The 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications

Reward Maximization for Embedded Systems with Renewable Energies

Clemens Moser, Jian-Jia Chen, and Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
Email: moser@tik.ee.ethz.ch, jchen@tik.ee.ethz.ch, thiele@tik.ee.ethz.ch

Abstract

Renewable energies can enable embedded systems to be functional indefinitely. In particular for small autonomous sensors, energy harvesting techniques have attracted much interest. This paper considers systems which provide services periodically with adjustable quality evaluated in terms of rewards. The reward garnered for one service is monotonically increasing and strictly concave with respect to the energy consumption of the service. There exist two major constraints which arise due to the burstiness of common energy sources: (1) The harvested energy is temporarily low and the service must be lowered or suspended. (2) During bursts, the harvested energy exceeds the battery capacity. To resolve these issues, we propose algorithms to derive optimal solutions which maximize the overall reward. Furthermore, we determine the minimum battery capacity necessary to optimally exploit a given power source. By applying real data recorded for photovoltaic cells as the harvested energy, simulations illuminate the merits of our algorithms.

1 Introduction

Power management has become an important system design issue for embedded systems since most embedded devices are powered by batteries. How to prolong the lifetime of battery-powered systems or how to reduce the energy consumption subject to performance or timing constraints has been studied extensively in the literature. This holds in particular for systems adopting dynamic voltage scaling [5, 25], dynamic power management [7, 11], and micro-architectural techniques for cache re-configuration [24].

For some services, the quality of the provided service depends on the amount of computation. Generally, the more computation required to provide a service, the more reward the system garners for the execution, such as the imprecise computation model [14] and the increasing reward with increasing service (IRIS) model [9, 22]. For most practical applications, such as image and speech processing, time-dependent planning, and multimedia applications, the reward function is usually a concave function of the amount of computation [3].

By specifying an energy constraint for scheduling to provide services with good quality in a certain interval, reward-based energy-efficient scheduling of real-time tasks has been studied in the literature, e.g., [2, 6, 8, 20, 21, 26, 27]. Specifically, Rusu et al. [20, 21] provide heuristic algorithms for scheduling multiple real-time tasks with different award. Alenawy and Aydin [2] later proposed heuristic algorithms for reward-based energy-efficient scheduling of periodic real-time tasks in off-line and on-line fashions. At the mean time, Chen et al. [6, 8] proposed approximation algorithms. Recently, researchers have started exploring reward-based energy-efficient scheduling of real-time tasks with multiple execution versions.

The possibility to harvest energy from the environment and to sustain everlasting operation has earned much interest recently. Wireless sensor networks is one area, where this approach is exceptionally interesting. Here, the energy generated by small solar panels suffices to execute most common data gathering applications. Consequently, numerous researchers have started to design energy harvesting circuits to efficiently convert and store solar energy [4, 10, 12, 18, 23].

In energy harvesting devices, the energy consumption of the system should depend on the energy harvested from the environment to maximize the performance instead of minimizing the energy consumption. An important observation is that most environmental power sources are not constant over time [19]. The solar energy generated by photovoltaic elements arrives in bursts, and has to be stored if, e.g., the device must be operational during night. Driven by solar energy, the main challenge for such a system is to optimize its performance while respecting the time-varying amount of energy. How to design and play out a given battery capacity becomes a key concern.

In this paper, we address, e.g., sensor nodes which are situated in an outdoor environment and are directly exposed to sunlight. Moreover, our results are applicable to other energy sources like vibrational energy which have some kind of periodic or predictable behaviour. To our

1533-2306/08 $25.00 © 2008 IEEE
DOI 10.1109/RTCSA.2008.25

best knowledge, this is the first paper to explore the maximization of system reward for energy harvesting systems with predictable energy sources.

This paper contains the following contributions:

- We formulate the *reward maximization on energy harvesting* problem, which is to maximize the rewards for a concave reward function due to constraints of renewable energy sources and the energy buffer.
- We propose polynomial-time algorithms that derive optimal assignments in energy consumption to maximize the overall reward.
- To provide insights for system designers, we show how to determine the minimum battery capacity and a sufficient prediction horizon for a given power source.
- Our results are supported by simulations using long-term measurements of photovoltaic energy.

The rest of this paper is organized as follows: Section 2 presents the related work to this study. Section 3 defines the system models and the studied problem. Section 4 provides the proposed algorithms as well as illustrative examples, while Section 5 shows the optimality of our proposed algorithms. Section 6 gives the related remarks for designing the embedded system. In Section 7, simulation results based on real data recorded for photovoltaic cells are presented. Section 8 concludes this paper.

2 Related Work

The problem addressed in [13] is to maximize the utilisation of *solar energy*, i.e., to minimize round-trip losses of the battery. Although the objective is completely different from the one in this paper, the work in [13] is one of the first to optimize *energy harvesting* systems.

Rusu et al. [20] explore the reward maximization for a set of real-time tasks with multiple versions for execution by applying energy harvesting devices. The execution frames are divided into two types, namely recharging frames and discharging frames. These two types of frames are then executed by applying their static schedules individually. If the scheduler observes more energy residual in the battery, three different approaches are proposed to distribute the additional energy for getting more system reward. Our paper focuses on a more fundamental problem to maximize the system reward globally, in which the energy consumption in all recharging frames and discharging frames might be different to achieve the global optimization.

In [15], it is pointed out that greedy scheduling disciplines are not suitable if tasks on a uniprocessor are processed using time as well as regenerative energy. An optimal scheduling algorithm which tries to avoid deadline violations is presented. In contrast, the application discussed in this paper requires sequential execution of a periodic task. Task preemption is not allowed, and also not necessary. Instead, we try to optimize the overall reward of an application.

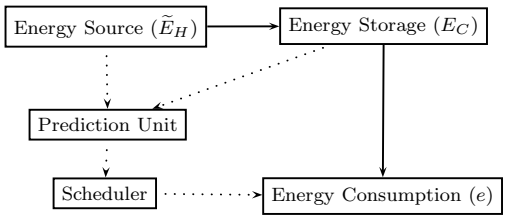


Figure 1. Illustration of the system model

The authors of [16] show that many optimization problems arising in energy harvesting systems can be modeled by the class of linear programs. A multiparametric linear programming approach is presented which solves optimization problems offline and stores large look-up tables for online usage. As objective, (piecewise) linear functions are possible whereas this paper focusses on concave objective functions. Moreover, we present very simple but optimal algorithms for a specific system dynamic which solely requires the storage of a few internal variables.

3 System Model and Problem Definition

This paper explores the reward maximization for a system equipped with energy harvesting devices, such as solar panels. The energy consumption of the system should depend on the energy harvested from the environment to maximize the performance instead of minimizing the energy consumption. The system model is depicted in Figure 1, in which the harvested energy from the energy source is stored in the energy storage, and the scheduler decides how to consume the energy based on the information provided by the prediction unit and the available energy in the energy storage. This section will present the energy harvesting model for the energy source, the energy storage model, the service model used for scheduling, and the problem considered in this paper.

3.1 Energy Harvesting Model

We are given an energy harvesting device, such as a solar panel, which generates energy depending on the environment. A prediction unit estimates the energies harvested in each of the next K frames in the future, where K is the *number of frames of the prediction horizon*. We assume that each frame has the same length and the basic time unit is the length of one frame. We denote $\widetilde{E}_H(k)$ the accumulated energy harvested in the k-th frame. For instance, a frame may correspond to one hour, and having $K = 24$ would correspond to a prediction horizon of one

day. How to determine a reasonable parameter K will be presented in Section 7 by means of simulation.

For the rest of this paper, we assume a perfect energy predictor. For a discussion about suitable energy prediction algorithms or how to handle prediction mistakes, the reader is referred to [17].

3.2 Energy Storage Model

The harvested energy is stored in the energy storage, e.g., in a supercapacitor or a battery. To store the energy $\widetilde{E}_H(k)$ harvested in the k-th frame, we assume an efficiency factor $0 \leq \alpha(\widetilde{E}_H) \leq 1$ which is usually a function of the harvested energy. Only $\alpha(\widetilde{E}_H) \cdot \widetilde{E}_H(k)$ amount of the harvested energy will be stored to the energy storage in the k-th frame. For brevity, let $E_H(k)$ be the amount of harvested energy that will be stored to the energy storage in the k-th frame, where $E_H(k)$ is $\alpha(\widetilde{E}_H) \cdot \widetilde{E}_H(k)$. For simplicity of presentation, for the rest of this paper, we, implicitly, denote $E_H(k)$ as the harvested energy in the k-th frame.

Let $E_{\max}$ be the maximum capacity of the energy storage and $E_C(k)$ be the energy in the energy storage at the end of the k-th frame. After the service of the k-th frame with energy consumption e_k, the residual energy in the energy storage is $\min\{E_{\max}, E_C(k-1) + E_H(k) - e_k\}$. That is, if $E_C(k-1)+E_H(k)-e_k$ is larger than $E_{\max}$, the system loses some amount, $E_C(k-1)+E_H(k)-e_k-E_{\max}$, of energy due to the capacity constraint.

3.3 Service Model

This paper targets at services for a variety of different applications. The higher the computation/workload/demand of the assigned service, the more reward the system gains for the execution, such as the imprecise computation model [14] and the increasing reward with increasing service (IRIS) model [9,22]. For each frame, we have to determine how to provide the service by the scheduler. The quality of the provided service is evaluated in each frame, in which the reward function is a strictly concave and increasing function of the amount of computation, such as image and speech processing, time-dependent planning, and multimedia applications.

The energy consumption for a given workload of provided service is assumed to be a convex function (when dynamic voltage scaling is adopted [6,25]) or a linear function (when the power consumption is a constant). Therefore, the reward function is a strictly concave and increasing function of the energy consumption. For the rest of this paper, we will only discuss the amount of energy consumption in each frame, while the required computation time to complete the service in a frame with the specified energy consumption can be derived by simple calculation.

Let $r(\epsilon)$ denote the reward for executing the service in a frame with energy consumption ϵ, where

- $r(\epsilon)$ is monotonically increasing in ϵ.
- $r(\epsilon)$ is strictly concave in ϵ, i.e.,

$$\alpha \cdot r(\epsilon_1) + (1-\alpha) \cdot r(\epsilon_2) < r\left(\alpha \cdot \epsilon_1 + (1-\alpha) \cdot \epsilon_2\right),$$

for any $\epsilon_1 \epsilon_2 \geq 0$, $\epsilon_1 \neq \epsilon_2$, and $1 > \alpha > 0$.

Based on the concavity of the reward function, the following lemma holds.

Lemma 1 *If $\epsilon_1 + \epsilon_2 = \epsilon_3 + \epsilon_4$ with $0 \leq \epsilon_1 < \epsilon_3, \epsilon_4 < \epsilon_2$, then $r(\epsilon_1) + r(\epsilon_2) < r(\epsilon_3) + r(\epsilon_4)$.*

Proof. Let α_3 be $\frac{\epsilon_2-\epsilon_3}{\epsilon_2-\epsilon_1}$ and α_4 be $\frac{\epsilon_2-\epsilon_4}{\epsilon_2-\epsilon_1}$. Since $\epsilon_1 < \epsilon_3, \epsilon_4 < \epsilon_2$, we have $0 < \alpha_3 < 1$ and $0 < \alpha_4 < 1$. Because $\epsilon_1+\epsilon_2 = \epsilon_3+\epsilon_4$, we know that $\alpha_3+\alpha_4 = 1$. Therefore, by the concavity, we conclude $r(\epsilon_1)+r(\epsilon_2) = (\alpha_3+\alpha_4)r(\epsilon_1)+(2-\alpha_3-\alpha_4)r(\epsilon_2) < r(\alpha_3\epsilon_1 + (1-\alpha_3)\epsilon_2) + r(\alpha_4\epsilon_1 + (1-\alpha_4)\epsilon_2) = r(\epsilon_3) + r(\epsilon_4)$. □

3.4 Problem Definition

We are given a predictor for K frames at time 0, in which the energy in the energy storage at time 0 is specified as $E_C(0)$ and the energy harvested in the k-th frame is $E_H(k)$. The k-th frame starts from time $k-1$ to time k. The problem is to find an *assignment* $\vec{e} = (e_1, e_2, \ldots, e_K)$ of energy consumption for these K frames so that the reward is maximized without violating the required energy constraint. The reward of an assignment $\vec{e}$ is $\sum_{k=1}^{K} r(e_k)$. Let $E_C(k, \vec{e})$ be the energy in the energy storage at time k by applying the assignment of energy consumption $\vec{e}$. After completing the last frame, we would like to reserve some amount E_ℓ of energy in the energy storage for future use, and, hence, a feasible assignment $\vec{e}$ must satisfy $E_C(K, \vec{e}) \geq E_\ell$. We denote the studied problem as the *reward maximization on energy harvesting* problem. Without loss of generality, we only explore the case that $E_C(0) - E_\ell + \sum_{i=1}^{K} E_H(i) \geq 0$ in this paper since there is no feasible solution for the other case.

We formally define the feasibility of an assignment for the reward maximization on energy harvesting problem as follows:

Definition 1 *[Feasible Assignment] An energy vector $\vec{e} = (e_1, \ldots, e_K)$ is feasible if*

(a) $E_C(k, \vec{e}) = \min\{E_{\max}, E_C(k-1, \vec{e}) + E_H(k) - e_k\}$, *where $E_C(0, \vec{e})$ is $E_C(0)$,*

(b) $E_C(k, \vec{e}) \geq 0$, $\forall 1 \leq k < K$, *and*

(c) $E_C(K, \vec{e}) \geq E_\ell$.

An assignment is said *optimal* for the reward maximization on energy harvesting problem if its reward is the maximum among all feasible assignments. We say there exists an *energy underflow* for an assignment $\vec{e}$ if there exists $E_C(k, \vec{e}) < 0$ for some $1 \leq k \leq K-1$ or $E_C(K, \vec{e}) < E_\ell$. On the other hand, an assignment $\vec{e}$ is said with *energy overflow* if there exists some k with $E_C(k-1, \vec{e}) + E_H(k) - e_k > E_{\max}$.

4 Proposed Algorithms

This section presents our proposed algorithms for the reward maximization on energy harvesting problem. For the sake of clearness, we will present an algorithm for energy storages with unlimited capacity. The algorithm is then extended to general cases by considering limited energy storage capacity.

4.1 A Greedy Algorithm for Unlimited Energy Storage Capacity

Based on Lemma 1, an optimal assignment for the reward maximization on energy harvesting problem should consume a constant amount of energy to maximize the achieved reward. However, the harvested energy might not be enough to support the energy consumption. For example, as shown in Figure 2(a), if K is 6 with $E_\ell = E_C(0) = 2, E_H(1) = 6, E_H(2) = 4, E_H(3) = 0, E_H(4) = 0, E_H(5) = 5, E_H(6) = 5$, Lemma 1 suggests to have an assignment $\vec{e}$ with $\frac{6+4+5+5}{6} = \frac{10}{3}$ unit of energy consumption for all these six frames. However, according to Definition 1, the resulting assignment is not feasible since there is an energy underflow with $E_C(4, \vec{e}) = -\frac{4}{3}$. Therefore, an optimal assignment for the reward maximization on energy harvesting problem should try to consume some constant amounts of energy without leading to energy underflow.

Let k be the index of the last frame that has been assigned so far, in which k is initialized as 0. For each j with $j = k+1, k+2, \ldots, K$, the maximum amount of energy that is allowed to consume from time k to time j is $E_C(k) + \sum_{i=k+1}^{j} E_H(i)$. If we decide to consume $E_C(k) + \sum_{i=k+1}^{j} E_H(i)$ amount of energy from time k to time j, an assignment, ignoring feasibility constraints, should consume $\frac{E_C(k)+\sum_{i=k+1}^{j} E_H(i)}{j-k}$ amount of energy for each of the frames from the $(k+1)$-th frame to the j-th frame. Let $\widetilde{e}_j$ be $\frac{E_C(k)+\sum_{i=k+1}^{j} E_H(i)}{j-k}$, $\forall j = k+1, k+2, \ldots, K$. Clearly, when $\widetilde{e}_j \geq \widetilde{e}_{k^*}$ for every index $k < j < k^*$, a partial assignment for the $k+1$-th frame to the k^*-th frame with a constant amount of energy consumption $\widetilde{e}_{k^*}$ will lead to a solution with $E_C(j) \geq 0$ for any $k < j \leq k^*$. Therefore, we find the maximum index k^* in which $\widetilde{e}_j \geq \widetilde{e}_{k^*}$ for every index $k < j < k^*$, and then assign energy consumption $\widetilde{e}_{k^*}$ to any j-th frame with $j = k+1, k+2, \ldots, k^*$. Then, we can update the index k as k^* and repeat the above procedure. However, since we have a constraint on the residual energy in the energy storage after completing the K-th frame, $\widetilde{e}_K$ should be revised as $\frac{E_C(k)-E_\ell+\sum_{i=k+1}^{K} E_H(i)}{K-k}$ to guarantee that the residual energy at time K is E_ℓ.

The proposed algorithm is presented in Algorithm 1, denoted by Algorithm Greedy-Incremental (GI) for the rest of this paper. With the initialization of k as 0, we calculate the values $\widetilde{e}_j$ in Step 3 and Step 4 for every $j = k+1, k+2, \ldots, K$. Then, we find the maximum index k^*, in which $\widetilde{e}_j \geq \widetilde{e}_{k^*}$ for every index $k < j < k^*$, in Step 5 with the assignment of energy consumption $\widetilde{e}_{k^*}$ for every of the frames from the $k+1$-th frame to the k^*-th frame. Clearly, $E_C(k^*)$ is 0 (E_ℓ, respectively) when k^* is less than K (k^* is equal to K, respectively). Then, the algorithm goes to the next loop by updating k as k^*. The time complexity of the algorithm is $O(K^2)$ with a proper implementation of the summation in Step 3 and Step 4 in Algorithm 1.

Algorithm 1 Greedy-Incremental (GI)

Input: K, $E_H(k)$ for $k = 1, 2, \ldots, K$, $E_C(0)$, E_ℓ;
Output: a feasible assignment of energy consumption for the K frames;
1: $k \Leftarrow 0$;
2: **while** $k < K$ **do**
3: let $\widetilde{e}_j$ be $\frac{E_C(k)+\sum_{i=k+1}^{j} E_H(i)}{j-k}$, $\forall j = k+1, k+2, \ldots, K-1$;
4: let $\widetilde{e}_K$ be $\frac{E_C(k)-E_\ell+\sum_{i=k+1}^{K} E_H(i)}{K-k}$;
5: $k^* \Leftarrow \max\{\arg\min_{k<\hat{k}\leq K}\{\widetilde{e}_{\hat{k}}\}\}$;
6: $e_j^* \Leftarrow \widetilde{e}_{k^*}$, $\forall k+1 \leq j \leq k^*$;
7: **if** $k^* = K$ **then**
8: $E_C(k^*) \Leftarrow E_\ell$;
9: **else**
10: $E_C(k^*) \Leftarrow 0$;
11: $k \Leftarrow k^*$;
12: return $\vec{e}^*$ as the solution;

Applying Algorithm GI to the example in the first paragraph in this subsection would lead to a solution as shown in Figure 2(b). When k is 0, we have $\widetilde{e}_1 = 8, \widetilde{e}_2 = 6, \widetilde{e}_3 = 4, \widetilde{e}_4 = 3, \widetilde{e}_5 = 3.4, \widetilde{e}_6 = \frac{10}{3}$, and, hence, e_1^*, e_2^*, e_3^*, and e_4^* are set to 3 since k^* is 4. Then, when k is 4, we have $\widetilde{e}_5 = 5, \widetilde{e}_6 = 4$. Therefore, e_5^* and e_6^* are set to 4. Figure 2(c) shows the stored energy $E_C()$ over time.

We have the following lemmas for the derived solution.

Lemma 2 *The derived solution from Algorithm Greedy-Incremental consumes energy non-decreasingly in these K frames.*

Proof. Suppose that $e_i^* > e_{i+1}^* = e_{i+2}^* = \cdots = e_j^* \neq e_{j+1}^*$ for some $1 \leq i \leq K$ for contradiction. (For brevity, we assume e_{K+1}^* is ∞.) Then, when determining the energy consumption e_i^*, we have $\widetilde{e}_i > \widetilde{e}_j$, which contradicts the selection of k^* in Algorithm Greedy-Incremental. □

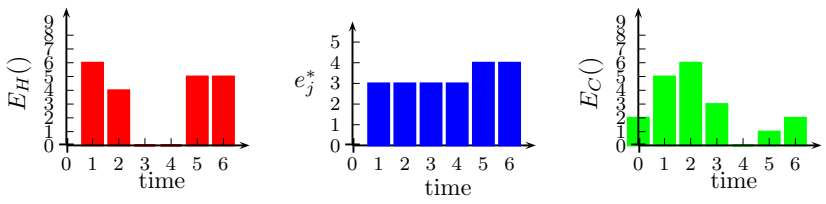
Lemma 3 *The derived solution from Algorithm Greedy-Incremental is feasible.*

Lemma 4 *The derived solution from Algorithm Greedy-Incremental consumes energy $E_C(0) - E_\ell + \sum_{i=1}^{K} E_H(i)$ in these K frames.*

Proof. The above lemmas come directly from the construction. □

4.2 An Algorithm for Limited Energy Storage Capacity

We now deal with systems with energy storage capacity constraints, i.e., $E_{\max} \neq \infty$. The derived solution $\vec{e}^*$ is



(a) $E_H()$ (b) The assignment (c) Energy in the energy storage

Figure 2. The solution derived by Algorithm Greedy-Incremental when K is 6 for specified $E_H()$.

guaranteed to have no energy underflow if there is no energy overflow. But, when $E_{\max} \neq \infty$, there might be some energy overflows in assignment $\vec{e}$, and will lead to some energy underflows since the derived solution tries to consume all the harvested energy. Consider the setting of $E_{\max} = 5$ for the example in the first paragraph of Section 4.1. The derived solution of Algorithm GI has energy overflow at time 2. Then, the derived solution $\vec{e}$ will later have energy underflow, e.g., at time 4 in this example. Therefore, if the resulting schedule has energy overflow, we should consume more energy to avoid energy overflow so that we can get more reward.

Beginning from the solution $\vec{e}^*$ derived from Algorithm GI, we revise the solution to prevent from energy overflow. The assignment $\vec{e}^*$ can be segmented into N *segments*, in which, in assignment $\vec{e}^*$, all the frames in a segment are with the same energy consumption and any two frames in different segments are with different energy consumptions. Suppose that k_n is the index of the frame at the end of the n-th segment for every $n = 1, 2, \ldots, N$, where k_0 is initialized as 0 for notational brevity. The proposed algorithm, denoted by Algorithm Recursive-Decomposition (RD) as shown in Algorithm 2, then revises $\vec{e}^*$ individually in these N segments by calling a recursive procedure subSeg().

The subSeg() procedure takes four parameters k', K', E'_C, and E'_ℓ as its input to derive a feasible assignment between the $(k'+1)$-th frame and the K'-th frame (from time k' to time K') with energy in the energy storage E'_C at time k' and E'_ℓ at time K'. When $K' - k'$ is 1, it is clear to execute the K'-th frame with energy consumption $E_H(K') + E'_C - E'_\ell$. When $K' - k'$ is more than 1, we have to check whether there will be energy underflow or energy overflow. Firstly, let $e^\flat$ be the average energy consumption for executing these $K' - k'$ frames, i.e., $e^\flat \Leftarrow \frac{E'_C - E'_\ell + \sum_{i=k'+1}^{K'} E_H(i)}{K'-k'}$ in Step 3 in Procedure subSeg in Algorithm 2. For every $k' < j < K'$, let $e_j^\sharp$ be the maximum average energy consumption from time j to time K', where $e_j^\sharp \Leftarrow \frac{E_{\max} - E'_\ell + \sum_{i=j+1}^{K'} E_H(i)}{K'-j}$ in Step 4 in Procedure subSeg in Algorithm 2. Similarly, let $\hat{e}_j$ be the maximum average energy consumption from time k' to time j, where $\hat{e}_j \Leftarrow \frac{E'_C + \sum_{i=k'+1}^{j} E_H(i)}{j-k'}$ in Step 5 in Procedure subSeg in Algorithm 2. Let $k^\sharp$ be the index j with the minimum $e_j^\sharp$ and $\hat{k}$ be the index j with the minimum $\hat{e}_j$, where ties are broken arbitrarily. If both $e_{k^\sharp}^\sharp$ and $\hat{e}_{\hat{k}}$ are no less than $e^\flat$, the procedure returns the assignment to consume energy $e^\flat$ from the $k'+1$-th frame to the K'-th frame. Otherwise these $K' - k'$ frames are divided into two sub-segments. If $e_{k^\sharp}^\sharp$ is less than $e^\flat$, the frames from the k'-th frame to the $k^\sharp$ frame are restricted to leave energy $E_{\max}$ in the energy storage at time $k^\sharp$ by calling subSeg$(k', k^\sharp, E'_C, E_{\max})$ and subSeg$(k^\sharp, K', E_{\max}, E'_\ell)$. Otherwise, if $\hat{e}_{\hat{k}}$ is less than $e^\flat$, the frames from the k'-th frame to the $\hat{k}$ frame are restricted to leave no energy in the energy storage at time $\hat{k}$ by calling subSeg$(k', \hat{k}, E'_C, 0)$ and subSeg$(\hat{k}, K', 0, E'_\ell)$.

Applying Algorithm RD on the example in Figure 2(a) with $E_{\max} = 5$, and $E_\ell = E_C(0) = 2$, the first segment with $k_1 = 4$ will be divided into two segments by taking $k^\sharp$ as 2 and $e_2^\sharp = 2.5$. As a result, $\vec{e}^\dagger$ is $(3.5, 3.5, 2.5, 2.5, 4, 4)$.

The time complexity of Algorithm (RD) is $O(K^2)$ since the time complexity for the n-th segment is $O((k_n - k_{n-1})^2)$, while $\sum_{n=1}^{N} O((k_n - k_{n-1})^2) = O(K^2)$.

We have the following lemma for the derived solution.

Lemma 5 *The derived solution from Algorithm Recursive-Decomposition is feasible.*

Proof. Since the assignment is determined in Step 2 and Step 15 in Procedure subSeg in Algorithm 2, it is clear that there is no energy overflow or energy underflow in the derived solution. □

5 Proofs for the Optimality of the Proposed Algorithms

This section provides the optimality of Algorithm GI and Algorithm RD. Due to space limitation, some proofs are only sketched. The following property from the concavity of the reward function must hold for any optimal solution:

Lemma 6 (Jumping of Energy Consumption) *For an optimal energy vector $\vec{e}$, (1) if $e_i < e_{i+1}$, $E_C(i, \vec{e})$ is 0; (2) if $e_i > e_{i+1}$, $E_C(i, \vec{e})$ is $E_{\max}$.*

Proof. Let $E_C(i, \vec{e})$ be γ. We prove this lemma by contradiction. Suppose that $e_i < e_{i+1}$ and $\gamma > 0$. Let $e_i^\perp$ be $\min\left\{e_i + \gamma, \frac{e_i + e_{i+1}}{2}\right\}$, while $e_{i+1}^\perp$ is $e_i + e_{i+1} - e_{i^\perp}$. Let

Algorithm 2 Recursive-Decomposition (RD)

Input: K, $E_H(k)$ for $k = 1, 2, \ldots, K$, $E_C(0)$, E_ℓ;
Output: a feasible assignment of energy consumption for the K frames;

let $\vec{e}^*$ be the solution derived from Algorithm 1;
divide the K frames into N segments and let k_n be the index of the frame at the end of the n-th segment for every $n = 1, 2, \ldots, N$, where k_0 is 0;
for $n = 1; n \le N; n \Leftarrow n + 1$ **do**
  let $e^\dagger_{(k_{n-1}+1)}, e^\dagger_{(k_{n-1}+2)}, \ldots, e^\dagger_{k_n}$ be the resulting assignment by calling SUBSEG(k_{n-1}, k_n, E'_C, E'_ℓ), where E'_C is $E_C(0)$ when $n = 1$, E'_C is 0 for any $n > 1$, E'_ℓ is 0 for any $n < N$, and E'_ℓ is E_ℓ when $n = N$;
return $\vec{e}^\dagger$ as the solution;

Procedure: SUBSEG()
Input: (k', K', E'_C, E'_ℓ);
Output: a feasible assignment of energy consumption for the frames from the $(k'+1)$-th frame to the K'-th frame;

if $K' - k' = 1$ **then**
  return the assignment by consuming $E'_C + E_H(K') - E'_\ell$ for the K'-th frame;
$e^\flat \Leftarrow \frac{E'_C - E'_\ell + \sum_{i=k'+1}^{K'} E_H(i)}{K' - k'}$;
$e^\sharp_j \Leftarrow \frac{E_{\max} - E'_\ell + \sum_{i=j+1}^{K'} E_H(i)}{K' - j}$ for every $k' < j < K'$;
$\hat{e}_j \Leftarrow \frac{E'_C + \sum_{i=k'+1}^{j} E_H(i)}{j - k'}$ for every $k' < j < K'$;
$k^\sharp \Leftarrow \arg_{k'<j<K'} \min \left\{ e^\sharp_j \right\}$;
$\hat{k} \Leftarrow \arg_{k'<j<K'} \min \{\hat{e}_j\}$;
if $e^\sharp_{k^\sharp} < e^\flat$ **then**
  SUBSEG($k', k^\sharp, E'_C, E_{\max}$);
  SUBSEG($k^\sharp, K', E_{\max}, E'_\ell$);
else if $\hat{e}_{\hat{k}} < e^\flat$ **then**
  SUBSEG($k', \hat{k}, E'_C, 0$);
  SUBSEG($\hat{k}, K', 0, E'_\ell$);
else
  return the assignment to consume $e^\flat$ from the $k'+1$-th frame to the K'-th frame;

$e^\perp_j$ be e_j for any $j \neq i, i+1$. It is clear that $E_C(j, \vec{e})$ and $E_C(j, \vec{e}^\perp)$ are the same for any $j \neq i$. By the definition of $e^\perp_i$, $E_C(i, \vec{e}^\perp)$ must be no less than 0, and hence $\vec{e}^\perp$ is feasible. Moreover, we know that $e_i < e^\perp_i, e^\perp_{i+1} < e_{i+1}$ with $e_i + e_{i+1} = e^\perp_i + e^\perp_{i+1}$. By Lemma 1, assignment $\vec{e}^\perp$ has more reward than assignment $\vec{e}$, which contradicts the optimality of $\vec{e}$.

Suppose that $e_i > e_{i+1}$ and $\gamma < E_{\max}$. Let $e^\perp_i$ be $\max\left\{e_i - E_{\max} + \gamma, \frac{e_i + e_{i+1}}{2}\right\}$, while $e^\perp_{i+1}$ is $e_i + e_{i+1} - e^\perp_i$. Similar to the argument of the first case, we will have contradiction to the optimality of assignment $\vec{e}$. □

Moreover, the following lemma shows that there is no energy waste for an optimal assignment.

Lemma 7 (Total Energy) *For an optimal energy vector $\vec{e}$, $\sum_{i=1}^K e_i = E_C(0) + \sum_{i=1}^K E_H(i) - E_\ell$.*

Proof. If $\sum_{i=1}^K e_i < E_C(0) + \sum_{i=1}^K E_H(i) - E_\ell$, assignment $\vec{e}$ is clearly sub-optimal since the reward function is an increasing function. On the other hand, if $\sum_{i=1}^K e_i > E_C(0) + \sum_{i=1}^K E_H(i) - E_\ell$, $\vec{e}$ is not feasible. Both contradict the optimality of $\vec{e}$. □

5.1 Optimality of Algorithm Greedy-Incremental

Based on Lemmas 2, 3, 6, and 7, we can prove the optimality of Algorithm GI when there is no limitation on the energy storage capacity.

Theorem 1 *Algorithm Greedy-Incremental derives optimal assignments for the reward maximization on energy harvesting problem when $E_{\max} = \infty$.*

Proof. Based on Lemma 6 and Lemma 7, an optimal assignment $\vec{e}^\perp$ must have non-decreasing energy consumption with $\sum_{i=1}^K e^\perp_i = E_C(0) + \sum_{i=1}^K E_H(i) - E_\ell$. Suppose that $\vec{e}^\perp$ is different from the assignment $\vec{e}^*$ derived by Algorithm Greedy-Incremental. By adopting the segmentation terminology for $\vec{e}^*$ at the beginning of Section 4.2. Suppose that n' is the first segment that $\vec{e}^*$ and $\vec{e}^\perp$ differs from each other. By definition, $e^*_{(k_{n'-1}+1)} = e^*_{(k_{n'-1}+2)} = \cdots = e^*_{k_{n'}}$. Let κ be the index of the first frame that $\vec{e}^*$ and $\vec{e}^\perp$ differs from each other.

If $e^*_\kappa < e^\perp_\kappa$, we have $\sum_{i=k_{n'-1}+1}^{k_{n'}} e^*_i < \sum_{i=k_{n'-1}+1}^{k_{n'}} e^\perp_i$, since $e^\perp_\kappa \le e^\perp_{\kappa+1} \le \cdots \le e^\perp_{k_{n'}}$ and, by Lemma 2, $e^*_\kappa \le e^*_{\kappa+1} \le \cdots \le e^*_{k_{n'}}$. Therefore, assignment $\vec{e}^\perp$ has some energy underflow, which contradicts the feasibility of $\vec{e}^\perp$.

If $e^*_\kappa > e^\perp_\kappa$, because $e^*_\kappa \le e^*_{\kappa+1} \le \cdots \le e^*_K$, we know that $E_C(k_{n'-1} + 1, \vec{e}^*) + \sum_{i=k_{n'-1}+1}^{k} e^\perp_i < \sum_{i=k_{n'-1}+1}^{k} e^*_i <^1 E_C(k_{n'-1} + 1, \vec{e}^*) + \sum_{i=k_{n'-1}+1}^{k} E_H(i)$ for any $k_{n'-1} + 1 \le k < K$ and $E_C(k_{n'-1} + 1, \vec{e}^*) + \sum_{i=k_{n'-1}+1}^{K} e^\perp_i < \sum_{i=k_{n'-1}+1}^{K} e^*_i <^2 E_C(k_{n'-1} + 1, \vec{e}^*) - E_\ell + \sum_{i=k_{n'-1}+1}^{K} E_H(i)$, where $<^1$ and $<^2$ come from Lemma 3. Therefore, there is energy waste in assignment $\vec{e}^\perp$, which contradicts the optimality of $\vec{e}^\perp$.

As a result, assignment $\vec{e}^*$ derived by Algorithm Greedy-Incremental is optimal for the reward maximization on energy harvesting problem. □

5.2 Optimality of Algorithm Recursive-Decomposition

We now show the optimality of Algorithm Recursive-Decomposition for the reward maximization on energy harvesting problem when there is limitation on the energy storage capacity. It is obvious that the derived assignment $\vec{e}^\dagger$ does not have energy waste, i.e., $\sum_{i=1}^K e^\dagger_i = E_C(0) + \sum_{i=1}^K E_H(i) - E_\ell$. The following lemma is needed to prove the optimality property in Lemma 6.

Lemma 8 *For procedure SUBSEG() with specified parameters k', K', E'_C, and E'_ℓ in Algorithm 2, (1) if $e^\sharp_{k^\sharp} \ge e^\flat$ and $\hat{e}_{\hat{k}} < e^\flat$, then, $e^\dagger_{\hat{k}} \le e^\dagger_{\hat{k}+1}$; (2) if $e^\sharp_{k^\sharp} < e^\flat$, $e^\dagger_{k^\sharp} \ge e^\dagger_{k^\sharp+1}$.*

Proof. For the first case, we will reach the contradiction by $\hat{e}_j < \hat{e}_{\hat{k}}$ for some $k < j < K'$ or $e^\sharp_j < e^\flat$ for some $k <$

$j < K'$. For the second case, if $e^\sharp_{k^\sharp} < e^\flat$ and $e^\dagger_{k^\sharp} < e^\dagger_{k^\sharp+1}$, we will have $e^\sharp_j < e^\sharp_{k^\sharp}$ for some $k < j < K'$. □

Based on Lemma 8, it is not difficult to see that the derived solution satisfies the optimality properties in Lemma 6. The following theorem states the optimality of Algorithm Recursive-Decomposition.

Theorem 2 *Algorithm Recursive-Decomposition derives optimal assignments for the reward maximization on energy harvesting problem.*

Proof. Based on Lemma 6 and Lemma 7, an optimal assignment $\vec{e}^\perp$ can only decrease (respectively, increase) energy consumption at time k when $E_C(k, \vec{e}^\perp)$ is $E_{\max}$ (respectively, 0). Suppose that $\vec{e}^\perp$ is different from the assignment $\vec{e}^\dagger$ derived by Algorithm Recursive-Decomposition. Let κ be the smallest index in which $e^\perp_\kappa$ is different from $e^\dagger_\kappa$. Let m be the index with $e^\dagger_{m-1} \neq e^\dagger_m = e^\dagger_{m+1} = \cdots = e^\dagger_\kappa$, where $e^\dagger_0$ is defined as ∞ for boundary condition. By definition, $E_C(m, \vec{e}^\perp)$ is equal to $E_C(m, \vec{e}^\dagger)$. Let β_1 (respectively, β_2) be the earliest index after κ with $E_C(\beta_1, \vec{e}^\dagger) = E_{\max}$ (respectively, $E_C(\beta_2, \vec{e}^\dagger) = 0$). If there does not exist β_1 (respectively, β_2), let β_1 (respectively, β_2) be K.

If $e^\dagger_\kappa < e^\perp_\kappa$ and $\beta_1 < \beta_2$, we know that $e^\dagger_i \geq e^\dagger_j$ for any $m \leq i < j \leq \beta_2$ due to Lemma 6. Moreover, for any $\kappa \leq i \leq \beta_2$, we have $e^\perp_i > e^\dagger_\kappa$ since it is impossible to have $E_C(i, \vec{e}^\dagger) = E_{\max}$. Clearly, we reach the conclusion that there is an energy underflow of $\vec{e}^\perp$. The proof is similar for the case that $e^\dagger_\kappa < e^\perp_\kappa$ and $\beta_1 \geq \beta_2$.

If $e^\dagger_\kappa > e^\perp_\kappa$ and $\beta_1 \geq \beta_2$, similarly, there will be some energy overflow of $\vec{e}^\perp$ or $\sum_{i=1}^K e^\perp_i < E_C(0) - E_\ell + \sum_{i=1}^K E_H(i)$, which contradicts the optimality of $\vec{e}^\perp$. If $e^\dagger_\kappa > e^\perp_\kappa$ and $\beta_1 < \beta_2$, there will be an energy overflow at time β_1 in $\vec{e}^\perp$.

As a result, assignment $\vec{e}^\dagger$ derived by Algorithm Recursive-Decomposition is optimal for the reward maximization on energy harvesting problem. □

6 Remarks

This section gives the related remarks for designing embedded systems with energy harvesting devices. We will first show how to derive the minimum energy storage capacity for optimality so that the designers can choose a proper energy storage. Then, we will present the energy buffering technique that is used to buffer the energy harvested in a frame. We will also provide remarks for systems with the maximum and the minimum energy consumption constraints in a frame.

To design the energy supply of an embedded system, it is important to estimate how to dimension the energy storage device. Given an initial energy $E_C(0)$, an energy source $E_H(k), 1 \leq k \leq K$ and a final energy constraint E_ℓ, we are interested in the minimum storage capacity $E_{\max}$ which is needed to achieve the maximum possible reward. We denote $E_{\max,\min}$ the minimum capacity $E_{\max}$ for which the optimal reward equals the reward for an unconstrained system which $E_{\max} = +\infty$. We can now determine

$$E_{\max,\min} = \max_{k=1,2,\ldots,K} \{E_C(0) + \sum_{j=1}^{k} (E_H(j) - e^*_j)\},$$

where $\vec{e}^*$ is the assignment calculated by Algorithm Greedy-Incremental. Note that this result, like all contributions of this paper, holds for an arbitrary concave reward function $r()$.

As already mentioned, a frame may last several hours in physical time. If now the energy storage is full, i.e., $E_C(k-1) = E_{\max}$ and $e_k = E_H(k)$ for some k, we may still have an energy overflow if $E_H(k)$ arrives *before* it is consumed by e_k. This conflict can be resolved by assuming the existence of small energy buffers (capacitors or supercapacitors) or a lower capacity $E'_{\max}$. In a similar way, problems can be resolved for the underflow problem.

In some systems, the service might have the requirement to consume at least some amount of energy consumption in a frame, and there might also be an constraint on the maximum energy consumption in a frame since there is no improvement in quality/reward for more energy consumption. We can revise the solution $\vec{e}^\dagger$ derived from Algorithm Recursive-Decomposition by setting the energy consumption of the k-th frame as the maximum energy consumption if $e^\dagger_k$ is greater than the maximum energy consumption constraint. The proofs in Section 5 can be extended to prove the optimality of the revised solution. If there exists $e^\dagger_k$ which is less than the minimum energy consumption constraint, it is not difficult to see that the input does not admit an feasible assignment.

7 Simulation Results

7.1 Simulation Setup

In our experiments, we used real measurements of solar light intensity $\left[\frac{W}{m^2}\right]$ recorded at [1] as input data $E_H()$. Certainly, to simulate a concrete system one would have to scale the measured power profile with the size, number and efficiency of the actually used solar panels. The data is sampled every 5 minutes, so we have a maximum of 288 samples per day. Since solar energy shows a similar pattern every 24 hours, multiples of a day are reasonable choices for the number of frames of the prediction horizon.

To establish the relationship between the parameter K and physical time, we introduce now two additional variables which are only used for simulation purposes. We denote f the *number of frames per day*. In all experiments, we used $f = 16$, i.e., the length of each frame is 1.5 hours. In addition, we denote d the *number of days of the prediction horizon* . Clearly, the parameter K can be computed as $K = d \cdot f$. At the end of this horizon, we

want the remaining energy $E_C(K)$ to be at least equal to the initial energy $E_C(0)$, i.e., $E_\ell = E_C(0)$.

The reward function $r(e)$ should reflect the quality of a service with energy consumption e, and is often deduced from the peculiarities of human perception. We performed simulations for a variety of reward functions, but, due to space limitation, only the results of one reward function are presented in this paper. We opted for the reward function

$$r(e) = \ln\left(0.01 + \frac{e}{1000}\right)$$

which assigns negative rewards (i.e., penalties) to energies $e < 990$. In particular, setting the service e to 0 is punished with an penalty of $\ln(0.01) \approx -4.61$. For an experiment, we repeated for N times to get the results.

7.2 Evaluations Compared to an Adversary Algorithm

Since there are no other algorithms available for the reward maximization on energy harvesting problem, we designed an adversary algorithm which can be found in the Appendix (Algorithm 3). What makes finding adversary algorithms tricky is that one has to find algorithms which are *feasible* and *competitive* at the same time. To this end, the constructed adversary algorithm constitutes the smartest solution an engineer would probably implement on a sensor node not beeing aware of the techniques described in this paper. The time complexity of the adversary algorithm is the same as that of Algorithm 2.

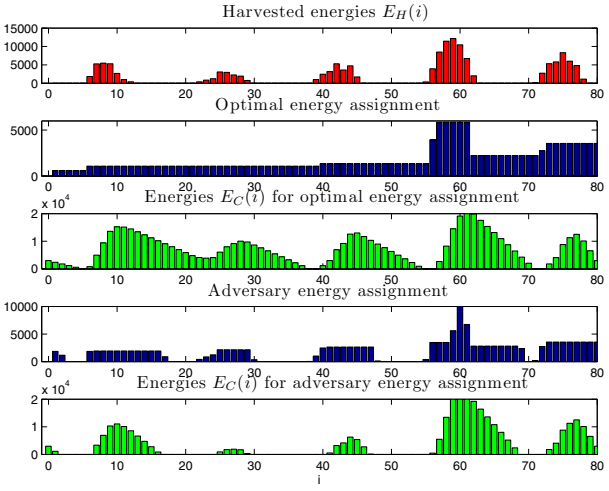


Figure 3. Comparison of assignments $\vec{e}^\dagger$ and $\vec{e}^u$ for $E_C(0) = E_\ell = 3000, E_{\max} = 20000, d = 5, f = 16$.

Figure 3 displays a comparison of the assignments generated by the adversary algorithm and Algorithm Recursive-Decomposition for $d = 5$ days. Both assignments start with an initial energy $E_C(0) = 3000$, the energy storage capacity is $E_{\max} = 20000$. Obviously, the optimal assignment $\vec{e}^\dagger$ manages to balance the energy consumption much better than the assignment $\vec{e}^u$ derived from the adversary algorithm. The latter has to suspend the service completely during the first four nights, which is clearly an unacceptable behaviour. Around frame 60, a burst of energy E_H is forcing $\vec{e}^u$ to increase the service to 10000, whereas $\vec{e}^\dagger$ shows only a moderate increase to 6000. As a result, the total reward for assignment $\vec{e}^\dagger$ amounts to 34.6; assignment $\vec{e}^u$ achieves a negative total reward of -57.1.

For Figure 4, we repeated the experiment $N = 20$ times for 100 consecutive days. The average reward during this time was 68.4 for the optimal and 12.2 for the adversary algorithm. So also in terms of average reward, the optimal assignment outperforms $\vec{e}^u$ significantly.

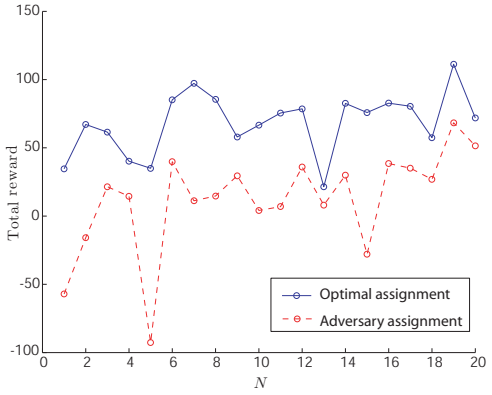


Figure 4. Rewards of assignments $\vec{e}^\dagger$ and $\vec{e}^u$ for $N = 50$ repetitions, $E_C(0) = E_\ell = 3000, E_{\max} = 20000, d = 5, f = 16$.

7.3 Choosing a Sufficient Parameters d and $E_{\max}$

A fundamental question one might ask when designing a system in a given environment is: How many days d should the horizon span to obtain reasonable rewards? For this purpose we simulated the Algorithm Recursive-Decomposition for different parameters $d \in \{1, 2, 3, 5, 10, 15, 30, 70, 105, 210\}$. To obtain a total simulated time of 210 days for each experiment, the experiments were repeated $N = \{210, 105, 70, 42, 21, 14, 7, 3, 2, 1\}$ times, respectively. For each experiment, we calculate the sum of rewards for 210 days, while the value is referred to as the *accumulated reward*.

As a matter of fact, the accumulated reward depends both on the number d of days of the prediction horizon *and* the energy storage capacity $E_{\max}$. In Figure 5 we see that the accumulated reward increases quickly with the parameters d and $E_{\max}$. The minimum energy capacity $E_{\max}$ required to optimally exploit this energy source

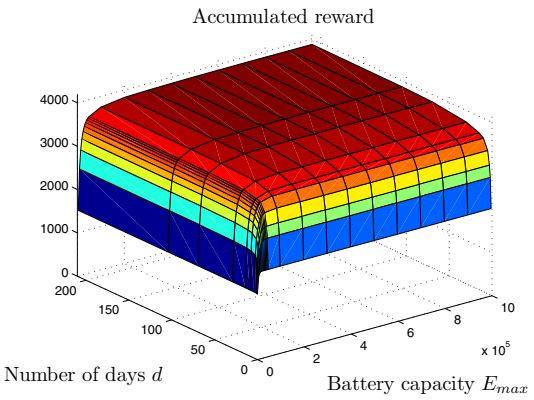


Figure 5. Accumulated reward for 210 days for $r(\vec{e}^{\dagger})$, $E_C(0) = E_\ell = 3000, f = 16$.

is $E_{\max,\min} = 759450$. Using this value for the battery, a horizon of $d = 15$ days is sufficient to achieve 93.4% of the maximum possible reward (i.e., the reward for $d, E_{\max} = \infty$). For this particular reward function, however, also smaller capacities $E_{\max}$ are possible to achieve a similar reward.

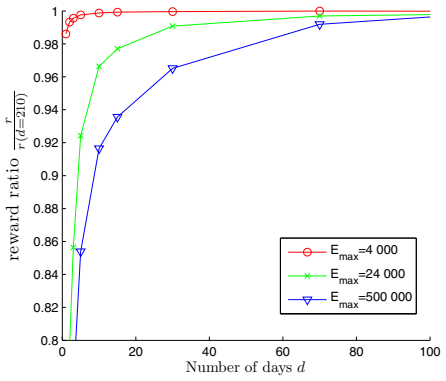


Figure 6. Convergence of the average reward $r(\vec{e}^{\dagger})$ over the number of days d, $E_C(0) = E_\ell = 3000, f = 16$.

In Figure 6, the accumulated rewards were normalized by the reward obtained by the experiment with the longest horizon, namely 210 days. If one chooses a smaller $E_{\max}$, it turns out that the reward converges faster towards its longterm average with increasing d. For a horizon of $d = 15$ days, a capacity of $E_{\max} = 500000$ will result in a reward of 93,7% of the reward for the same capacity with $d = 210$ days. For smaller capacities $E_{\max} = 24000$ and 4000, the ratio increases to 97,7 % and 99.9 %, respectively. The reason for this behaviour is that Algorithm 2 is getting more and more nearsighted with smaller capacity $E_{\max}$: Due to the capacity constraint, local maxima are computed to avoid energy overflows. Hence, for small energy storage capacities $E_{\max}$, the total reward can hardly be improved by increasing the prediction horizon d.

8 Conclusions

We have been studying energy harvesting systems which receive their energy from an environmental source, e.g., solar energy. Instead of performing classical power management techniques which try to save energy subject to performance constraints, such a device primarily has to tune its performance according to the underlying energy source. In this paper, we identify and solve the reward maximization on energy harvesting problem. As rewards, we opted for functions which are monotonically increasing and concave with the energy consumption. These rewards may be reasonable metrics for many applications where the subjective quality saturates with increasing effort. We provide polynomial-time algorithms to calculate assignments which optimally level out the available energy. For measurements of solar energy – which is probably the most prominent and powerful energy source – we perform simulations which demonstrate significant improvements compared to naive approaches. Furthermore, design parameters like battery capacity or duration of the prediction horizon can be derived with the help of our methods.

References

[1] Bern university of applied sciences, engineering and information technologies, photovoltaic lab: Recordings of solar light intensity at Mont Soleil from 01/01/2002 to 31/09/2006. www.pvtest.ch, March, 2007.

[2] T. A. Alenawy and H. Aydin. On energy-constrained real-time scheduling. In *EuroMicro Conference on Real-Time Systems (ECRTS'04)*, pages 165–174, 2004.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez. Optimal reward-based scheduling for periodic real-time tasks. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99)*, pages 79–89, 1999.

[4] D. Brunelli, L. Benini, C. Moser, and L. Thiele. An Efficient Solar Energy Harvester for Wireless Sensor Nodes. In *Design, Automation and Test in Europe (DATE 08)*, Munich, Germany, March 10-14 2008.

[5] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In *RTCSA*, pages 28–38, 2007.

[6] J.-J. Chen and T.-W. Kuo. Voltage-scaling scheduling for periodic real-time tasks in reward maximization. In *the 26th IEEE Real-Time Systems Symposium (RTSS)*, pages 345–355, 2005.

[7] J.-J. Chen and T.-W. Kuo. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *ICCAD*, pages 289–294, 2007.

[8] J.-J. Chen, T.-W. Kuo, and C.-L. Yang. Profit-driven uniprocessor scheduling with energy and timing constraints. In *ACM Symposium on Applied Computing*, pages 834–840, 2004.

[9] J. K. Dey, J. F. Kurose, and D. F. Towsley. On-line scheduling policies for a class of IRIS (increasing reward with increasing service) real-time tasks. *IEEE Transactions on Computers*, 45(7):802–813, 1996.

[10] J. Hsu, A. Kansal, J. Friedman, V. Raghunathan, and M. Srivastava. Energy harvesting support for sensor networks. In *SPOTS track at IPSN 2005*, 2005.

[11] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, pages 275–280, 2004.

[12] X. Jiang, J. Polastre, and D. E. Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, pages 463–468, UCLA, Los Angeles, California, USA, April 25-27 2005.

[13] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *Trans. on Embedded Computing Sys.*, 6(4):32, 2007.

[14] J. W.-S. Liu, K.-J. Lin, W.-K. Shih, A. C.-S. Yu, C. Chung, J. Yao, and W. Zhao. Algorithms for scheduling imprecise computations. *IEEE Computer*, 24(5):58–68, May 1991.

[15] C. Moser, D. Brunelli, L. Thiele, and L. Benini. Real-time scheduling for energy harvesting sensor nodes. In *Real-Time Systems*, volume 37, pages 233–260, Norwell, MA, USA, 2007. Kluwer Academic Publishers.

[16] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management in energy harvesting systems. In *DATE '07: Proceedings of the Conference on Design, Automation and Test in Europe*, pages 773–778, NY, USA, 2007. ACM Press.

[17] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Robust and Low Complexity Rate Control for Solar Powered Sensors. In *Design, Automation and Test in Europe (DATE 08)*, Munich, Germany, March 10-14 2008.

[18] C. Park and P. Chou. Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *Proceedings of the Sensor and Ad Hoc Communications and Networks. SECON '06.*, volume 1, 2006.

[19] S. Roundy, D. Steingart, L. Frechette, P. K. Wright, and J. M. Rabaey. Power sources for wireless sensor networks. In *Wireless Sensor Networks, First European Workshop, EWSN 2004, Proceedings*, Lecture Notes in Computer Science, pages 1–17, Berlin, Germany, January 19-21 2004. Springer.

[20] C. Rusu, R. Melhem, and D. Mosse. Maximizing the system value while satisfying time and energy constraints. In *IEEE 23th Real-Time System Symposium*, pages 246–255, Dec. 2002.

[21] C. Rusu, R. Melhem, and D. Mossé. Multiversion scheduling in rechargeable energy-aware real-time systems. In *EuroMicro Conference on Real-Time Systems (ECRTS'03)*, pages 95–104, 2003.

[22] W.-K. Shih, J. W.-S. Liu, and J.-Y. Chung. Algorithms for scheduling imprecise computations with timing constraints. *SIAM J. Computing*, 20(3):537–552, June 1991.

[23] F. Simjee and P. H. Chou. Everlast: long-life, supercapacitor-operated wireless sensor node. In *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 2006. ACM Press.

[24] C.-L. Yang and C.-H. Lee. Hotspot cache: joint temporal and spatial locality exploitation for i-cache energy reduction. In *ISLPED*, pages 114–119, 2004.

[25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382, 1995.

[26] H.-S. Yun and J. Kim. Reward-based voltage scheduling for fixed-priority hard real-time systems. In *International Workshop on Power-Aware Real-Time Computing*, 2004.

[27] H.-S. Yun and J. Kim. Reward-based voltage scheduling for hard real-time systems with energy constraints. In *International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA)*, pages 416–435, 2004.

Appendix

The following algorithm averages the energy consumption for the remaining frames. Only if energy overflows or underflows happen, recalculations of the energies are performed. As shown in Algorithm 3 in Steps 4 and 5, possible energy underflows for the next frame $k+1$ are avoided by reducing the energy consumption to $E_C(k)+E_H(k+1)$. Analogously, the energy consumption is increased in Step 11 to prevent the stored energy $E_C(k+1)$ from overflowing. If such an overflow is avoided, however, we consume more energy than initially planned and we might end up with an infeasible schedule. Hence, a recalculation for the remaining frames becomes necessary to obtain a feasible schedule. This is done in Steps 12-14 by again averaging the remaining energy. Finally, the reward of the schedule can be improved by recalculating the energies also for energy underflows (Steps 6-8).

Algorithm 3 Adversary

1: $k \Leftarrow 0$;
2: $e_j^a = \frac{E_C(0)+\sum_{i=1}^{K} E_H(i)-E_\ell}{K}$, $\forall j = 1, \ldots, K$;
3: **while** $k < K$ **do**
4: **if** $E_C(k) + E_H(k+1) - e_{k+1}^a < 0$ **then**
5: $e_{k+1}^a \Leftarrow E_C(k) + E_H(k+1)$;
6: **for** $i = k+2; i \leq K; i \Leftarrow i+1$ **do**
7: $e_j^a \Leftarrow \frac{\sum_{i=k+2}^{K} E_H(i)-E_\ell}{K-k-1}; \forall j = k+2, .., K$;
8: **if** $E_C(k) + E_H(k+1) - e_{k+1}^a > E_{\max}$ **then**
9: $e_{k+1}^a \Leftarrow E_C(k) + E_H(k+1) - E_{\max}$;
10: **for** $i = k+2; i \leq K; i \Leftarrow i+1$ **do**
11: $e_j^a \Leftarrow \frac{E_{\max}+\sum_{i=k+2}^{K} E_H(i)-E_\ell}{K-k-1}; \forall j = k+2, .., K$;
12: $E_C(k+1) = E_C(k) + E_H(k+1) - e_{k+1}^a$;
13: $k \Leftarrow k+1$;
14: return $\vec{e}^a$ as the solution;

Acknowledgements

The work presented in this paper was partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. In addition, this research has been supported by grants from ROC National Science Council NSC-096-2917-I-564-121 and the European Network of Excellence ARTIST2.