Battery Aware Stochastic QoS Boosting in Mobile Computing Devices

Hao Shen, Qiuwen Chen and Qinru Qiu
Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, New York, USA
{hshen01, qchen14, qiqiu}@syr.edu

Abstract

Mobile computing has been weaved into everyday lives to a great extend. Their usage is clearly imprinted with user's personal signature. The ability to learn such signature enables immense potential in workload prediction and resource management. In this work, we investigate the user behavior modeling and apply the model for energy management. Our goal is to maximize the quality of service (QoS) provided by the mobile device (i.e., smartphone), while keep the risk of battery depletion below a given threshold. A Markov Decision Process (MDP) is constructed from history user behavior. The optimal management policy is solved using linear programing. Simulations based on real user traces validate that, compared to existing battery energy management techniques, the stochastic control performs better in boosting the mobile devices' QoS without significantly increasing the chance of battery depletion.

Key words: mobile, battery, energy, Markov Decision Process

1. Introduction

Mobile computing has been weaved into everyday lives for communication, sensing, controlling and entertainment to a great extend. Many of the applications running on mobile devices can be configured into different levels of quality of service (QoS). For example, by increasing the synchronization frequency between the mobile device and the email server, an email application can receive incoming mails more promptly; by boosting the duty cycle of built-in sensors such as GPS, more accurate environment information can be gathered. The increase of QoS of an application running on mobile computing device always associates with extra energy dissipation [5]. While the progress of battery technology still cannot keep up with the increasing energy demand of the computing devices, traditional energy management of mobile device aims at minimizing energy dissipation. The common practice is to adopt a conservative QoS configuration to trade for longer battery life.

The energy in a mobile computing device is not simply expenditure but rather a dynamic flow that has generation and consumption. For example, most users recharge their smartphones every night. Recent study shows that about 72% of smartphone users will recharge their phone before the battery is low. The advances in energy harvesting techniques also make it possible for future generation smartphones to scavenge ambient RF or solar energy from environment when they are available. Given that the energy is replenishable, it is not necessary to overemphasize on energy saving. A more challenging research topic is how to exploit the potential of future battery recharge to deliver higher QoS. User behavior and preference plays an important role in determining the availability of external energy resources [11]. It is clear that the amount of incoming energy is a stochastic process that is strongly influenced by user behavior.

Smartphone usage and energy management have been considered in many previous researches. Authors of [7] conducted a thorough study on the diversity in smartphone usage and discovered immense diversity among users. These discoveries laid the basis of the user centric mobile device management. Authors of [4], [6], [8], [9] and [10] focus on context-aware mobile device power and performance management. Among these works, reference [4] is the most similar to ours as it assigns excessive energy to boost the performance. It considers the remaining battery energy at the time of battery charge as a random variable and predicts the lower bound of this value with required confidence. The predicted remaining energy is considered as an extra and will then be redistributed to applications to increase their QoS proportionally.

Both [9] and [6] showed that predicting the battery level of mobile device is difficult. The former achieves only 40% average accuracy in battery level prediction during one-day period, while the later predicts the battery charging opportunity at merely 37% accuracy in average and 84% accuracy at best when the model parameters are optimized empirically. Clustering users according to their charging preference can reduce the average prediction error from 60% to about 25% [9]. However, the peak error is still as high as 38% and it always coincides with the initiation of battery charge, which means poor prediction when the battery is low. The low accuracy can be explained by discoveries in [4] and [11], which shows that in terms of battery use and recharge behavior, significant variation exists not only across different users, but also within individual user across his/her own pattern.

In this paper, we investigate more sophisticated models based on neural network for battery prediction. Our results confirmed that deterministic prediction of battery level usually has low accuracy and hence not suitable to guide energy management. We then apply stochastic control to solve the energy management problem. Markov Decision Process (MDP) based and Q-learning based management policies are evaluated and compared.

The research in this paper is enabled by the smartphone usage traces collected by the Livelab [3] project. The traces record various information including battery change, charging time, application usage, IO statistics, Cell Tower ID, Wifi availability and etc. for 34 users over 6~12 months. All of our analyses are carried out on this set of traces.

2. Battery Level Prediction Using Neural Networks

In this section, we study the potential of predictive energy management. The basic idea is to periodically predict the remaining energy level at the time of battery charge and distribute this extra energy to boost the QoS of applications. The key of predictive energy management is the accuracy of battery energy prediction.

We refer to remaining energy level at the beginning of battery charge as our *target variable*, because it is what we need to predict. Three neural networks with different input vectors are trained and tested to predict the next target variable. The first model makes the prediction simply based on current time and battery level and is referred as 2-input model. Compared to the first model, the second model has 24 more input variables that represent the battery level changes of past 24 hours. It is referred as 26-input model. Similarly, the last model, has 5 more input variables than the first model. These 5 variables give the battery level at the beginning of 5 recent battery charges, in other words it uses the 5 recent values of the target variable to predict the sixth one. The model is referred as 7-input model. All three models are trained using the first half of the collected data of different users and tested using the second half of the data.

Figure 1 gives the prediction error for 34 users. The prediction is made every hour and the reported error is the average error of the testing set. As we can see, the 2-input and 26-input model have higher accuracy than the 7-input model. The prediction error of the former ranges from 15% to 22%, which is a little better than the results reported in [6] and [9]. However, there is no significant improvement. The 7-input model has the worst accuracy. This indicates the lack of strong temporal correlation in the target variable.

*This work is supported in part by NSF under grant CNS-0845947

Figure 2 plots the battery level at the beginning of 100 consecutive battery charges. The sequence has large temporal variations and is hard to be predicted based on its previous values. We also found that the prediction error reduces as the prediction time gets closer to the next battery charge. Figure 3 gives the relation between prediction error of the 2-input model and the time to next battery charge. As we can see, when predicted 2 hours ahead of next battery charge, the average absolute error is 15% of overall battery capacity and the relative error is 50%. These numbers increase to 25% and 250% if the prediction is made 17 hours ago. Unfortunately, prediction at earlier time is more important as it provides higher reward.

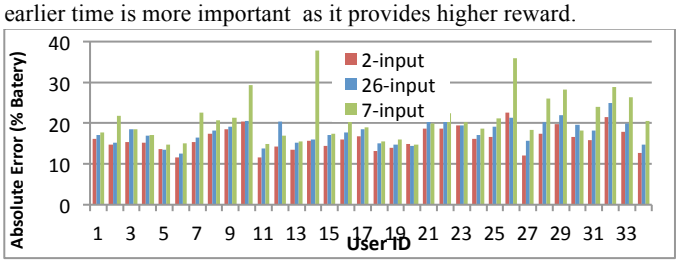


Figure 1 Average prediction error.

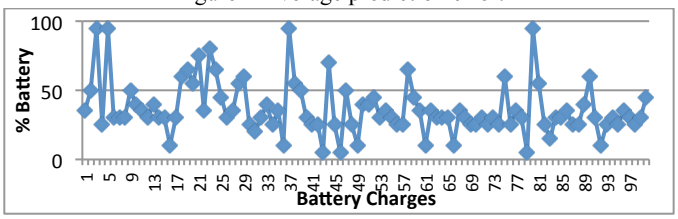


Figure 2 Battery level at the beginning of 100 battery charges

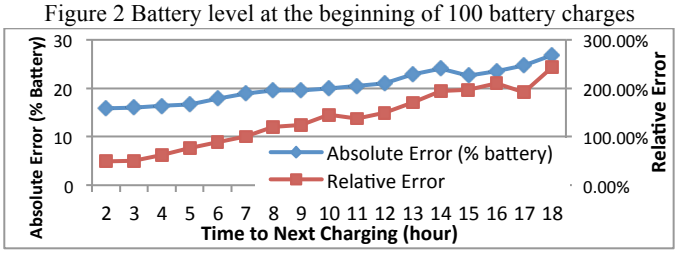


Figure 3 Prediction error vs. time to next battery charge

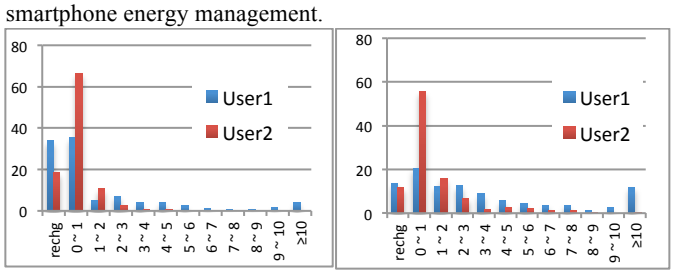
3. Stochastic Control for Smartphone Energy Management

3.1 State space

Using real user usage statistics, [7] confirmed that the usage behavior of different users could be described as the same mathematical model, though probably with different parameters. From [11], we can also see that though very different from user to user, the distribution of recharging level for each user has certain fixed patterns. These findings motivate us to explore stochastic control for the smartphone energy and QoS management. We consider smartphone battery change as a Markov Decision Process (MDP) [1][13] and consider QoS settings of the phone as control actions. The objective is to maximize the average QoS level while keeping the possibility of battery depletion under given threshold. Different users have different recharging behaviors. Some users (Type-A in [11]) charge the phone regularly regardless of the battery level. Some other users (Type-B in [11]) charge the phone only when the battery is low. Such psychological effect is hard to model. In our work, we target at those users whose activities are relatively independent to the battery level.

The first step of model construction is to identify the state space of the MDP. The MDP tracks the change of battery; therefore the first feature that we included is the current battery level itself. Secondly, we found that time is highly correlated to the phone usage and user charging behavior. For the same user, the battery charging usually happens around the same time and the phone usage during the day is also quite stable. For example, Figure **4** gives the histogram of the battery change for two different users during 2~3am(Figure **4**(a)) and 2-3pm (Figure **4**(b)). The left most set of data in both figures gives the percentage of time that the smartphone is recharging. It is labeled as "rechg". The next 11 sets of data give the percentage of time that the smartphone consumes 0~1%, 1~2%,…., 9~10%, and 10~100% of battery energy during the recorded time period. As we can see, both users have higher possibility to charge their phone during 2~3am than 2~3pm. The chance to have nonzero battery energy dissipation is higher during 2~3pm than 2~3am. We also see that User1 has more intensive smartphone usage than User2 and also charges more often as a consequence.

Figure **4** further confirms the rationale of using stochastic model for smartphone energy management.



(a) Usage during 2~3am (b) Usage during 2~3pm

Figure 4 Battery change histogram

In addition to time and battery level, we are also interested to find out if other features, such as battery change rate, current and previous location, phone sleep time, should be included in the state space. We have found that they are all highly correlated to the time and battery level, thus do not provide much new information.

Correlations between future phone usage and previous phone usage are also calculated. We use hourly battery change rate $\Delta B(t)$ and smartphone sleep time $T_{sleep}(t)$ to represent phone usage during time slot *t*. The duration of each time slot is set to 1 hour. Figure 5 gives the correlations between $\Delta B(t)$ and $\Delta B(t-i)$, as well as the correlations between $\Delta B(t)$ and $T_{sleep}(t-i)$, $0 \le i \le 3$. As we can see, the battery change rate and phone sleep time in even one hour ago has low correlation with current battery change rate. And the correlation keeps on reducing when the distance in time increases. This indicates that the previous phone usage does not provide much help in predicting the future battery change either. Including it in the state space will not improve the model accuracy but add model complexity.

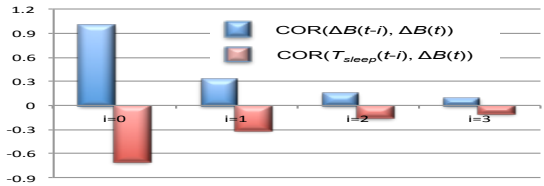


Figure 5 Temporal correlation of phone usage.

We denote the state vector of the MDP as $(t, B(t))$, where *t* is the time at the beginning of current time slot and $B(t)$ is the battery level at time *t*. A state *i* is a *depletion* state if $B(t) = 0$, which indicates the depletion of the battery. Associated to each state *i*, there is a set of *K* actions $a_i^k \in A(i) = \{a_i^0, a_i^1, a_i^2 \ldots, a_i^{K-1}\}$. Each action corresponds to a QoS level of the phone and we assume that the average power consumption of the phone at the *k*th QoS level is known. The reward of state *i*, denoted as $r(i, a_i^k)$, is set proportional to the chosen QoS level a_i^k.

3.2 MDP training and solving

By observing the history of smartphone activities, we train the MDP model for each user. The training process is to determine the

transition probability ($p_{i,j}(a_i^k)$) from state i, to state j under action a_i^k and the reward $r(i, a_i^k)$.

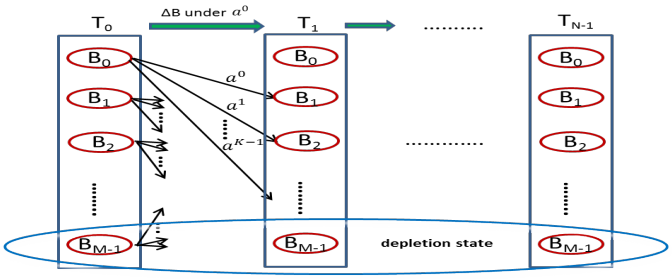


Figure 6 MDP training process

The detail of our MDP model training is shown in Figure 6. Each red circle in the figure represents a distinct state. The entire state space is divided into N groups. Each group corresponds to a specific time (i.e. t) in the state vector. With each group there are M states corresponding to M battery levels (i.e. B in the state vector). B_{M-1} is the battery depletion state which should be avoided. From a state in group T_n, the system will go to a state in group T_{n+1}. If n=N-1, then the system will go to a state in group T_0. The training sequence is collected from system without any QoS boost, in other words, only the action a^0 is taken in the original training trace and the battery change ΔB under a^0 is recorded. We assume 70% hourly battery increase rate during battery charging. Because we target only at users whose activities are independent to battery level, we assume the same usage activity happens regardless of the current status of battery. The recorded information ΔB reflects the workload activities, from which we can estimate the battery change for other QoS settings $a^k, k \neq 0$. The amount of battery change decides which state the system will transit into. In this way, for every remaining battery state B belonging to the same group and every action a of this state, we can calculate the next battery state B' and reward r. Then after training, the transition probability ($p_{i,j}(a_i^k)$) can easily be calculated and the reward $r(i, a_i^k)$ is calculated as the average of the rewards received under this state action pair.

The optimal policy is found by solving the mathematical program as following [2]:

$$\max \sum_{i \in S} \pi_i \sum_{a \in A(i)} f(i,a) r(i,a) \qquad (1)$$

subject to

$$\pi_j = \sum_{i \in S} \pi_i \sum_{a \in A(i)} p_{ij}(a),\ j \in S \quad (2) \qquad \sum_{a \in A(i)} f(i,a) = 1, \quad (3)$$

$$f(i,a) \geq 0, \quad (4) \qquad \sum_{i \in S_{Depletion}} \pi_i < L, \quad (5)$$

where π_i is the stationary distribution for state i. $f(i,a)$ is the probability that an action a is taken if the state of the system is i and it is the set of variables that we need to optimize. $r(i,a)$ is the reward received if action a is taken in state i. $p_{ij}(a)$ is the transition probability from state i to state j, given that action a is taken at state i. S is the state space of the system. Equation (2) constraints the balance of the state probability and transition probability. Equation (3) specifies that probabilities of all actions taken in a state should add up to 1. The constraint (5) specifies that the overall probability of those depletion states should be less than L. L is a small number possibly given by the user based on their specific tolerance of battery depletion. Higher tolerance usually will lead to more performance boost opportunities.

By introducing a set of new decision variables x_{ia}, $x_{ia} = \pi_i f(i,a), i \in S, a \in A(i)$, the above non-linear problem can be transformed into a linear one and solved.

4.Implementation and Evaluation

4.1. Implementation

We use the real user traces from the Livelab project [3] as we mentioned before. We use four-way cross-validation [14] in the experiments. Traces of every user are partitioned into 4 equal sized subsets, 3 of which form the training set and 1 of which will be the testing set. The final result is the average of all tests. A simulator is implemented using C++ to evaluate the policy. If the user tolerance of battery depletion (i.e. L in constraint (5)) is too low, no feasible solution can be found. A best effort solution will be used instead.

It is assumed that a phone has three QoS levels corresponding to 1x, 1.5x and 2x of the default setting. The energy dissipation of the phone is assumed to be proportional to its QoS level and the original energy dissipation when no boost was performed. This assumption is used only to simplify the experiment setup. How to adjust the QoS of different applications and what is the relationship between the QoS and power consumption are nontrivial problems outside the scope of this paper [5].

In addition to the MDP based approach, three reference approaches are also simulated. The first one is Q-learning based approach[12] which generally shares the same underlying model with MDP but is an online learning method. We refer to this method as '*ML*' in our simulation. The second one is based on [4], which profiles the histogram distribution of the remaining energy at the time of battery recharge. The profiled information will be used to predict the lower bound of the remaining energy and the QoS of the phone will be raised proportionally based on the estimation. A confidence level is determined based on the profiled distribution that specifies the probability that the prediction is correct. We refer to this method as '*HIST*' in our simulation. The profiling is performed on the training set and the policy is tested on the testing set. The third reference policy is prediction-based approach, which predicts the remaining energy using the 2-input neural network as described in Section 2. Similar to HIST, the QoS will be raised based on the prediction. We refer to this method as '*Nnet*'.

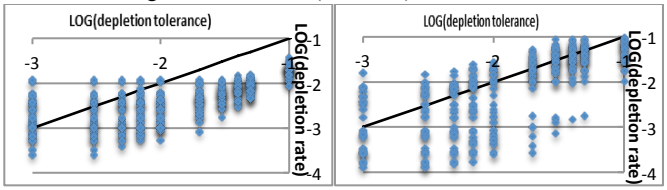
4.2. Evaluation Result

Different level of user tolerance of battery depletion leads to different potential of QoS boost. For the MDP method, varying L in Equation (5) gives the indication of different tolerance of the battery depletion. For example, L=0.01 means that the user can tolerate 1% chance of battery depletion during the entire smartphone usage in exchange for performance boost. For HIST, the confidence level is set to be $1 - tolerance$. For ML method, the battery depletion tolerance is tracked using a feedback control method by dynamically changing the penalty of the battery depletion state. All the results following are based on four-way cross validation.

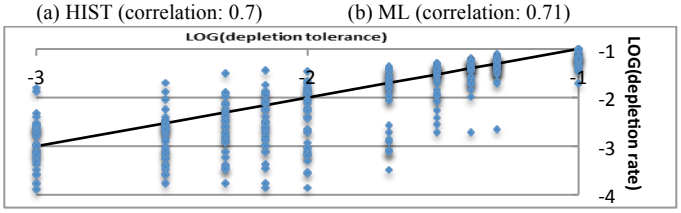
For all the 34 users, we vary the depletion tolerance from 10% to 0.1% and recorded the amount of QoS boosts and actual battery depletion rate. The results are shown in Figure 7.

Figure 7 (a)~(c) compares the actual battery depletion rate (Y-axis) with the depletion tolerance (X-axis) for all 34 users under different management algorithms. The performance of Nnet is not shown here, because there is no way to integrate the user depletion tolerance with the prediction based management. The black line in the figure represents the ideal cases where the actual depletion exactly meets the constraint. The points above the line correspond to systems that are under-constrained and have depletion violation while the ones below the line are systems over-constrained. Note that both X and Y axes are logarithmic, therefore the difference between the actual and the constraint is magnified when depletion tolerance is low. As shown in the figure, in most cases all 3 algorithms tend to over-constrain than under-constrain points. When depletion tolerance is high (i.e. loose constraint), HIST is more conservative than ML and MDP and all cases using HIST are over-constrained. The correlation between actual depletion and depletion tolerance is calculated and

given in the figure. The higher correlation means more précised management. As we can see, the system using MDP management achieves the highest correlation (i.e, 0.78).

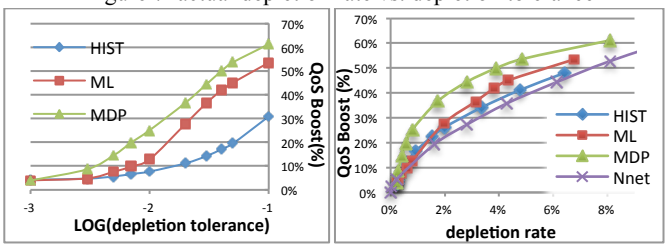


(a) HIST (correlation: 0.7) (b) ML (correlation: 0.71)



(c) MDP (correlation: 0.78)

Figure 7 actual depletion rate vs. depletion tolerance



(a) (b)

Figure 8 QoS Boosts vs. (a) depletion tolerance (b) actual depletion rate

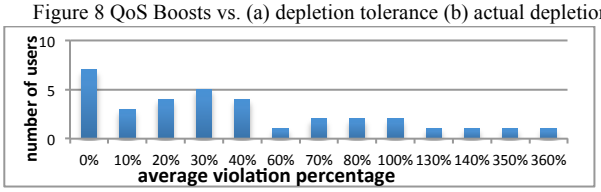


Figure 9 . MDP violation percentage histogram

Figure **8** (a) shows the percentage QoS boost under different depletion tolerance compared to nominal case without any QoS boost. The results reported here is the average of all 34 users. As we can see, MDP gives the most QoS increase and the HIST gives the least. The low violation and low performance boost shows that the HIST is much more conservative than MDP and ML especially when the depletion tolerance is loose. Figure **8** (b) shows the relation between the QoS boosts and the actual battery depletion rate collected from simulations. In the figure, the curves at the upper left are better than the curves in the lower right. Again, the results reported here is the average of all 34 users. Note that this figure shows the tradeoff between QoS boost and the actual battery depletion rate. Those data points with similar X values might not correspond to the same depletion tolerance constraint. For Nnet, the tradeoff curve is obtained by varying the amount of energy that is distributed for QoS boosting as different portions of the predicted target variables. As we can see, the MDP method gives the highest QoS improvement than others with the same battery depletion, while the Nnet gives the lowest QoS improvement. It shows that the stochastic control method outperforms others as it achieves better QoS and energy reliability tradeoffs by better tracking different user's battery usage and recharge patterns.

One of the concerns for the stochastic approaches is that they still have a few violations (under-constraints) when depletion tolerance is not very tight while the HIST has not violation at all as shown in Figure 7. The foremost reason is that setting confidence level of HIST to $(1 - tolerance)$ is very conservative and will prevent QoS boost. On the other hand, the phone usage and battery charging pattern for some users are not always consistent during all the time and it is hard to capture their behavior using simple MDP models as we did. Figure **9** shows the histogram distribution of average degree of violation for those 34 users under the MDP based energy management. The average degree of violation is calculated as $\max\left[0, \frac{depletion_{actual} - depletion_{tolerance}}{depletion_{tolerance}}\right]$. Please note that the degree of violation is defined as the relative increase of battery depletion compared to the tolerance. For example, if the depletion tolerance is 1% and the actual depletion is 2%, then the degree of violation is 100%. The X-axis in Figure **9** gives the range of average degree of violation, 0% stands for the range [0%, 10%]. The Y-axis is the number of users whose average depletion rate falls in the corresponding range. For instance, among 34 users, there are 7 users whose average degree of violation is between 0%~10% and 23 users whose degree of violation is less than 50%. As we can see, the majority of users have reasonable degree of violations, only some "inconsistent" users deteriorate the results greatly. For those users, more sophisticated models may need to be developed.

5. Conclusions

In this paper, we aim at increasing the QoS of mobile devices considering the fact that many users recharge the battery before depletion. Neural network model that predicts the remaining energy at next battery charge is first investigated. The results show that accurate prediction is difficult. Then we present a stochastic framework for QoS boosting under the user specified battery depletion tolerance. The model is trained using real user traces and the framework is simulated and compared with existing approaches.

6.References

[1] Y. Tan and Q. Qiu, "A Framework of Stochastic Power Management Using Hidden Markov Model," *DATE'08*, pp.92-97, 2008

[2] D. Bello and G. Riano, "Linear Programming solvers for Markov Decision Processes," *SIEDS'2006*, pp.90-95, Apr 2006

[3] C. Shepard, A. Rahmati, C. Tossell, L. Zhong and P. Kortum, "Livelab: Measuring Wireless Networks and Smartphone User in the Field," *ACM SIGMETRICS Perfrom. Eval. Rev.*, vol.38, no.3. Dec 2010

[4] N. Banerjee, A. Rahmati, M. D. Corner, S. Rollins and L. Zhong, "Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems," *UbiComp'07*, pp.217-234, 2007

[5] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," *SOSP'99*, pp.48-63, 1999

[6] N. Ravi, J. Scott, L. Han and L. Iftode, "Context-aware Battery Management for Mobile Phones," *PerCom 2008*, pp.224-233, Mar 2008

[7] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan and D. Estrin, "Diversity in Smartphone Usage," *MobiSys'10*, pp.179-194, 2010

[8] Y. S. Lee and S. B. Cho, "An Efficient Energy Management System for Android Phone Using Bayesian Networks," *ICDCSW'2012*, pp.102-107, Jun2012

[9] E. A. Oliver and S. Keshav, "An Empirical Approach to Smartphone Energy Level Prediction," *UbiComp'11*, pp.345-354, 2011

[10] T. Yan, D. Chu, D. Ganesan, A. Kansal and J. Liu, "Fast App Launching for Mobile Devices Using Predictive User Context," *MobiSys'12*, pp.113-126, 2012

[11] A. Rahmati, A. Qian and L. Zhong, "Understanding Human-Battery Interaction on Mobile Phones," *MobileHCI'07*, pp.265-272, 2007

[12] H. Shen, Y. Tan, J. Lu, Q. Wu and Q. Qiu, "Achieving Autonomous Power Management Using Reinforcement Learning," *TODAES*, vol.18, issue.2, 2013

[13] Wiki "Markov Decision Process " : http://en.wikipedia.org/wiki/Markov_decision_process

[14] Jiawei Han and Micheline Kamber, "Data Mining Concepts and Techniques," Second Edition, Morgan Kaufmann Publishers, March 2006