A methodology for energy performance testing of smartphone applications

Article · June 2012

DOI: 10.1109/IWAST.2012.6228978

CITATIONS
15

READS
79

4 authors, including:

Rajesh Palit
North South University
39 PUBLICATIONS **249** CITATIONS
SEE PROFILE

Ajit Singh
Indira Gandhi Agricultural University
47 PUBLICATIONS **487** CITATIONS
SEE PROFILE

A Methodology for Energy Performance Testing of Smartphone Applications

Abdulhakim Abogharaf
University of Waterloo
Ontario, Canada
ajabogha@uwaterloo.ca

Rajesh Palit
University of Waterloo
Ontario, Canada
rpalit@uwaterloo.ca

Kshirasagar Naik
University of Waterloo
Ontario, Canada
snaik@uwaterloo.ca

Ajit Singh
University of Waterloo
Ontario, Canada
asingh@uwaterloo.ca

***Abstract*—Smartphones are becoming increasingly popular among users. They are equipped with enormous number of applications, and those applications drain the smartphones' batteries. Moreover, battery capacity is significantly restricted due to constraints on size and weight of the device. It is important for smartphone applications to be energy efficient. Thus, a methodology to conduct energy performance testing is needed for two reasons: (i) evaluate the power consumption of a single application on a given device; (ii) compare the power consumption of different smartphones or platforms running the same application. In our earlier work "Selection and execution of user level test cases for energy cost evaluation of smartphones" (Proceedings of the *6th* AST, 2011), we have developed a testing methodology that significantly reduces the number of test cases. In addition, we have introduced the concepts of *primary* and *standalone* test configurations. However, ordering of the executions of those two kinds of tests is non-trivial, and it was not studied in that paper.**

In this paper, we introduce a methodology to effectively interleave the identification of those two kinds of test configurations in order to reduce the total number of configurations. We express the methodology in the form of a detailed flow chart that application developers can easily follow. We apply the methodology to a specific smartphone, namely *HTC Nexus One* smartphone in order to illustrate the process of this methodology. The total number of test configurations obtained by the given methodology is the same as the number predicted by numerical expressions.

***Keywords*-Smartphones; testing; energy performance.**

I. Introduction

The rapid growth of smartphones has brought millions of mobile applications to "app stores." Many of these applications make use of smartphone cameras, Global Positioning System (GPS) and Internet access, requiring considerably large amount of power. Due to constraints on size and weight of handheld devices, battery capacity is significantly restricted and becomes the bottleneck of smartphones' usability. Therefore, having power-efficient mobile applications is important for prolonging the battery recharging time.

Unlike their computer counterparts, smartphones face a new type of abnormal application behaviors which are called energy bugs (ebugs) [1]. Compared to traditional application bugs, ebugs, such as no-sleep bug, cause unexpected amount of high energy consumption where the application continues to run seemingly well. This makes ebug detection a much more difficult task than traditional bug detection. For this reason, a reliable methodology to evaluate the power consumption of applications is needed.

The requirements in energy performance testing are different than those in functional testing. Whereas functional testing most likely treats the parameters (the input) equally and fairly, energy performance testing puts more weight on particular parameters that mostly impact the power consumption. In functional testing, all parameters are equally expected to cause a bug; however, in energy performance testing, the parameters that control the hardware (e.g. GPS) are more expected to cause ebugs. Also, in energy performance testing, the relationship between parameters (dependency, for example) can be considered to reduce the number of required test configurations.

Due to the enormous number of phone configurations and application contents, an immense number of test cases exists [2]. Therefore, a selection technique reducing the number of test cases is desired. For this reason, we have proposed an approach to select user-level test cases for energy cost evaluation of smartphone applications [3]. In the current work, we explain the methodology by means of a detailed flow chart and apply it to a smartphone.

The concepts of parameters, configurations, application contents, and test cases are at the core of this work. We explain these terms in the following:

- *Parameters:* The settings of smartphone parameters affect the power consumption of applications. For example, the power consumption of a music player application is affected by the level of the device volume. Most of the parameters have various states representing different values. Some of those parameters are display size, Bluetooth, and brightness level. Tables II to IV in the appendix show some parameters with their states for five different smartphones, namely, *BlackBerry 9700*, *HTC HD2*, *HTC Nexus One*, *Nokia E71*, and *iPhone 3GS*. In the tables, 'Yes' means the parameter is available on that particular smartphone, 'No' implies that the parameter is not available, and 'Alternative' denotes that a similar parameter is available. Some parameters take discrete values, and others take continuous values.
- *Configurations:* A configuration of a smartphone is an instance of all parameters for which a device is used

to run an application [2]. Since the total number of all possible combinations of parameters' values is vastly large, there is an enormous number of configurations.

- *Application contents:* Application contents represent the settings of the application under test. Since we are not evaluating the functionality of the application, only application contents that affect the energy performance metric are considered. For example, video playing application supports different file formats (contents), and these different file formats require different amount of processing time. Also, they use different amount of memory. So that, the type of video file affects the power consumption of a video player application.
- *Test cases:* A test case represents input and output of an experiment. A device configuration and application contents are the input of the test case. The output is the power consumption. We are not interested in the user output (the service given by the application). We observe the power consumption during the execution time and consider it as a test output. Since there is enormous number of configurations and a large number of application contents, the number of possible test cases is very large.

In this work, we make the following contributions beyond our previous paper [3]:

- We clearly explain the methodology of how to effectively select test configurations in order to evaluate the energy performance of a smartphone's application. By "effectively" we mean how to select exactly the required number of test configurations. A detailed flow chart is used to describe this testing methodology.
- We apply the methodology to a specific smartphone, namely *HTC Nexus One*.

Application developers and testers can follow this process to select a reduced set of configurations in order to evaluate the energy performance of their applications. Also, they can use this process to compare applications' behaviors on deferent devices and platforms.

II. Background

Our previous work [3] is one of the basic studies that were done in the area of energy performance testing for smartphone applications. In that work, a methodology of test selection technique was proposed to have a reduced number of configurations which is used to obtain the energy behavior of an application. Since this work is mainly based on our previous work, a brief summary of the previous work is given next.

Assume that there are m parameters in a smartphone d grouped in the set $G^d = \{B_1, B_2, \cdots, B_m\}$. A specific state of a parameter B_j is b_j^l, and it has $\eta(B_j)$ states. For example, the Bluetooth parameter B_{33} (refer to Table III in the appendix) has a state b_{33}^1 = 'ON', and number of states for this parameter is $\eta(B_{33}) = 2$. A configuration β_i represents an instance of all settable parameters of a smartphone. A smartphone application A_i has a type of content C_i. Number of contents for a particular application A_i is denoted by $\eta(C_i)$. For example, a video player application A_1 supports only two file format: $c_1^1 = MPEG$ and $c_1^2 = Flash$. Number of contents for this application is $\eta(C_1) = 2$.

The number of test cases N_c is expressed as:

$$N_c = S_d \times \sum_{i=1}^{M} X_d(A_i) \times \eta(C_i), \tag{1}$$

where S_d is the number of configurations for a smartphone d. M is the number of chosen applications. $X_d(A_i)$ indicates whether or not application A_i is available in the phone d.

The number of all possible configurations on a smartphone d is expressed as:

$$S_d = \prod_{j=1}^{m} \eta(B_j) \tag{2}$$

This number is extremely large in any smartphone [3], and it is not feasible to consider all the configurations of a smartphone while testing. To reduce number of configurations, firstly we categorize the smarphone parameters as follows:

- *Basic Parameters* (G_0^d): This group contains all the parameters whose values are fixed. A user cannot change or adjust the values of those parameters. Processor, Memory and size of display fall in this group. Table II in the appendix shows some basic parameters of the five smartphones mentioned earlier. Those parameters are kept aside during the testing process. However, when we compare two or more smartphones on energy performance, the differences in these parameters must be noticed during the analysis of the test results.
- *Active Parameters* (G_1^d): This group contains the parameters that the user can control and adjust their values. Basically, these parameters are some utility programs on top of operating system (OS), which control hardware components. Applications for controlling volume of a device and brightness of the display are examples of active parameters. Table III in the appendix illustrates some of the active parameters. The active parameters are further divided into two groups:
  - *Primary Parameters* (G_1^p): A subset of active parameters are included in a set named as primary parameters. These active parameters appear to consume a significant proportion of the total power consumptions. So that they strongly impact the power consumption of an application, and their variation yield more informative test results.
  - *Standalone Parameters* (G_1^s): These parameters are the rest of the active parameters. With the primary

parameters, standalone parameters are considered for performing standalone configurations.

- *Passive Parameters* (G_2^d): This group contains the rest of the parameters whose values are also adjustable by the users. They are also utility programs incorporated in the OS or from third parties. For example, WiFi Hotspot and Bluetooth Discoverable are programs which use WiFi and Bluetooth interfaces, respectively. They do not control the hardware components, but they use hardware components, and thus, they affect the test outcomes. During the testing process, we need to take out the effect of these applications by keeping their values fixed throughout the testing process. Table IV in the appendix shows some of passive parameters.

We consider only the active parameters for the device configurations, omit the basic parameters and fix the passive parameters to certain values. The number of active parameters is m_A. The total number of test cases is given as:

$$N_C = (S_d^P + S_d^S) \times \sum_{i=1}^{M} X_d(A_i) \times \eta(C_i), \tag{3}$$

where S_d^P is the number of primary configurations, and S_d^S is the number of standalone configurations. Primary configurations are obtained by considering primary parameters only. Standalone configurations are obtained by considering standalone parameters one by one with the primary parameters. The number of the primary configurations S_d^p is expressed as follows:

$$S_d^P = \begin{cases} 1 + \sum\limits_{\forall B_i \in G_1^p} [\eta(B_i) - 1] + \frac{k(k-1)}{2} & , \mathit{Ind}(G_1^p) \\ \prod\limits_{\forall B_i \in G_1^p} \eta(B_i) & , \text{Otherwise}, \end{cases} \tag{4}$$

where k is number of parameters in the primary set G_1^p, and $Ind(G_1^p)$ means that all the parameters in the primary set are independent of each other [3]. Two parameters are said to be independent of each other if their energy costs are additive. In our previous work [3], a special case of S_d^P was given for $|G_1^p| = 1$.

The number of standalone configurations S_d^s related to a standalone parameter B_j is Q_j which can be expressed as:

$$Q_j = \begin{cases} \eta(B_j) - 1 + k & , \mathit{Ind}(B_j\ , G_1^p) \\ [\eta(B_j) - 1] \left[1 + \sum\limits_{\forall B_i \in G_1^p} (\eta(B_i) - 1) \right] & , \mathit{Ind}(G_1^p) \\ [\eta(B_j) - 1] \prod\limits_{\forall B_i \in G_1^p} \eta(B_i) & , \mathit{Otherwise}, \end{cases} \tag{5}$$

where $Ind(B_j, G_1^p)$ means the standalone parameter B_j is independent of all primary parameters G_1^p. In our previous work [3], a special case of Q_j was given for $|G_1^p| = 1$. The number of standalone configurations S_d^s is expressed as:

$$S_d^S = \sum_{\forall B_j \in G_1^s} Q_j \tag{6}$$

Our approach uses a test bench to measure the power consumption (the output of the test) of the application under test for every test case. As shown in Fig. 1, a smartphone is powered by a power supply with high precision current measurement unit. This power supply is connected to a computer in order to read and store the measurements. During the experiment, we disconnect the battery's power interface and power the smartphone from the power supply; however, the battery communication interface should stay connected to the smartphone. Also the battery has to have enough power during all experiments so that no power-saving state takes place.

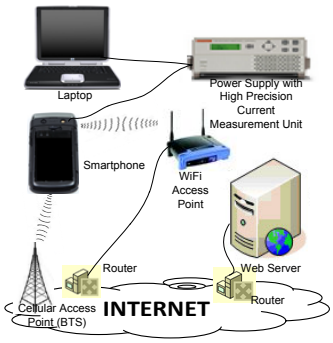

Figure 1. Experimental setup

Our previous work only shows the reduced number of configurations. It only shows how many primary and standalone configurations we have. However, it does not explain how to get those configurations. Next section shows how to get the reduced set of configurations by means of a detailed flow chart and by applying the process to a specific phone.

III. Description of the Methodology

In this section, *HTC Nexus One* smartphone will be considered as an example to explain the flow chart shown in Fig. 2 and to show how to fill up the configuration table shown in Table I.

1) As a first step, all the phone's parameters have to be grouped in a set G^d. *HTC Nexus One* smartphone has 21 parameters that affect its power consumption (See Tables II to IV for some of these parameters). However, parameters in G^d are categorized into three sets, G_0^d, G_1^d and G_2^d. The set G_0^d includes the basic parameters. For *HTC Nexus One* phone, there are six basic parameters. The set G_1^d includes the active parameters that control the hardware components. *HTC Nexus One* smartphone has five active parameters ($m_A = 5$). The set G_2^d contains the passive parameters that use, but do not

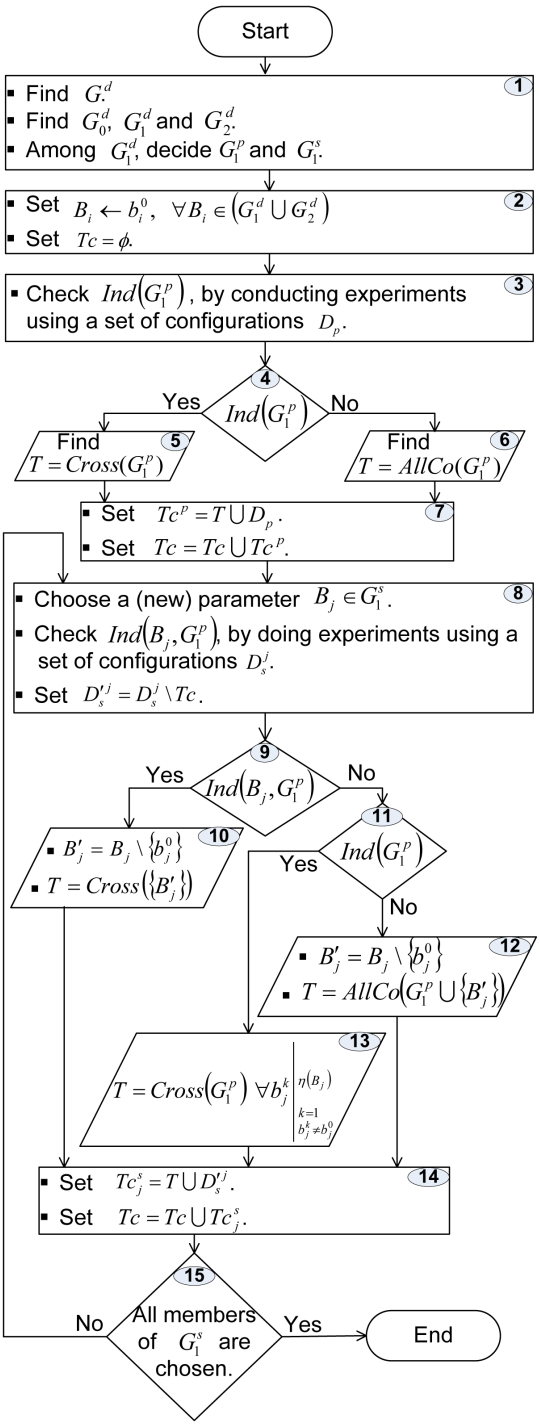

Figure 2. Flow chart for filling out the configuration table, (*AllCo* and *Cross* are explained in the text, and T is a local variable)

Stage	Configurations	Primary parameters $k = 2$		Stand alone Parameters $m_A - k$		
		Data Access Mode $\eta(B_{34}) = 3$	Volume $\eta(B_{31}) = 16$	GPS $\eta(B_{35}) = 2$	Bluetooth $\eta(B_{33}) = 2$	Brightness $\eta(B_{32}) = 5$
Primary configurations	C_1^t	Off	0	Off	Off	0%
	C_2^t	Off	15	Off	Off	0%
	C_3^t	3G	0	Off	Off	0%
	C_4^t	3G	15	Off	Off	0%
	C_5^t	WiFi	0	Off	Off	0%
	C_6^t	Off	1	Off	Off	0%
	C_7^t	Off	2	Off	Off	0%
	C_8^t	Off	3	Off	Off	0%
	C_9^t	Off	4	Off	Off	0%
	C_{10}^t	Off	5	Off	Off	0%
	C_{11}^t	Off	6	Off	Off	0%
	C_{12}^t	Off	7	Off	Off	0%
	C_{13}^t	Off	8	Off	Off	0%
	C_{14}^t	Off	9	Off	Off	0%
	C_{15}^t	Off	10	Off	Off	0%
	C_{16}^t	Off	11	Off	Off	0%
	C_{17}^t	Off	12	Off	Off	0%
	C_{18}^t	Off	13	Off	Off	0%
	C_{19}^t	Off	14	Off	Off	0%
Stand alone configurations	C_{20}^t	3G	0	On	Off	0%
	C_{21}^t	3G	15	On	Off	0%
	C_{22}^t	WiFi	0	On	Off	0%
	C_{23}^t	3G	0	Off	On	0%
	C_{24}^t	3G	15	Off	On	0%
	C_{25}^t	WiFi	0	Off	On	0%
	C_{26}^t	3G	0	Off	Off	100%
	C_{27}^t	3G	15	Off	Off	100%
	C_{28}^t	WiFi	0	Off	Off	100%
	C_{29}^t	3G	0	Off	Off	25%
	C_{30}^t	3G	0	Off	Off	50%
	C_{31}^t	3G	0	Off	Off	75%

Table I
CONFIGURATIONS TABLE FOR *HTC Nexus One* SMARTPHONE ($m_A = 5$)

control, the hardware components. There are ten passive parameters in *HTC Nexus One* phone.

The parameters in G_1^d are further classified into two groups, G_1^p and G_1^s. The primary parameters G_1^p appear to consume a large proportion of energy. The rest of active parameters are called standalone parameters G_1^s. Techniques for selecting primary parameters G_1^p are beyond the scope of this work. For the *HTC Nexus One* smartphone, only Data Access Mode and Volume are considered in G_1^p; that is $G_1^p = \{B_{31}, B_{34}\}$ and $k = |G_1^p| = 2$.

2) After categorizing the parameters, all settable parameters (G_1^d and G_2^d parameters) should be set to initial values $b_j^0 \Big|_{B_j \in (G_1^d \cup G_2^d)}$. Passive parameters are kept fixed to those initial values during the whole experiment. Now, we need to initiate the set of configurations $Tc = \phi$. This set will include all the selected configurations.
3) In this step, we need to check the dependency among all primary parameters G_1^p. In order to do so, we need to conduct some experiments. Two parameters are said to be independent if their energy costs are additive [3]. The dependency has to be checked for each and every pair of the primary parameters G_1^p. The function $Ind(G_1^p)$ takes the set of the parameters G_1^p and checks the dependency among each and every pair of the parameters in that set. The outcome of

this function is either "*True*" if all the parameters are independent of each other or "*False*" otherwise. To check the dependency for a pair of parameters, at least four configurations are required. However, same configurations can be used to check the dependency for many pairs. Although being mainly used to check the dependency, those configurations are also used to fill up the configuration table. The set D_p represents the set of configurations used to check the dependency among G_1^p. In our example, the set of configurations D_p consists of the first four configurations in the Table I. That is $D_p = \{C_1^t, C_2^t, C_3^t, C_4^t\}$.

4) In here, we take a decision about how to get the primary configurations depending on the results of the dependency check. If all parameters in G_1^p are independent (that is $Ind(G_1^p) = true$), we need to go to step (5) to get the primary configurations. Otherwise, we get the primary configurations via step (6). In our example, we assumed that the two primary parameters we have are independent on each other; that is $Ind(G_1^p) = true$.
5) In this step, we get a set of configurations T representing a subset of the primary configurations Tc^p. Since the primary parameters are independent, we need to choose configurations only across each of the primary parameters [3]. The function $Cross(G_1^p)$ takes the primary parameters G_1^p and returns a set of configurations T by going across each of the primary parameters. In our example, by going across both of the primary parameters, we get $T = \{C_1^t \sim C_3^t, C_5^t \sim C_{19}^t\}$.
6) since the primary parameters are not independent, we need to consider all possible combinations of the primary parameters in order to get the set of configurations T. The function $AllCo(G_1^p)$ takes the primary parameters and returns a set of configurations T by considering all possible combinations of those parameters.
7) Some of the configurations in the set T were already considered in the set of configurations D_p used to check the dependency among primary parameters. The set of all primary configurations Tc^p can now be gotten by the union of T and D_p; that is $Tc^p = T \bigcup D_p$. Then, the set Tc has to be updated by considering the primary configurations Tc^p in it. In our example, the set of the primary configurations $Tc^p = \{C_1^t, C_2^t, ..., C_{19}^t\}$. We can verify the number of primary configurations by applying Eq. 4. Now, we update the set Tc; that is $Tc = Tc \bigcup Tc^p = Tc^p = \{C_1^t, C_2^t, ..., C_{19}^t\}$.
8) From now on, we will be dealing with the standalone parameters. We choose one (new) standalone parameter B_j, and then we check the dependency between the standalone parameter B_j and the primary parameters G_1^p. In our example, we chose GPS parameter (B_{35}) as a first standalone parameter. The function $Ind(B_j, G_1^p)$ checks the dependency between the standalone parameter and each of the primary parameters. It returns "*True*" if the standalone parameter is independent of all of the primary parameters. Otherwise, it returns "*False*". We need to conduct some experiments using a set of configurations D_s^j in order to check dependency. The set D_s^j contains the configurations used to check the dependency between the standalone parameter B_j and each of the primary parameters G_1^p. We used the configurations $\{C_3^t, C_4^t, C_{20}^t, C_{21}^t\}$ to check the dependency between the GPS parameter and the volume parameter, and the configurations $\{C_3^t, C_5^t, C_{21}^t, C_{22}^t\}$ to check the dependency between the GPS parameter and the Data Access Mode parameter. Thus $D_s^{35} = \{C_3^t, C_4^t, C_5^t, C_{20}^t, C_{21}^t, C_{22}^t\}$. The set $D_s'^j$ is a subset of D_s^j, which represents the configurations in D_s^j that are not yet included in the Tc. That is $D_s'^{35} = D_s^{35} \backslash Tc = \{C_{20}^t, C_{21}^t, C_{22}^t\}$.
9) At this point, a decision about how to get the standalone configurations is made based on the results of the dependency check. If the standalone parameter B_j is independent of all the primary parameters, we go to step (10). Otherwise, we go to step (11). In our example, we assumed that standalone parameters are independent of all primary parameters.
10) At this point, we eliminate the initial state of the standalone parameter B_j from its set of states because the initial state was already considered in the primary stage, and we call the new set of states B_j'. Since $Ind(B_j, G_1^p) = True$, we get standalone configurations related to B_j by going only across B_j'. In our example, only the configuration C_{22}^t is considered ($T = \{C_{22}^t\}$).
11) In this case, we need to look again at the dependency among the primary parameters, which was already checked in step (3). In order to get the standalone configurations, we go to step (13) if the primary parameters are independent, and we go to step (12) otherwise.
12) We eliminate the initial value b_j^0 of the standalone parameter B_j and form a new set of states B_j'. After that, we get the set of configurations T related to B_j by considering all possible combinations between the primary parameters G_1^p and the set $\{B_j'\}$.
13) For this case, we basically consider the primary configurations with the rest of the states of the standalone parameter B_j in order to get the standalone configurations related to B_j. In other words, to get the set of configurations T, we apply the function $Cross(G_1^p)$ with each state of the standalone parameter B_j except the initial state.
14) At this point, we have already got the set of configurations T. We can get the set of standalone configurations Tc_j^s corresponding to the standalone parameter B_j by taking the union between the set T and the set $D_s'^j$. The set Tc is updated by considering the standalone

configurations Tc_j^s. In our example, the standalone configurations corresponding to the GPS parameter is $Tc_{35}^s = T \bigcup D_s'^{35} = \{C_{20}^t, C_{21}^t, C_{22}^t\}$, and the set of configurations $Tc = Tc \bigcup Tc_{35}^s = \{C_1^t, C_2^t, ..., C_{22}^t\}$. We can verify that $Q_{35} = |Tc_{35}^s|$ via Eq. 5.

15) At this point, we check if there is a standalone parameter which has not been chosen yet. If so, we go back to step (8). Otherwise, we end. In out example, and by assuming that the rest of the standalone parameters (Bluetooth (B_{33}) and Brightness (B_{32})) are also independent of G_1^p, we get $Tc_{33}^s = \{C_{23}^t \sim C_{25}^t\}$ and $Tc_{32}^s = \{C_{26}^t \sim C_{31}^t\}$. The configurations $\{20 \sim 31\}$ represent all standalone configurations (see Eq. 6).

From the above application for *HTC Nexus One* phone, and referring to Table I, we notice that: $m_A = 5$; $k = 2$; $\eta(B_{34}) = 3$; $\eta(B_{31}) = 16$; $\eta(B_{35}) = 2$; $\eta(B_{33}) = 2$; $\eta(B_{32}) = 5$; $G_1^p = \{B_{34}, B_{31}\}$; $Ind(B_{35}, G_1^p) = True$; $Ind(B_{33}, G_1^p) = True$, and $Ind(B_{32}, G_1^p) = True$. By referring to Eqs. ($4 \sim 6$), now we can get: $S_d^p = 19$; $Q_{35} = 3$; $Q_{33} = 3$; $Q_{32} = 6$; $S_d^s = 12$, and $S_d = 31$.

IV. Related Work

Since the Exhaustive testing [4] cannot be practically applied, many approaches were proposed to reduce number of test cases into a much smaller set with comparable results.

One of the techniques is the pairwise testing [5]. For each and every pair of states which belong to different parameters, there is at least one test case that contains both of those states. Pairwise testing has become an important tool in a tester's toolbox.

However, an application could misbehave only with three, or more combinations. Study results found that all the known application misbehaviors are caused by interactions among 6 or fewer parameters [6]. Another Technique called t-wise testing which requires that every combination of any t parameter values must be tested at least once [7].

A more practical and efficient approach in examining applications than t-wise testing is the variable strength interaction testing [8]. This technique focuses on a specific set of parameters and apply a higher strength testing on them without ignoring the rest. This is required because interactions do not often exist uniformly between parameters. Some parameters have strong interactions with each other while others may have few or none interactions.

All of the aforementioned techniques were proposed for functional testing. Their main aim is to maximize the chance of finding errors and bugs in the application under test. However, our interest is energy performance testing for smartphone applications. In evaluating the power consumption, we need to put more weight on the parameters that seem to consume more energy, and we need to make use of the dependency between the parameters.

V. Conclusion

Smartphones are increasingly getting more popular among all users. They have computer-like capabilities and provide sophisticated services to the users. Nonetheless the enormous growth in smartphones' ubiquity and their applications and services, their utility has been severely limited by their battery life. Since the growth of the battery is not keeping up with rapid evolution of other smartphone's components, it is very important to make smartphones applications energy efficient. For that reason, energy performance testing is sorely essential to find energy bugs (ebugs) in smartphones' applications and to compare application behaviors in different smartphones.

This work explains the methodology of our energy performance testing. Application developers and testers can follow this process to select test configurations and conduct experiments in order to evaluate the energy performance of their applications . A detailed flow chart was used to explain the process steps, and a specific smartphone, namely *HTC Nexus One*, was used as an example to illustrate the process. As potential future work, the impact of application contents might be studied and considered for selecting test cases.

References

[1] A. Pathak, Y. C. Hu, and M. Zhang, "Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, pp. 5:1–5:6.

[2] R. Arya, R. Palit, and K. Naik, "A methodology for selecting experiments to measure energy costs in smartphones," in *2011 7th International, Wireless Communications and Mobile Computing Conference (IWCMC)*, 2011, pp. 2087–2092.

[3] R. Palit, R. Arya, K. Naik, and A. Singh, "Selection and execution of user level test cases for energy cost evaluation of smartphones," in *Proceeding of the 6th International Workshop on Automation of Software Test*, 2011, pp. 84–90.

[4] K. Y. Cho, S. Mitra, and E. McCluskey, "Gate exhaustive testing." IEEE Computer Society, 2005, pp. 771–777.

[5] J. Czerwonka, "Pairwise testing in real world," in *Proceedings of 24th Pacific Northwest Software Quality Conference*, 2006.

[6] R. Kuhn, Y. Lei, and R. Kacker, "Practical combinatorial testing: Beyond pairwise," vol. 10, no. 3, May-June 2008, pp. 19 –23.

[7] D. Cohen, S. Dalal, M. Fredman, and G. Patton, "The AETG system: an approach to testing based on combinatorial design," in *IEEE Transactions on Software Engineering*, vol. 23, no. 7, Jul 1997, pp. 437–444.

[8] M. Cohen, P. Gibbons, W. Mugridge, C. Colbourn, and J. Collofello, "A variable strength interaction testing of components," in *Computer Software and Applications Conference*, Nov. 2003, pp. 413–418.

APPENDIX

B_i	Parameters	Description	BB 9700	HTC Nexus One	Nokia E71	HTC HD2	iPhone 3GS
01	Display	Size of display	480 x 360 pixels, 2.44”	480 x 800 pixels, 3.7”	320 x 240 pixels, 2.36”	480 x 800 pixels, 4.3”	320 x 480 pixels, 3.5”
02	Operating System (OS)	Name of the OS	BlackBerry OS	Android	Symbian	Windows CE	iPhone OS 3
03	Battery Capacity	Battery type and capacity	Li-Ion 1500 mAh	Li-Ion 1400 mAh	Li-Po 1500 mAh	Li-Ion 1230 mAh	Li-Ion 1250 mAh

Table II
SOME OF SMARTPHONES’ BASIC PARAMETERS.

B_i	Parameters	Description	BB 9700	HTC Nexus One	Nokia E71	HTC HD2	iPhone 3GS
31	Volume	This option allows the user to change the volume level of the device	*Yes* (0,1,…,10)	*Yes* Option 1: Sounds (Silent) Option 2: Volume levels: (0,1,…,15)	Yes volume levels (0,1,…,10)	*Yes* Option 1: Sounds (Silent) Option 2: Volume levels (0,1,…,15)	Yes volume levels (0,1,…,16)
32	Brightness	This option allows the user to change the brightness level of the device	*Yes* (0, 10,…,100)	*Yes* Continuous (0 to 100%)	*Yes* (0, 25,…,100)	*Yes* Continuous (0 to 100%)	*Yes* Continuous (0 to 100%)
33	Bluetooth	This option allows the user to turn on/off their Bluetooth connection	*Yes* (ON/OFF)	*Yes* (ON/OFF)	*Yes* (ON/OFF)	*Yes* (ON/OFF)	*Yes* (ON/OFF)
34	Data Access Mode	This option allows the user to select from WiFi/EDGE/3G connections	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*

Table III
SOME OF SMARTPHONES’ ACTIVE PARAMETERS.

B_i	Parameters	Description	BB 9700	HTC Nexus One	Nokia E71	HTC HD2	iPhone 3GS
61	Network Selection Mode	This option lets the mobile device to select the network manually or automatically	*Yes* (Auto/Manual)	*Yes* (Search automatically)	*Yes* (Manual/Auto)	*Yes* (Auto - Select/Deselect)	*Alternative* (Auto - Select / Deselect)
62	WiFi Settings - Network Notification	This option prompts the user whenever any WiFi network is available	*Alternative* Prompt when manual connection or login is required	*Yes* (ON/OFF)	*Yes* Option 1: Show WiFi availability (Yes/No) Option 2: Scan for Networks (every 1 to 10 min)	*No*	*Yes* (ON/OFF)
63	Portable WiFi Hotspot	This option leads the mobile to act as a WiFi hotspot	*No*	*Yes* Portable WiFi Hotspot (Select/Deselect)	*No*	*Alternative* Internet sharing (Select/Deselect)	*Alternative* (Setup Internet Tethering)
64	Screen Timeout	This option allows the user to set the display timeout for the screen	*Yes* (10 sec to 2 Min)	*Yes* (15 sec to 30 min)	*Yes* (5 to 90 sec)	*Yes* (1 to 10 min on battery power) and (1 to 30 min on external power)	*Yes* (1 to 5 min or Never)
65	Automatic Dim Backlight	This option dims the display light automatically	*Yes* (ON/OFF)	*No*	*Yes* (5 to 60 sec)	*Yes* (10 sec to 5 min on battery power) and (1 to 10 min on external power)	*No*
66	Security - Firewall	The user can enable or disable the Firewall	*Yes* (Enabled / Disabled)	*No*	*No*	*No*	*No*

Table IV
SOME OF SMARTPHONES’ PASSIVE PARAMETERS.