices of colors acceptable enough, and (iii) the extent to which the original developers of commercial apps would be willing to account for GEMMA's recommendations.

2. OPTIMIZING ENERGY CONSUMPTION OF GUIS WITH GEMMA

This section describes GEMMA, a multi-objective search-based optimization technique that suggests possible color alternatives achieving a tradeoff between three desirable and possibly conflicting objectives: (i) reducing energy consumption on OLED displays; (ii) increasing contrast between adjacent GUI elements; and (iii) preserving a consistency in the color usage with respect to the original design (*i.e.,* proposing color schemas that are close to the one adopted by developers in the original app's GUI). In addition to nearly optimizing the objectives mentioned above, GEMMA can also verify that the proposed GUIs satisfy some specified constraints. For instance, the current implementation of GEMMA does not consider GUIs' color schemas as valid in which adjacent elements have the same color or colors having a very low contrast between them, in order to promote the readability of the recommended GUIs.

To solve the optimization problem, first, we obtain a power model of the corresponding display, relating pixel colors to power consumption (Section 2.1). Then, we identify GUI elements composing each screen (Section 2.2). Finally, we define a multi objective Genetic Algorithm (GA) aimed at finding near-optimal solutions to the stated problem (Section 2.3).

2.1 Estimating the Power Consumption

The power consumption of OLED screens can be estimated by using the color components of each pixel [11, 12, 24, 39]. According to the specific pixel matrix of the screen, each color in a pixel is rendered by using a combination of *red*, *green*, and *blue* sub-pixels at different levels (*a.k.a.*, intensity) in the standard RGB (sRGB) color model. In the case of OLED screens the power consumption of sub-pixels depends on the color component level. The power consumption of an OLED pixel is a linear function of its linear RGB values, *i.e.,* the gamma decoding result from its values in the standard RGB space [11, 12]. Therefore, given a power profile $P_{\langle color \rangle}(level)$ that returns the power consumption of a color level, the power consumption of a *pixel* in the position $\langle x, y \rangle$ with $color_{x,y}$ and components *Red*, *Green*, and *Blue* is estimated as in Equation 1, and the total power consumed by the screen when painting a GUI is the sum of pixel power consumption over all the $X \times Y$ pixels in the screen (Equation 2).

$$P(color_{x,y}) = P_{\langle R \rangle}(R_{x,y}) + P_{\langle G \rangle}(G_{x,y}) + P_{\langle B \rangle}(B_{x,y}) \quad (1)$$

$$TP(GUI) = \sum_{x=1}^{X} \sum_{y=1}^{Y} P(color_{x,y}) \quad (2)$$

The power profile of the color components represented by the $P_{\langle color \rangle}$ functions is screen specific. In fact, this characteristic of the OLED screen has been exploited before to build power models that estimate the consumption of GUIs [11, 12, 24, 39]. In our case we built the power model for a SUPER AMOLED screen (1080 × 1920 pixels) in a Samsung Galaxy S4, by using a monsoon power monitor [30]. To define each $P_{\langle color \rangle}$ function, we measured the current drawn by the screen when painting it with all the pixels set to black (*i.e., idle* period) during 30 seconds, and full screen with all the pixels set to the same primary color component for each $level \in [1, 255]$ during 30 seconds (*i.e., measurement* period). We are not considering the effect of the screen brightness, consequently, we set the display brightness to the maximum value. For both, idle and measurement periods we computed the average current of a pixel dividing the raw value, measured with the power monitor, by the total number of pixels in the screen (1080 × 1920). After collecting all the values, we subtracted the current in $idle_k$ from the current in the corresponding $measurement_k$ to account only for the current drawn by the screen pixels. Finally, we removed noise by using the Tukey's smoother implemented in *R* [37].

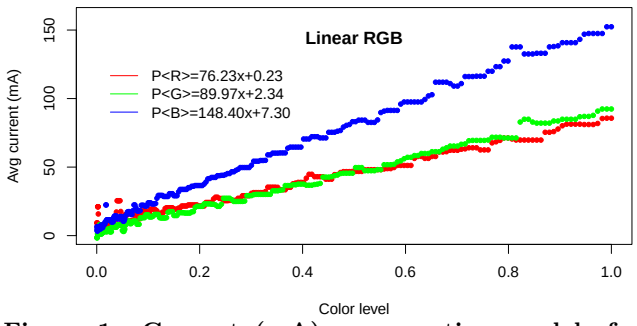

Figure 1: Current (mA) consumption models for primary colors of the Galaxy S4 SUPER AMOLED screen in linear RGB color space.

To derive the power model, we followed the same procedure previously adopted in [11, 12]. With the 255 measurements representing the power profile of a primary color in sRGB, we transformed the intensities to linear RGB, and then we found the linear function describing the power consumption (Fig. 1). The linear RGB simplifies the construction of the model without loss of information [11]. With the power models using as input the color intensities in linear RGB, it is possible to estimate the current drawn by all the pixels in the Galaxy S4 AMOLED screen when a specific GUI is painted, by using Equation 2. The model in Fig. 1 shows current measurements in mA (milli Amperes); the current is proportional to power (Watts) because the voltage is constant during the measurements; thus, there is no need to plot the results in terms of Watts. These measurements are consistent with the ones by Dong *et al.* [11, 12] for OLED screens.

2.2 Extracting Color Composition from GUIs

Using the power model to estimate the power consumption of GUIs requires extracting pixel-based representations of the GUIs. This extraction can be done online when the target GUIs are displayed on the device's screen [11], or offline by relying on screenshots [39] or server-side code [24]. The representation is usually the GUI-based color histogram or a set of tuples ⟨pixel, red-level, green-level, blue-level⟩. GEMMA relies on dynamic analysis of an app to extract GUI-based information, since the GUI is shown on the device (or emulator), and the Android *View Server* running on the device can be queried to collect features of the GUI components such as location, dimensions, text, and component's hierarchy. With this information available, the extraction can be focused on detecting the color scheme (*i.e.,* individual colors and composition) instead of detecting shapes representing GUI components. Moreover, the information provided by the *View Server* allows us to distinguish components with images, components with/without text, and containers. This is particularly important to avoid transformation/distortion of images/logos that belong to the GUI or even to third-parties.

In terms of representation, GEMMA extracts bag-of-color-pixels (BOCP) and bag-of-color-components (BOCC) of Android app GUIs. A BOCP is a collection of pixels with the same color in the GUI, and a BOCC is a collection of components having pixels with the same color in the GUI. The BOCPs/BOCCs extraction procedure starts with getting a snapshot of the current GUI in the device, and the components in the GUI, by querying the *View Server* with the *UI Automator utility* in the Android SDK. The snapshot contains an XML-based representation of the GUI hierarchy (including location, type, dimensions, and text for each component), and a screenshot of the GUI. To identify the representative colors of each component we used a clustering-based quantization on the color histogram of the component. Only components different from `ImageView` and `ImageButton` were considered. Pixels belonging to each component are detected by traversing the GUI hierarchy in a bottom-up fashion, *i.e.,* deepest level is processed first. A pixel in the screen is assigned to only one component, and the priority for this assignment is based on the depth of the GUI hierarchy. These heuristics allow assigning pixels according to the z-index in the GUI; if there are components inside a container, the pixels are first assigned to the components, and then, the leftover pixels are assigned to the container. The set of pixels assigned to a component/container are then used to derive a color histogram.

Top-k colors (*i.e.,* centroids) are detected in the histogram of each component while avoiding gradients and font shadows. For that, GEMMA traverses the sorted histogram—from high to low frequency— looking for changes in the contrast ratio [38] higher than a threshold r. For instance, let us assume a text edit component with dark gray gradient (background) and white fonts; a lot of "grays" will surface at the beginning of the sorted histogram, and then white will appear. However, the contrast ratio between the grays is going to be low (close to one), and the ratio between grays and white is going to be higher. Thus, the top-1 color in the histogram is the centroid for the first cluster, then a new cluster is detected when the contrast ratio is higher than r. This procedure is applied until k centroids are detected. The values we adopted for k and r are reported in Section 3.3.

Finally, the colors of the GUI components are discretized (*i.e.,* color quantization) using the color centroids in the histogram. It means that the pixels are assigned the closest quantized color (*i.e.,* color centroid) to the original pixel color. The BOCPs is a hash-map structure for all the pixels in the GUI, in which the key is a quantized color $color^*$ and the value is a list of pixels assigned to that color. The BOCCs is a hash-map too, in which the key is a quantized color $color^*$ and the value is a set of the components associated with the pixels in BOCPs[$color^*$].

2.3 Multi-objective Optimization Model

In the following, we describe the GA component in GEMMA. The values we use for the GA parameters are reported in Section 3.3. The GA has been implemented using *jMetal* [13].

Representation. Our representation contains a gene for each BOCP, *i.e.,* each gene will define the color to all the pixels assigned to that BOCP. We indicate a generic solution of our optimization problem as S, and with $S[i]$ we denote the color this solution assigns to the *i-th* BOCP.

Multi-Objective Genetic Algorithm. Rather than evaluating each solution according to a single fitness function, a multi-objective GA produces *frontiers* composed of *Pareto-optimal* solutions. A solution X is said to be *Pareto-optimal* if-and-only-if it is non-dominated by any other solution within the search space, *i.e.,* if-and-only-if no other solution Y exists which would improve one of the objective functions, without worsening other objectives. All the solutions that are not dominated by any other solution are said to form a *Pareto-optimal set*, while the corresponding *objective vectors* (containing the values of the objective functions) are said to form a *Pareto frontier*. Identifying a Pareto

frontier is useful because the software engineer can use the frontier to make a well-informed decision that balances the trade-offs between the different objectives. In our context, one could select color choices achieving the lowest energy consumption, having the highest contrast, being the closest to the original design, or a compromise among these objectives. The specific multi-objective GA we use for GEMMA is the Non-dominated Sorting Genetic Algorithm(NSGA)-II [10]. NSGA-II tries to ensure diversity in the evolving populations to avoid the situation where populations have been filled only with dominating solutions (because of the elitism effect, *i.e.,* best solutions are preserved).

Initial Population Generation. The initial population of each GA execution uses a color palette composed of 512 colors, including m colors from the BOCPs, white, black, a palette in equidistant color harmony with $(512 - m - 2)/3$ colors, a monochromatic palette of $(512 - m - 2)/3$ colors with difference in saturation and brightness, and the equidistant harmony palette, but with random saturation and brightness for each of the colors. The starting point of the equidistant harmony and monochromatic palette (*i.e.,* initial values for the tones) are randomly selected. Our choice for a diverse palette that include equidistant harmonies and monochromatic scale is to provide the GA with colors that can drive it to producing visually appealing GUIs.

Equidistant harmony is the rule for color composition in the HSB/HSV (Hue Saturation Brightness/Value) color space that derives a color palette with the same saturation and brightness, but modulating the tone (*i.e.,* hue) by taking proportional spaces in the HSB/HSV color wheel. Because the tone in the HSB/HSV space varies from 0°to 359°, and each integer value corresponds to one color (*e.g.,* 0° is red, 120° is green, and 240° is blue), equidistant colors can be derived by dividing the hue space in a target number of colors. For instance, an equidistant harmony of 20 colors, *i.e.,* a difference of (360/20)°between each color, starting at 10° and fixed values for saturation S and brightness B would be $\langle 10°, S, B\rangle$,$\langle 28°, S, B\rangle$,$\langle 46°, S, B\rangle$,...,$\langle 334°, S, B\rangle$, $\langle 352°, S, B\rangle$. Equidistant harmonies can also derive into palettes with value and saturation contrast by changing randomly the saturation and brightness of the colors in the original equidistant harmony. Monochromatic palettes can be generated by using the same hue but varying the saturation and brightness randomly. In the three composition schemes we used a brightness range [0.2, 0.9] to avoid colors that are too bright or too dark.

Genetic Operators. In order to evolve GA individuals, we use *selection*, *crossover* and *mutation* operators. The *crossover* operator is the *one-point* crossover. Given two *parents*, a one-point crossover cuts them—*i.e.,* the array of BOCP—in a random position p, and then all genes after position p are exchanged to produce the offspring. Crossover is applied to individuals of the population with a probability p_{cross}. The *mutation* operator is the uniform bit-flip mutation that, with probability p_{mut}, changes the color for one BOCP to a different color randomly selected from the palette. The *selection* operator is the binary tournament selection operator described by Deb *et al.* for NSGA-II [10]. A binary tournament selection holds a competition between two solutions, selecting the firstly ranked one with a probability p, and the second one with probability $p \cdot (1 - p)$.

Fitness functions and constraints. Let us consider an app having ns screens SCR_w with $w = 1, 2, \ldots, ns$. For each of these screens, let us assume that it has been estimated that, on average, the *h-th* screen is being displayed for a percentage pt_h of the total application usage time. Such estimates can be obtained by profiling application usages, and are needed so to prioritize low energy consumption on screens that are displayed for a long duration of the overall usage time. That is, having bright colors on a splash screen appearing for a few seconds is not as crucial as having them on the main application GUI. Now, let us also assume that each screen has a $X \times Y$ resolution, and that, given a pixel in position $\langle x, y\rangle$ of the screen h, the function $B(x, y, h)$ returns the index of the BOCP that the pixel belongs to. Therefore, given a GA individual solution S, the color that such a solution assigns to a pixel $\langle x, y\rangle$ in screen h is $S[B(x, y, h)]$.

The first fitness function (to be minimized) is the *Energy Consumption Fitness (ECF)*. Given a solution S, it computes its estimated consumption per unit of time based on Equation 1, as in the following:

$$ECF(S) = \sum_{h=1}^{ns} p_h \cdot \sum_{x=1}^{X} \sum_{y=1}^{Y} P(S[B(x, y, h)]) \quad (3)$$

where $ECF(S)$ sums the estimated power consumptions of all the pixels in all the screens, weighting them for the estimated fraction of time p_h for a particular screen in use.

The second fitness function (to be maximized) is the *contrast fitness (CF)*, which relies on the GUI model extracted as explained in Section 2.2. Let us consider that each screen h is composed of nc_h components $C_{i,h}$, the $Adj(C_{i,h})$ function returns the set of adjacent components to $C_{i,h}$, and $BC(C_{i,j})$ returns the bag of pixels to which the component belongs.

$$TCon(C_{i,h}) = \sum_{C_{j,h} \in Adj(C_{i,h})} Con\left(S[BC(C_{i,h})], S[BC(C_{j,h})]\right) \quad (4)$$

$$CF(S) = \sum_{h} \sum_{i=1}^{nc_h} TCon(C_{i,h}) \quad (5)$$

That is, $CF(S)$ sums, for each screen the contrast between each component $C_{i,h}$ in the screen h and all its adjacent components $C_{j,h} \in Adj(C_{i,h})$, using the function $Con(a, b)$, that returns the contrast between two colors a and b using a formula defined by W3C [38]:

$$Con(a, b) = \left| \frac{299 \cdot red_a + 587 \cdot green_a + 114 \cdot blue_a}{1000} - \frac{299 \cdot red_b + 587 \cdot green_b + 114 \cdot blue_b}{1000} \right| \quad (6)$$

The third fitness function (to be minimized) is the *design fitness (DF)* and it aims at producing GUIs with a set of colors that are close to the ones used in the original design. Basically, while it can be acceptable to swap colors with respect to the original design—*e.g.,* to use the darker color for the background and the brighter color for the text or for small components—or to have relatively similar colors, changing completely the palette might result in a GUI with a choice of colors deviating drastically from the designer's intention. For this reason, the idea behind the third fitness function for a solution S is to use the distance between each color $S[i]$ in S and the closest color $Or[j]$ in the original palette. To compute the distance between two colors a and b, we use, according to existing literature [36], the Euclidean distance along the green, red, and blue components of the color. Given $d(a, b)$ the distance between two colors a and b, and $Or^*(color)$ is a function that, given a color, returns the closest color in the original palette:

$$\alpha(color) = \begin{cases} 1 & \text{if } BOCCs[color] \neq BOCCs\,[Or^*(color)] \\ 2 & \text{otherwise} \end{cases} \quad (7)$$

$$DF(S) = \sum_{i=1}^{n} \alpha(S[i]) \cdot d(S[i], Or^*(S[i])) \quad (8)$$

Namely, for each color in $S[i]$, *DF(S)* sums the distance to the closest color $Or^*(S[i])$ according to a penalization factor (α) defined by Equation 7. The rationale here is that we want to penalize twice color differences when they occur on the same components (*i.e.,* same BOCC), though this can be a conflicting objective with respect to the *ECF(S)*. Therefore, if the color $S[i]$ is used in S for the same set of components with color $Or^*(S[i])$ in the original design, the distance is penalized twice.

Besides the three fitness functions, we also define a constraint aimed at avoiding solutions with low contrast. Specifically, we consider one constraint violation every time there is a pair of adjacent components ($C_{i,h}$, $C_{j,h}$) when:

$$Con(S[BC(C_{i,h})], S[BC(C_{j,h})]) < CnTh \quad (9)$$

where $CnTh$ is a contrast threshold. Constraints are handled using the mechanism defined in NSGA-II [10]. Constraint violations influence the binary tournament selection, and this is done using an additional domination principle named constraint domination. Namely, (i) feasible solutions (not violating any constraints) are ranked better than infeasible ones, (ii) feasible solutions are ranked in terms of the objectives dominance, and (iii) infeasible solutions are ranked based on the number of violated constraints. As a consequence of the latter point, solutions violating constraints are not necessarily discarded, but, for the sake of diversity, they survive. Clearly, once the resulting Pareto front has been obtained when the evolutionary algorithm stops, solutions violating constraints are discarded in case they are still present.

3. EMPIRICAL STUDY DESIGN

The *goal* of the study is to evaluate GEMMA in terms of (i) the energy savings that could be achieved by running Android apps when adopting the GUI color design recommended by GEMMA; (ii) the colorfulness of the GUIs that GEMMA produces as assessed by mobile apps users; and (iii) GEMMA's suitability in an industrial context, when applied to minimize GUI energy consumption of existing commercial apps. The *context* of the study consists of 25 apps from the Google Play and F-droid markets, 85 mobile app users, and three software companies (for a total of three project managers involved). The *quality focus* is the effectiveness of GEMMA in producing GUIs aimed at reducing the energy consumption of Android apps while also keeping a visually attractive color composition and ensuring sufficient contrast.

All the materials used in our study are publicly available in our replication package [25].

3.1 Research Questions

In the context of our study we formulated the following three research questions (RQ):

RQ$_1$: *To what extent is* GEMMA *able to optimize the GUI energy consumption, contrast, and design objectives?* This RQ aims at investigating the effectiveness of GEMMA in reducing the energy consumption of the apps' GUIs while keeping a high contrast between its underlying components and staying as close as possible to the original design. Such investigation is merely performed in terms of the metrics used to compute the fitness functions defined in Section 2.

RQ$_2$: *Are the color compositions generated by* GEMMA *visually attractive as perceived by Android users?* This RQ focuses on the colorfulness of the GUIs generated by GEMMA, as it is perceived by humans instead of being evaluated in terms of metrics. In visual aesthetics, colorfulness is the factor used to evaluate elements related to individual colors and compositions [31]. We do not focus on the whole concept of visual attractiveness; a complete appraisal of visual aesthetics should include other facets, such as simplicity, diversity, and craftsmanship that are related to the GUI layout and design concepts [31]. Therefore, we focus on the colorfulness only, because GEMMA only generates alternative color schemes and keeps the GUI layout intact.

RQ$_3$: *Would actual developers of mobile applications consider changing colors in an app as recommended by* GEMMA*?* For an approach like GEMMA a successful technological transfer is the main target objective. In the context of this research question we investigate the industrial applicability of GEMMA with the help of three software companies developing Android apps.

Note that our three research questions have a high degree of complementarity in GEMMA's evaluation. The first two research questions quantitatively analyze the efficacy of GEMMA in producing energy saving GUI color schemas while still keeping those GUIs visually attractive for the apps' users. These two aspects are then qualitatively investigated in an industrial context in our **RQ**$_3$.

3.2 Context Selection

The list of the 25 apps considered in our study is reported in our replication package [25]. It is worth noting that, while all 25 apps have been used in the context of the first two research questions, the investigation performed in the software companies (*i.e.,* **RQ**$_3$) is clearly limited to the set of five apps they developed and provided to us for the study. As for the remaining 20 apps considered in our study we targeted a diverse set of apps in terms of domain categories. Two of the authors inspected Google Play and F-droid to randomly select a set of native apps (*i.e.,* non-HTML based) to use in the context of our study. The selected set of apps cover ten categories (*e.g.,* productivity, weather, *etc.*).

In the context of our **RQ**$_2$, we involved 85 app users[2] asking for their opinion about the look and feel of the GUIs generated by GEMMA. Since **RQ**$_2$ only aims at assessing how visually attractive the generated color schemes are, the only requirement we had for participants is to have some basic experience with mobile apps.

As for **RQ**$_3$, we involved three Italian companies, namely GenialApps [2], Next [4] and IdeaSoftware [3], that provided us a total of five apps to use in the context of our study. All the companies have a multi-year experience in mobile apps development and each of them count between ten and 20 developers. As it will be detailed later, we performed semi-structured interviews with people from these companies with the aim of gathering qualitative feedback about the GUIs generated by GEMMA. In particular, two project managers (from GenialApps and Next) and one developer (from MediaStudio) participated in our study.

[2]Including 50 BS, 6 MS, and 14 PhD students; 14 industrial developers, 1 professional graphical designer and 6 others [25]

3.3 Data Collection

To answer our RQs we applied GEMMA on the 25 apps in order to generate color schemas for GUIs. As explained in Section 2, GEMMA is able to work on multiple GUIs of the same app. Since each of the 25 apps includes a specific number of different GUIs (or GUI states), for the sake of consistency in our data analysis we decided to consider three GUIs per app as the objective of GEMMA's optimization process. In particular, we always consider the main GUI of the app (*i.e.,* the one visualized when the app is launched on the device) plus two randomly selected GUI states.

We used the following settings for GEMMA[3]: *1) Extracting Color Composition from GUIs (Section 2.2)*: $k = 3$, to extract three different colors from each GUI component: background, text (or second main color), and the third color for the border. As for the contrast ratio r, we found 1.6 to be a common lower bound for the contrast between colors in borders and background as well as background and text in the Android apps considered in our study.

2) GA parameters (Section 2.3): $p_{cross} = 0.9$; $p_{mut} = 1/|BOCP|$, where $|BOCP|$ is the number of bags of pixels identified in a GUI. Such parameters were the default ones in *jMetal* [13]; we at least checked whether a lower crossover rate or higher mutation did not produce better results than ours. The other choices were *population size=50* and *GA number of evaluations=50,000* (larger populations and higher number of evaluations did not produce better results). Finally, we set the minimum contrast ratio between adjacent components ($CnTh$) to 4, as we found this value to be sufficient to ensure readability of a text over a background.

For each app (*i.e.,* for each of its three considered GUIs) we run GEMMA 30 times to account for the randomness of the multi-objective GA [8]. At each run we stored for each solution in the Pareto front respecting the contrast ratio constraint (see Section 2) as well as for the original color schema of the app: (i) its energy consumption (ECF function), (ii) its contrast ratio (CF function), and (iii) its "distance" from the original design (DF function)—see Section 2.

To answer $\mathbf{RQ_2}$ we designed an online survey, aiming at collecting the participants' opinion about the aspect of the GUIs generated by GEMMA. We wanted to verify if the energy saving GUIs generated by GEMMA are also attractive in terms of colorfulness. The first part of our survey included three questions aimed at gathering information about the background of the study's participants. The second part of the questionnaire asked participants for their feedback about the colorfulness of the original apps' GUIs as well as of those recommended by GEMMA. Since GEMMA provides as output a set of solutions (*i.e.,* all those in the Pareto front respecting the contrast ratio constraint) and that we run GEMMA 30 times on each app, we needed to make a selection of the recommended GUIs to be shown to the survey's participants. For each of the 25 apps involved, we showed to participants: (i) the original three selected GUIs; (ii) the GEMMA's solution having the lowest energy consumption across those generated in the 30 executions; and (iii) the GEMMA's solution having the median energy consumption across those generated in the 30 executions. Thus, for each app A, participants examined, in a single page of the survey, three sets of GUIs, each one containing a different color schema for the three GUIs selected for A. For each of them, participants had to answer the following questions from the colorfulness evaluation as suggested by Moshagen and Thielsch [31]: 1) *Is the color composition visually attractive?*; 2)*Do the colors match?*; 3) *Is the choice of colors botched?*; and 4) *Are the colors appealing?*

The possible answers for each question were presented with a four-point Likert scale: 1=absolutely not, 2=more no than yes, 3=more yes than no, 4=absolutely yes [31]. Each participant evaluated the colorfulness of five apps in order to reduce the participants' workload. Specifically, we created five groups of apps and we assigned round-robin each participant to a specific group aiming at balancing the number of evaluations for each app. Given the 85 participants to our study, each app has been evaluated by at least 15 participants. All GUIs for the same app were shown in the same page to allow an easy comparison between the different GUIs by participants. The order in which the apps were evaluated by the participants was randomized as well. Finally, note that none of the participants was aware of the experimental goals nor which of the GUIs in each page was the original one.

To address $\mathbf{RQ_3}$ we conducted semi-structured interviews with the project managers of the three involved companies. The interviews lasted for two hours with each company. Given that we had sufficient time available and only a few apps for each interview, for each app we showed the whole set of solutions provided by GEMMA in a single run, *i.e.,* the run among the 30 we executed that provided the largest set of solutions. This choice was made in order to have a larger set of alternative designs to discuss with the companies. We accompanied each of the shown solutions with information about its energy consumption and potential saving with respect to the original design.

During the interview, for each solution we asked the participant to answer the following question: "*Given the energy saving provided by this design with respect to the original design, would you adopt it in your app?*", using a score on a four-point Likert scale: 1=absolutely no, 2=no, 3=yes, 4=absolutely yes. In addition, we also asked questions to understand the principal reasons why the participants liked or not the suggested colors and which are the components for which the participants noticed a poor choice of colors. The interviews were conducted by one of the authors, who annotated the provided answers as well as additional insights about the GEMMA's points of strength and weakness that emerged during the interviews.

3.4 Data Analysis

To answer $\mathbf{RQ_1}$ we show boxplots of the energy consumption, the contrast ratio, and the distance from the original design (measured as reported in Section 2) obtained by running GEMMA 30 times on each app. We report such information for six categories of solutions: having the lowest and the median energy consumption, having the highest and the median contrast ratio, and having the lowest and the median distance from the original design (*i.e.,* six solutions for each run). We statistically compare the energy consumption and the contrast ratio of the generated solutions with those of the original design of the considered apps by using the Wilcoxon test [9]. The results are considered statistically significant at $\alpha = 0.05$. Since we perform multiple tests (the original GUI of each app is compared against different categories of

[3]Due to space limitations we explain how we chose the settings without reporting calibration details.

Table 1: ECF and CF: Wilcoxon test (adjusted p-values) and Cliff's delta (d).

Test	adj. p-value	d
ECF		
original design *vs* **lowest ECF**	<0.01	0.77 (Large)
original design *vs* **median ECF**	<0.01	0.57 (Large)
original design *vs* **highest CF**	<0.01	0.35 (Medium)
original design *vs* **median CF**	<0.01	0.58 (Large)
original design *vs* **lowest DF**	<0.01	0.49 (Large)
original design *vs* **median DF**	<0.01	0.57 (Large)
CF		
original design *vs* **lowest ECF**	<0.01	-0.07 (Negligible)
original design *vs* **median ECF**	<0.01	-0.35 (Medium)
original design *vs* **highest CF**	<0.01	-0.53 (Large)
original design *vs* **median CF**	<0.01	-0.34 (Medium)
original design *vs* **lowest DF**	<0.01	-0.25 (Small)
original design *vs* **median DF**	<0.01	-0.34 (Medium)

generated GUIs), we adjust our p-values using the Holm's correction procedure [19]. We also estimate the magnitude of the observed differences by using the Cliff's Delta (or d), a non-parametric effect size measure for ordinal data [15]. Cliff's d is considered negligible for $d < 0.10$ (positive as well as negative values), small for $0.10 \leq d < 0.33$, medium for $0.33 \leq d < 0.474$, and large for $d \geq 0.474$ [15].

As for $\mathbf{RQ}_2$, we must firstly note that our goal is not to determine whether GUIs recommended by GEMMA are considered more attractive than the original GUIs, but, rather, whether they are acceptable enough, and to know what is the "price to pay" in terms of visual aesthetics for the energy saving solutions provided by GEMMA. To this aim, we report boxplots of the ratings assigned by participants to the colorfulness of the different categories of GUIs. Also, we statistically compare the ratings assigned to the original GUIs of the 25 apps to those assigned to generated solutions with lowest and median energy consumption by performing the same statistical analysis used in $\mathbf{RQ}_1$.

Finally, for $\mathbf{RQ}_3$ we qualitatively discuss the outcomes of the semi-structured interviews.

4. STUDY RESULTS

This section reports the analysis of the results for the three research questions formulated in Section 3.1.

$\mathbf{RQ}_1$: To what extent is GEMMA able to optimize the GUI energy consumption, contrast, and design objectives? Boxplots in Fig. 2 show the distribution of values for the three fitness functions described in Section 2.3 exhibited by the *original design*[4] and by solutions in the Pareto front obtained at each GEMMA's run having the: *lowest ECF*, *median ECF*, *highest CF*, *median CF*, *lowest DF*, and *median DF*. In addition, Table 1 reports the results of the Wicoxon test (adjusted p-values) and the Cliff's d effect size. We compared the values for the ECF (energy consumption) and CF (contrast ratio) fitness functions between the original designs and the different categories of solutions considered in this research questions. The design achieving the best value for a fitness function (*i.e.,* the lowest value for ECF or the highest value for CF) is highlighted in **bold** for each test.

Fig. 2-(a) highlights that the design solutions generated by GEMMA have a lower energy consumption as compared to the original design. In particular, picking on each Pareto front the solution having the lowest value for ECF ensures a mean energy saving of 66% (median 89%). In other words, the most energy efficient colour schemas recommended by GEMMA consume, on average, one third of the original color design. Interestingly enough, the other categories of solutions proposed by GEMMA and considered in this research question offer a substantial energy savings as compared to the original design. For instance, the solution ensuring the highest contrast level between the GUI components (*Highest CF* in Fig. 2) offers a mean energy saving of 18% (median 8%) with respect to the original design. The statistical comparison in terms of energy consumption (ECF) between the original design and the solutions proposed by GEMMA (top part of Table 1) always highlight a statistically significant difference in favor of the GEMMA's solution accompanied by a large effect size for all categories of solutions but the one ensuring the highest contrast level (medium effect size).

The results in Fig. 2-(b) highlight that, besides ensuring a lower energy consumption, the color schemas recommended by GEMMA also help in improving the contrast ratio between the GUI's components (*i.e.,* higher values for the CF function). The GEMMA's solutions providing the best contrast ratio are able to obtain a mean improvement of the contrast of 61% (median 54%). It is surprising to see how GEMMA's solutions having the lowest energy consumption are able to improve the contrast of the original design (mean +16%, median=21%). While this result does not ensure the pleasantness of the color schemas generated by GEMMA (this aspect is investigated in $\mathbf{RQ}_2$ and $\mathbf{RQ}_3$), at least it should ensure the readability of the generated GUIs (*i.e.,* high contrast between the GUIs components). The results of the statistical test (bottom part of Table 1) show that the contrast ratio ensured by the GEMMA' color schemas is significantly higher with respect to the original ones. The effect size ranges between negligible (for the solutions ensuring the lowest ECF values) and large (for the solutions having the highest CF). While the negligible effect size achieved for solutions in the *lowest ECF* group might seem like a negative result, it is worth noting that such color schemas are able to achieve a mean reduction of energy consumption of 66%, while also being able to improve its contrast ratio (even with a negligible effect size).

An example of a solution generated by GEMMA for one of the subject apps in our study is shown in Fig. 3 flanked by the original design (left side). The proposed solution offers energy consumption savings of up to 53% as well as an increase in terms of contrast ratio by 31%.

Finally, by looking at Fig. 2-(c), it is interesting to understand what happens to the DF fitness function. Remember that this function equals zero when the evaluated solution (*i.e.,* color schema) exactly matches the original design. In all categories of solutions we selected and reported, there are solutions having zero as value for DF (as visible by the lower whisker of the boxplots "touching" the zero value). This indicates that DF function in GEMMA lets the GA search for solutions that very close to the original design.

$\mathbf{RQ}_2$: Are the color compositions generated by GEMMA visually attractive as perceived by Android users?

Boxplots in Fig. 4 show the distribution of answers for the visual aesthetics-related questions from the 85 participants. The boxplots summarize the answers regarding the *original design*, *lowest ECF*, and *median ECF* solutions of the 25 considered apps. In addition, Table 2 reports the results of the Wicoxon test (adjusted p-values) and the Cliff's d effect size. Fig. 4 highlights that the solutions generated

[4]For the original design there is no value for the DF, since this fitness function reflects the distance of the color schema of a solution from the original design.

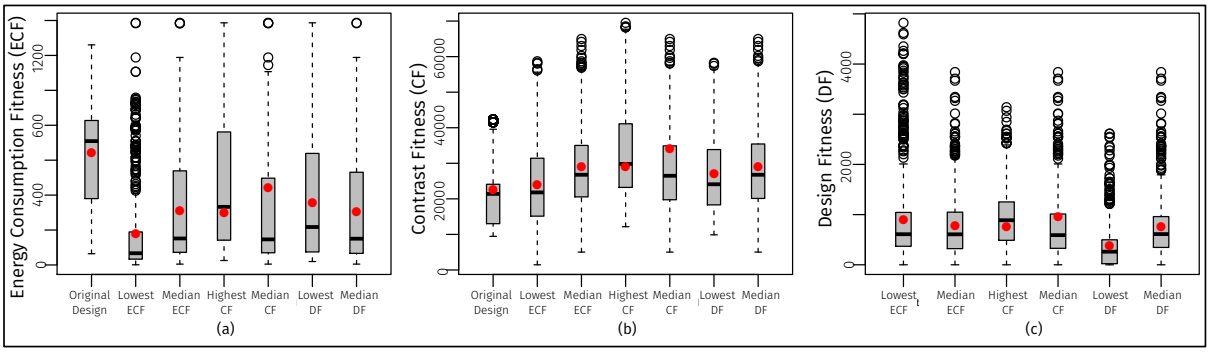

Figure 2: Boxplots of ECF, CF, and DF for the original design and for solutions provided by GEMMA.

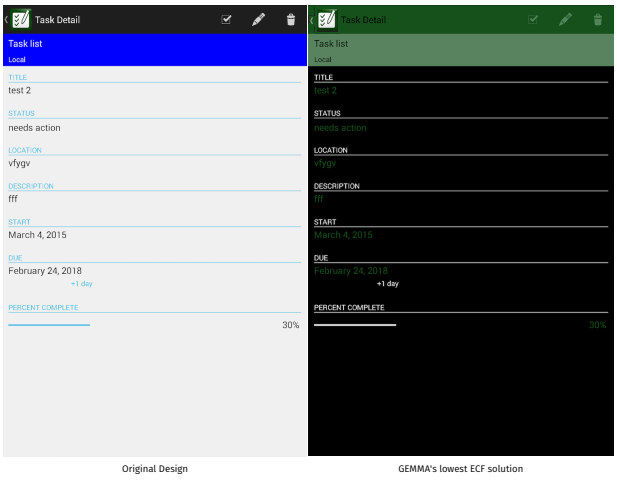

Figure 3: Original design *vs* GEMMA's solution.

by GEMMA have a "cost" in terms of visual aesthetics as perceived by end users. Answers from the participants to the colorfulness factor—described by the four questions presented in Section 3.3—show a low evaluation for the solutions with the lowest ECF (*i.e.,* the one saving the maximum amount of energy). Although end users do not totally dislike the *lowest ECF* solutions, the difference with the original design is statistically significant with a large/medium effect size. By looking at the comments left by participants and justifying their low ratings for the solution having the lowest ECF, it appears clear that most of them simply do not like GUIs having dark colors as main base: *"Black backgrounds can look very good but not when there is a lot of text"; "I prefer other colors than black for the mobile applications".*

Instead, participants found the *median ECF* solutions more visually appealing than the *lowest ECF* ones. The differences reported between the *original designs* and the *median ECF* solutions are small for the investigated colorfulness factors. Indeed, Fig. 4 highlights how the median and average answers for the four questions are "close" when comparing the *original designs* with the *median ECF* solutions. This is also confirmed by the statistical analysis reported in Table 2: while there is a significant difference in the participants' ratings in favour of the apps' *original design* when comparing it with the *median ECF* solutions, such a difference only results in a small effect size. Remember that, as shown in the context of RQ_1, solutions having a *median*

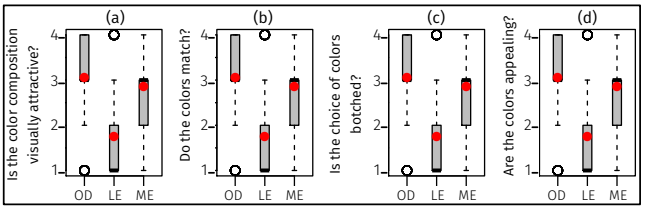

Figure 4: RQ_2: Boxplots of answers provided by participants. OD=Original Design, LE=Lowest ECF, ME=Median ECF.

Table 2: RQ_2: Wilcoxon test (p-value) and Cliff's delta (d).

Test	adj. p-value	d
Is the color composition visually attractive?		
original design *vs* lowest ECF	<0.01	-0.67 (Large)
original design *vs* median ECF	<0.01	-0.13 (Small)
Do the colors match?		
original design *vs* lowest ECF	<0.01	-0.65 (Large)
original design *vs* median ECF	<0.01	-0.16 (Small)
Is the choice of colors botched?		
original design *vs* lowest ECF	<0.01	0.33 (Small)
original design *vs* median ECF	<0.01	0.09 (Small)
Are the colors appealing?		
original design *vs* lowest ECF	<0.01	-0.55 (Large)
original design *vs* median ECF	<0.01	-0.13 (Small)

ECF are still able to achieve a substantial energy savings with respect to the original design (42%, on average). This illustrates GEMMA's ability to balance multiple objectives, *i.e.,* generating energy-saving GUIs that use colors close to the original composition and being accepted by end users.

RQ_3: Would actual developers of mobile applications consider changing colors in an app as recommended by GEMMA**?**

In the following, we report the results of the interviews we conducted with project managers of three Italian companies aiming at analyzing the practical applicability of GEMMA in a real development context. As explained in Section 3.3, we asked the participants to answer the following question: "*Given the energy saving provided by this design with respect to the original design, would you adopt it in your app?*", using a score on a four-point Likert scale: 1=absolutely no, 2=no, 3=yes, 4=absolutely yes.

Next. The first person we interviewed was Nicola Noviello, the project manager of Next. Such a company provided us with two apps, namely *Bollate* [1] and *Petrella* [5]. The two apps are from the category "Places and travels", and provide

tourist with information on two Italian towns. Before starting the interview, Nicola confirmed that GEMMA might have a high practical applicability even if it would be important to assess which percentage of the overall app energy consumption is actually amenable to its GUI, and thus could be optimized by GEMMA. This is something we plan to explore more in the future. As for the evaluation of the solutions provided by GEMMA for the Bollate app, we showed to Nicola eleven design alternatives, asking him if he would adopt them in his app. None of them was considered acceptable (answer *"absolutely not"*) despite the average energy saving around 85%. The reason why Nicola discarded such alternatives is represented by the color of the background. All the alternatives provided by GEMMA had a black background. In the context of Nicola's experience, "*it is hard to make attractive an interface with a black background*". Specifically, Nicola claimed that he uses a black background only if it is explicitly required by the customer. Otherwise, he designs apps with more vivid colors even if he recognized that this could negatively impact the energy consumption of the app. This suggests an interesting future direction to explore, *i.e.,* usage of interactive GAs, where the developer can be integrated in the evolutionary loop and set personalized constraints (*e.g.,* do not use this color for this component) during the generation of solutions. The results were different for the Petrella app. Among the five different alternatives, Nicola considered three as good alternatives to the original design, claiming that he would consider their adoption (answer *"yes"*). It is worth noting that the solution considered the best by Nicola is also the one that provides the highest energy savings (71%). For this app, Nicola particularly appreciated the recommended solutions due to their high similarity with the original design. This characteristic allows obtaining substantial energy savings just by performing small adjustments to the original GUI. In addition, Nicola claimed that "*even if the colors of these alternatives are less vivid than the original design, I believe that the proposed combinations—with some smalls adjustments— will make the app attractive*".

Genialapps. The second person we interviewed was Giuseppe Socci, the project manager of Genialapps. The app that they provided was Sing Happy Birthday Songs (HBS) [6], which has around 2,000 ratings on the Google Play market and a number of installations between 100K and 500K. The app is from the category "Music and Songs" and it can be used to send happy birthday wishes with a personalized phone call. Giuseppe was particularly interested in GEMMA. He claimed that such an approach might have an high industrial impact. We provided him with a set of 46 different alternatives. Among them, 29 were discarded as not adoptable as alternative to the original design (answer *"absolutely not"*). The reason why Giuseppe did not like those alternatives is that "*the colors are too dull and in a joyful app, like HBS, colors must be vivid*". The other 17 alternatives were considered good ones, with three of them gathering an *"absolutely yes"* to the posed question (see Section 3.3). One of these three solutions provides an energy savings of 63%. Such an alternative, again with some small adjustments, was considered by Giuseppe an excellent solution for his app. During the interview, Giuseppe also pointed out an important issue that approaches like GEMMA should take into account. GEMMA is particularly useful "*for apps that are used for a rather long period of time (e.g., social network app). Although it is true that many drops make the ocean,*

Figure 5: IdeaSoftware app: Original design *vs* GEMMA's solution (excerpts).

and therefore, in our case, many apps not optimized used for a short time still consume a lot of energy, it is also true that developers are not very likely to change the GUI aiming at saving battery. The reason is that this non-functional requirement is quite difficult to sell, unless the app is used for rather long time periods and thus the customer has a tangible evidence of the energy saved." This consideration is perfectly inline with our weighted model for power consumption, in which we give more importance to screens that are used more. As well as Nicola, Giuseppe is also interested in "*the actual impact that the colors of the GUI have on the overall energy consumption of the app*". He also pointed out that "*in the case of HBS, since it has a lightweight logic (the server is in charge of the application logic) I suppose that the influence of the GUI is considerable*".

IdeaSoftware. The last person we interviewed was Luciano Cutone, co-founder and the project manager of IdeaSoftware. The company requested to anonymize both of the apps that they provided for our study. In the following, we refer to such apps as App1 and App2. Both apps have thousands of installations on the Google Play market. Luciano was particularly interested in GEMMA, that he defined as "*a tool that should be integrated in an IDE and used daily*". This confirms the high industrial impact that GEMMA might have. As for App1, we provided to Luciano a set of 15 different alternatives. Among them, 8 were not considered acceptable (answer *"absolutely not"*) by Luciano because "*the combination of colors was not exciting*". The other 7 solutions were instead considered as good alternative to the original design (answer *"absolutely yes"*). Luciano particularly liked one solution that is reported, compared to the original design, in Fig. 5. Due to the need for anonymization, we only report a portion of the original and recommended GUIs. Luciano claimed "*I would definitively use this combination of colors in my app. The final result is excellent and I really like the effect of the GUI with a black background. This helps in saving battery and makes the app more elegant. I will propose the new combination of colors for the next release of the app*". Such a consideration highlights different points of view by Luciano and Nicola about black background emphasizing that the choice of colors is subjective and, as mentioned before, an interactive version of GEMMA could be worthwhile. For the App2, we only provided three different solutions. In addition, the average energy savings for such solutions is around 10%, due to the already energy efficient original design. Indeed, as pointed out by Luciano, when designing the GUI the developers used low bright colors in order to (i) make the app more professional and (ii) save the battery of the phone since the app was designed to be used continuously during the day. Such considerations confirmed that the energy consumption is a problem relevant for industry when the app is used frequently. Nevertheless, among the three solutions, Luciano classified two of them as good alternatives (answer *"yes"*).

5. THREATS TO VALIDITY

Threats to *construct validity* concern the relationship between theory and observation, and in this work are mainly related to the measurements we performed in our study. One major threat is that the measures of contrast and pleasant design—*i.e.,* the *CF(S)* and *DF(S)* objectives—which we use in the optimization process and quantitatively compare with values of the original colors ($\mathbf{RQ}_1$) might not represent a proxy of what actually is perceived by users. $\mathbf{RQ}_2$, within its generalizability limitations, is intended to mitigate such a threat. Another possible threat is related to the fact that we used power models to estimate energy consumption. So far, such models have been already used in previous studies and considered reliable enough, and our models are consistent with the ones by Dong *et al.* [11, 12] for OLED screens.

Threats to *internal validity* are related to factors, internal to our study, that can influence our results. We have used GA settings that are frequently used in the research literature [10, 13, 14], and calibrated the number of evolutions so that longer evolutions did not produce better results. Of course, we cannot exclude that a better calibration would produce better results. Last, but not least, we accounted for GA randomness by executing GEMMA 30 times [8].

Threats to *conclusion validity* concern the relationship between experimentation and outcome. For $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$, as described in Section 3.4, we have quantitatively assessed our results using appropriate statistical procedures. Instead, $\mathbf{RQ}_3$ is exploratory in nature, intended to collect some preliminary feedback about the practical applicability of GEMMA in a real development context.

Threats to *external validity* are related to the generalizability of our findings. We are aware that our results have to be interpreted carefully because (i) they may depend on the specific device for which we extracted the color power model; (ii) results of $\mathbf{RQ}_2$ may depend on people's preferences, and other people might have different opinion on the chosen colors, and (iii) although we applied GEMMA on 25 different apps, we are aware than GEMMA might produce different results in terms of energy consumption reduction and colorfulness on a different set of apps.

6. RELATED WORK

GEMMA is mostly related to previous work on energy optimization of GUIs in mobile apps as well as approaches for detecting energy greedy units in such apps.

6.1 Improving Energy Consumption of GUIs

OLED screens are suitable for optimizing energy consumption of GUIs in apps, because the power consumed by OLED screens depends on the combination of color levels in the screen's sub-pixels. Therefore, power models of OLED screens estimate the energy by combining individual consumption of the color sub-pixels [20, 41]. These models are particularly useful, since they allow researchers to estimate energy consumption without using expensive power monitors. Thus, power models have been used for energy saving visualization of sequential data [40], design of energy adaptive displays [20], and design of color-adaptive browsers [11].

As for mobile apps, representative works are the ones by Dong and Zhong [12] for Windows mobile apps, Li *et al.*'s approach [24] for mobile web applications, and Wan *et al.*'s work [39] on screenshots of Android apps. The three approaches and GEMMA are compared in Table 3. While we share with these approaches the goals of minimizing energy consumption through properly choosing colors and using color power models, GEMMA introduces novel aspects related to (i) multi-objective optimization, (ii) considering contribution of different screens based on their usage duration, and (ii) ensuring a pleasant and consistent choice of colors.

Table 3: GUI-based energy optimization in Android.

Approach	Opt. problem	Color palettes
[12]	*min* energy, *s.t.* similar contrast than in original GUI	1) Predefined themes, 2) Monochromatic palette, 3) black assigned to most frequent color in original GUI, then subsets of R,G,B assigned to the rest of pixels
[24, 39]	*min* energy, *s.t.* keep color distance between neighboring HTML elements	Background replaced with dark colors, randomly selected colors are assigned to the rest of pixels
GEMMA	*min* energy, *max* contrast, *min* distance to original design, *s.t.* contrast > 4	Palette contains 512 colors: original + black + white + equidistant harmonies and monochromatic palette)

6.2 Detecting Energy Bugs in Mobile Apps

Energy bugs and energy hot spots in Android apps—at different granularities—have been catalogued extensively. For instance, Pathak *et al.* [32, 34] describe a taxonomy of energy bugs that depends on the hardware, software, or external conditions. Kwon and Tilevich [21] focused on cloud offloading energy consumption. Pathak *et al.* [33] and Linares-Vásquez *et al.* [26] reported energy greedy Android APIs. Hao *et al.* [17] and Li *et al.* [22] identified energy greedy apps, meanwhile Li *et al.* [23] focused on energy measurement at code statement level. Liu *et al.* [27] detected energy bugs related to misuses of wake locks and sensors. Finally, Rasmussen *et al.* [35] measured the impact of ads in apps on energy consumption. All the approaches mentioned in this section detect different kinds of energy-greedy units in the source code, yet not specifically related to the screen usage. For this reason, GEMMA can be considered complementary to those approaches: in other words, reducing the energy footprint of mobile apps concerns considering various aspects, including, among others, screen usage and colors.

7. CONCLUSION AND FUTURE WORK

We presented GEMMA, a multi-objective approach for generating energy-friendly color schemas for Android app GUIs. The multi-objective optimization balances the energy reduction objective with other objectives related to contrast and closeness to the original design.

GEMMA successfully generated designs for 25 Android apps with significant reduction in energy consumption. While the empirical evaluation highlighted that solutions with the highest energy savings are usually not preferred by end-users, mainly because of the dark background, GEMMA still generated solutions that achieve a good energy reduction while being acceptable by end-users. Also, by evaluating the solutions on five commercial apps, we confirmed that some project managers and developers are ready to account for GEMMA's recommendations in future app releases. In the future work, we are planning on relying on a more precise approach for reverse engineering app GUIs, which would account for a proper choice of colors for GUI elements such as text fields and buttons.

8. REFERENCES

[1] Bollate. https://play.google.com/store/apps/details?id=com.comunicazione360.bollate.

[2] Genial apps website. \url{http://www.genialapps.eu/portale/}.

[3] Mediastudio website. http://www.mediastudio.it/.

[4] Next website. http://www.nextopenspace.it/.

[5] Petrella. https://play.google.com/store/apps/details?id=it.nextlabs.platform.petrella.

[6] Sing happy birthday songs. http://happybirthdayshow.net/en/.

[7] Weplant. https://play.google.com/store/apps/details?id=it.nextlabs.platform.weplanet\&hl=it.

[8] A. Arcuri and L. Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1–10, New York, NY, USA, 2011. ACM.

[9] W. J. Conover. *Practical Nonparametric Statistics*. Wiley, 3rd edition edition, 1998.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182 – 197, 2002.

[11] M. Dong and L. Zhong. Chameleon: A color-adaptive web browser for mobile oled displays. *IEEE Transaction on Mobile Computing*, 11(5):724–738, May 2012.

[12] M. Dong and L. Zhong. Power modeling and optimization for oled displays. *IEEE Transaction on Mobile Computing*, 11(9):September, 2012.

[13] J. J. Durillo and A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[14] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition, 1989.

[15] R. J. Grissom and J. J. Kim. *Effect sizes for research: A broad practical approach*. Lawrence Earlbaum Associates, 2nd edition edition, 2005.

[16] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan. Estimating Android applications' CPU energy usage via Bytecode profiling. In *GREENS'12*, pages 1–7, 2012.

[17] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan. Estimating mobile application energy consumption using program analysis. In *ICSE'13*, pages 92–101, 2013.

[18] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. Campbell, and S. Romansky. Greenminer: a hardware based mining software repositories software energy consumption framework. In *MSR'14*, pages 12–21, 2014.

[19] S. Holm. A simple sequentially rejective Bonferroni test procedure. *Scandinavian Journal on Statistics*, 6:65–70, 1979.

[20] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan. Energy-adaptive display system designs for future mobile environments. In *International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, 2003.

[21] Y. K.won and E. Tilevich. Reducing the energy consumption of mobile applications behind the scenes. In *ICSM'13*, pages 170–179, 2013.

[22] D. Li, S. Hao, J. Gui, and W. Halfond. An empirical study of the energy consumption of android applications. In *International Conference on Software Maintenance and Evolution (ICSME'14)*, page to appear, 2014.

[23] D. Li, S. Hao, W. G. J. Halfond, and R. Govindan. Calculating source line level energy information for android applications. In *ISSTA'13*, pages 78–89, 2013.

[24] D. Li, A. H. Tran, and W. Halfond. Making web applications more energy efficient for oled smartphones. In *International Conference on Software Engineering (ICSE'14)*, pages 573–538, 2014.

[25] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk. Replication package. http://www.cs.wm.edu/semeru/data/GEMMA/.

[26] M. Linares-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 *11th IEEE Working Conference on Mining Software Repositories (MSR'14)*, pages 2–11, 2014.

[27] Y. Liu, C. Xu, S. Cheung, and J. Lu. Greendroid: Automated diagnosis of energy inefficiency for smartphone applications. *IEEE Transactions on Software Engineering*, Preprint, 2014.

[28] Y. Liu, C. Xu, and S. C. Cheung. Where has my battery gone? finding sensor related energy black holes in smartphone applications. In *PerCom'13*, pages 2–10, 2013.

[29] I. L. Manotas-Gutiérrez, L. L. Pollock, and J. Clause. Seeds: a software engineer's energy-optimization decision support framework. In *ICSE'14*, pages 503–514, 2014.

[30] Monsoon-Solutions. Power monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[31] M. Moshagen and M. T. Thielsch. Facets of visual aesthetics. *Human-Computer Studues*, 68:689–709, 2010.

[32] A. Pathak, Y. Hu, and M. Zhang. Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices. In *Hotnets'11*, page Article No 5, 2011.

[33] A. Pathak, Y. Hu, and M. Zhang. Where is the energy spent inside my app? fine grained energy accounting on smartphones with eprof. In *EuroSys'12*, pages 29–42, 2012.

[34] A. Pathak, A. Jindal, Y. Hu, and S. P. Midkiff. What is keeping my phone awake? characterizing and detecting no-sleep energy bugs in smartphone apps. In *MobiSys'12*, pages 267–280, 2012.

[35] K. Rasmussen, A. Wilson, and A. Hindle. Green mining: energy consumption of advertisement blocking methods. In *GREENS'14*, pages 38–45, 2014.

[36] G. Sharma. *Digital Color Imaging Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 2002.

[37] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[38] W3C. Contrast ratio definition. `http://www.w3.org/WAI/ER/WD-AERT/#color-contrast`.

[39] M. Wan, Y. Jin, D. Li, and W. G. J. Halfond. Detecting display energy hotspots in android apps. In *8th IEEE International Conference on Software Testing, Verification and Validation (ICST'15)*, page to appear, 2015.

[40] J. Wang, X. Lin, and C. North. Greenvis: Energy-saving color schemes for sequential data visualization on oled displays. Technical Report TR-12-09, Department of Computer Science, Virginia Tech, 2012.

[41] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, and A. Yla-Jasski. A system-level model for runtime power estimation on mobile devices. In *International Conference on Green Computing and Communications*, pages 27–34, 2010.

[42] J. Zang, A. Musa, and W. Le. A comparison of energy bugs for smartphone platforms. In *MOBS'13*, 2013.