Getting More Out of Energy-harvesting Systems: Energy Management under Time-varying Utility with PreAct

Kai Geissdoerfer
Networked Embedded Systems Lab
TU Dresden, Germany
kai.geissdoerfer@tu-dresden.de

Raja Jurdak
Distributed Sensing Systems Group
CSIRO, Australia
raja.jurdak@csiro.au

Brano Kusy
Distributed Sensing Systems Group
CSIRO, Australia
brano.kusy@csiro.au

Marco Zimmerling
Networked Embedded Systems Lab
TU Dresden, Germany
marco.zimmerling@tu-dresden.de

ABSTRACT

Careful energy management is a prerequisite for long-term, unattended operation of solar-harvesting sensing systems. We observe that in many applications the utility of sensed data varies over time, but current energy-management algorithms do not exploit prior knowledge of these variations for making better decisions. This paper presents PreAct, the first energy-management algorithm that exploits time-varying utility to optimize application performance. PreAct's design combines strategic long-term planning of future energy utilization with feedback control to compensate for deviations from the expected conditions. We implement PreAct on a low-power microcontroller and compare it against the state of the art on multiple years of real-world data. Our results demonstrate that PreAct is up to 53 % more effective in utilizing harvested solar energy and significantly more robust to uncertainties and inefficiencies of practical systems. These gains translate into an improvement of 28 % in the end-to-end performance of a real-world application we investigate when using PreAct.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; **Sensors and actuators**; **Embedded software**.

KEYWORDS

Energy management, Energy allocation, Energy harvesting, Dynamic power management, Solar power, Energy prediction

ACM Reference Format:
Kai Geissdoerfer, Raja Jurdak, Brano Kusy, and Marco Zimmerling. 2019. Getting More Out of Energy-harvesting Systems: Energy Management under Time-varying Utility with PreAct. In *The 18th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2019) (IPSN '19), April 16–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3302506.3310393

IPSN '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6284-9/19/04…$15.00
https://doi.org/10.1145/3302506.3310393

1 INTRODUCTION

As wireless sensing systems transition from research to industry long-term, unattended operation becomes paramount to their success. In many scenarios, solar energy harvesting is a viable approach for battery-supported sensing devices. An essential component of these systems is an energy-management algorithm that dynamically adjusts the application duty cycle so that the energy consumed does not exceed the energy harvested, known as *energy neutrality*.

Motivation. In many applications, the sensing events of interest are not uniformly distributed over time. For example, ecosystem restoration uses microclimate data to infer ecosystem health and to predict restoration success [19]. An important microclimate parameter is soil moisture dynamics, whose utility for the application is high after rainfalls and low otherwise. An energy-harvesting sensing system should exploit the prior knowledge of the seasonal cycles of rainfall to conserve energy when utility is low so that more energy is available when utility is high.

However, current energy-management algorithms do not take advantage of this opportunity. Reactive algorithms adjust the application duty cycle in response to deviations of the battery's state of charge (SoC) from a target SoC [20, 33]. This couples the duty cycle with the harvested energy: The system consumes energy when it becomes available, regardless of whether utility is high or low. Predictive algorithms, on the other hand, try to reduce variations in application duty cycle by forecasting energy availability and using the battery as temporary energy storage [5, 6]. In doing so, these algorithms implicitly assume that utility is constant over time.

Contribution and road-map. This paper presents PreAct, a new energy-management algorithm that exploits prior knowledge of an application's time-varying utility. In PreAct, users specify utility as a function of time whose values range from 0 (lowest utility) to 1 (highest utility). Based on predictions about future harvested energy and feedback on the battery's current SoC, PreAct dynamically determines the optimized application duty cycle that satisfies the utility specification under the given battery capacity constraints.

This paper makes the following conceptual contributions:

- We introduce the concept of *time-varying utility* for long-term solar energy-harvesting systems. Sec. 2 proposes an intuitive specification in the form of a time-dependent *utility function*, and describes how an application should utilize energy over time to satisfy a given utility function under

energy-neutrality and battery-capacity constraints. This includes the definition of *effectiveness*, a new metric that we introduce to capture how well energy utilization and utility function match up. We specify the corresponding optimization problem that only a clairvoyant could solve offline with full knowledge of all future system states.
- We design PreAct, the first online energy-management algorithm that approximates the optimal offline solution in the face of real-world uncertainties, such as weather, charging inefficiencies, and self-discharge. As detailed in Sec. 3, PreAct combines strategic planning to maximize effectiveness with feedback control to automatically compensate for deviations from the predicted conditions. In this way, PreAct essentially integrates the benefits of predictive and reactive algorithms into a novel solution that exploits time-varying utility to achieve a significantly better system performance.

Sec. 4 describes our prototype implementation of PreAct on the TI MSP432P401R platform, which comes with an ARM Cortex-M4F microcontroller unit (MCU) running at 48 MHz clock rate. We then evaluate PreAct on eleven years of real-world solar traces from all major climate zones in Sec. 5 and on two years of data from a real-world application, monitoring soil moisture dynamics for mine rehabilitation, in Sec. 6. Our key findings are as follows:

- The energy and processing overhead of PreAct is negligible in practice: One execution of PreAct on the MSP432P401R takes on average 6.0 ms and consumes on average 85.3 µJ.
- PreAct achieves an improvement of up to 53 % in effectiveness compared with the state-of-the-art energy-management algorithms LT-ENO [6] and ENO-MAX [33].
- Even without prior knowledge of time-varying utility, PreAct outperforms the state of the art as it can better cope with battery capacity constraints as well as real-world uncertainties and inefficiencies. For example, the effectiveness of LT-ENO degrades by up to 58 % under unexpected conditions, whereas PreAct maintains the same high performance.
- PreAct's higher effectiveness translates into 28 % lower reconstruction error of soil water content dynamics in our real-world case study. This also validates the faithfulness of the effectiveness metric to application-level performance.

In light of our contributions, we review related work in Sec. 7, and end the paper in Sec. 8 with brief concluding remarks on the scope of PreAct and possible directions for future work.

2 A CASE FOR TIME-VARYING UTILITY

We argue that in many sensing applications the events of interest are not uniformly distributed over time. Rather the utility of sensed data varies over time, and domain experts can identify at least coarse-grained periods of low and high utility. We are the first to demonstrate that such prior knowledge can be exploited by an energy-management algorithm to increase the application-level performance of long-term solar energy-harvesting systems.

Motivating examples. As mentioned above, the utility of monitoring soil-moisture dynamics is clearly high after rainfall events and low otherwise [19]. Although individual rainfall events are difficult to predict, the likelihood of rainfall increases during wet seasons, which exhibit a distinct periodicity that is well known from past observations. Another example is biodiversity monitoring in the rainforest, where most areas experience flooding from April to September [27]. During this period, little activity is expected from land-based animals that move to higher ground. Similarly, avalanche forecasting based on snow-level monitoring has high utility only from October to April in the northern hemisphere [12].

Table 1: Key terms used in the description.

Term	Description
$u(n)$	User-defined utility
$e_h(n)$	Harvested energy
$\hat{e}_h(n)$	Predicted harvested energy
$soc(n)$	Battery's state of charge
$soc^*(n)$	Ideal state of charge
$soc_t(n)$	Target state of charge
$e_{util}(n)$	Energy utilization
$e^*_{util}(n)$	Ideal energy utilization
c_{bat}	Battery capacity

Utility function. For a domain expert, an intuitive way to specify utility is in the form of a *utility function* $u(n)$ with values ranging from 0 (lowest utility) to 1 (highest utility). If no prior knowledge about time-varying utility is available, or if the utility is expected to be constant over time, the utility function $u(n)$ can be set to any constant value $0 < u(n) \leq 1$. The index n represents an application-specific time period, such as hour, day, or week. Table 1 lists these and other key terms we use in the description.

As an illustrative example, Fig. 1a plots the utility function $u(n)$ and the daily harvested solar energy $e_h(n)$ over one year for an application with a seasonal utility pattern, like the soil-moisture dynamics application. The utility function u follows a seasonal trend: it is low from day 100 to day 250 and high otherwise. The harvested energy e_h follows the opposite trend and exhibits weather-related short-term variations.

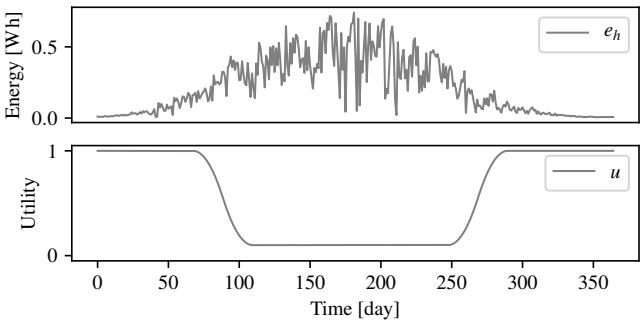
Ideal energy utilization. The application performance of an energy-harvesting sensing system depends on its duty cycle. The duty cycle can be controlled in software and relates to application-specific parameters such as wake-up interval, sampling rate, etc. Thus, an energy-management algorithm determines both the energy utilization and the resulting application performance by dynamically adjusting the duty cycle at runtime.

Ideally, to maximize application performance, we want the energy utilization to be proportional to the user-defined utility function. We thus define the *ideal energy utilization*

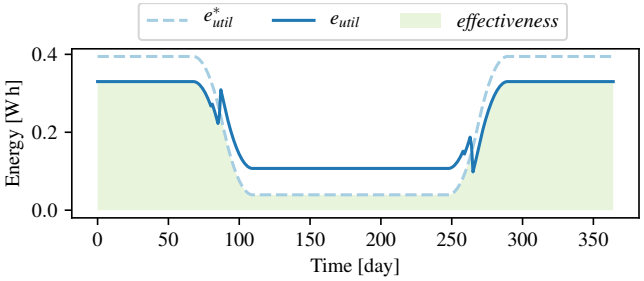
$$e^*_{util}(n) = k \cdot u(n) \tag{1}$$

To satisfy the energy-neutrality condition, the average energy utilization of the system must be equal to the average amount of harvested energy in the long run (*i.e.*, $\bar{e}^*_{util} = \bar{e}_h$). With this requirement in mind, the constant k in (1) is given by $k = \bar{e}_h/\bar{u}$. The dashed curve in Fig. 1b shows the ideal energy utilization that corresponds to the utility function shown in Fig. 1a.

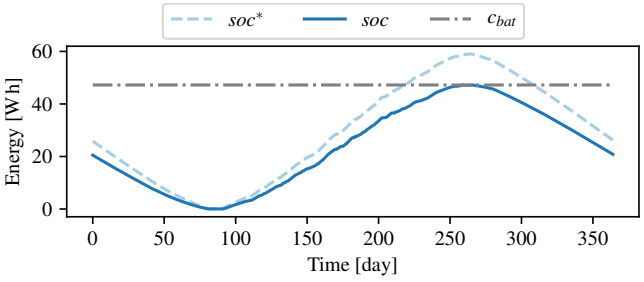
Using the battery as temporary storage, the system may sustain a high duty cycle also when the currently harvested energy is low. However, the amount of energy that can be stored is limited by the capacity of the battery c_{bat}. Looking at the example in Fig. 1a, where



(a) Energy availability in terms of daily harvested energy $e_h(n)$ and utility function $u(n)$. *The two may be misaligned, as illustrated here.*



(b) Ideal energy utilization $e^*_{util}(n)$, optimized energy utilization $e_{util}(n)$, and effectiveness of $e_{util}(n)$ as illustrated by the shaded area.



(c) Battery capacity c_{bat} and progression of ideal SoC $soc^*(n)$ and optimized SoC $soc(n)$. *Clearly, soc^* is infeasible under the given c_{bat}.*

Figure 1: Example scenario illustrating the problem of long-term energy management under time-varying utility.

utility function and energy availability are misaligned, we observe in Fig. 1c that the ideal energy utilization e^*_{util} would require an *ideal SoC* soc^* that is larger than the battery capacity. This indicates that the ideal energy utilization is not always feasible.

Effectiveness metric. To quantify how well a *feasible energy utilization* $e_{util}(n)$ matches up with the ideal energy utilization $e^*_{util}(n)$ over a given time period $i \leq n \leq j$, we introduce a new metric called *effectiveness* as follows:

$$eff(i,j) = \frac{\sum_{n=i}^{j} \min\left(e_{util}(n), e^*_{util}(n)\right)}{\sum_{n=i}^{j} e^*_{util}(n)} \tag{2}$$

Effectiveness is 0 if no energy is utilized, and 1 if the utilized energy is always at least as high as the ideal utilization. The shaded area in Fig. 1b shows the proportion of utilized energy e_{util} that contributes to effectiveness over the one-year period in our example scenario. As the plot shows, the effectiveness metric does not reward

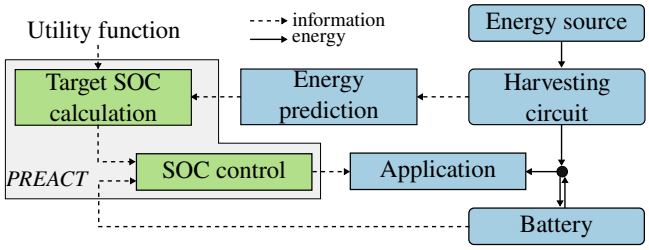


Figure 2: Architecture of PreAct. *Given a user-defined utility function and long-term predictions of future harvested energy, PreAct periodically determines the application duty cycle that maximizes effectiveness under the given battery capacity constraints.*

over-provisioning (*e.g.*, from day 100 to day 250) because any excess energy would be better spent during periods of higher utility.

The effectiveness metric is application-agnostic and thus allows for a generic treatment of the energy-management problem under time-varying utility. Nevertheless, our evaluation results in Sec. 6 indicate that it strongly correlates with the performance metrics of real-world sensing applications.

Optimization problem. The problem of energy management for a given utility function $u(n)$ and battery capacity c_{bat} can be formulated as a constrained optimization problem: *For any period n, $i \leq n \leq j$, find the application duty cycle $dc(n)$ that maximizes effectiveness $eff(i,j)$ under the constraint that the battery's SoC $soc(n)$ never exceeds the battery capacity c_{bat}.*

With full knowledge of future harvested energy, a solution to this problem can be found using standard optimization methods. For instance, $e_{util}(n)$ in Fig. 1b is an *optimized energy utilization* for the utility function $u(n)$ and energy availability $e_h(n)$ shown in Fig. 1a subject to the battery capacity c_{bat} in Fig. 1c. This optimized energy utilization has an effectiveness of 85 %, while current energy-management algorithms [6, 33] would only achieve an effectiveness of 62 % in the example scenario. Thus, the concept of time-varying utility we propose here enables significant improvements.

The need for an online algorithm. However, to harness these performance gain in practice, an online algorithm that approximates the offline solution to the above optimization problem is needed. This is because (*i*) future energy availability can only be predicted with limited accuracy, (*ii*) energy-harvesting systems are subject to various real-world effects and uncertainties like temperature-dependent charging efficiencies and battery self-discharge, and (*iii*) solving the high-dimensional optimization problem on a resource-constrained MCU is either infeasible or incurs high overhead. We next present an online algorithm that addresses these challenges.

3 PREACT

This section describes the design of PreAct,[1] a new energy-management algorithm that exploits prior knowledge about an application's time-varying utility to optimize the performance of solar energy-harvesting systems.

[1] https://preact.nes-lab.org/

3.1 Overview

Fig. 2 shows the main parts of PreAct and how they interact with the other components of an energy-harvesting sensor node. PreAct takes as input a utility function defined by the user and long-term predictions of future harvested energy. It outputs the duty cycle of the application so that effectiveness is maximized under the given battery capacity constraints. Although PreAct determines the duty cycle, the application has full control over how, when, and for what to use this "budget," such as periodically reading out sensors and wireless communication when an interesting event has been sensed.

During system operation, PreAct executes with a certain period. In the following, we assume a period of one day. This matches the diurnal pattern of solar energy and is practical for many applications, but other periods are possible. Given a period of one day, PreAct executes at the end of every day and performs two consecutive steps:

1) compute the target SoC for the next day based on the utility function and the predicted harvested energy;
2) use a controller to determine the application duty cycle for the next day based on the computed target SoC and feedback about the battery's current SoC.

Steps 1) and 2) represent the PREdictive and reACTive elements PreAct combines to form an effective, yet highly robust energy-management algorithm. Its effectiveness stems from the strategic planning in step 1) based on expected conditions. Its robustness stems from step 2) that compensates for any deviations between expected and actual conditions. Such deviations are inevitable in practice and may be due to a variety of different factors, including inaccurate predictions of future harvested energy, uncertainty in the energy consumed by the application, battery self-discharge, etc. PreAct implicitly compensates for any such factors, which makes it highly robust and widely applicable in real systems.

Before delving into the details of PreAct concerning steps 1) and 2), we briefly outline the existing approach we use to obtain long-term predictions of future harvested energy.

3.2 Prediction of Harvested Energy

Similar to prior long-term energy-management algorithms [5, 6], we use the model-based approach of Buchli et al. [7]. To the best of our knowledge, it is the only lightweight approach that does not require Internet connectivity or additional sensors.

According to the astronomical model in [7], the energy harvested on day n can be predicted by

$$e_{in}(n) = A_{panel} \cdot \eta_{panel} \cdot \eta_{cc} \cdot ghi(doy, \varphi, \lambda, \Omega) \tag{3}$$

where A_{panel} is the solar panel size, η_{panel} is the efficiency of the panel, and η_{cc} is the efficiency of the charge controller. The last term in (3) denotes the amount of energy arriving on a plane surface, called global horizontal irradiance (GHI), which is a function of the day of the year doy, the position on earth specified by latitude φ and longitude λ, and a parameter Ω that accounts for environmental conditions.

To compensate for deviations from the expected conditions, the approach scales the predictions by comparing historical values of actual harvested energy e_h to the corresponding predictions e_{in} obtained via (3) over the past M days:

$$\hat{e}_h(n+i) = \frac{\sum_{j=1}^{M} e_h(n-j)}{\sum_{j=1}^{M} e_{in}(n-j)} \cdot e_{in}(n+i) \tag{4}$$

Using (4) we can provide PreAct with predictions of the energy harvested on any future day $n+i$, $i \geq 1$, where n is the current day.

As an example, Fig. 3 shows a real trace of daily harvested energy over the course of one year and the corresponding one-year-ahead prediction obtained via (4) for $i = 1, \ldots, 365$. The prediction faithfully approximates the mean harvested energy and follows the seasonal trend. This facilitates strategic planning, as described next.

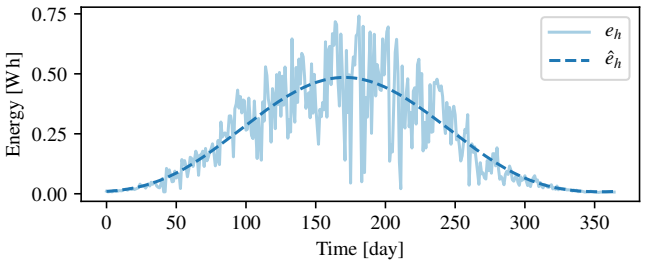


Figure 3: A real trace of daily harvested energy e_h and the corresponding one-year-ahead prediction $\hat{e}_h$.

3.3 Computation of Target SoC

When PreAct executes at the end of each day, it leverages the one-year-ahead prediction of harvested energy to compute the target SoC for the next day. The objective is to maximize effectiveness with respect to the user-defined utility function under the given battery capacity constraints. The procedure consists of the following steps:

- compute the ideal SoC $soc^*(n+i)$ for $i = 1, \ldots, 365$ that results from harvesting the predicted energy $\hat{e}_h(n+1)$ and consuming the ideal energy $e^*_{util}(n+i)$ given by (1) from the current day n up to one year into the future;
- adjust the computed ideal SoC to satisfy the battery capacity constraints and to avoid wasting harvested energy because of a flat or fully charged battery.

The target SoC for the next day is then given by the first data point of the adjusted ideal SoC, namely $soc_t(n+1)$. We describe both steps in more detail below.

Compute ideal SoC. To compute the ideal SoC for all future days $n+i$, $i = 1, \ldots, 365$, we exploit the fact that, for an ideal battery, the SoC at the end of day $n+1$ is given by the sum of the SoC at the end of day n and the difference between the harvested and utilized energy on day $n+1$:

$$soc(n+1) = soc(n) + e_h(n+1) - e_{util}(n+1) \tag{5}$$

We can iteratively apply this expression to compute the SoC at any future day. Note that the above reasoning assumes an ideal battery; any non-idealities (self-discharge, aging effects, etc.) are compensated by PreAct's feedback controller, as described in Sec. 3.4.

Since we are interested in the ideal SoC, we replace e_{util} with the ideal energy utilization e^*_{util} from (1) and e_h with the predicted

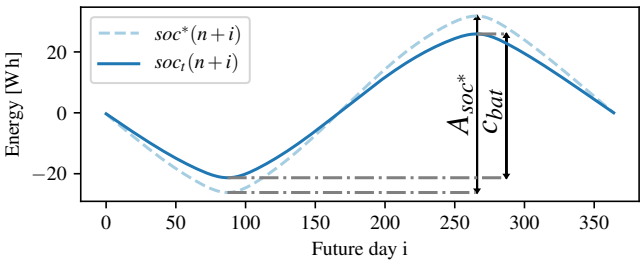


Figure 4: The peak-to-peak amplitude A_{soc^*} of the ideal SoC $soc^*(n+i)$ exceeds the battery capacity c_{bat}, whereas the target SoC $soc_t(n+i)$ fits into the battery.

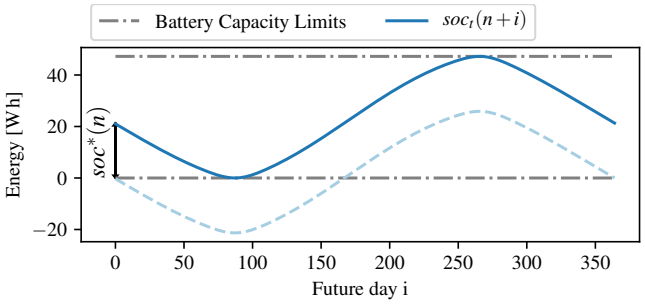


Figure 5: The initial value of the ideal SoC $soc^*(n)$ is determined such that the target SoC takes only values between 0 and c_{bat} with maximum safety margins to these limits.

harvested energy $\hat{e}_h$ from (4):

$$soc^*(n+i) = soc^*(n) + \sum_{j=1}^{i} \left(\hat{e}_h(n+j) - e^*_{util}(n+j) \right) \tag{6}$$

The only unknown in (6) is the initial value of the ideal SoC $soc^*(n)$, which we determine below. Revisiting our example scenario from Sec. 2, we plot in Fig. 4 the progression of the ideal SoC for an initial value of $soc^*(n) = 0$.

Adjust ideal SoC. To check whether the ideal SoC from (6) exceeds the battery capacity constraints throughout the one-year horizon, we compute the resulting peak-to-peak amplitude, which is independent of the initial value of $soc^*(n)$, as follows:

$$A_{soc^*} = \max_{1 \le i \le 365} (soc^*(n+i)) - \min_{1 \le i \le 365} (soc^*(n+i)) \tag{7}$$

Fully discharging a battery can cause irreversible damage and modern battery-management circuits therefore disconnect the load when the voltage drops below a certain threshold. Therefore, $0 \le soc \le c_{bat}$ refers to the practically usable portion of the battery. If the peak-to-peak amplitude A_{soc^*} is larger than the battery capacity c_{bat}, we need to scale the ideal SoC by c_{bat}/A_{soc^*} to satisfy the battery capacity constraints:

$$soc_t(n+i) = \begin{cases} \frac{c_{bat}}{A_{soc^*}} \cdot soc^*(n+i) & \text{if } A_{soc^*} > c_{bat} \\ soc^*(n+i) & \text{otherwise} \end{cases} \tag{8}$$

Using (8) PreAct can compute the target SoC $soc_t(n+i)$ for any day in the one-year horizon (*i.e.*, for $i = 1, \ldots, 365$), where the ideal SoC $soc^*(n+i)$ is given by (6). The only missing piece is the initial value $soc^*(n)$. To compute it, we again assume it to be 0 when applying the following expression:

$$soc^*(n) = \begin{cases} -\min\limits_{1 \le i \le 365} (soc_t(n+i)) & \text{if } A_{soc^*} \ge c_{bat} \\ -\min\limits_{1 \le i \le 365} (soc_t(n+i)) + \frac{c_{bat} - A_{soc^*}}{2} & \text{otherwise} \end{cases} \tag{9}$$

The reasoning behind (9) is as follows: If the amplitude of the target SoC is equal to the battery capacity, we set the initial value so that it ranges exactly between the battery capacity limits 0 and c_{bat} over the one-year horizon. This can be seen by looking at the target SoC shown in Fig. 5 for our example scenario. Otherwise, the initial value given in (9) ensures maximum safety margins between the target SoC and the battery capacity limits.

PreAct uses the above procedure to strategically plan one year in advance, which is essential to maximize effectiveness under the given battery capacity constraints. Nevertheless, it only feeds the target SoC of the next day $soc^*(n+1)$ into the SoC controller, which is detailed in the next section. This approach is conceptually similar to finite-horizon optimization used in model-predictive control.

3.4 SoC Controller

The task of PreAct's controller is to compute the application duty cycle that yields the target SoC by the end of the next day when it executes again. This is non-trivial because the relation between application duty cycle and energy consumption is complex and difficult to capture accurately. Further, the harvested energy on the next day may differ from the prediction due to unexpected weather conditions, and a battery's SoC is subject to real-world effects like temperature-dependent charging efficiencies, self-discharge, etc.

As a natural solution, we use a proportional-integral-derivative (PID) controller that automatically compensates for such effects. Specifically, using feedback in form of the measured SoC of the battery $soc(n)$, the controller predicts the control error at the end of the next day with respect to the target SoC $soc_t(n+1)$ as follows:

$$e_d = \left(soc(n) + \hat{e}_h(n+1) - soc_t(n+1) \right) / c_{bat} \tag{10}$$

The first sum extrapolates the SoC on the next day based on the predicted harvested energy, and the divider decouples the magnitude of the control error from the battery capacity.

Using the current control error e_d given by (10), the sum of all previous and the current control error, and the change in control error since the last execution of PreAct, the controller computes the current control output $\phi(n)$ via

$$\phi(n) = \left[K_p \cdot e_d(n) + K_i \cdot \sum_{0}^{n} e_d(i) + K_d \cdot (e_d(n) - e_d(n-1)) \right] \tag{11}$$

The PID coefficients K_p, K_i, and K_d in (11) can be determined using established methods; for example, we use the Nelder-Mead method [24] in the evaluation of PreAct in Sec. 5.

Finally, we need to ensure that the application duty cycle takes only values between 0 and 1:

$$dc(n+1) = \begin{cases} 0 & \text{if } \phi(n) < 0 \\ 1 & \text{if } \phi(n) > 1 \\ \phi(n) & \text{otherwise} \end{cases} \tag{12}$$

PreAct sets the application duty cycle to $dc(n+1)$ given by (12) until it executes again at the end of the next day.

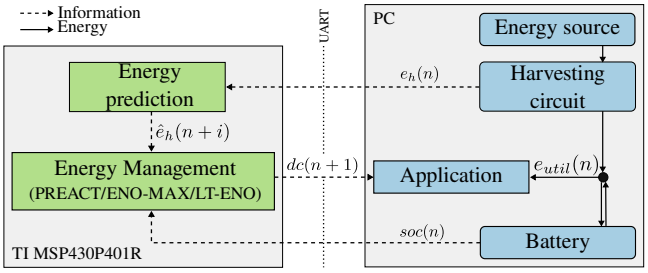


Figure 6: Trace-based simulations using an implementation of PreAct, LT-ENO, and ENO-MAX running on an MCU and a simulator running on a PC.

4 IMPLEMENTATION

We prototype PreAct on the TI MSP432P401R platform, which features an ultra-low-power 32-bit ARM Cortex-M4F MCU with a floating-point unit running at 48 MHz clock rate. For comparison, we also implement two state-of-the-art energy-management algorithms, LT-ENO [6] and ENO-MAX [33], on the same platform (see Sec. 5.1.1 for a detailed description of the configuration parameters of the three algorithms). Our implementations use only standard libraries and consist of 51 (ENO-MAX), 156 (PreAct), and 159 (LT-ENO) lines of C++ code. Using gcc 6.3.1 with optimization level `-O3`, the compiled code of the algorithms has a memory footprint of 0.67 kB–1.8 kB in flash and 0.06 kB–1.8 kB in RAM (without stack).

5 EVALUATION

We evaluate and compare PreAct to the state-of-the-art algorithms LT-ENO [6] and ENO-MAX [33] on a large dataset of 11 years of real solar radiation traces. Our main research questions are:

- What is the runtime overhead when executing the energy-management algorithms on a real sensor node? (Sec. 5.2)
- How well can the algorithms adapt to different application-dependent system dimensions in terms of harvestable solar energy and available battery capacity (Sec. 5.3)
- To what extent is the performance affected by the alignment between utility function and harvested energy? (Sec. 5.4)
- How robust are the algorithms to real-world inefficiencies and uncertainties of energy-harvesting systems? (Sec. 5.5)

This part of our evaluation is complemented by a real-world case study in Sec. 6 that evaluates the performance gains of PreAct over the state of the art in terms of an application-level metric.

5.1 Experimental Setup and Methodology

5.1.1 Setup and Approach. As shown in Fig. 6, the experimental setup consists of a TI MSP430P401R-based sensor node and a PC, connected to each other via UART. The sensor node executes the energy-management algorithm (*i.e.*, our implementation of PreAct, LT-ENO, or ENO-MAX) and the energy prediction (only LT-ENO and PreAct). On the PC runs a trace-based simulator that takes a real solar radiation trace as input and accounts for all major components of a solar energy-harvesting system (panel, harvesting circuit, battery, etc.) and their non-idealities. For each day n in the solar trace, the simulator determines the energy harvested on that day $e_h(n)$ and the SoC at the end of that day $soc(n)$. Based on these two inputs, the sensor node executes the energy-management algorithm to compute the application duty cycle for the next day $dc(n+1)$, and informs the trace-based simulator accordingly. This process repeats for each day in the solar trace.

Table 2: The five locations from which we use 11 years of real solar radiation data for our evaluation.

Location	Climate Zone	Latitude [°]	Longitude [°]
Mamirauá	A	-2.2	-65.7
Alice Springs	B	-23.6	133.9
Berlin	C	52.5	13.4
Harbin	D	45.8	126.5
Ivujivik	E	62.4	-77.19

We note that our evaluation approach goes far beyond the one that has been used by previous work on long-term energy management for solar energy-harvesting systems. Similar to previous work, we also use trace-based simulations to enable a fair comparison of different algorithms, and consider multiple years of real solar radiation data from different climate zones to be able to draw meaningful conclusions; these two evaluation goals are extremely difficult to achieve in a real deployment. On top of that, however, (*i*) we are the first to actually implement and execute the energy-management algorithms on a real sensor node to evaluate their runtime overhead, and (*ii*) we simulate several real-world effects whose impact on performance has been neglected by previous work [5, 6, 17, 33], including inaccurate SoC estimations, battery aging, and self-discharge.

In the following, we describe the solar traces, utility functions, algorithms, and performance metrics we use. Afterward, we detail the implementation of the trace-based simulation.

5.1.2 Real Solar Radiation Traces. We select the five locations listed in Table 2 to reflect all five major climate zones according to [25]. For each location, we use the daily average global horizontal irradiance over an 11-year period (1989 to 1999). The data were obtained from the NASA Langley Research Center Atmospheric Science Data Center Surface Meteorological and Solar Energy (SSE) web portal supported by the NASA LaRC POWER Project [10].

5.1.3 Utility Functions. We consider three utility functions:

- *Uniform* matches applications with a time-invariant utility or with a utility that has unknown temporal variations.
- *Seasonal*, shown in Fig. 9, is representative of applications with a seasonal utility pattern, such as environmental monitoring. Here, utility may be high during a contiguous portion of the year and low during the remainder of the year.
- *Weekends*, shown in Fig. 10, matches the typical weekly routine of humans. For example, the utility of an application like beach water quality monitoring may be high on Saturday and Sunday but low on the other (working) days of the week.

5.1.4 Compared Algorithms. Besides PreAct, we have evaluated four state-of-the-art energy-management algorithms, two predictive [6, 17] and two reactive [20, 33] approaches. Below, we present results for the best algorithm from each of the two classes.

- LT-ENO is the state-of-the-art predictive approach for solar energy-harvesting systems and assumes a *time-uniform*

utility [6]. Based on the energy-prediction algorithm from Sec. 3.2, it dynamically determines a stable duty cycle that allows for fully charging the battery during times of energy surplus to sustain operation during times of energy deficit. To this end, LT-ENO relies on a well-dimensioned battery determined according to a capacity planning method [6]. We use this method to allow for a fair comparison. Further, as recommended by the authors, we use a window size of $M = 63$ for the dynamic scaling of the solar predictions.

- ENO-MAX is the best known reactive method for long-term energy management. It uses a control algorithm to determine the application duty cycle based on the battery capacity and the current SoC such that energy-neutral operation is achieved over a configurable time frame. We set $\alpha = 1/180$ (targeting energy-neutral operation over 180 days), which performed best in our evaluation. To reduce duty cycle variance, the controller output is passed through an exponentially weighted moving average filter with parameter β. We follow the authors' suggestion and set $\beta = 0.25$.
- PreAct is the new energy-management algorithm described in Sec. 3. Its ability to follow a given utility function depends on the PID coefficients. We fit one set of PID coefficients to each of the three utility functions using the Nelder-Mead method [24], and use them throughout the evaluation. PreAct's only other parameter is the window size M for the solar predictions: we use $M = 63$ as for LT-ENO.

LT-ENO requires a known linear relation between duty cycle and energy consumption of the application. We consider

$$e_{util}(n) = e_h^{max} \cdot dc(n) \tag{13}$$

This models an application that consumes energy proportional to the duty cycle, where a duty cycle of 1 corresponds to consuming the maximum energy e_h^{max} harvested on a single day during the 11-year period captured by a given solar trace in our data set.

5.1.5 Performance Metrics. We evaluate the overhead in terms of the *time* and *energy* needed for one algorithm execution on the TI sensor platform (*i.e.*, overhead per day). We measure the execution time with an oscilloscope by toggling a GPIO pin on the target MCU and the energy consumption using TI's EnergyTrace method for a constant supply voltage of 3.3 V. For a given utility function and solar radiation trace, we report the performance of an algorithm in terms of its *effectiveness* computed over the entire trace using (2).

5.1.6 Simulator Details. We implement the trace-driven simulator in about 700 lines of Python code. It simulates a solar energy-harvesting system consisting of a solar panel, a harvesting circuit, a lithium-ion battery, and an application that utilizes energy according to (13). Given a solar radiation trace as input, the simulator scales the amount of harvestable energy according to the panel size, the panel efficiency, and the efficiency of the harvesting circuit. We model a modern harvesting integrated circuit that can drive a load directly from harvested energy and can charge the battery with excess energy. The amount of harvested energy and the current SoC of the battery are measured and provided to the energy-management algorithm. If the battery is fully charged and the application draws less power than what could be supplied, then no energy is harvested. We consider a solar panel with an area of 25 cm^2 and an efficiency of 5 % [14]. The harvesting circuit works with an efficiency of 70 % [32]. Further, the system and the battery operate at a constant voltage of 3.3 V, and the initial SoC is 50 %.

Moreover, our simulator accounts for all major sources of inefficiency and uncertainty in a real energy-harvesting system:

- *Charging efficiency:* Lithium-ion batteries have charging and discharging efficiencies of 90 % or higher depending on temperature and current [16, 21]. We therefore scale the energy that flows into and out of the battery by 0.9.
- *Capacity degradation:* Several studies suggests that battery calendar aging is significant over the course of multiple years, causing the capacity of common lithium-ion batteries to decrease by 1 %–8 % per year [11, 18]. To account for aging effects, we decrease the battery capacity by a fixed amount each day that corresponds to an annual degradation of 5 %.
- *Self-discharge:* Self-discharge is typically modeled as a constant leakage current [17] or an average percentage of SoC [15]. We adopt the model of a constant energy loss proportional to the SoC and decrease the SoC by a fixed percentage each day, corresponding to a self-discharge of 3 % in SoC per month [30].
- *SoC uncertainty:* Obtaining precise measurements of the battery's SoC is difficult; for example, voltage-only SoC estimators achieve an accuracy of 95 % [4]. We model SoC uncertainty using $soc(n) \cdot (1 + \epsilon)$, where ϵ is measurement noise and normally distributed with mean 0.03 and variance 0.02.
- *Consumption uncertainty:* Similarly, it is difficult to accurately predict the application's energy consumption based on the duty cycle. In Global Positioning System (GPS) tracking, for example, the time to get a fix depends on the satellite constellation and other factors that are hard to predict. We model energy consumption uncertainty using $e_{util}(n) \cdot (1 + \gamma)$, where $e_{util}(n)$ is given by (13) and the uncertainty γ is normally distributed with mean -0.05 and variance 0.05.

Whenever we refer to the *ideal model* below, these five real-world effects do not influence the simulated amount of harvested energy during a day $e_h(n)$ and the simulated SoC at the end of a day $soc(n)$.

5.2 Run-time Overhead

We start by looking at the runtime overhead of our prototype implementations of the three energy-harvesting algorithms.

Finding: *PreAct and the two comparison algorithms have a negligible overhead in terms of energy consumption and execution time.*

Settings. We feed our evaluation framework with the solar trace from Harbin, which features both seasonal (long-term) and weather-related (short-term) variations. We perform one run over the entire trace for each algorithm, and measure execution time and energy consumption per algorithm execution on the sensor node.

Results. Table 3 lists the average execution time and the average energy consumption per execution of the three energy-management algorithms. We see that, although PreAct has the largest overhead, the 85.3 µJ it consumes (per day) is still negligible, considering that this is less than 0.001 % of the energy harvested on any day in our solar traces. The execution times are on par with the transmission time of a single packet using a low-power IEEE 802.15.4 radio. We thus conclude that all three algorithms have negligible overhead.

Table 3: Average execution time and average energy consumption for one execution of PreAct, LT-ENO, and ENO-MAX on the TI MSP430P401R platform. *The overhead (per day) of all three energy-management algorithms is negligible in practice.*

Algorithm	Execution Time [ms]	Energy Consumption [µJ]
PreAct	6.0	85.3
LT-ENO	3.2	43.6
ENO-MAX	0.8	9.7

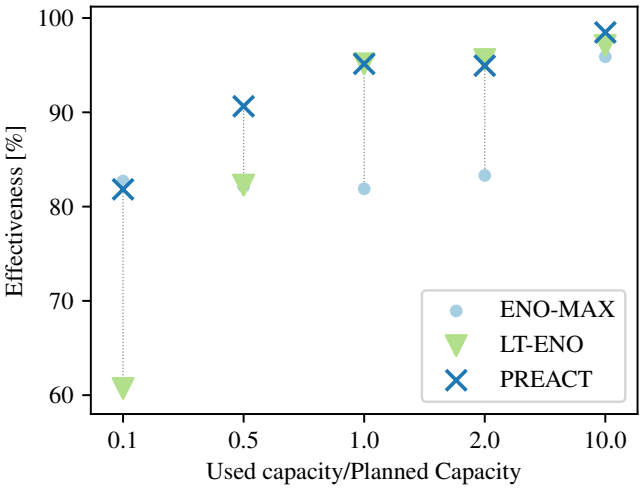


Figure 7: Effectiveness of PreAct, LT-ENO, and ENO-MAX for different ratios of used battery capacity to planned battery capacity. *PreAct performs well across a wide range of system dimensions, achieving 35 % higher effectiveness in some cases.*

5.3 Impact of System Dimensioning

The ability to exploit the battery as temporary energy storage for realizing a certain energy utilization depends on the ratio of harvested energy to available battery capacity. In practice, however, the battery capacity and solar panel size are often constrained or predefined by the application scenario and the available hardware.

Finding: *Unlike LT-ENO and ENO-MAX, PreAct performs well for a wide range of battery capacity constraints and achieves up to 35 % higher effectiveness than the two state-of-the-art algorithms.*

Settings. To systematically evaluate this aspect, we use the ideal model. We use again the solar data from Harbin and consider a uniform utility function to allow for a fair comparison against LT-ENO. To test different ratios of harvested energy to available capacity, we fix the panel size and vary the battery capacity. Using the capacity planning method employed by LT-ENO [6], we determine the *planned capacity* (7500 mAh). Then, we determine five *used capacities* that correspond to 0.1×, 0.5×, 1×, 2×, and 10× of the planned capacity, and simulate each algorithm for these five capacities.

Results. Fig. 7 plots the effectiveness of the three algorithms for the five ratios of used capacity to planned capacity. We see that PreAct performs well across all ratios, whereas LT-ENO and ENO-MAX perform poorly for certain ratios. In fact, PreAct's effectiveness consistently ranges between 82 % and 98 %, which is in some cases 35 % higher than the effectiveness of LT-ENO and ENO-MAX.

A closer look at the results in Fig. 7 reveals that LT-ENO underperforms for under-dimensioned capacity. In this case, the battery is fully charged most of the time, so LT-ENO cannot benefit from times of energy surplus. ENO-MAX, instead, takes the battery capacity into accounts and is therefore able to compromise between providing a uniform duty cycle and keeping the SoC within limits. PreAct's ability to compromise between realizing the ideal energy utilization in the long run versus immediately spending the harvested energy makes for an excellent performance across a much wider range of system dimensions.

Table 4: Alignment between utility function and harvested energy. *A value of 100 % means that the two are perfectly aligned.*

Function	Mamiraua	Alice Springs	Berlin	Harbin	Ivujivik
Uniform	88.7 %	88.0 %	69.8 %	79.2 %	61.2 %
Seasonal	63.4 %	71.3 %	35.1 %	46.5 %	26.2 %
Weekends	81.0 %	81.3 %	67.3 %	75.8 %	59.8 %

5.4 Impact of Alignment between Utility Function and Harvested Energy

Realizing a desired application duty cycle according to the utility function becomes more difficult if the utility function and the energy availability in terms of harvested energy are misaligned. We investigate next how this (mis-)alignment affects the performance of the three energy-management algorithms.

Finding: *PreAct achieves 81 %–98 % effectiveness across all alignments between utility function and harvested energy in our dataset, whereas LT-ENO and ENO-MAX perform significantly worse if they are misaligned. Overall, PreAct improves performance by up to 53 %.*

Settings. We use the ideal model and additionally exclude the effect of limited battery capacity (to be fair toward LT-ENO) by using a battery that has 10× the planned capacity for each location. One way to quantify the alignment between a utility function and energy availability according to a given solar trace is to compute effectiveness via (2) assuming that any harvested energy is directly utilized, that is, $e_{util}(n) = e_h(n)$ where $e_h(n)$ is given by (3).

Table 4 lists the alignment for all 15 combinations of utility functions and solar traces in our dataset. We find the lowest value for the seasonal utility function in Ivujivik (26.2 %) far in the north, where energy is mainly available in summer. The highest value is found for the uniform function in Mamiraua (88.7 %), which is close to the equator and thus enjoys uniform energy availability throughout the year. We test each algorithm against all 15 combinations.

Results. Fig. 8 plots effectiveness against alignment, including a linear fit for each algorithm to better observe the trends. Looking at the trends, we find that all algorithms generally perform better as the alignment increases. PreAct achieves significantly higher effectiveness than the comparison algorithms in all cases with an alignment factor of 75 % or lower. Overall, it performs up to 53 % and 45 % better than LT-ENO and ENO-MAX because it strategically controls the application duty cycle to match the utility function.

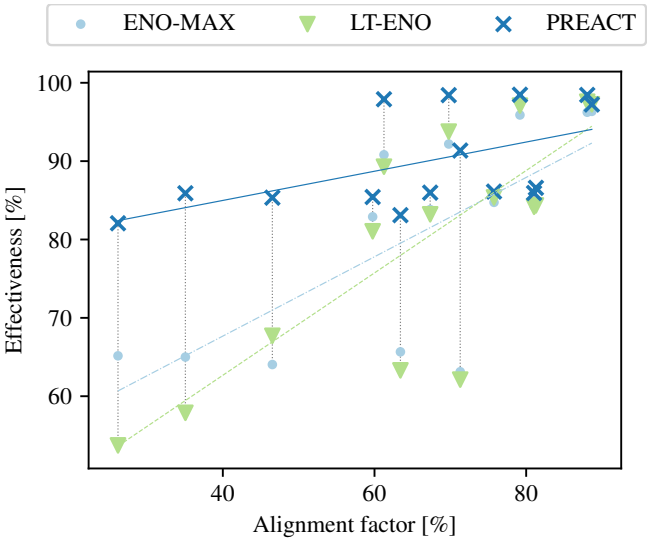


Figure 8: Effectiveness of PreAct, LT-ENO, and ENO-MAX depending on the alignment between utility function and harvested energy. *PreAct performs well even if the alignment is very low, outperforming LT-ENO and ENO-MAX by up to* 53 %.

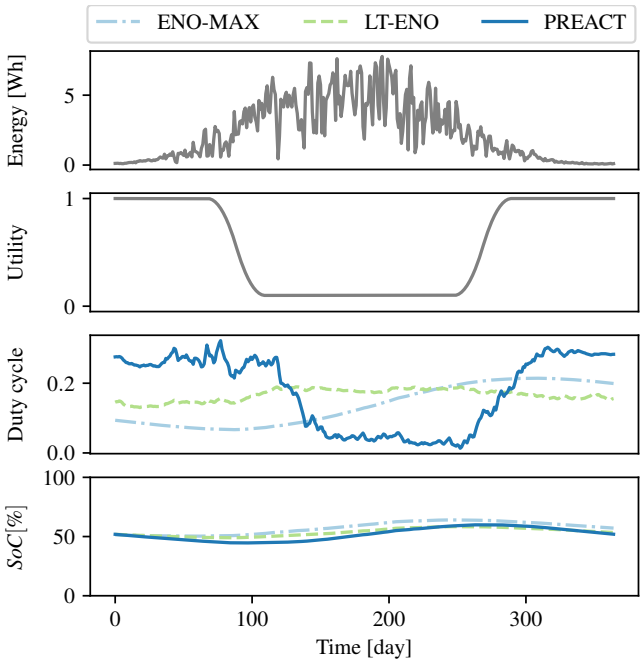


Figure 9: Harvested energy, utility function, duty cycle, and SoC over time for the combination with the lowest alignment in Table 4. *PreAct provides a significantly higher duty cycle when utility is high while recharging the battery when utility is low.*

This behavior can be seen in Fig. 9, which plots duty cycle and SoC over time for the combination with the lowest alignment. We see that PreAct achieves the highest duty cycle when utility is high while recharging the battery when utility is low—and this occurs regardless of the considerable misalignment. In Fig. 10 we see that PreAct is also able to follow the fine-grained *weekends* utility

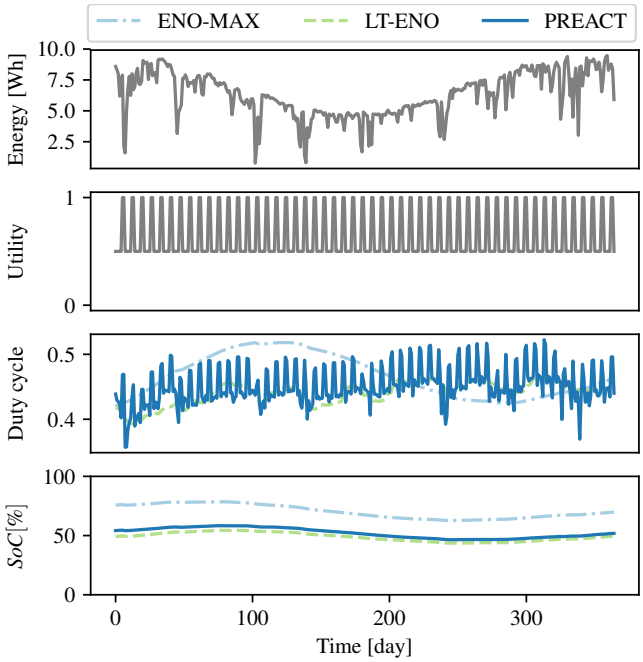


Figure 10: Harvested energy, utility function, duty cycle, and SoC over time for the weekends utility function in Alice Springs. *Even though PreAct employs strategic long-term planning to maximize effectiveness, it can follow fine-grained utility functions.*

function. However, the more aggressive setting of the controller leads to an increased sensitivity towards weather-related variations, evident from the low duty cycle around day 10, where the harvested energy is significantly reduced due to cloud coverage.

In most cases, LT-ENO and ENO-MAX perform similarly and the corresponding trends have almost the same slope. ENO-MAX's goal is to reduce duty cycle variance, but its approach has previously been found to be incapable of dealing with long-term variations of solar energy [6]. In some cases, the remaining variations in ENO-MAX's budget *coincidentally* align with utility so that effectiveness is higher than with LT-ENO.

5.5 Impact of Uncertainties and Inefficiencies

Uncertainties such as inaccurate SoC estimations and inefficiencies like battery self-discharge are present in any real-world energy-harvesting system. We evaluate the robustness of the three energy-management algorithms to such effects.

Finding: *PreAct is highly robust to various real-world uncertainties and inefficiencies. As a result, its effectiveness is only slightly lower than without inefficiencies, while effectiveness of LT-ENO can be reduced by up to* 58 %.

Settings. For a fair comparison against LT-ENO and ENO-MAX, we consider the uniform utility function. Based on solar data from Harbin, we perform several runs in which we activate none (ideal model), all, or only one of the five sources of inefficiency and uncertainty (see Sec. 5.1.6). To quantify the impact of each individual effect on the performance of the three energy-management algorithms, we report below the decrease in effectiveness with respect to the ideal model (*i.e.*, without any uncertainty or inefficiency).

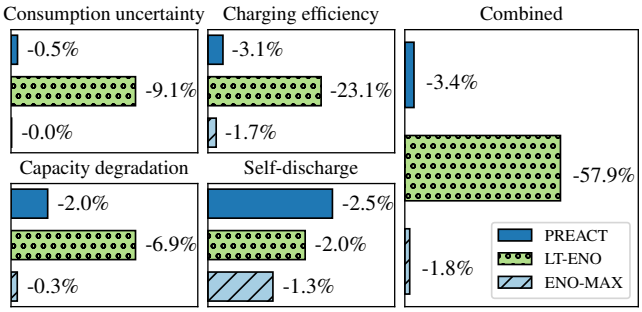


Figure 11: Decrease in effectiveness under different real-world effects with respect to the ideal model, for PreAct, LT-ENO, and ENO-MAX. *PreAct is highly robust to various sources of uncertainty and inefficiency, while LT-ENO is significantly affected.*

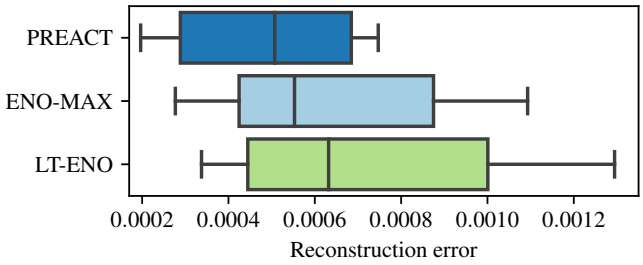


Figure 12: Reconstruction error in soil water dynamics computed across all ten sensors nodes and the whole observation period, for PreAct, LT-ENO, and ENO-MAX. The plot shows minimum, maximum, median, and 25th and 75th percentiles. *PreAct improves the average application-level performance by up to 28 % compared with two state-of-the-art algorithms.*

Results. Fig. 11 shows the decrease in effectiveness of the three algorithms under energy consumption uncertainty, charging inefficiency, capacity degradation, or self-discharge and when all effects affect the system at the same time (combined). The decrease in effectiveness under SoC estimation uncertainty was below 1 % for all algorithms, so we do not show it in Fig. 11.

We find that the decrease for PreAct is very low across the board. When combining all non-idealities, PreAct's performance degrades by only 3.5 %, which is on par with the 1.8 % decrease of ENO-MAX. Both PreAct and ENO-MAX are highly robust to inefficiencies and uncertainties thanks to the underlying feedback control mechanism. For example, in the case of energy consumption uncertainty, the two methods automatically adapt the application duty cycle to the current conditions without relying on error-prone estimates of the actual energy consumption. This robustness is a key asset for enabling multi-year, unattended operation of solar energy-harvesting sensing systems in which an accurate characterization of all hardware-related effects is next to impossible.

By contrast, the performance of LT-ENO decreases by 57.9 % when its strong assumptions are not met. This may lead to suboptimal performance in real-world scenarios, where future harvested energy and system parameters cannot be accurately predicted.

6 REAL-WORLD CASE STUDY

The previous section mainly explored the performance of PreAct in terms of effectiveness depending on various impact factors. We now consider a real-world application to evaluate and compare PreAct's performance in terms of an actual application-level metric.

Finding: *By exploiting prior knowledge of the wet season in sub-tropical Eastern Australia, PreAct increases the performance of a microclimate sensor for mine rehabilitation by up to 28 % compared with LT-ENO and ENO-MAX. Further, the performance of the microclimate sensor correlates with effectiveness, which validates the faithfulness of the effectiveness metric to application-level performance.*

Application scenario. Soil moisture dynamics has been identified as a reliable indicator of rehabilitation success of land that was used for surface mineral extraction [28]. Using microclimate sensors that measure soil water content, it is possible to quantify water retention time, which is an important factor for tree growth [19]. Clearly, this retention time is only visible in soil water after rainfall. Therefore, measuring soil water content exhibits a distinct time-varying utility, where utility is high when there has been rain and low otherwise.

We use data from a real deployment of ten sensor nodes in an open-cut surface mine in Eastern Australia from 09/2012 to 07/2014. Each sensor node measured solar radiation and soil water content periodically every 5 min and transmitted both readings to a sink node using its low-power wireless radio. Soil moisture dynamics is the first derivative of the soil water content measurements reported by the sensor nodes; it captures the rate of water discharge from the soil, which is highly correlated with soil quality and vegetation regrowth. Using these data as ground truth, we now evaluate the performance of this application if the ten wireless sensor nodes were harvesting solar energy and taking energy-management decisions using PreAct, LT-ENO, or ENO-MAX. The duty cycle determined by the energy-management algorithms directly affects the rate with which each sensor measures and reports soil water content.

Application-level metric. We quantify the application-level performance using the *reconstruction error*, defined as the mean squared error (MSE) between the ground truth soil moisture dynamics and the (reconstructed) soil moisture dynamics obtained when the sampling rate is controlled by an energy-management algorithm.

Settings. We consider a solar panel size of 15 cm^2, a battery capacity of 3300 mAh, and a baseline power draw of 1 mW. The energy consumed by a sensor node for obtaining and transmitting one soil water content reading is 1.5 J. To specify a utility function, we note that sub-tropical climate in Eastern Australia has a characteristic wet season with an increased chance of precipitation. Specifically, using data from the Australian Bureau of Meteorology [13], we compute the utility function shown in Fig. 13 based on monthly average rainfall of the years 2001 to 2011.

Results. Fig. 12 shows the reconstruction error computed across all ten sensor nodes and over the whole observation period of almost two years. We see that PreAct outperforms LT-ENO by 28 % and ENO-MAX by 23 % in terms of average reconstruction error. The improvements are even larger when looking at the maximum error.

Fig. 13 shows, for one of the nodes, the ground truth soil water content measurements and the sampling rate chosen by the energy-management algorithms. PreAct provides the highest sampling rate of all algorithms when utility is high and most rainfall events

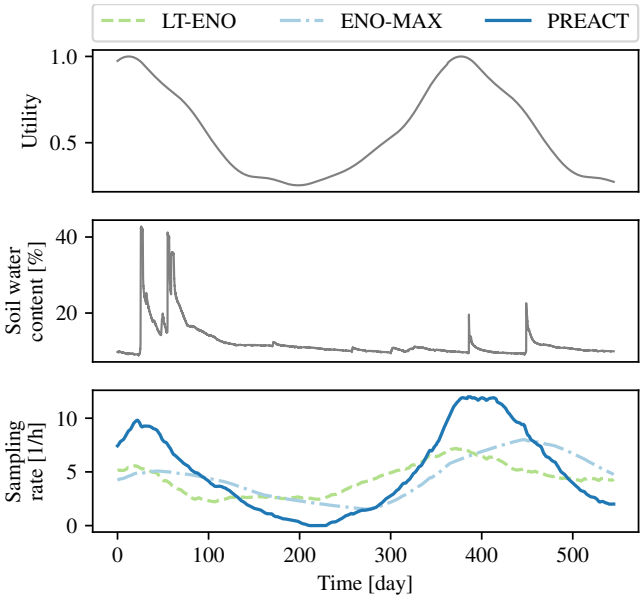


Figure 13: Utility function, ground truth of soil water content measurements, and sampling rate chosen by PreAct, LT-ENO, and ENO-MAX for one of the ten deployed microclimate sensors in our real-world case study. *By exploiting prior knowledge of the wet season through the utility function, PreAct can sustain the highest sampling rate during and after rainfall events. This allows for a much more accurate reconstruction of soil water dynamics compared with LT-ENO and ENO-MAX.*

occur, as visible from the spikes in soil water content. This explains the significantly lower reconstruction error when using PreAct.

We also determine Spearman's correlation coefficient between the reconstruction error and the effectiveness of each algorithm across all ten sensor nodes. We find that there is a strong correlation of 0.62 ($p = 0.0002$) between the application-level metric and the effectiveness metric, although rainfall events are hard to predict precisely and the utility function in this example is only a rough estimate. This validates the usefulness of the effectiveness metric both as an optimization objective in the design of energy-management algorithms and as an application-agnostic performance metric when evaluating and comparing energy-management algorithms.

One may think that simply increasing the sampling rate when rainfall is detected would be a viable approach in this scenario, similar to adaptive sampling algorithms for purely battery-powered sensor nodes [1]. However, the system can only sustain a higher average sampling rate during and after rainfall events if it has proactively planned for this increased energy utilization during the wet season by taking into account predictions on future harvested energy, the application's time-varying utility, and the battery capacity constraints. PreAct addresses this need with a novel solution that is also highly robust if the conditions do not evolve as predicted.

7 RELATED WORK

Reactive algorithms. Vigorito et al. consider a linear-quadratic tracking problem, keeping the SoC at a configurable level to avoid depletion and waste when the battery cannot store excess energy [33]. By passing the output of the controller through a exponentially weighted moving average (EWMA) filter, they reduce short-term variations of the duty cycle. We implemented this method and found it to be adaptive and robust to non-idealities. However, it couples system performance with seasonal energy availability, leading to underperformance when utility is misaligned.

Le et al. use a PID controller to maintain a favorable, constant SoC [20]. By adapting the wake-up period of the node to deviations from this set-point, their system achieves long-term energy neutrality. Unlike their approach, the set-point in PreAct is time-dependent and takes into account energy predictions and utility.

The near-optimal energy management scheme proposed in [3] learns daily patterns of energy input and output, and optimizes energy allocation over a 24-hour horizon. By re-distributing unexpected surplus and deficit energy over short time intervals, the allocated energy is close to the optimal, offline solution.

Reactive algorithms are fundamentally unable to consider time-varying utility. PreAct borrows their key idea and combines it with a predictive, strategical approach. This way, PreAct inherits the advantages of reactive methods, adaptivity and robustness, yet optimizes utilization according to a long-term objective.

Predictive algorithms. Kansal et al. propose power management for energy-harvesting sensor networks, introducing basic terms and outlining specific challenges [17]. Their approach to split the day into time slots and to make predictions on them based on historical values was adopted and refined by other researchers [8, 9, 26]. Various authors incorporated predictions of harvested energy into short-term energy management algorithms [2, 22, 23, 29]. This short-term approach performs sub-optimally in the long run when application utility does not match energy availability [6].

The authors of [31] achieve short-term energy neutrality with high robustness using a novel reward function for reinforcement learning. They find that including short-term weather predictions further reduces deviations from energy-neutral operation.

The authors of [5, 6] are the first to propose an optimization for a prediction horizon corresponding to the long-term variations in the solar energy source. After dimensioning the system according to a capacity planning method [7], the run-time algorithm from [6] uses a lightweight optimization procedure to iteratively determine the duty cycle for a day to uniformly balance predicted energy input and utilization over a one-year horizon. By contrast, PreAct allows for a time-varying utility, and exploits this to conserve energy when utility is low to optimize performance when utility is high.

In [5] the authors pick up on their previous work and formulate a model predictive control problem for guaranteed minimum energy utilization. The method only calculates the optimized utilization, without detailing, how it can be realized. The approach to calculate a target SoC within a limited prediction horizon is similar to our method. However, the underlying objective function and approximation of the solution are fundamentally different.

8 DISCUSSION AND CONCLUSIONS

This paper shows how prior knowledge of the time-varying utility of sensed data can be used to optimize the performance of energy-harvesting systems. We thus presented PreAct, the first

energy-management algorithm that takes advantage of this opportunity. PreAct exploits prior knowledge about time-varying utility when planning future energy utilization if such knowledge is available, but significantly improves performance compared with prior approaches even if it is not, as demonstrated by our evaluation. In addition, PreAct reactively tracks the determined target SoC to automatically compensate for deviations from predicted conditions. Extensive trace-based simulations show that compared with current long-term energy-management algorithms PreAct achieves significantly better application performance and is also more robust to real-world uncertainties and inefficiencies.

Finally, PreAct is versatile to fit a broad range of applications and harvesting modalities. In this paper, we focused on solar energy harvesting and long-term sensing applications, which are widely adopted in real deployments. More generally, the underlying concepts and PreAct's design are applicable to other energy sources and applications operating on different timescales as long as the following two requirements are met:

- The incoming energy must be predictable over the planned optimization horizon. The predictions can be rather coarse-grained because PreAct's feedback mechanism automatically compensates for these and other inaccuracies.
- The energy storage must be large enough to compensate for deviations between harvested energy and application utility.

For example, PreAct may be used in a health-monitoring application with kinetic energy harvesting, where characteristic human motion patterns and the application utility can be predicted and future energy utilization can be optimized over a one-day horizon using a relatively small battery. Exploring other harvesting sources, different timescales, and other energy-storage technologies is an interesting direction for future work. Moreover, we believe it would be possible to provide probabilistic guarantees on PreAct's energy neutrality and minimum application utility based on a formal analysis that accounts for statistical properties of the energy source.

ACKNOWLEDGMENTS

We are grateful to Lothar Thiele for thoughtful feedback on an earlier manuscript, and to our anonymous reviewers and shepherd for valuable comments, which helped improve the quality of this paper. This work was supported in part by the German Research Foundation (DFG) within the Cluster of Excellence cfaed (grant EXC 1056) and the Emmy Noether project NextIoT (grant ZI 1635/2-1).

REFERENCES

[1] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. 2007. Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications. In *Proc. of the 4th IEEE Int. Conf. on Mobile Adhoc and Sensor Systems (MASS)*.

[2] Y. Bao, X. Wang, X. Liu, S. Zhou, and Z. Niu. 2014. Solar radiation prediction and energy allocation for energy harvesting base stations. In *Proc of the IEEE Int. Conf. on Communications (ICC)*.

[3] G. Bhat, J. Park, and U. Y. Ogras. 2017. Near-optimal Energy Allocation for Self-powered Wearable Systems. In *Proc. of the 36th IEEE Int. Conf. on Computer-Aided Design (ICCAD)*.

[4] B. Buchli, D. Aschwanden, and J. Beutel. 2013. Battery State-of-Charge Approximation for Energy Harvesting Embedded Systems.. In *Proc. of the 10th European Conf. on Wireless Sensor Networks (EWSN)*. Springer.

[5] B. Buchli, P. Kumar, and L. Thiele. 2015. Optimal power management with guaranteed minimum energy utilization for solar energy harvesting systems. In *Proc. of the 11th IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*.

[6] B. Buchli, F. Sutton, J. Beutel, and L. Thiele. 2014. Dynamic Power Management for Long-term Energy Neutral Operation of Solar Energy Harvesting Systems. In *Proc. of the 12th ACM Conf. on Embedded Network Sensor Systems (SenSys)*.

[7] B. Buchli, F. Sutton, J. Beutel, and L. Thiele. 2014. Towards Enabling Uninterrupted Long-Term Operation of Solar Energy Harvesting Embedded Systems. In *Proc. of the 11th European Conference on Wireless Sensor Networks (EWSN)*. Springer.

[8] A. Cammarano, C. Petrioli, and D. Spenza. 2012. Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *Proc. of the 9th IEEE Int. Conf. on Mobile Ad-Hoc and Sensor Systems (MASS)*.

[9] A. Cammarano, C. Petrioli, and D. Spenza. 2016. Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks. *IEEE Sensors Journal* 16, 17 (2016).

[10] NASA Atmospheric Science Data Center. Surface meteorology and Solar Energy. https://eosweb.larc.nasa.gov/sse/. Accessed: 2017-02-20.

[11] M. Ecker, J. B. Gerschler, J. Vogel, S. KÃďbitz, F. Hust, P. Dechent, and D. U. Sauer. 2012. Development of a lifetime prediction model for lithium-ion batteries based on extended accelerated aging test data. *Journal of Power Sources* 215 (2012).

[12] V. V. Fireteanu, A. Tudor-Serban, I. S. Sacala, and M. A. Moisescu. 2014. Avalanche Prediction Based on Snow Level Monitoring Using Wireless Sensor Networks. *Applied Mechanics and Materials* 656 (2014).

[13] Australian Government Bureau for Meteorology. Climate Data Online. http://www.bom.gov.au/climate/data/. Accessed: 2018-03-21.

[14] M. A. Green, Y. Hishikawa, E. D. Dunlop, D. H. Levi, J. Hohl-Ebinger, and A. W. Y. Ho-Baillie. 2017. Solar cell efficiency tables (version 51). *Progress in Photovoltaics: Research and Applications* 26, 1 (2017).

[15] J. Jeong and D. Culler. 2012. Predicting the Long-Term Behavior of a Micro-Solar Power System. *ACM Trans. on Embedded Computing Systems (TECS)* 11, 2 (2012).

[16] J. Jiang and C. Zhang. 2015. *Fundamentals and application of lithium-ion batteries in electric drive vehicles.* John Wiley & Sons.

[17] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. 2007. Power Management in Energy Harvesting Sensor Networks. *ACM Trans. on Embedded Computing Systems (TECS)* 6, 4 (2007).

[18] P. Keil, S. F. Schuster, J. Wilhelm, J. Travi, A. Hauser, R. C. Karl, and A. Jossen. 2016. Calendar Aging of Lithium-Ion Batteries I. Impact of the Graphite Anode on Capacity Fade. *Journal of The Electrochemical Society* 163, 9 (2016).

[19] B. Kusy, C. Richter, S. Bhandari, R. Jurdak, V. J. Neldner, and M. R. Ngugi. 2015. Evidence-based landscape rehabilitation through microclimate sensing. In *Proc. of the 12th IEEE Int. Conf. on Sensing, Communication and Networking (SECON)*.

[20] T. N. Le, O. Sentieys, O. Berder, A. Pegatoquet, and C. Belleudy. 2012. Power manager with PID controller in energy harvesting wireless sensor networks. In *Proc. of the IEEE Int. Conf. on Green Computing and Communications (GreenCom)*.

[21] K. Li and K. J. Tseng. 2015. Energy efficiency of lithium-ion battery used as energy storage devices in micro-grid. In *Proc. of the 41st IEEE Ann. Conf. of the Industrial Electronics Society (IECON)*.

[22] Y. Li, Z. Jia, and X. Li. 2014. Task Scheduling Based on Weather Forecast in Energy Harvesting Sensor Systems. *IEEE Sensors Journal* 14, 11 (2014).

[23] C. Moser, L. Thiele, D. Brunelli, and L. Benini. 2007. Adaptive Power Management in Energy Harvesting Systems. In *Proc. of the Conference on Design, Automation and Test in Europe (DATE)*.

[24] J. A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7, 4 (1965).

[25] M. C. Peel, B. L. Finlayson, and T. A. McMahon. 2007. Updated world map of the Köppen-Geiger climate classification. *Hydrology and Earth System Sciences* 11, 5 (2007).

[26] J. R. Piorno, C. Bergonzini, D. Atienza, and T. S. Rosing. 2009. Prediction and management in energy harvested wireless sensor nodes. In *Proc. of the 1st Int. Conf. on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE)*.

[27] Project Providence. Project Providence - Biodiversity monitoring in the Amazon rainforest. http://www.projectprovidence.org/. Accessed: 2018-03-19.

[28] M. R. Ngugi, V. J. Neldner, D. Doley, B. Kusy, D. Moore, and C. Richter. 2015. Soil moisture dynamics and restoration of self-sustaining native vegetation ecosystem on an open-cut coal mine. *Restoration Ecology* 23, 5 (2015).

[29] J. Rodway and P. Musilek. 2016. Wireless sensor networks with pressure-based energy forecasting: A simulation study. In *Proc. of the 29th IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE)*.

[30] Sanyo Denki K.K. Lithium ion. http://www.rathboneenergy.com/articles/sanyo_lionT_E.pdf. Accessed: 2017-09-21.

[31] S. Shresthamali, M. Kondo, and H. Nakamura. 2017. Adaptive Power Management in Solar Energy Harvesting Sensor Node Using Reinforcement Learning. *ACM Trans. on Embedded Computing Systems (TECS)* 16, 5 (2017).

[32] J. Taneja, J. Jeong, and D. Culler. 2008. Design, Modeling, and Capacity Planning for Micro-solar Power Sensor Networks. In *Proc. of the 7th ACM Int. Conf. on Information Processing in Sensor Networks (IPSN)*.

[33] C. M. Vigorito, D. Ganesan, and A. G. Barto. 2007. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In *Proc. of the 4th IEEE Int. Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*.