Runtime Adaptation of Applications Using Design Of Experiments: A Smartphone-Based Case Study

Frank Maker, Rajeevan Amirtharajah, *Member, IEEE*, and Venkatesh Akella

***Abstract*—We consider the problem of adapting embedded software to heterogeneous devices where it is impractical to obtain a system-level power model for each target platform and operating environment. Our solution leverages the emerging capability of measuring power consumption at run-time using a built-in battery monitoring unit (BMU). We use a statistically rigorous design of experiments (DoE) methodology to efficiently characterize the power consumption configuration space of software online instead of constructing a system-level power model offline. This approach is simple, low-cost, and permits software to dynamically select suitable parameters at run-time to satisfy energy or performance constraints. We illustrate this approach using a camera trap application as a case study deployed on a Nokia N80 smartphone.**

***Index Terms*—Design of experiments, embedded software, heuristic algorithms, predictive models, table lookup.**

I. Introduction

A. Problem Statement

Our goal in this work is to create "write once, adapt everywhere" embedded software that automatically *adapts* software parameters online *without* domain knowledge to satisfy constraints on energy consumption, performance, or both. This allows devices to adjust performance based on available energy or vice versa. These constraints may come from user demands, the operating system, or a power management controller. In addition to time-varying constraints (Assumption A1), we assume the following:

- A1: application has sufficient heterogeneity (i.e., diversity) that developing a universal power model is impractical;
- A2: exhaustively evaluating the software configuration space online would deplete limited system energy;.
- A3: application tolerates an initial training period where constraints may not be satisfied;
- A4: system has a built-in power measurement capability that consumes negligible power.

B. Related Work & Challenges

One key distinction between this and related embedded software power optimization work is Assumption A2. This is particularly important on embedded systems with substantial heterogeneity such as smartphones, single-purpose devices and sensor networks. Smartphone heterogeneity stems from applications running on devices by different manufacturers, platforms, software versions, and varying user demands. Single-purpose devices (cameras, video recorders, smart watches, etc.) have similar heterogeneity as their software platforms converge. Sensor network heterogeneity results from software running on multiple generations of nodes and variations in the harvested energy (solar, wind, etc.) available.

Most embedded software power models are generated offline using software predictors. Predictor values are recorded along with system power using benchtop power measurement equipment while running benchmarks. Models are fitted to the recorded data using linear regression [1], [2]. These models have two drawbacks: 1) the benchmarks used to build each model can have limited accuracy for nonbenchmarked applications; and 2) analytical models are unable to capture the multiple states of components which impact power consumption. System call finite state machine models [3] can overcome these limitations, but require significant development time and system knowledge, which makes targeting heterogeneous devices with the same model intractable.

C. Key Contributions

Online modeling with built-in power monitoring [1], [4] enables application benchmarking at run-time. However, previous work has not considered how to manage the energy of online power modeling itself. If this energy is well-managed then embedded software can build a run-time model to adapt software configurations to different energy and performance constraints. We propose a novel technique using built-in power monitoring (Assumption A5) and Design of Experiments to efficiently survey the configuration space of software and to initialize a look-up table (LUT). This table is then consulted to find configurations that satisfy constraints at run-time. By using a LUT, our model captures multiple component states for better accuracy. We will demonstrate how this very simple strategy can guarantee efficient configuration space coverage.

There are several applications which satisfy assumptions A1-A5 and can use this technique such as: camera traps, video playback, and wireless data transmission. To evaluate the performance of our approach, we present a case study using a camera trap in Section II. Section III details how DoE was used for online modeling. Section IV compares the performance of DoE to random sampling and heuristic configuration searches, and lastly in Section V we present our conclusions.

Manuscript received January 21, 2013; revised April 28, 2013; accepted September 07, 2013. Date of publication January 21, 2014; date of current version May 23, 2014. This work was supported by the Nokia and NSF CAREER Award #0547113. This manuscript was recommended for publication by M. Balakrishnan.

The authors are with the Department of Electrical and Computer Engineering, University of California, Davis, Davis, CA 95616 USA (e-mail: flmaker@ucdavis.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2014.2301692

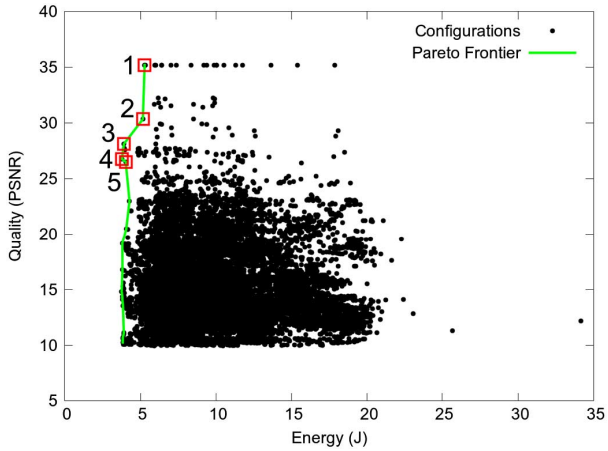

Fig. 1. Configuration space (energy × quality).

II. Camera Trap Case Study

A camera trap is a battery powered device with a motion sensor to trigger photographs. Their use has grown exponentially in biology field research with 19% more publications using camera traps each year since the 1990's [5]. They are used for evaluating population size and diversity, nest ecology[1], wildlife photography, detecting rare species, studying human-built structure occupation, and habitat monitoring [6].

Existing traps use a static configuration to capture photos, however we modeled a camera trap which uses online adaptation with one of three constraints: energy, image quality, and energy efficiency (quality/energy). These constraints reflect the tradeoff between energy and quality for each use case. For population size and nest ecology (to maximize image count) we used energy constraints. For wildlife viewing and rare species detection we used quality constraints. For the remaining use cases we employed energy efficiency to search for pareto optimal solutions[2] which had the largest energy efficiency values in our configuration space (see pareto frontier in Fig. 1).

Next we discuss how a camera trap can use one of these constraints, energy, to perform online adaptation. The user first deploys the camera trap, programs the deployment duration using a built-in dial, and presses the start button. This button press triggers a DoE survey to build a look-up table for the new location using a built-in battery monitor unit (BMU) to measure the energy consumption of each configuration. Afterwards, when the motion sensor is triggered an energy constraint is calculated using the deployment duration, remaining system energy, and a preprogrammed range (e.g., $\pm 10\%$).

The system next tries to retrieve a configuration from the LUT that satisfies the constraint within the specified range. This table is small and can be easily cached in main memory because it contains only each configuration's parameter values, image quality, and energy requirement. If no configuration is returned, an iterative search is performed for a satisfactory configuration. This search is limited to evaluating a fixed number of configurations to avoid an exhaustive search. If a suitable configuration is still not found, the closest configuration is used instead and the same constraint in future requests will trigger further searching.

[1]Nest ecology is the study of how organisms construct and use their nests.

[2]The lowest energy configurations for quality constraints are pareto optimal.

This camera trap scenario fulfills the assumptions (A1–A5) required for online adaptation.

- A1: Motion trigger events generate energy constraints which will vary over time based on the number of triggered photos and the battery energy depleted while idle.
- A2: Traps are often relocated and parameters must adapt to new conditions such as lighting conditions, depth of field, and background scenery [7]. These conditions correspond to selecting camera parameters such as white balance, zoom, and image compression, respectively.
- A3: We estimated the initial training period impact using a published camera trap study [7] with Equations 1 and 2. The trap captured 2.6 images per day, 30 μHz (f), over 30 days. Because of the low image frequency most of the system energy was used while idle. We estimated this usage (corresponding to system overhead) to be 50 mW (P_{sys}) by dividing the battery energy, 129.6 kJ ($E_{\max}$), by the deployment duration. According to the manufacturer, each capture (including system overhead) requires 4.86 J (E_{task}) and 0.5 s (T_{task}) [8]. We can then calculate the deployment time (T) based on the number of training configurations (n_C)

$$T \leq \frac{E_{\max} - n_C(E_{\text{task}} - P_{\text{sys}}T_{\text{task}})}{P_{\text{sys}} + f(E_{\text{task}} - P_{\text{sys}}T_{\text{task}})}. \quad (1)$$

  Since T_{task} is negligible, Equation 1 can be simplified

$$T \leq \frac{E_{\max} - n_C E_{\text{task}}}{P_{\text{sys}} + f E_{\text{task}}}. \quad (2)$$

  Assuming an exhaustive search over a 7-bit configuration space (128 configurations) then $T = 29.8$ days and exhaustive characterization has negligible impact. However, if we assume a 13-bit configuration space (8,192 configurations), then $T = 20.8$ days and there is a 31% reduction in operating lifetime.
- A4: Deployments last from days to several weeks [7], providing sufficient time for training.
- A5: We used a benchtop power monitor, however new cameras traps could use a built-in BMU.

Online adaptation's efficacy for camera traps depends on the energy balance between capturing images and system overhead for each particular use case. For example, rare species detection may require only 2.6 images per day [7], representing 0.29% of the battery energy. However, nest ecology would demand a higher image rate, for example one image every 30 seconds thus consuming 50% of the battery. The scope for energy savings lies between these two scenarios.

A. Experimental Setup

Camera trap images are usually stored for later collection [7], however newer traps transmit photos wirelessly [6]. Since most devices currently lack this feature, we modeled a wireless camera trap using a smartphone instead. To verify that a smartphone was an appropriate power model, we measured the energy for capturing and transmitting an image using the same configuration on three smartphones: HTC Magic, Google Nexus One, and Nokia N80. Each required 610, 522 and 1074 mW of power and 3.69, 5.41, and 7.14 s, resulting in 2.25, 2.83, and 7.67 J of energy respectively. Idle power ranged from 1 to 5 mW, which

is an order of magnitude smaller than the camera trap. However, the energy capacity was also an order of magnitude smaller, resulting in a simliar operational time.. For example, the Nokia N80 with an 820 mAh battery stored 10.9 kJ, required 5 mW when idle, and would last 25.28 days, which was similar to the durations from Assumption A3 in the previous section.

To compare configurations we used a standard test image (Lena), under the same conditions, and quantified picture quality using peak signal-to-noise ratio (PSNR). To select a reference image, or "signal," for calculating PSNR we employed a focus group of five people to rank reference images based on perceived quality. The selected reference images were then removed from our experiments, since they have infinite PSNR values. Energy measurements were averaged continuously using a general purpose interface bus (GPIB) controlled power supply, stored in a SQLite database for simulations, and required approximately a month.

Fig. 1 shows the range of energy and quality values measured (19 200 total) for the camera trap application on a Nokia N80 smartphone. The majority of the configurations, 97%, consumed between 4–21 J of energy and had picture qualities between 10–24 PSNR, with an additional 296 configurations with higher PSNR outside this cluster. The most efficient configurations (i.e., the highest quality with least energy) were found along the left-hand side (Pareto frontier).

The best fit frontier line had a slope of 12.42 meaning the most efficient solutions required one joule of energy for each 12.42 increase in PSNR. The five highest efficiency configurations are labeled in Fig. 1. These configurations shared some settings (Bluetooth, no flash, 640 x 480) that match intuition based on knowledge of the application domain, but others (*Exposure* set to *Night* versus *Backlight* versus *Auto*) did not.

III. Design of Experiments Approach

Design of Experiments (DoE) is a statistical methodology used in the natural sciences, agriculture, and manufacturing to design a series of experiments when they are subject to time financial constraints [9]. DoE efficiently samples the experimental space to maximize the information obtained and offers a statistically rigorous alternative to searches based on heuristics. This survey can be used to establish domain knowledge automatically. In computer engineering it is primarily used for computer architecture design space exploration (DSE) due to the extensive search time required [10].

Each variable in a DoE experiment is called a *factor*. Factors with a natural numerical ordering are *quantitative* factors and those without are *qualitative* factors. We chose a design in this work that uses all combinations of the minimum and maximum values for each factor. This approach is known as a 2^k *full factorial design* where k is the number of factors with 2 levels (minimum and maximum) used. We used a full factorial design because it is the most comprehensive for factors with two levels, however other designs such as randomized blocks, Latin squares, and Graeco-Latin squares can be used if nuisance factors are present. If too many experiments are required for a full factorial, a fractional factorial approach, such as a Plackett–Burman design, can reduce the number of experiments at the expense of not detecting higher order interactions [9].

TABLE I
Configuration Space Parameters

Param.	**Setting**				
Radio	Bluetooth	Wi-Fi			
Exposure	Auto	Center	Backlight	Night	
Flash	Auto	Forced	Red-Eye	None	
Color	RGB24	RGB16	RGB12	JPEG	
Size	640x480	800x600	1280x960		
White Bal.	Auto	Cloudy	Daylight	Fluor.	Tungsten
Zoom	0	8	16	...	72

To use DoE in our application we mapped experiments to configurations and factors to camera settings. However, our configuration space (see Table I) had only two quantitative parameters (zoom and size) and the remaining qualitative parameters had to be evaluated individually. For radio and flash the highest and lowest settings were clear: Bluetooth versus Wi-Fi and None versus Forced respectively. For color mode, we used the largest uncompressed setting, RGB24, and the only compressed setting, JPEG. For exposure and white balance mode we did not identify any extrema and instead distributed the values across each experiment. In Design 1 (D1) we used all 32 combinations of the quantitative parameters (zoom, size, radio, flash, color) with the whitebalance value changed every experiment and exposure every other experiment. In Design 2 (D2), Design 1 was repeated four times to cover more quantitative-qualitative interactions. Lastly, in Design 3 (D3) all 640 combinations ($2^5 \times 5 \times 4$) of all values were used.

IV. Results and Discussion

How well does DoE perform? Table II shows the results of the designed surveys averaged over ten simulations of 500 image captures for randomly generated energy, quality, or energy efficiency constraints (fulfilling Assumption A1). Constraints were uniformly distributed between the extrema values (see Fig. 1) and were satisfied if a configuration was found within 500 search iterations and $\pm 10\%$ of the constraint. D1, with only 32 configurations, satisfied 56% of the energy and energy efficiency constraints and 81% of the quality constraints. Energy constraints had the lowest success rate because 37% of the values lie within range of only three outliers at 23.03, 25.66, and 34.15 J. D3, with a table size of 640, satisfied over 96% of the efficiency and 100% of the image quality constraints, but required an additional six hours and 4.7 kJ. The surveys performed well considering they did not require domain knowledge and were easy to deploy.

Why not just initialize the table with a random survey? A random survey is a simpler alternative to DoE for sampling a configuration space without domain knowledge. We compared our designed survey (D2) to an equal size random survey (Random) to compare their coverage of the configuration space. Results from both surveys are shown in Table II. Random performed slightly better on energy constraints because of its ability to discover outliers however, D2 was more successful for efficiency and quality constraints by covering a larger range of values. We also considered the limits of satisfiable constraints for each survey. The heat maps in Fig. 2 identify the number of configurations which satisfied energy and quality constraints within $\pm 10\%$ using both surveys. Random covered

TABLE II
RESULTS FROM 10 SIMULATIONS WITH 500 CAPTURES

	Survey & Algorithm	Success (%)	Table Size	Duration (h)	Energy (J)
Energy	D1	56	32	0.3	303
	D2	57	128	1	1,206
	D3	57	640	7	5,915
	Random	60	128	1	1,296
	D2 + Genetic Alg.	65	699	8	7,945
	D2 + Hill Climbing	64	1,782	21	24,625
	D2 + Random Greedy	85	16,176	176	162,590
	D2 + Random New	88	19,199	209	192,854
	D2 + Exhaustive	100	19,200	210	192,871
Efficiency	D1	56	32	0.3	303
	D2	81	128	1	1,206
	D3	96	640	7	5,915
	Random	58	128	1	1,286
	D2 + Genetic Alg.	94	302	3	2,912
	D2 + Hill Climbing	94	860	9	9,972
	D2 + Random Greedy	97	5,703	62	57,194
	D2 + Random New	97	6,103	66	61,433
	D2 + Exhaustive	100	19,200	210	192,871
Quality	D1	81	32	0.3	303
	D2	100	128	1	1,206
	D3	100	640	7	5,915
	Random	78	128	1	1,267
	D2 + Genetic Alg.	100	128	1	1,205
	D2 + Hill Climbing	100	128	1	1,205
	D2 + Random Greedy	100	128	1	1,205
	D2 + Random New	100	128	1	1,205
	D2 + Exhaustive	100	128	210	192,871

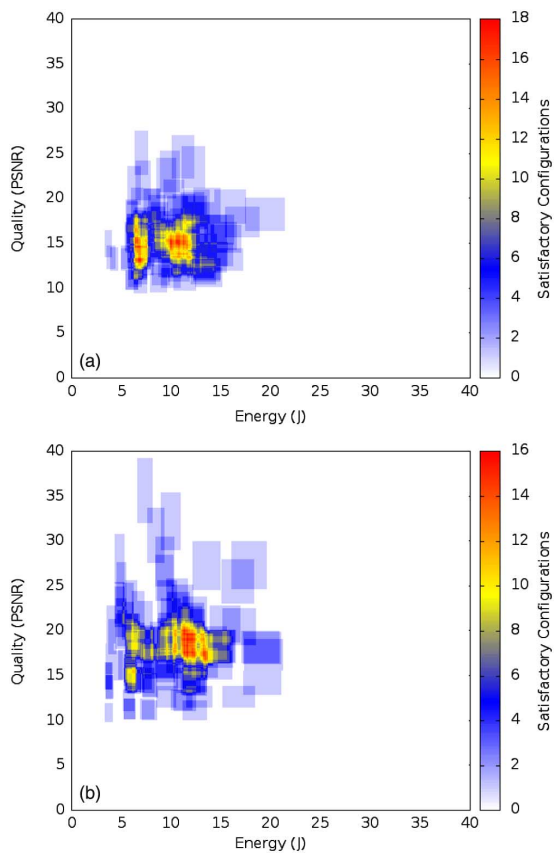

Fig. 2. Designed versus Random configuration space survey (a) Random Survey of 128 Configurations (b) Designed Survey of 128 Configurations (D2).

the highest density regions, but could discover outliers whereas our designed survey covered a larger range.

Does postinitialization search improve performance? Next we evaluated our DoE survey with a limited heuristic search to satisfy constraints when a request cannot be satisfied from the initial survey. We considered five different heuristics: 1) genetic algorithm; 2) hill climbing; 3) random greedy; 4) random new (required each random configuration to be previously unevaluated); and 5) exhaustive search.

For energy constraints the nonrandom searches (genetic algorithm and hill climbing) were 7.5% more successful than each DoE survey, requiring an additional 2–19 kJ (2%–15% battery life) and 1–14 hours (0.1%–1.9% deployment time). Both Random Greedy and Random New were 29.5% more successful on average than D1, D2, and D3. Both random searches were also more successful, but used 155–185 kJ more energy (120%–142% battery life), and 169–202 more hours (23%–28% deployment time), than the largest survey (D3). For energy efficiency constraints D3 performed better than nonrandom searches and within 1% of both random searches using 51–56 kJ less energy and 55–59 less hours. Quality constraints were easily satisfied by D2 without searching and therefore the energy and time requirements were the same.

V. CONCLUSION

In summary, we presented a DoE approach for runtime adaptation of embedded software on heterogeneous devices without domain knowledge. We outlined the requirements for this technique and a representative camera trap case study. A designed survey with only 32 configurations covered the configuration space better than random sampling, fulfilled 56% of energy and energy efficiency constraints, and 81% of quality constraints. For energy efficiency and quality constraints a 640 configuration survey was only 1% less successful than the best search and required 56 kJ less energy and 59 hours less. For energy constraints the best search satisfied 31% more constraints than the same survey, but evaluated 99.99% of the search space. An extension of this work has also been published with a sensitivity analysis to resource constraints and platform heterogeneity [11].

REFERENCES

[1] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign Syst. Synth. (CODES/ISSS '10)*, New York, NY, USA, Oct. 2010, p. 105.

[2] M. Dong and L. Zhong, "Self-constructive high-rate system energy modeling for battery-powered mobile systems," *Interface*, 2011.

[3] A. Pathak *et al.*, "Fine-grained power modeling for smartphones using system call tracing," in *Proc. 6th Conf. Comput. Syst. (EuroSys '11)*, New York, NY, USA, 2011, p. 153.

[4] W. Jung *et al., DevScope: A Nonintrusive and Online Power Analysis Tool for Smartphone Hardware Components*, 2012.

[5] J. Rowcliffe and C. Carbone, "Surveys using camera traps: Are we looking to a brighter future?," *Animal Conserv.*, vol. 11, pp. 185–6, Jun. 2008.

[6] K. U. K. Allan *et al.*, "Evaluating Types and Features of Camera Traps in Ecological Studies: A Guide for Researchers," in *Camera Traps in Animal Ecology: Methods and Analyses*. Berlin, Germany: Springer-Verlag, 2010, ch. 3.

[7] R. Kays *et al.*, "Camera traps as sensor networks for monitoring animal communities," in *Proc. 4th IEEE Int. Workshop Practical Issues Building Sensor Netw. Appl. (SenseApp '09)*, 2009, pp. 811–8.

[8] Reconyx, Inc., "Professional research camera traps," Jul. 2012.

[9] D. C. Montgomery, *Design and Analysis of Experiments*. Hoboken, NJ, USA: Wiley, 2001.

[10] D. Sheldon *et al.*, "Soft-core processor customization using the design of experiments paradigm," in *Proc. 2007 Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.

[11] F. Maker *et al.*, "MELOADES: Methodology for long-term online adaptation of embedded software for heterogeneous devices," *J. Syst. Arch.*, vol. 59, no. 8, pp. 643–655, Sep. 2013.