# Near-Optimal Energy Allocation for Self-Powered Wearable Systems

Ganapati Bhat, Jaehyun Park and Umit Y. Ogras
School of Electrical Computer and Energy Engineering, Arizona State University, Tempe, AZ
Email: {gmbhat, jpark244, umit}@asu.edu

***Abstract*—Wearable internet of things (IoT) devices are becoming popular due to their small form factor and low cost. Potential applications include human health and activity monitoring by embedding sensors such as accelerometer, gyroscope, and heart rate sensor. However, these devices have severely limited battery capacity, which requires frequent recharging. Harvesting ambient energy and optimal energy allocation can make wearable IoT devices practical by eliminating the charging requirement. This paper presents a near-optimal runtime energy management technique by considering the harvested energy. The proposed solution maximizes the performance of the wearable device under minimum energy constraints. We show that the results of the proposed algorithm are, on average, within 3% of the optimal solution computed offline.**

## I. Introduction

Advances in low power sensor, processor and wireless communication technologies enable a wide range of wearable applications. For instance, small form factor and low cost IoT devices offer a great potential for non-invasive healthcare services which are not limited to any specific time or place [3, 7]. Exciting, and possibly pervasive, applications include health monitoring, activity tracking and gesture-based control [29]. However, small form factor and low cost constraints severely limit the battery capacity. Therefore, harvesting ambient energy and optimal energy allocation are crucial for the success of wearable IoT devices.

Energy limitation is one of the major problems faced by wearable applications. Bulky batteries are heavy and inflexible, while small printed batteries have modest (3.16–29.6 mWh/cm$^2$) capacity [24, 25, 41], which requires frequent charging. Therefore, it is imperative to exploit ambient energy sources such as light, motion and heat. Recent studies show that photovoltaic (PV) cells can provide 0.1 mW/cm$^2$ (indoor) – 100 mW/cm$^2$ (outdoor) power [39]. Similarly, human motion and heat can generate 0.73 mW/cm$^3$ [10] and 0.76 mW/cm$^2$ power at $\Delta$T = 10 K [22], respectively. Energy harvesting can be particularly effective for wearable devices, since they are inherently personalized. For example, the device can easily learn the *expected* energy generation and consumption patterns based on daily activities. Therefore, we adopt energy harvesting as the primary source. At the same time, the intermittent nature and current source behavior of the energy sources necessitate an energy storage element, such as a battery and super capacitance [31]. In this work, we utilize rechargeable flexible batteries as a reinforcement to provide a smooth quality of service and backup, in case the harvested energy falls significantly below expectations. The batteries we employ offer 148 mWh capacity at a 12$\times$35 mm$^2$ footprint, have 2 mm thickness, and weigh 1.7 g [30].

The primary goal of this work is to provide *recharge-free* wearable IoT devices that maximize the quality of service (QoS). To achieve this goal, we propose a dynamic energy optimization framework with a finite time horizon. The proposed framework channels the generated power between the battery and the IoT device, while enforcing minimum and target energy constraints to guarantee recharge-free operation. The fundamental components of the proposed framework are illustrated in Figure 1 and described below.

**Inputs and objective:** The inputs to our optimization framework are the initial battery energy and the expected energy harvested pattern. In addition, we also specify the minimum battery level allowed at any point in time and the battery energy target at the end of the day. The minimum energy constraint ensures that the battery always has a reserve to perform emergency tasks. Similarly, the energy level target ensures that the battery will have a desired level of charge at the end of each day. Our goal is to optimize the work performed by the IoT device, called the utility, under the battery level constraints. We measure the utility using an increasing function of the energy allocated to the IoT device. This choice captures the fact that more energy allocation would lead to a larger utility. At the same time, it is more general than simply maximizing the allocated energy itself, since allocating more energy may have a diminishing rate of return.

**Dynamic optimization with 24-hour horizon:** The first component of the proposed solution is a finite horizon dynamic optimization formulation, as represented by the green patterned box in Figure 1. We set the finite time horizon as 24 hours, since the energy harvesting pattern and user activities are repeated on a daily basis with potential day-to-day variation. The 24-hour horizon is divided into equal intervals, e.g., one hour or one minute epochs. We derive a closed-form solution that gives the optimal energy allocations for each time interval during the day by using Karush-Kuhn-Tucker (KKT) conditions [23]. The optimality of this solution is guaranteed if the expected energy harvesting pattern matches with the actual generated energy. However, there are inter-day and inter-interval variations in the generated energy due to environmental conditions. Therefore, we also need to perturb the energy allocations computed using expected values.

**Perturbation in each interval:** The energy allocations computed at the beginning of each day deviate from their optimal values due to uncertainties in the harvested energy and load conditions. Therefore, we also perform runtime optimization by taking the differences in the expected and actual energy values into account. For example, suppose that one-day horizon is divided into 24 one-hour intervals, and the energy harvested during the first hour is less than the assumed value. We compute this difference at the end of the first hour. Then, we reflect it in the energy allocations computed for the rest of the intervals on that day. In this way, the deviation from the optimal allocations are rectified at every interval. As a result, we continuously adapt to the changes in the environmental conditions with negligible runtime overhead.

**Learning the daily patterns:** Throughout the day, we keep track of harvested energy in each interval, and use this data to find the *expected* energy harvesting pattern. Similarly, user motion patterns reveal low and high activity periods (e.g., sleep and exercise times). Daily averages of this data are fed to the proposed framework. Then, this data is used to guide the energy allocations, such as allocating

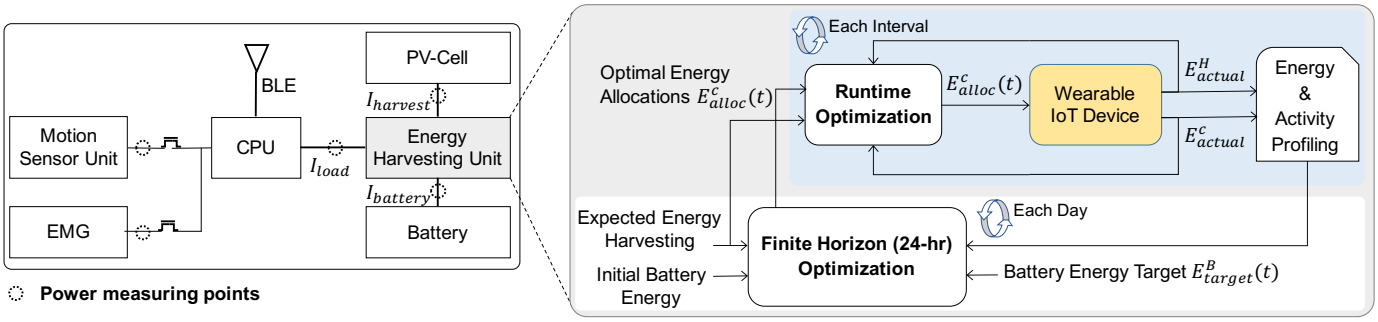



Fig. 1: The proposed hardware architecture and energy harvesting framework. The energy harvesting unit channels the generated current between the IoT device and battery. Our proof of concept prototype uses PV-cell as the ambient energy source, but the proposed framework can work with multiple energy sources.

minimum energy during sleep, as described in Section IV-C.

We demonstrate the proposed framework using the hardware prototype presented in Section V-A. Our prototype employs flexible PV-cells to harvest energy from ambient light. The effectiveness of our optimization algorithm is evaluated for different user activities and energy harvesting patterns obtained from an online database [2, 38]. The proposed runtime algorithm is near optimal, since the actual harvested energy in a given interval may be different than the expected value. Therefore, we compare our results with the maximum achievable utility computed using an oracle and an offline optimization algorithm [12, 13]. We show that the utility obtained by our runtime optimization approach is within 3% of the optimal utility, which is not feasible since it assumes an oracle. Moreover, our results converge to the optimal solution as the difference between the harvested energy and its expected value diminish.

*The major contributions of this work are as follows:*

- We present a closed-loop solution for finding the optimal energy consumption of a self-powered IoT device when the amount of harvested energy is known a priori,
- Since the actual harvested and consumed energy may differ from their expected values, we propose a novel runtime algorithm with constant time complexity for setting the energy consumption in finite horizon,
- We demonstrate that our results are, on average, within 3% of the optimal solution computed offline for a wide range of practical scenarios using a hardware prototype. We also show that the proposed algorithm incurs negligible power consumption and execution time penalty.

The rest of the paper is organized as follows: We review the related work in Section II. We present the preliminaries and the proposed algorithm in Section III and Section IV, respectively. Finally, we discuss the experimental results in Section V, and summarize the conclusions in Section VI.

## II. Related Work

Wearable IoT devices have recently attracted significant attention due to advances in sensing, low-power processing, communication protocol and radio technologies [15, 26]. In particular, flexible hybrid electronics technology offers a great potential for sensor-rich wearable applications [5, 14, 21].

Limited battery capacity of wearable devices has led to the study of energy harvesting. Major components of an energy harvesting system are the energy source, storage, harvesting circuit and harvesting-aware power management [31, 34]. Solar energy harvesting using PV-cells is one of the most promising techniques adopted by many recent studies [1, 28, 31]. Body heat and motion can also generate energy with the help of thermoelectric [16, 35] and piezoelectric sensors [17, 32], respectively.

Energy harvesting aware power management for wireless sensor nodes has been studied extensively in recent years [9, 20, 40]. In particular, the work in [20] presents a general framework for including energy harvesting in power management decisions. The authors maximize the duty cycle of a sensor node using a linear program formulation. To avoid solving a linear program at runtime, the authors also present a low-complexity heuristic to solve the linear program. Similarly, a linear quadratic tracking based algorithm that adapts the duty cycle of the sensor node is presented in [40]. The authors minimize the deviation of the battery level from a specified target. However, these solutions do not consider the application requirements when tuning the duty cycle of the nodes.

Concurrent task scheduling and dynamic voltage frequency scheduling is proposed to increase the lifespan of energy harvesting systems in [27]. At the beginning of each time interval, their algorithm refines the solar irradiance estimation and adjusts the task scheduling, but it is unable to correct future energy allocations. To achieve long-term recharge-free operation, a design-time capacity planning and runtime adjustment method is presented in [6]. Their methodology derives the battery capacity that can satisfy uninterrupted operation for a year. During runtime, the duty ratio of the device is changed based on the daily operation history. However, this approach only reacts to the harvested energy variations, thus leaving room for improvement.

In wearable IoT applications, energy can be optimized by considering the user activity and application characteristics. Our proposed approach learns the energy harvesting and user activity patterns. We first calculate the optimal energy allocation using a closed-form formula, assuming expected harvesting pattern. Then, we propose a novel runtime algorithm that both revises the optimal allocation dynamically and redistributes the slack from the previous intervals.

## III. Preliminaries and Overview

We divide the one-day horizon into $T$ equal intervals. For example, the battery energy illustration in Figure 2 assumes $T = 24$, i.e., each interval is one hour long. The proposed approach does not put any constraints on the level of granularity, provided that the overhead of the runtime energy allocation calculations is negligible[1].

**Energy constraints:** The battery energy at the beginning of any interval $t$ is denoted as $E^B_t$ for $0 \leq t \leq T-1$. The proposed

[1]Our implementation runs with one-minute intervals without any significant overhead.

approach can work with multiple ambient sources such as a PV-cell, thermoelectric generator and a piezoelectric device. In our experiments, we use a commercial PV-cell as the ambient energy source [8]. Suppose that the harvested and consumed energies in interval $t$ are given by $E_t^H$ and $E_t^c$, respectively. As illustrated in Figure 2, the battery energy dynamics can be expressed as:

$$E_{t+1}^B = E_t^B + \eta_t E_t^H - E_t^c, \quad 0 \le t \le T-1 \tag{1}$$

where $\eta_t$ is used to model the losses of the battery and power management circuitry including the PV cell and voltage converters. The efficiency is time varying since it is a function of generated current. Regardless of the harvested energy, the IoT device should have enough reserves to perform an emergency task, such as detecting a fall and sending an emergency signal. Therefore, we set a minimum battery level constraint $E_{min}$. Similarly, we constrain the energy level at the end of the day from below, such that there is sufficient reserve for the next day. Hence, the constraints on the battery energy level are given as:

$$E_T^B \ge E_{target} \quad \text{and} \quad E_t^B \ge E_{min} \ \ \forall t \ \ 0 \le t \le T-1 \tag{2}$$

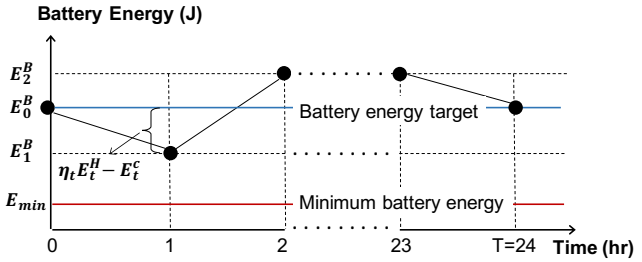



Fig. 2: Illustration of the battery level computation for $T = 24$ hr horizon.

**Driver applications and the utility function:** Although the proposed framework does not depend on any particular application, we consider health monitoring and activity tracking as the driver applications. We monitor the user activity using a motion sensor unit that integrates an accelerometer and a gyroscope. We also employ circuitry for real-time acquisition of physiological signals such as electromyography (EMG) and electrocardiogram (ECG). These signals are sampled and processed by a micro-controller unit (MCU). The processing results are transmitted to a personal device, such as a smartphone, using Bluetooth Low Energy (BLE) protocol.

The energy requirement of the target application is determined primarily by three factors. The first one is the active power consumption $P_{act}(f_t)$ as a function of the processing speed $f_t$ during interval $t$. In our driver applications, this includes sampling the sensors, processing the data in real-time, and potentially transmitting data through BLE connection. The other factors are the duty ratio $\rho_t$, i.e., the percentage of time the application is active, and the idle power consumption $P_{idle}$. With these definitions, the average application power consumption in a given interval can be written as:

$$P_t = \left[\rho_t P_{act}(f_t) + (1-\rho_t) P_{idle}\right] \tag{3}$$

A given target application needs a minimum duty ratio $\rho_{min}$ and operating frequency $f_{min}$ to accomplish its performance requirements. For example, it may need to guarantee a certain number of measurements per unit time. We use these requirements to compute the minimum energy $M_E$ that should be allocated for each period. Allocating more energy can improve the QoS by delivering higher throughput, while less energy allocation means lower QoS. We define

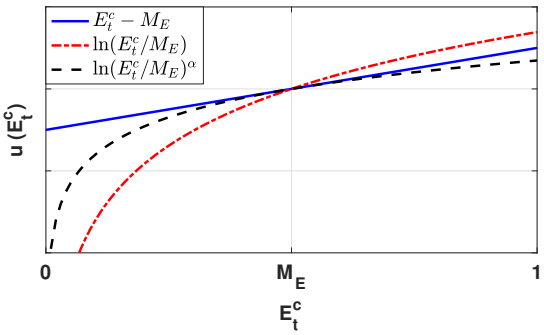



Fig. 3: Illustration of the application utility function.

a utility function that expresses the quality of service in terms of $M_E$ to capture this behavior. For illustration, a linear utility function $E_t^c - M_E$ is plotted in Figure 3. In general, allocating more energy has a diminishing rate of return, while allocation under $M_E$ degrades quality at a faster rate. Hence, we employ a parameterized and generalized the utility function that captures this behavior as illustrated in Figure 3:

$$u(E_t^c) = \ln\left(\frac{E_t^c}{M_E}\right)^\alpha \tag{4}$$

where the parameter $\alpha$ is used to tune the utility function for a specific user or application. We note that the algorithm presented next works with *any utility function that is concave and increasing*. The major parameters used in this paper are summarized in Table I.

## IV. Optimal Energy Management

### A. Problem Formulation

Our goal is to maximize the utility over a one-day horizon under the energy constraints explained in Section III. Hence, we can formulate the optimization problem using Equations 1–4 as follows:

$$\begin{aligned} \textit{maximize} \quad & U(E_0^c, E_1^c \dots E_T^c) = \sum_{t=0}^{T-1} \beta^t \ln\left(\frac{E_t^c}{M_E}\right)^\alpha \\ \textit{subject to} \quad & E_{t+1}^B = E_t^B + \eta_t E_t^H - E_t^c \qquad 0 \le t \le T-1 \\ & E_{t+1}^B \ge E_{min} \qquad 0 \le t \le T-1 \\ & E_T^B \ge E_{target} \end{aligned} \tag{5}$$

In this formulation, we compute the total utility as the sum of the utilities in each interval. A positive discount factor $0 < \beta^t \le 1$ is added to enable bias against distant intervals.

The optimal solution to the problem given in Equation 5 can be found offline using dynamic programming [4]. However, it requires solving a set of $T$ nonlinear equations, which is computationally expensive for a runtime algorithm. Furthermore, it relies on the knowledge of the energy that will be harvested in the future intervals (i.e., $E_t^H$ for $0 \le t \le T-1$). In what follows, we propose a two-step solution based on two insights that enable us to overcome these challenges. The proposed solution leads to a near-optimal runtime algorithm with a complexity of $\mathcal{O}(1)$, i.e., the complexity does not grow with the time horizon or the number of intervals.

### B. Optimal Closed-Form Solution with Relaxed Constraints

The proposed solution relies on two key insights:

**Key insight-1:** We can derive a closed-form analytical solution to this optimization problem, if we *tentatively* ignore the minimum energy constraint. Obviously, the revised solution is not guaranteed to satisfy the minimum energy constraint $E_t^B \ge E_{min}$. However, we can enforce it at runtime at the expense of loss in optimality.

TABLE I: Summary of the major parameters

| Symbol | Description |
|---|---|
| $T$ | Number of control intervals in the finite horizon |
| $\beta > 0$ | Discounting factor for utility |
| $E_{min}, E_{target}$ | Minimum and target battery energy constraints |
| $P_t$ | Power consumption of the IoT device in interval $t$ |
| $\rho_t, f_t$ | Duty ratio and frequency of the IoT device in interval $t$ |
| $M_E$ | Minimum energy required for positive utility |
| $\alpha$ | A positive parameter to control the shape of the utility function |
| $E_t^H, E_t^c$ | Harvested and consumed energy in interval $t$ |
| $E_t^B$ | Battery energy at the beginning of interval $t$ |
| $\Delta_t^h, \Delta_t^c$ | Deviation from the expected values of harvested and consumed energy |

Therefore, we find the closed-form solution at the beginning of each day. Then, the energy allocations are adjusted at the beginning of each interval, as described in Section IV-C.

**Key insight-2:** We cannot rely on the knowledge of energy that will be harvested or consumed throughout the day. However, we can learn the expected patterns by profiling the generated energy during each time interval. This enables us to derive the optimal allocation for each interval at the beginning of each day by utilizing the expected values. Similarly, the actual energy consumption may be different than the optimal allocation, as detailed in Section IV-C. Therefore, we compare the actual generated and consumed energies with their expected values. Then, we use the difference to perturb the energy allocations for the remaining intervals, as described in Section IV-C. Since we relax the $E_{min}$ constraint, there may be time intervals during which the battery level drops below the minimum threshold. Furthermore, the proposed approach can over- or under-allocate energy due to unexpected changes in the harvested energy, unlike an oracle-based offline optimization. However, these effects do not propagate beyond one interval, since the proposed approach rectifies over- and under-allocations at the beginning of the next control interval. For the continuity of the discussion, we first summarize the closed-form solution with relaxed constraints below.

**Closed-form solution:** When we relax the minimum energy constraint and assume expected values for the harvested energy, the optimal energy allocation for each interval can be found as follows:

$$\text{First interval}: \quad E_0^c = \frac{E_0^B - E_{target} + \sum_{t=0}^{T-1} \eta_t E_t^H}{1 + \beta + \beta^2 + \ldots + \beta^{T-1}} \tag{6}$$

$$\text{Subsequent intervals}: E_{t+1}^c = \beta E_t^c \qquad 0 \leq t \leq T-1$$

The derivation is presented in the Appendix. Note that the denominator can be computed a priori, and the total expected energy that will be harvested is available through profiling. Therefore, this closed-form equation enables us compute the energy allocations with constant time complexity. Next, we explain how we employ this solution to design a runtime algorithm.

### C. Near-Optimal Runtime Solution

This section presents our novel algorithm that builds on top of the closed-form solution given by Equation 6. The proposed algorithm perturbs the optimal allocations found using the expected energy values and enforces the minimum energy constraints at runtime.

*1) Uncertainty in Expected Energy Values:* The actual energy harvested at runtime may differ from the expected value due to factors such as environmental conditions. Efficiency in storing the harvested energy also adds to the uncertainty, since it varies with the load. We represent the difference between the actual energy generation and the expected value by $\Delta_t^H$. $\Delta_t^H > 0$ ($\Delta_t^H < 0$) means that actual energy harvested during interval $t$ is larger (smaller) than the expected value for that interval.

An IoT device uses the energy allocation target for a given interval $t$ to compute the average power consumption allowed in that interval. Then, it finds the duty ratio and operating frequency using Equation 3 as summarized in Section IV-C4. However, the actual energy consumed at the end of the interval may be different from the target. We subtract the actual consumption from the allocated energy to find the difference $\Delta_t^c$. Similar to the difference in the harvested energy, $\Delta_t^c > 0$ means a surplus, $\Delta_t^c < 0$ means that more energy than the allocated target is consumed. Hence, the difference between the expected energy accumulation and the actual values can be written as:

$$\Delta_t = \Delta_t^H + \Delta_t^c \quad 0 \leq t \leq T-1 \tag{7}$$

When $\Delta_t$ is positive, the energy surplus can be used during the remaining intervals. Otherwise, the consumed energy is more than the allocated target. Therefore, the deficit should be reflected in the remaining intervals.

*2) Perturbation of the Allocated Energy Values:* We need to adjust two quantities to account for the unpredictable dynamic variations. First, the optimal solution given in Equation 6 needs to be corrected in light of the new data available at the end of each interval. Second, the over or under expenditure in the previous interval should be distributed to future intervals.

**Correcting the Future Allocations:** Suppose that we adjust the optimal allocation at the beginning of the time interval $t$. The difference in expected and actual energy accumulated over earlier intervals $\{\Delta_0, \Delta_1, \ldots, \Delta_{t-1}\}$ are known at this point. Therefore, the adjusted allocation for interval $t$ can be found using Equation 6 as:

$$E_t^c = \beta^t \frac{E_0^B - E_{target} + \sum_{k=0}^{T-1} \eta_t E_k^H + \sum_{k=0}^{t-1} \mathbf{\Delta_k}}{1 + \beta + \beta^2 + \ldots + \beta^{T-1}}$$

Since we are interested in a computationally efficient recursive solution, we can re-arrange the terms to express $E_t^c$ in terms of $E_{t-1}^c$ and $\Delta_{t-1}$ only:

$$E_t^c = \beta\big(\beta^{t-1} \frac{E_0^B - E_{target} + \sum_{k=0}^{T-1} \eta_t E_k^H + \sum_{k=0}^{t-2} \Delta_k}{\sum_{k=0}^{T-1} \beta^k} + \frac{\beta^{t-1} \mathbf{\Delta_{t-1}}}{\sum_{k=0}^{T-1} \beta^k}\big)$$

$$E_t^c = \beta\big(E_{t-1}^c + \frac{\beta^{t-1} \mathbf{\Delta_{t-1}}}{\sum_{k=0}^{T-1} \beta^k}\big) \tag{8}$$

Hence, Equation 8 corrects the future allocations based on the most up-to-date energy generation and consumption information after each interval.

**Redistributing the Surplus/Deficit:** In addition to correcting the future allocations, we need to account for deviations from the revised optimal values in the past intervals. For example, assume that the optimal allocation for interval $t-1$ was computed as 10 mAh, but the harvested energy in interval $t-1$ turned out to be significantly lower than the expected value. Suppose that the optimal allocation in interval $t-1$ is corrected as 6 mAh in light of the new measurements. Equation 8 corrects the future allocations, but it does not claim back 4 mAh overspent in the previous interval. In other

words, Equation 8 alone does not make up for over-consumption, or reclaim the underutilized energy allocations in the *previous intervals*. Therefore, we need to distribute $\Delta_{t-1}$ to the remaining intervals $[t, T-1]$. A straightforward uniform distribution is not sufficient, since any adjustment introduced at time $t$ affects the future allocations due to the recursive rule in Equation 8.

Suppose that we add a correction term to Equation 8 as follows:

$$E_t^c = \beta\big(E_{t-1}^c + \frac{\beta^{t-1}\Delta_{t-1}}{\sum_{k=0}^{T-1}\beta^k}\big) + a_t\Delta_{t-1}$$

where $a_t$ is a normalization coefficient that will ensure that the perturbations in the remaining intervals will add up to precisely $\Delta_{t-1}$. By grouping the terms with $\Delta_{t-1}$, we obtain:

$$E_t^c = \beta E_{t-1}^c + \big(\frac{\beta^t}{\sum_{k=0}^{T-1}\beta^k} + a_t\big)\Delta_{t-1} \tag{9}$$

Since the perturbation term will be multiplied with $\beta$ in each future interval (due to the $\beta E_{t-1}^c$ term), the sum of the perturbations from the current interval through the last one can be written as:

$$\sum_{k=t}^{T-1}\beta^{k-t}\big(\frac{\beta^t}{\sum_{k=0}^{T-1}\beta^k} + a_t\big)\Delta_{t-1} = \Delta_{t-1}$$

By solving this equation, we can find $a_t$ as:

$$a_t = \begin{cases} \frac{1-\beta}{1-\beta^{T-t}} - \frac{\beta^t}{\sum_{k=0}^{T-1}\beta^k} & 0 < \beta < 1 \\ \frac{1}{T-t} - \frac{1}{T} & \beta = 1 \end{cases} \tag{10}$$

*3) User Activity and Minimum Energy Constraint:* Profiling the energy consumption and user activity reveal specific periods with low or high activities. For example, it is possible to identify sleep and exercise periods. The proposed approach enables us to easily introduce new equality constraints based on this information. More precisely, we set $E_t^c = M_E$ for intervals $t$ that fall during the sleep duration. Similarly, one can allocate a certain maximum value during expected exercise periods. We note that over-allocation does not have a significant drawback since unutilized allocations are distributed to future periods. However, under-allocation may hurt the utility if the interval duration is long (e.g., one hour). Therefore, low activity regions should be selected conservatively. Since these constraints can be introduced as pre-allocation, they do not change the formulation.

The final consideration is enforcing the minimum energy constraint. Equation 9 can cause the battery energy drain below $E_{min}$, since this constraint was relaxed to find a closed-form solution. Therefore, we project the remaining battery energy $E_{t+1}^B$ at runtime using Equation 1, and compare it against $E_{min}$ before committing to a solution. If there is a violation, we allocate the maximum energy that satisfies $E_{t+1}^B = E_{min}$. That is, the allocation becomes:

$$E_t^c = \begin{cases} \beta E_{t-1}^c + \big(\frac{\beta^t}{\sum_{k=0}^{T-1}\beta^k} + a_t\big)\Delta_{t-1} & E_{t+1}^B \geq E_{min} \\ E_t^B + \eta_t E_t^H - E_{min} & otherwise \end{cases} \tag{11}$$

where $E_{t+1}^B$ and $a_t$ and are given by Equations 1 and 10, respectively.

*4) Summary of the Proposed Algorithm:* We conclude this section with a step-by-step description of the runtime operation:

1) *At the beginning of each day*: Compute the allocation for the first interval $E_0^c$ using Equation 6.
2) *For each interval* $0 \leq t \leq T-1$: Divide the energy allocation $E_t^c$ by the interval duration to find the target power consumption $P_t$. Then, use Equation 3 to find the duty ratio $\rho_t$. If there are multiple allowed frequency levels $f_t$, we use the most energy efficient $f_t$. However, any feasible combination is acceptable.
3) *During each interval* $0 \leq t \leq T-1$: Keep track of actual harvested and consumed energy. Compute $\Delta_t$ at the end of the interval by finding the difference between the expected and measured values.
4) *Before the start of each interval* $1 \leq t \leq T-1$: Use Equation 11 to find the next allocation $E_t^c$. If $t = T-1$ stop, otherwise increment $t$ and go to step 2.

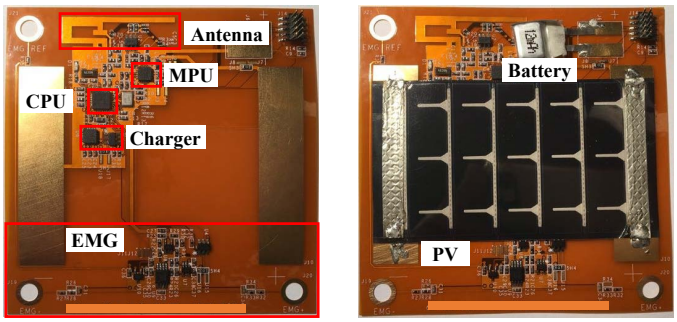
## V. Experimental Evaluation

### A. Experimental Setup

**IoT Device Parameters:** We employ the prototype shown in Figure 4 to demonstrate the proposed algorithm under realistic scenarios. It consists of an MPPT charger (TI BQ25504 [36]), a microprocessor (TI CC2650 [37]), a motion sensor unit (InvenSense MPU-9250 [19]), and EMG circuitry. We use a PV-cell from FlexSolarCells SP3-37 [8] as the energy-harvesting device and a 12 mAh Li-Po battery GMB 031009 [11] as the storage element. We have probes to measure the power consumption of different components, as illustrated in Figure 1. These measurements are used to validate the power model given in Equation 3 as a function of the duty ratio and frequency. We also determined the IoT device parameters, such as $E_{min}$ and $M_E$, listed in Table II, based on these measurements.

TABLE II: Parameter values used during evaluations

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $E_{min}$ | 0.75 mAh | $E_{target}$ | 8 mAh |
| $P_{idle}$ | 2.2 mW | $M_E$ | 0.6 mAh |
| $T$ | 24 | $\alpha$ | 1 |




(a) (b)

Fig. 4: Prototype (a) front view, (b) back view

**Energy Harvesting Model:** The harvested energy is determined by the PV-cell and the radiation intensity, which is a function of observation time and location. I-V characteristics of SP3-37 are measured by varying the radiance from 100 to 1000 $W/m^2$ with the help of a halogen lamp. Then, this empirical data is used to model the maximum generated power as a function of radiation. This model enables us to compute the harvested energy, if the radiation is known. To find the radiation, we first estimate the position of the sun at a given date and time using Sandia's Ephemeris model [33]. Then, we convert the position information to radiation using Ineichen's model [18]. These three models are used by our algorithm to predict the energy that will be harvested during the day. We compare our results to an offline optimal algorithm implemented using the CVX package [13] and an oracle. The oracle uses the actual radiation, which is measured at every minute on the NREL Solar Radiation Research Laboratory's baseline measurement system [2].

**User Activity Model:** The energy consumption varies as a function of the user activity. To evaluate a wide range of scenarios, we use

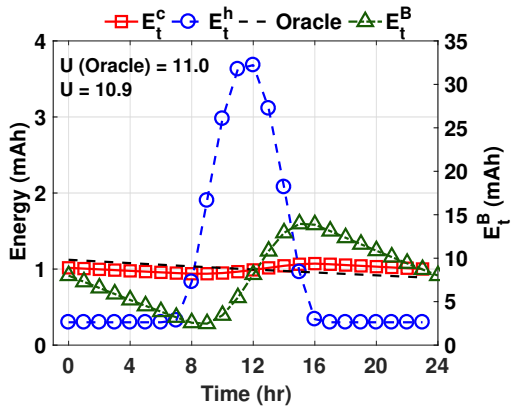



Fig. 5: Energy allocation in January without learning the user pattern.

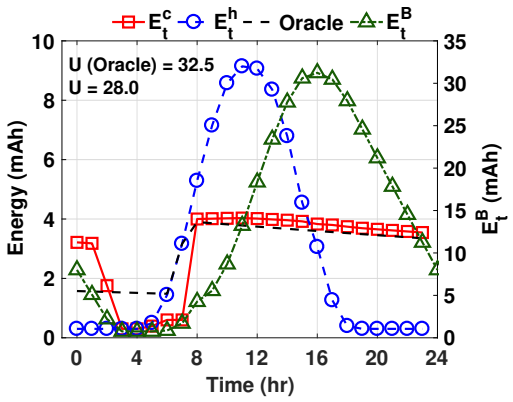



Fig. 6: Energy allocation in July without learning the user pattern.

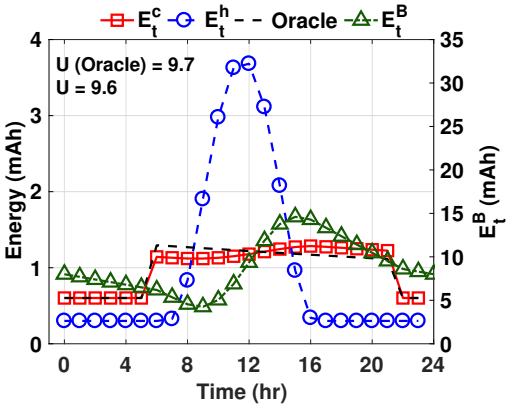



Fig. 7: Energy allocation in January after learning the user pattern.

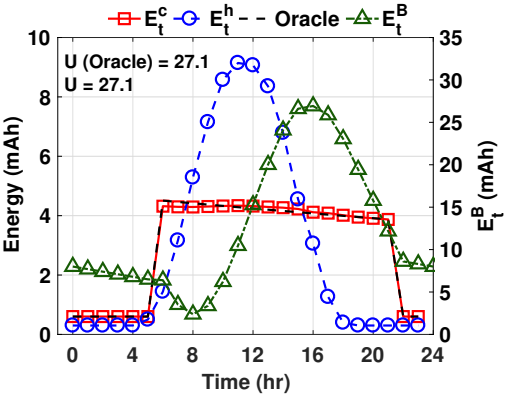



Fig. 8: Energy allocation in July after learning the user pattern.

different user activity patterns from the American Time Use Survey conducted by the US Department of Labor [38]. This survey contains the time a user spends for various activities. In our evaluations, we use five activity categories {*sleep*, *work*, *exercise*, *leisure*, *others*}. When the user is asleep, we allocate $M_E$ to the corresponding interval. Otherwise, we use the proposed approach to find the optimal allocation.

### *B. Energy Allocation Over Time*

We first illustrate the operation of the proposed algorithm for a specific user and date. Figure 5 shows the energy-harvesting profile, battery energy and optimal allocations on January 1$^{\text{st}}$ for user-1. The energy harvesting profile (blue ∘ markers) shows that there is little to none energy generation until 8 AM. During this period, the allocated energy (red □ markers) is supplied by the battery, whose stored charge drops continuously (green △ markers). Once the harvested energy exceeds the energy allocated within an interval (around 10 AM), the battery energy starts recovering. We observe that our results match very closely with the result of the offline optimization that uses an oracle (dotted lines). We do not see a significant difference in the allocated energy throughout the day, since the battery capacity is sufficient to absorb the variation in the harvested energy. However, we observe a dramatically different behavior for July, as shown in Figure 6. The peak harvested energy is about 2.5× larger in July than January (∼3.5 mAh versus ∼9 mAh), and it spans a wider range. Therefore, the proposed algorithm allocates aggressively at the beginning of the day, relying on the energy that will be generated later. However, it hits the minimum battery energy constraint at 4 AM, unlike the offline optimization that accounts for $E_{min}$ from the beginning. As soon as the battery energy drops to $E_{min}$, the proposed algorithm starts allocating *sub-optimally* only the harvested energy to the IoT device. This continues until the harvested energy becomes sufficiently large to power the IoT device and charge the battery (8 AM). While the allocation during the rest of the day closely follows the optimal allocation, the IoT device is under-powered from 3 AM to 8 AM. As a result, the loss in utility with respect to the oracle is larger compared to that obtained for January. This demonstrates the cost of neglecting the minimum energy constraint at the beginning of the day.

Next, we analyze the results on same days by taking the user activity into account. We identify the periods of low activity, primarily the intervals categorized as *sleep*, and constrain the allocations in those intervals as $E_t^c = M_E$. We add the same constraint to the offline optimization for fairness. Comparing Figure 5 to Figure 7 shows that the algorithm starts allocating less energy at night. As a result, more energy is reserved for higher activity intervals, which leads to more than 30% increase in the utility during those intervals. Like before, the results match very closely with the offline optimization results. Incorporating the user activity leads in even more savings in the results obtained for July. When we account for user activity, the proposed algorithm does not over-allocate at the early hours, since there is little activity during night. Therefore, the battery energy does not hit to $E_{min}$, and our results coincide with the oracle results, as shown in Figure 8.

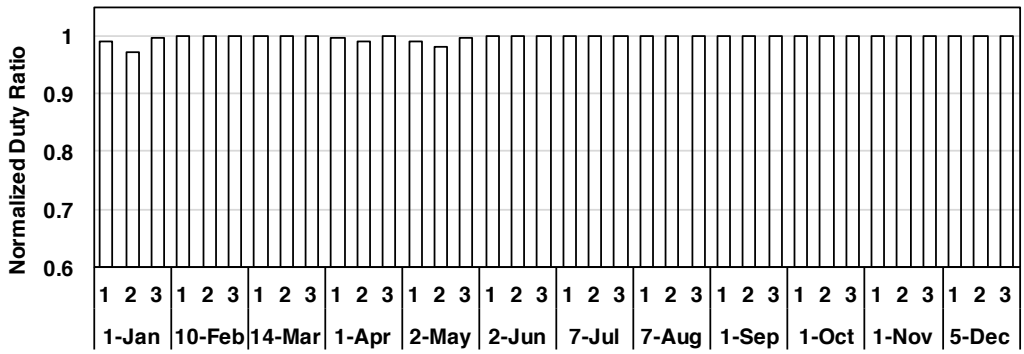



Fig. 9: Comparison of the proposed solution to offline optimization for three users over 12 months.

### C. Comparison to Offline Optimization

Improving the duty ratio is an important end goal. Therefore, this section compares the duty ratio obtained with the proposed approach against the offline optimization results, which employ an oracle. We performed the comparisons for three different users from the US Department of Labor [38] database over 12 months. Figure 9 summarizes the normalized duty ratio (our results divided by the offline optimization results). We observe that the duty ratio provided by our approach is, on average, within 1% of the duty ratio achieved by the oracle. Moreover, the largest loss in optimality in the duty is less than 5%. We observe a bigger loss under two conditions. First, when the variation between the expected and actual energy generation is large, the results of the proposed algorithm degrade, as anticipated. Second, when the peak-to-peak variation in the harvested energy becomes comparable to the battery capacity ($\sim$25% of $E_{target}$), the proposed algorithm hits the $E_{min}$ target, as shown in Figure 6.

## VI. Conclusions and Limitations

Wearable IoT devices have a great potential to enable health monitoring, activity tracking and gesture-based control applications. However, they face severe energy limitations due to weight and cost constraints. Therefore, harvesting energy from ambient sources, such as light and body heat, and using it optimally is critical for their success. This paper presented a near-optimal runtime algorithm for self-powered wearable IoT devices. The proposed approach is based on two observations that lead to near-optimal results with constant time complexity. First, we obtain a closed-form solution for the optimization problem by relaxing the minimum battery energy constraint. Then, we use the expected energy that will be harvested throughout the day to solve the relaxed finite horizon optimization problem. Finally, we account for the deviations from the expected values and enforce the minimum energy constraints at runtime. We demonstrate that our results are on average within 3% of optimal values computed offline using an oracle. The results degrade as the peak-to-peak variation in the harvested energy and deviation from the expected values increase. However, the degradation in the utility is small when the battery capacity can absorb the peak-to-peak variations.

**Acknowledgment:** This work was supported by NSF CAREER award CNS-1651624.

## References

[1] C. Alippi and C. Galperti, "An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1742–1750, 2008.

[2] A. Andreas and T. Stoffel, "NREL Solar Radiation Research Laboratory (SRRL): Baseline Measurement System (BMS); Golden, Colorado (Data); NREL Report No. DA-5500-56488," 1981, http://dx.doi.org/10.5439/1052221, accessed 5 August 2017.

[3] H. Banaee, M. U. Ahmed, and A. Loutfi, "Data Mining for Wearable Sensors in Health Monitoring Systems: A Review of Recent Trends and Challenges," *Sensors*, vol. 13, no. 12, pp. 17 472–17 500, 2013.

[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[5] G. Bhat *et al.*, "Multi-Objective Design Optimization for Flexible Hybrid Electronics," in *Proc. of Int. Conf. on Comput.-Aided Design*, 2016.

[6] B. Buchli, F. Sutton, J. Beutel, and L. Thiele, "Dynamic Power Management for Long-Term Energy Neutral Operation of Solar Energy Harvesting Systems," in *Proc. Conf. on Embedd. Network Sensor Syst.*, 2014, pp. 31–45.

[7] V. Custodio, F. J. Herrera, G. López, and J. I. Moreno, "A Review on Architectures and Communications Technologies for Wearable Health-Monitoring Systems," *Sensors*, vol. 12, no. 10, pp. 13 907–13 946, 2012.

[8] FlexSolarCells, "SP3-37 Datasheet," 2013, http://www.flexsolarcells.com/index_files/OEM_Components/Flex_Cells/specification_sheets/01_FlexSolarCells.com_PowerFilm_Solar_SP3-37_Specification_Sheet.pdf, accessed 5 August 2017.

[9] B. Gaudette, V. Hanumaiah, S. Vrudhula, and M. Krunz, "Optimal Range Assignment in Solar Powered Active Wireless Sensor Networks," in *Proc. IEEE Infocom*, 2012, pp. 2354–2362.

[10] M. Geisler *et al.*, "Human-Motion Energy Harvester for Autonomous Body Area Sensors," *Smart Materials and Structures*, vol. 557, no. 1, p. 012024, 2017.

[11] GMB, "031009 datasheet," 2009, http://www.gmbattery.com/Datasheet/LIPO/LIPO-031009-12mAh.pdf, accessed 5 August 2017.

[12] M. Grant and S. Boyd, "Graph Implementations for Nonsmooth Convex Programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, 2008, pp. 95–110.

[13] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, Version 2.1," 2014, http://cvxr.com/cvx, accessed 5 August 2017.

[14] U. Gupta, J. Park, H. Joshi, and U. Y. Ogras, "Flexibility-aware Systems on Polymer: Concept to Prototype," *IEEE Trans. on Multi-Scale Comput. Syst.*, vol. 3, no. 1, pp. 36–49, 2017.

[15] M. A. Hanson *et al.*, "Body Area Sensor Networks: Challenges And Opportunities," *Computer*, vol. 42, no. 1, p. 58, 2009.

[16] D. C. Hoang, Y. K. Tan, H. B. Chng, and S. K. Panda, "Thermal Energy Harvesting From Human Warmth for Wireless Body Area Network in Medical Healthcare System," in *Int. Conf. on Power Electron. and Drive Syst.*, 2009, pp. 1277–1282.

[17] G.-T. Hwang *et al.*, "Self-Powered Cardiac Pacemaker Enabled by Flexible Single Crystalline PMN-PT Piezoelectric Energy Harvester," *Advanced materials*, vol. 26, no. 28, pp. 4880–4887, 2014.

[18] P. Ineichen and R. Perez, "A New Airmass Independent Formulation for the Linke Turbidity Coefficient," *Solar Energy*, vol. 73, no. 3, pp. 151–157, 2002.

[19] InvenSense, "Motion Processing Unit," 2016, https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250, accessed 5 August 2017.

[20] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *ACM Trans. Embedd. Comput. Syst.*, vol. 6, no. 4, p. 32, 2007.
[21] Y. Khan *et al.*, "Flexible Hybrid Electronics: Direct Interfacing of Soft and Hard Electronics for Wearable Health Monitoring," *Advanced Functional Materials*, vol. 26, no. 47, pp. 8764–8775, 2016.
[22] S. J. Kim, J. H. We, and B. J. Cho, "A Wearable Thermoelectric Generator Fabricated on a Glass Fabric," *Energy & Environmental Science*, vol. 7, no. 6, pp. 1959–1965, 2014.
[23] H. W. Kuhn and A. W. Tucker, "Nonlinear Programming," in *Proc. of the Second Berkeley Symp. on Mathematical Statistics and Probability*. University of California Press, 1951, pp. 481–492.
[24] R. Kumar *et al.*, "All-Printed, Stretchable Zn-Ag2O Rechargeable Battery via Hyperelastic Binder for Self-Powering Wearable Electronics," *Advanced Energy Materials*, 2016.
[25] W. Lao-atiman, T. Julaphatachote, P. Boonmongkolras, and S. Kheawhom, "Printed Transparent Thin Film Zn-MnO2 Battery," *J. of the Electrochemical Soc.*, vol. 164, no. 4, pp. A859–A863, 2017.
[26] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A Survey On Wireless Body Area Networks," *Wireless Networks*, vol. 17, no. 1, pp. 1–18, 2011.
[27] X. Lin, Y. Wang, N. Chang, and M. Pedram, "Concurrent Task Scheduling and Dynamic Voltage and Frequency Scaling in a Real-Time Embedded System With Energy Harvesting," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 11, pp. 1890–1902, 2016.
[28] J. Park *et al.*, "Flexible PV-cell Modeling for Energy Harvesting in Wearable IoT Applications," *ACM Trans. Embedd. Comput. Syst.*, 2017.
[29] M. Patel and J. Wang, "Applications, Challenges, and Prospective in Emerging Body Area Networking Technologies," *IEEE Wireless Commun.*, vol. 17, no. 1, 2010.
[30] PowerStream. PGEB021235 40 mAH - Rechargeable Lithium Polymer Cells. http://www.powerstream.com/thin-lithium-ion.htm, accessed 16 July 2016.
[31] V. Raghunathan *et al.*, "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems," in *Proc. of Int. Symp. on Information Processing in Sensor Networks*, 2005, p. 64.
[32] S. Saadon and O. Sidek, "Micro-Electro-Mechanical System (MEMS)-Based Piezoelectric Energy Harvester for Ambient Vibrations," *Procedia-Social and Behavioral Sci.*, vol. 195, pp. 2353–2362, 2015.
[33] Sandia National Laboratories, "Sandia's Ephemeris Model," 2017, https://pvpmc.sandia.gov/modeling-steps/1-weather-design-inputs/sun-position/sandias-code/, accessed 5 August 2017.
[34] S. Sudevalayam and P. Kulkarni, "Energy Harvesting Sensor Nodes: Survey and Implications," *IEEE Commun. Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.
[35] Y. K. Tan and S. K. Panda, "Energy Harvesting From Hybrid Indoor Ambient Light and Thermal Energy Sources for Enhanced Performance of Wireless Sensor Nodes," *IEEE Trans. on Ind. Electron.*, vol. 58, no. 9, pp. 4424–4435, 2011.
[36] Texas Instruments, "BQ25504," 2015, http://www.ti.com/lit/ds/symlink/bq25504.pdf, accessed 5 August 2017.
[37] Texas Instruments, "CC2650," 2016, http://www.ti.com/lit/ds/symlink/cc2650.pdf, accessed 5 August 2017.
[38] US Department of Labor, "American Time Use Survey," 2015, https://www.bls.gov/tus/, accessed 25 July 2017.
[39] A. Valenzuela, "Energy Harvesting for No-Power Embedded Systems," 2008, http://focus.ti.com/graphics/mcu/ulp/energy_harvesting_embedded_systems_using_msp430.pdf, accessed 19 July 2016.
[40] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *Proc. of IEEE Comm. Society Conf. on Sensor, Mesh and Ad Hoc Comm. and Networks*, 2007, pp. 21–30.
[41] M. Wendler, G. Hübner, and I. M. Krebs, "Development of Printed Thin and Flexible Batteries," *Int. Circ. Graphic Ed. Res.*, vol. 4, pp. 32–41, 2011.

## APPENDIX: DERIVATION OF EQUATION 6

To solve the optimization problem given in Equation 5, we first evaluate the Lagrangian of the objective function as:

$$L = \sum_{t=0}^{T-1} \beta^t \big[ \ln \big(\frac{E_t^c}{M_E}\big)^\alpha - \lambda_t \big(E_{t+1}^B + E_t^c - E_t^B - \eta_t E_t^H\big)\big] + \sum_{t=0}^{T-1} \beta^t \mu_t [E_{t+1}^B - E_{min}] + \beta^{T-1}\mu_{T-1}[E_T^B - E_{target}] \quad (12)$$

Using the Lagrangian, we can write the first-order conditions as:

$$\frac{\partial L}{\partial E_t^c} : \beta^t \left[\frac{\alpha M_E}{E_t^c} - \lambda_t\right] = 0 \qquad 0 \le t \le T-1 \quad (13)$$

$$\frac{\partial L}{\partial E_{t+1}^B} : -\beta^t \lambda_t + \beta^t \mu_t + \lambda_{t+1}\beta^{t+1} = 0 \quad 0 \le t \le T-1 \quad (14)$$

$$\frac{\partial L}{\partial E_T^B} : -\beta^{T-1}\lambda_{T-1} + \beta^{T-1}\mu_{T-1} = 0 \quad i.e.,\ \lambda_{T-1} = \mu_{T-1} \quad (15)$$

In addition to the first-order conditions above, the Karush-Kuhn-Tucker (KKT) conditions are $\mu_t \ge 0$, $\lambda_t \ge 0$, and:

$$\mu_t (E_{t+1}^B - E_{min}) = 0 \qquad 0 \le t \le T-1 \quad (16)$$

$$\mu_{T-1}(E_T^B - E_{target}) = 0 \quad (17)$$

Since $u(E_t^c)$ is concave, Equations 13-17 give the necessary and sufficient conditions for the optimality [23]. We present a step-by-step solution below.

**1. Boundary Condition:** Lagrangian multipliers $\lambda_t$ can be found using Equation 13 as :

$$\lambda_t = \frac{\alpha M_E}{E_t^c} \quad 0 \le t \le T-1 \quad (18)$$

Combining this relation with Equation 15, we can conclude that $\lambda_{T-1} = \mu_{T-1} \neq 0$ . Hence, the complementary slackness given by Equation 17 implies $E_T^B = E_{target}$.

**2. Recursion over $\mathbf{E_t^c}$:** We can use Equation 14 to derive a recursion rule for $\lambda_t$ and combine it with Equation 18 as follows:

$$\beta\lambda_{t+1} = \lambda_t + \mu_t \qquad 0 \le t \le T-1$$

$$\frac{\alpha\beta M_E}{E_{t+1}^c} = \frac{\alpha M_E}{E_t^c} + \mu_t \qquad 0 \le t \le T-1 \quad (19)$$

We plug the boundary condition $E_T^B = E_{target}$ to this recursive relation. Then, the energy allocations in earlier interval can be solved using the Equation 5 and the KKT condition given by Equation 16.

Solving $T$ nonlinear equations at runtime is not efficient. However, *tentatively ignoring* the minimum energy constraint (*key insight 1*), enables us to eliminate $\mu_t$ from Equation 19. That is, we can set $\mu_t = 0$ in equations 16 and 19. Similarly, $E_t^H$ is not known a priori, but we use the expected values (*key insight 2*). The proposed algorithm presented in Section IV-C enables us to make up for these choices at runtime.

**3. Closed-form Solution:** After setting $\mu_t = 0,\ 0 \le t \le T-1$, Equation 19 reduces to:

$$E_{t+1}^c = \beta E_t^c \quad (20)$$

We can re-arrange the battery energy dynamics in Equation 5, and combine with this relation as follows:

$$E_0^c = E_0^B - E_1^B + \eta_0 E_0^H, \quad \beta E_0^c = E_1^B - E_2^B + \eta_2 E_2^H, \ldots$$
$$\beta^{T-1} E_0^c = E_{T-1}^B - E_{target} + \eta_{T-1} E_{T-1}^H$$

Note that $E_{target}$ in the last equation comes from the boundary condition. When we summing up these $T$ equations, $E_1^B - E_{T-1}^B$ cancel each other. Hence, we find $E_0^c$ as:

$$E_0^c = \frac{E_0^B - E_{target} + \sum_{t=0}^{T-1} \eta_t E_t^H}{1 + \beta + \beta^2 + \ldots + \beta^{T-1}} \quad (21)$$

Combining Equation 20 and Equation 21 gives the closed form solution summarized in Equation 6. □