Power Management in Energy Harvesting Embedded Systems with Discrete Service Levels

Clemens Moser
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH) Zurich
cmoser@tik.ee.ethz.ch

Jian-Jia Chen
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH) Zurich
jchen@tik.ee.ethz.ch

Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH) Zurich
thiele@tik.ee.ethz.ch

ABSTRACT

Power management has been a critical issue in the design of embedded systems due to the limited power supply. To prolong the lifetime, energy minimization has been studied under performance constraints in the past decade. The emerging embedded systems with the capability to harvest energy from the environment have recently triggered the revision of power management to improve the quality of service dynamically. As the available power/energy of an electronic device changes over time and is limited by many environmental factors, the system has to decide *when* to change to *which* service level to provide better quality of service without wasting the harvested energy. In this paper, we explore how to maximize the quality of service, in terms of system rewards, of a periodic application with discrete levels in an energy harvesting system. To decide service levels in a time horizon, this paper presents algorithms to derive optimal solutions if the future harvested energy is known. In addition, we present efficient algorithms to derive near-optimal solutions approximately. Our work is supported by simulation results which are based on long-term measurements of the power generated by real solar cells.

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process Management–Scheduling

General Terms

Algorithms, Performance

Keywords

Energy harvesting systems, Embedded systems, Power management, Reward maximization, Solar cells.

1. INTRODUCTION

The performance of battery-driven embedded systems has been constrained by their limited power supply. Power management has played an important role in modern embedded system design to prolong the battery lifetime without sacrificing too much performance. To impose the performance constraint, applications are usually associated with a soft or hard deadline to be met. The emerging technology of energy harvesting has earned much interest recently to provide a means for sustainable embedded systems [6]. Since the deployment cost might be very expensive, the possibility to harvest energy from the environment could provide sustainable services for data gathering applications. Among renewable energy resources, energy harvesting with solar panels is one of the most popular applied technologies, and there have been many energy harvesting circuits that are designed to efficiently convert and store solar energy, as shown in [4, 11, 16].

As the harvested energy is highly dependent on the environment, the energy consumption of the system should be adjusted to maximize the performance instead of minimizing the energy consumption. Most environmental energy sources are not constant over time [14]. For example, the harvested energy of a solar cell at a sunny noon is much higher than that at dawn. As a result, the solar energy generated by photovoltaic elements arrives in bursts, and has to be stored so that the device can still be operated at night. Distinct from the assumption on the amount of energy constraint in a certain time interval in [2,15], the main challenge for such a system is to optimize its performance while respecting the time-varying amount of energy. Exploiting solar energy for performance optimization has been studied recently. Kansal et al. [5] explore how to maximize the utilization of solar energy by minimizing the round-trip losses of the battery. Moser et al. [8] develop lazy-scheduling to avoid deadline violation in energy-harvesting systems. Liu et al. [7] extend the results in [8] to reduce the rate of deadline misses by taking into account processors with the capability of dynamic voltage scaling. Under the assumption that the reward is a strictly concave, increasing, and continuous function of energy consumption, Moser et al. [9] develop algorithms to derive schedules which can optimally maximize the reward received by the system for time-varying renewable energy.

To the best of our knowledge, all works for reward maximization in energy-harvesting systems so far consider the adaptation of *continuous* parameters to maximize the utility of energy harvesting systems. These parameters represent, e.g., the instantiation rates of tasks, the amount of computation or simply the duty cycle of a sensor node. In this paper, we take the first step to obtain a more realistic application model using a *finite set of discrete levels* of service. Specifically, we explore how to allocate service levels when the number of the available service levels is fixed and the reward function is an arbitrary function. The objective is to maximize the achieved system reward without wasting or exhausting the dynamically available energy of the battery. Namely, this paper focuses on

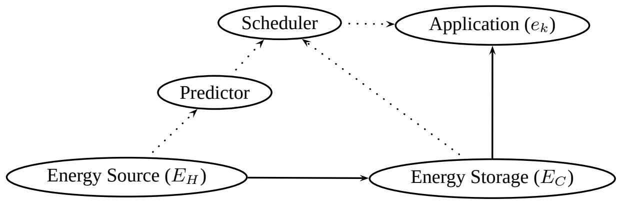

Figure 1: Energy and control flow in the system model.

the same problem studied in [9], but considers practical issues, in which most systems should have only some different service levels. Our contribution in this paper is as follows: We develop algorithms to derive optimal solutions which maximize the overall reward. Moreover, we propose efficient algorithms to derive near-optimal solutions approximately. Simulations illuminate the merits of our algorithms by applying real data recorded for photovoltaic cells as the harvested energy.

2. SYSTEM MODEL

This section presents the system model, including energy harvesting model for the energy source, the energy storage model, the service application models, and the problem definition. The system overview is depicted in Figure 1, where the energy harvested from the energy source is stored in the energy storage, and the scheduler makes decisions for consuming the energy based on the information by the predictor and the available energy in the energy storage.

Energy Harvesting Model. This paper explores embedded systems equipped with an energy harvesting device. The amount of harvested energy is highly dependent on the surrounding environment. For instance, for sensor nodes deployed in a certain environment, a predictor is used to predict the amount of harvested energy in the future [13]. This paper adopts existing predictors to predict the harvested energy in the future for making scheduling decisions. Specifically, we assume that the energy harvesting device has a prediction unit to estimate the amounts of energy harvested in each of the next K prediction intervals in the future. A prediction interval is also referred to as a *frame* for brevity. The *prediction horizon* is the interval length of the predicted interval of the predictor. Each frame is assumed to have the same length and is the basic time unit for scheduling. We denote $\widetilde{E}_H(k)$ the accumulated energy harvested that is predicted for the k-th frame.

For the rest of this paper, we will present our results based on a perfect predictor. For solar cells, one can apply diverse estimation algorithms or even external information from the weather forecast to achieve marginal miss prediction. The results here can also be applied for predictors with prediction mistakes. For detailed discussions about how to choose a suitable energy prediction algorithm or how to cope with prediction mistakes, please refer to [10].

Energy Storage Model. The energy storage, e.g., a supercapacitor or a battery, stores the energy harvested from the environment. As energy storage is not a perfect process, there might be some loss of energy. The *efficiency factor* defined as the actually stored energy divided by the harvested energy can be used to model the storage process. Usually, the efficiency factor, denoted by $\eta(\widetilde{E}_H)$, is a function of the harvested energy, and, by definition, is between 0 and 1. As a result, only $\eta(\widetilde{E}_H) \times \widetilde{E}_H(k)$ amount of the harvested energy will be stored to the energy storage in the k-th frame. The amount of the harvested energy that will be stored to the energy storage in the k-th frame is denoted by $E_H(k)$, where $E_H(k)$ is $\eta(\widetilde{E}_H) \times \widetilde{E}_H(k)$.

The energy storage is constrained by the maximum capacity $E_{\max}$. If the energy storage is full, the additional harvested energy dissipates. Suppose that $E_C(k)$ is the energy in the energy storage at the end of the k-th frame. After the k-th frame with energy consumption e_k, the energy in the energy storage is $\min\{E_{\max}, E_C(k-1)+E_H(k)-e_k\}$. In other words, if $E_C(k-1)+E_H(k)-e_k$ is larger than $E_{\max}$, the system dissipates $E_C(k-1)+E_H(k)-e_k-E_{\max}$ amount of energy, which is a waste.

Service Model. We investigate how to achieve performance maximization for periodically executed services in environmentally powered systems. To this purpose, the scheduler receives predictions of the future energy $E_H(k)$ generated in the next $1 \leq k \leq K$ frames. For every frame k, the scheduler has to assign exactly *one* service level s_k from a set $\{1, 2, 3, \dots, M\}$ of possible service levels. At this, a service level s_k is associated with a corresponding reward r_{s_k} and an energy consumption e_{s_k} which is drawn from the energy storage. The problem is to find an assignment $\vec{s} = (s_1, s_2, \dots, s_K)$ of service levels for these K frames such that the sum of rewards $\sum_{k=1}^{K} r_{s_k}$ is maximized while respecting the required energy constraints.

For the rest of this paper, we will only consider the energy consumption e_{s_k} of a service in each frame k. Note, however, that our service model covers many applications which are of practical relevance. On the one hand, in a computation-based service, a more precise computation usually requires more computation time and energy and will consequently receive a greater amount of reward. Moreover, our model also covers rate-based service models which are typical for, e.g., wireless sensor nodes which wake up and sample data with a certain rate (also called duty-cycle). In any case, the required computation time or rate of a service can be derived by simple calculation from the specified energy consumption e_{s_k}.

Without loss of generality, we order the service levels $\{1, \dots, M\}$ from the lower reward value to the higher reward value, i.e., $r_1 < r_2 < \cdots < r_M$, where M is the number of available service levels. Note that if $e_i \geq e_j$ for $i < j$, it is clear that the i-th service level is energy-inefficient, since it consumes more (or the same) energy but gains less reward. Therefore, without loss of generality, we assume that $e_1 < e_2 < \cdots < e_M$ after eliminating energy-inefficient service levels.

Problem Definition. We are given the prediction for K frames at time 0. 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 $E_H(k)$. For brevity, the k-th frame starts from time $k-1$ to time k. Let $E_C(k, \vec{s})$ be the energy in the energy storage at time k by applying the assignment of service levels $\vec{s}$. After the last frame, we require to have at least energy E_ℓ in the energy storage which can be used in the future. We now define the feasibility of an assignment for the discrete reward maximization on energy harvesting problem as follows:

DEFINITION 1. *[Feasible Assignment] A service vector $\vec{s} = (s_1, s_2, \dots, s_K)$ for all these K frames is feasible if*

(a) $s_k \in \{1, 2, 3, \dots, M\}, \forall 1 \leq k \leq K$,

(b) $E_C(k, \vec{s}) = \min\{E_{\max}, E_C(k-1, \vec{s}) + E_H(k) - e_{s_k}\}$, *where $E_C(0, \vec{s})$ is $E_C(0)$,*

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

(d) $E_C(K, \vec{s}) \geq E_\ell$.

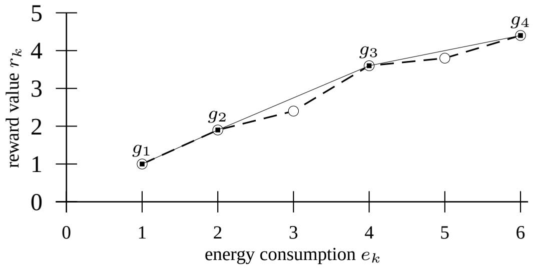

Figure 2: An example for eliminating energy-inefficient service levels in Algorithm 1 Greedy.

An assignment is said *optimal* for the discrete reward maximization on energy harvesting problem if its reward is the maximum among all feasible assignments. For an input instance of the discrete reward maximization on energy harvesting problem, we can easily verify whether there exists a feasible assignment or not by simply assigning all the frames at the lowest service level. For the rest of this paper, we will only focus our discussions on the cases with the existence of feasible assignments.

3. SERVICE ALLOCATION ALGORITHMS

The "brute force" way of maximizing the overall reward is infeasible when the number M of service levels and the number K of prediction intervals are large. If we are given a feasible service vector $\vec{s} = (s_1, s_2, \ldots, s_K)$, it is easy to verify the feasibility and compute the overall reward in $O(K)$ time. Unfortunately, there are M^K possible ways to select a service vector $\vec{s}$ for the K frames. Hence, determining an optimal service by enumerating all possible vectors, checking their feasibility and computing the rewards takes time $O(K \cdot M^K)$ and is infeasible when K and M are large.

This section presents a greedy algorithm extended from the algorithm in [9]. The greedy algorithm first derives an energy consumption profile for the prediction horizon by assuming any energy consumption in $[e_1, e_M]$ is available, and then allocates services in the prediction horizon without violating the battery capacity constraint. Then, we will present algorithms to derive optimal solutions with higher time complexity and approximation algorithms with adjustable time complexity.

3.1 Greedy Algorithm

In [9], Moser et al. develop an algorithm, called Algorithm Recursive-Decomposition (RD), to derive an optimal service assignment when the service levels are in a continuous spectrum and the reward function is concave. In particular, in each frame k, the energy consumption e_k is continuously adaptable in the interval $[e_1, e_M]$, yielding a reward r_k which is continuous in $[r_1, r_M]$. Suppose that the assignment by applying Algorithm RD is $\vec{s}$. The greedy algorithm for discrete service levels simply applies Algorithm RD as a subroutine to determine the assignment for discrete service levels by rounding down the energy consumption. A naïve extension of Algorithm RD is to choose service level $s_k^\dagger$ for the k-th frame such that $e_{s_k^\dagger} \leq e_{s_j} < e_{s_k^\dagger+1}$. This guarantees the feasibility of the derived assignment $\vec{s}^\dagger$, but loses optimality since it is possible to increase the service level for some frames to increase the system reward under the energy constraint.

Therefore, to optimize the system reward, we need to be more careful. Some available discrete service levels might be energy-inefficient. That is, service level j might consume much more energy but have marginal improvement on the reward. Choosing such

Algorithm 1 Greedy

```
Input: $K$ frames, $M$ service levels, $E_C(0)$, $E_H()$, $E_\ell$, $E_{\max}$;
Output: a feasible assignment of service levels $\vec{s}^*$ for the $K$ frames;
1: $E_C(k) \leftarrow 0$ for $k = 1, \ldots, K$;
2: let $\{1, \ldots, M'\}$ denote the service levels after the elimination described in (1) and (2);
3: for $k \leftarrow 1$; $k \leq K$; $k \leftarrow k+1$ do
4:     let $\vec{s}$ be the solution derived from algorithm Recursive-Decomposition (RD) in [9] with ($E_C(k-1)$, $E_\ell$, $E_{\max}$, $E_H(\kappa)$ for $\kappa = k, \ldots, K$) as input;
5:     $s_k^* \leftarrow \arg_{1 \leq i \leq M'} (e_i \leq e_{s_1} < e_{i+1})$;
6:     if $(E_C(k) + E_H(k) - e_{s_k^*}) > E_{\max}$ then
7:         $E_C(k) \leftarrow E_{\max}$;
8:     else
9:         $E_C(k) \leftarrow E_C(k) + E_H(k) - e_{s_k^*}$;
```

a service level is not worthwhile. The greedy algorithm will greedily discard those energy-inefficient service levels. The first step of the greedy algorithm is to eliminate those service levels j with

$$\frac{r_j - r_{j-1}}{e_j - e_{j-1}} < \frac{r_{j+1} - r_j}{e_{j+1} - e_j}. \tag{1}$$

By repeating above procedure, all energy-inefficient service levels can be removed. Suppose that the available i-th service level after elimination is g_i with reward r_{g_i} and energy consumption e_{g_i}, where M' is the number of available service levels after elimination. Therefore, for $2 \leq j \leq M'-1$, we know that

$$\frac{r_{g_j} - r_{g_{j-1}}}{e_{g_j} - e_{g_{j-1}}} \geq \frac{r_{g_{j+1}} - r_{g_j}}{e_{g_{j+1}} - e_{g_j}}. \tag{2}$$

Figure 2 illustrates the above procedure for eliminating energy-inefficient service levels. By applying algorithms for the convex hull problem, the elimination can be done in $O(M \log M)$ [3].

After eliminating the energy-inefficient service levels, our proposed greedy algorithm iteratively decides the service level of a frame based on the solution derived from Algorithm RD. That is, for the k-th iteration, Algorithm RD is applied to derive the service level of the k-th frame by assuming that the service levels of the first $k-1$ frames have been determined, and the greedy algorithm greedily chooses the service level g_j with $e_{g_j} \leq e_{s_k} < e_{g_{j+1}}$ for the k-th frame, where s_k is the service level derived from Algorithm RD for the k-th frame. The pseudo-code of the greedy algorithm is presented in Algorithm 1. The time complexity is $O(K^3 + (K+M)\log M)$, including the overhead for eliminating energy-inefficient service levels.

3.2 Sufficient Storage Capacity

This subsection presents a polynomial-time algorithm to derive optimal assignments under the assumption that the storage capacity is sufficient to store the harvested energy in the prediction horizon, e.g., $E_C(0) + \sum_{k=1}^{k'} (E_H(k) - e_1) \leq E_{\max}$ for $k' = 1, \ldots, K$. To reduce the search space, the following theorem is needed.

THEOREM 1. *If the storage capacity is sufficient and there is a feasible assignment, there exists an optimal assignment $\vec{s}^\dagger$ with non-decreasing energy consumption in the time horizon.*

PROOF. Let $\vec{s}$ be an optimal assignment, in which there is an index $1 \leq i \leq K-1$ such that $s_i > s_{i+1}$. By the definition of feasible assignments and the assumption of sufficient storage capacity, clearly, swapping the service level of the i-th frame and

Algorithm 2 Search

```
Input: $\{x_1, x_2, \ldots, x_{m-1}\}$, $m$, $M$, $X$, $\vec{s}^*$;
Output: a feasible assignment $\vec{s}^*$;
1: if $m = M$ then
2:     $s_k \leftarrow \arg_j \left(\sum_{i=0}^{j-1} x_i < k \leq \sum_{i=0}^{j} x_i\right), \forall k = 1, \ldots, K$;
3:     if $\vec{s}$ is feasible and $\sum_{k=1}^{K} r_{s_k} \geq \sum_{k=1}^{K} r_{s_k^*}$ then
4:         $\vec{s}^* \leftarrow \vec{s}$;
5:     return $\vec{s}^*$;
6: for $i \leftarrow 0$; $i \leftarrow K - X$; $i \leftarrow i + 1$ do
7:     $x_m \leftarrow i$;
8:     $\vec{s}^* \leftarrow$ Search($\{x_1, \ldots, x_m\}$, $m + 1$, $M$, $X - x_m$, $\vec{s}^*$);
```

the $i+1$-th frame is still an optimal assignment. As a result, we can transform optimal assignment $\vec{s}$ to another feasible assignment with the same reward value so that the energy consumption is non-decreasing in the time horizon. □

Suppose that x_i is the number of frames allocated with service level i, where $x_i \in \{0, 1, 2, \ldots, K\}$. For notational brevity, let x_0 be 0. Therefore, the service level of the k-th frame is assigned with service level j that satisfies $\sum_{i=0}^{j-1} x_i < k \leq \sum_{i=0}^{j} x_i$. Based on Theorem 1, when the storage capacity is sufficient, finding the optimal assignment is equivalent to the following programming:

$$\begin{aligned}
\text{maximize} & \sum_{i=1}^{M} x_i r_i \\
\text{s.t.} & \sum_{i=1}^{M} x_i = K \\
& x_i \in \{0, 1, 2, \ldots, K\}, \qquad i = 1, 2, \ldots, M \\
& s_k = \arg_j \left(\sum_{i=0}^{j-1} x_i < k \leq \sum_{i=0}^{j} x_i\right), \quad k = 1, 2, \ldots, K \\
& \vec{s} \text{ is a feasible assignment.}
\end{aligned}$$

The number of feasible solutions under the constraints $\sum_{i=1}^{M} x_i = K$ and $x_i \in \{0, 1, 2, \ldots, K\}\ \forall i = 1, 2, \ldots, M$ is at most $\binom{K+M-1}{M-1}$, which is bounded by $O(\frac{(K+M)^{M-1}}{2^{M-1}})$. Algorithm 2 presents an algorithm by applying recursiveness to derive the optimal solution, in which Search($\emptyset, 1, M, K, \vec{s}^*$) is applied by initialing $\vec{s}^*$ to execute all the K frames at the lowest service level. Since checking whether an assignment is feasible or not can be done in $O(K+M)$, the time complexity of Algorithm 2 is $O(\frac{(K+M)^M}{2^{M-1}})$, which is efficient when M is small.

3.3 Insufficient Storage Capacity

3.3.1 Dynamic Programming for Optimal Solutions

We now present a dynamic programming approach to derive optimal solutions when all the reward values are integers. Suppose that $E(k, \rho)$ is the maximum energy residual in the energy storage after servicing the k-th frame with total reward (for the first k frames) no less than ρ. For notational simplicity, let $E(k, \rho)$ be $-\infty$ if $\rho < 0$ or $\rho > kr_M$. Moreover, by the definition of feasible assignments, the total reward of the first k frames must be at least kr_1. Hence, when $\rho < kr_1$

$$E(k, \rho) = -\infty. \tag{3}$$

Suppose that j_ρ is the service level j with $r_{j-1} \leq \rho \leq r_j$, where r_0 is assumed 0 here for calculating j_ρ. Furthermore, when ρ is

Algorithm 3 Dynamic Programming (DP)

```
Input: $K$ frames, $M$ service levels, $E_C(0)$, $E_H()$, $E_\ell$, $E_{\max}$;
Output: assignment $\vec{s}$;
1: for $\rho \leftarrow 0$; ; $\rho \leftarrow \rho + 1$ do
2:     for $k \leftarrow 1$; $k \leq K$; $k \leftarrow k + 1$ do
3:         calculate $E(k, \rho)$ by applying (3) and (4) when $k = 1$ or
           (3) and (5) when $k > 1$;
4:         if $E(k, \rho) < 0$ then
5:             $E(k, \rho) \leftarrow -\infty$;
6:     if $E(K, \rho) \leq E_\ell$ and $E(K, \rho - 1) \geq E_\ell$ then
7:         backtrack the dynamic programming table to return the
           assignment $\vec{s}$ that constructs $E(K, \rho)$;
```

larger than r_M, let j_ρ be M. The boundary condition of $E(1, \rho)$ for $\rho \geq r_1$ is as follows:

$$E(1, \rho) = \begin{cases} \min\{E_{\max}, \hat{e}\}, & \text{if } \hat{e} = E_C(0) + E_H(1) - e_{j_\rho} \geq 0 \\ -\infty, & \text{otherwise.} \end{cases} \tag{4}$$

Then, for $k \geq 2$ and $\rho \geq kr_1$, the value of $E(k, \rho)$ can be calculated by the following recursive function:

$$\begin{aligned} & E(k, \rho) = \\ & \min\left\{E_{\max}, \max_{j=1,2,\ldots,M} \{E(k-1, \rho - r_j) + E_H(k) - e_j\}\right\} \end{aligned} \tag{5}$$

Note that, if $E(k, \rho)$ is less than 0, we have to set $E(k, \rho)$ to $-\infty$ to indicate that achieving total reward no less than ρ is not possible for the first k frames.

Based on (4) and (5), we can calculate the maximum ρ such that $E(K, \rho) \geq E_\ell$. The dynamic programming is illustrated in Algorithm 3. The following theorem shows the complexity of the algorithm and the optimality of the derived assignments.

THEOREM 2. *When the reward values are integers, Algorithm 3 derives optimal assignments for the discrete reward maximization on energy harvesting problem with time complexity $O(MKR_{opt})$ and space complexity $O(KR_{opt})$, where R_{opt} is the reward value of an optimal assignment.*

PROOF. By the optimal substructure of the recursive function in (5), it is not difficult to see the optimality. Moreover, the loop of ρ in Algorithm 3 stops when ρ is equal to $R_{opt} + 1$. To construct an entry for $E(k, \rho)$ with $k \geq 1$ and $\rho \geq 0$, it takes $O(M)$. As a result, the time complexity is $O(MKR_{opt})$ and the space complexity is $O(KR_{opt})$. □

3.3.2 Dynamic Programming for Approximated Solutions

The dynamic programming in Algorithm 3 requires the assumption that all the reward values are integers and has high complexity for deriving an optimal assignment when the optimal reward value is large. We now present an approximated solution for deriving approximated solutions, which works for any arbitrary reward values. The approximation scheme is based on a technique in rounding down the reward values of each service level. Throughout this section, let ϵ be a user-specified real number between 0 and 1 for the tolerance of approximated solutions. Specifically, by choosing a smaller ϵ, the approximated algorithm will derive a solution which is guaranteed to be more close to the optimal solution, but the complexity will be higher. Therefore, how to choose a proper ϵ to trade the performance with the running time is a task left for system designers. Then, for each service level j, we derive the *rounded reward* r'_j of service level j as follows:

$$r'_j = \left\lfloor \frac{r_j}{\epsilon \cdot r_M} \right\rfloor . \tag{6}$$

Then, by revising the dynamic programming described in (4) and (5) for the rounded rewards, we can have the same dynamic programming as follows:

$$\tilde{E}(k,\rho) = \tag{7}$$
$$\min\left\{E_{\max}, \max_{j=1,2,\ldots,M}\left\{\tilde{E}(k-1,\rho-r'_j)+E_H(k)-e_j\right\}\right\}$$

$$\tilde{E}(1,\rho) = \begin{cases} \min\{E_{\max}, \hat{e}\}, & \text{if } \hat{e} = E_C(0)+E_H(1)-e_{j'_\rho} \geq 0 \\ -\infty, & \text{otherwise,} \end{cases} \tag{8}$$

where j'_ρ is the service level j with $r'_{j-1} \leq \rho \leq r'_j$, where r'_0 is assumed 0 here for calculating j'_ρ. Note that the other boundary conditions in Section 3.3.1 must also be applied for $\bar{E}(k,\rho)$.

Suppose that R' is the maximum value with $\tilde{E}(K,R') \geq E_\ell$. By back-tracking the dynamic programming table, we can derive an assignment $\vec{s}^\sharp$ such that the sum of the rounded reward of $\vec{s}^\sharp$ is R' and the residual energy in the energy storage is no less than E_ℓ at the end of the K-th frame. The following theorem shows that the quality of the derived solution $\vec{s}^\sharp$ of the above dynamic programming is not too far away from the optimal assignment, even in the worse case.

THEOREM 3. *Deriving $\vec{s}^\sharp$ takes $O(\frac{K^2M}{\epsilon})$ time complexity and $O(\frac{K^2}{\epsilon})$ space complexity, and for any input instance with feasible assignment $\vec{s}$,*

$$\frac{1}{1-\epsilon}\sum_{k=1}^{K} r_{s^\sharp_k} \geq \sum_{k=1}^{K} r_{s_k}.$$

PROOF. By Equation (6), we know that $\frac{r_j}{\epsilon \cdot r_M} - 1 \leq r'_j \leq \frac{r_j}{\epsilon \cdot r_M}$. Therefore,

$$\frac{1}{\epsilon \cdot r_M}(\sum_{k=1}^{K} r_{s^\sharp_k}) - K \leq \sum_{k=1}^{K} r'_{s^\sharp_k} \leq \frac{1}{\epsilon \cdot r_M}(\sum_{k=1}^{K} r_{s^\sharp_k}).$$

In terms of rounded rewards, we know that assignment $\vec{s}^\sharp$ is the optimal. Consequently, we have $\sum_{k=1}^{K} r'_{s^\sharp_k} \geq \sum_{k=1}^{K} r'_{s_k}$ and

$$\frac{1}{\epsilon \cdot r_M}(\sum_{k=1}^{K} r_{s^\sharp_k}) \geq \sum_{k=1}^{K} r'_{s_k} \geq \frac{1}{\epsilon \cdot r_M}(\sum_{k=1}^{K} r_{s_k}) - K.$$

As a result, $\sum_{k=1}^{K} r_{s^\sharp_k} \geq \sum_{k=1}^{K} r_{s_k} - \epsilon r_M K \geq (1-\epsilon)\sum_{k=1}^{K} r_{s_k}$, where the last inequality comes from $\sum_{k=1}^{K} r_{s_k} \leq r_M K$.

Since the optimal rounded reward R'_{opt} is at most Kr'_M, the time complexity is $O(K^2 r'_M M) = O(\frac{K^2M}{\epsilon})$, where the space complexity is $O(K^2 r'_M) = O(\frac{K^2}{\epsilon})$. $\square$

4. SIMULATIVE EVALUATION

In this section, we evaluate our approaches by means of simulation. Measurements of solar light intensity $[\frac{W}{m^2}]$ during nearly 5 years recorded at [1] serve as harvested energy input $E_H(t)$. The time interval between two samples is 5 minutes, so we could extensively test and compare our algorithms for long time scales. Of course, one would have to scale the measured power profile in [1] with the size, number and efficiency of the actually used solar panels. However, we expect the influence of this scaling on our qualitative results to be negligible.

Choosing battery capacity and prediction horizon. For system designers, a fundamental question is how to dimension the battery of the embedded system. Besides the different service levels of the application, one has to measure a characteristic trace of the environmental energy source (e.g. solar energy). Let $E_{\max,\min}$ denote the minimum battery capacity $E_{\max}$ to optimally exploit the given energy source. Using Algorithm 3 with $E_{\max} = +\infty$, we can compute $E_{\max,\min}$ as the maximum value of the stored energy $E_C(k)$ for all $k = 1,\ldots,K$. This value can now be computed offline before deployment of the embedded system.

Another question of practical relevance is how to set the length of the prediction horizon. For solar energy, in order to optimize the system behaviour in a long-term perspective, the length of the prediction horizon should be several days. We opted for 8 frames per day and varied the number of days. For a prediction horizon of 5 days, e.g., we have to compute the service levels for $K = 40$ frames. We choose an exemplary set of $M = 5$ service levels with energy consumptions $e \in \{1000, 3000, 5000, 6000, 8000\}$ and reward values $r \in \{4, 5, 12, 13, 18\}$. The simulation begins with an initial energy $E_C(0) = E_\ell = 5000$ and a battery capacity $E_{\max} = 10000$. Note that, as the reward values are integer in our simulations, Algorithm 3 derives the optimal solutions, and, hence, we will not report the performance of Algorithm 2.

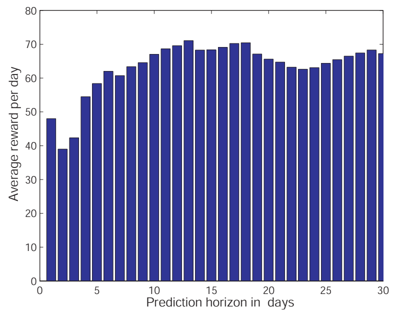

Figure 3: Convergence of the average reward for increasing length of the prediction horizon using 8 frames per day.

A sufficient prediction horizon can now be determined by a simulation as displayed in Figure 3. After 13 days, the average reward reaches a first peak. In contrast to [9] where the case of continuous service levels was studied, the average reward in Figure 3 is not monotonically increasing. However, using a prediction horizon of 13 days (i.e. $K = 104$) is a reasonable choice, since the average reward remains in an interval of $\pm 5\%$ of the long-term average.

Comparisons of the Proposed Algorithms. In the following we compare the performance of the different algorithms presented in this paper. We experienced that for large values of M, i.e., for applications with many service levels, the Greedy Algorithm 1 closely approximates the optimal solution derived by Algorithm 3. However, for the example with only $M = 5$ service levels in the previous section, Algorithm 1 can only roughly approximate the optimal solution. As displayed in Figure 4, the average reward of Algorithm Greedy 1 is converging towards $\approx 63\%$ of the optimal reward value obtained with Algorithm DP. This is one of the worst performances of Algorithm Greedy 1 we could simulate. As mentioned at the beginning of Section 3, this problem cannot be solved in acceptable time using a full search.

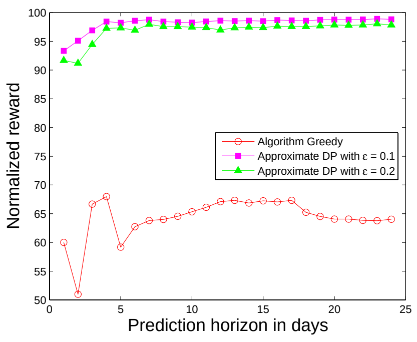

Figure 4: Normalized reward in % for Algorithm Greedy 1 and two approximated DP solutions with $\epsilon = 0.1$ and $\epsilon = 0.2$.

On the other hand, Figure 4 also displays two approximation of the dynamic programming with rounding factor $\epsilon = 0.1$ and $\epsilon = 0.2$. They compute service levels whose rewards are only a few percent from the optimal ones.

We also evaluated the computational complexity of our algorithms for numerous parameters. Instead of measuring the running time on a certain platform, we decided to choose a platform-independent evaluation of our algorithms. For this purpose, we counted the number of operations (both arithmetic as well as logic operations) which have to be executed for each algorithm. As depicted in Figure 5, Algorithm 1 Greedy is only more efficient for short prediction horizons. For our implementations of the algorithms, however, Algorithm 1 Greedy even has a higher computational load for longer prediction horizons. It becomes obvious, that the two approximation algorithms can substantially reduce the computation load.

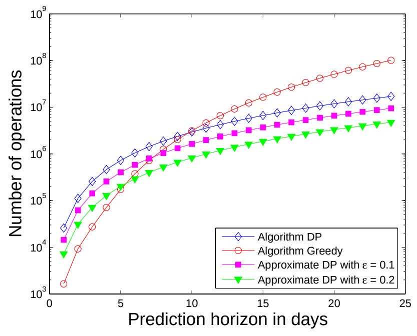

Figure 5: Computational overhead of Algorithm 1 and 3, and two approximated DP solutions with $\epsilon = 0.1$ and $\epsilon = 0.2$.

For the relevant horizons, we counted at most several millions of operations for all algorithms. In consideration of the fact that current sensor node platforms like the Tmote Sky [12] or the BTnode [12] are operated with frequencies between 4 MHz and 16 MHz, we believe that our algorithms are executed efficiently on common sensor nodes.

In summary, Algorithm Greedy 1 gives quite poor solutions for some parameter configurations. For a practical implementation, one would rather choose an suitable dynamic programming approximation. In doing so, one can easily trade performance versus computation time and, unlike for Algorithm Greedy 1, we can give performance guarantees for worst-case scenarios. Concerning the memory requirement for internal variables, we found that the introduced overhead is neglectable for all presented algorithms. Like that, our algorithms can be implemented efficiently on embedded systems, even on resource-constrained systems, e.g., sensor nodes.

5. CONCLUSIONS

This paper explores how to optimize the performance of energy harvesting systems with discrete service levels. We show that greedy algorithms extended from Moser et al. [9] by using continuous service levels do not work very well, even if the scheduling decision is done carefully. When the energy storage is sufficient, we propose a recursive algorithm to derive optimal solutions. For general cases with integer reward values, we propose algorithms based on dynamic programming to derive optimal solutions. For trading the time complexity with the quality of the derived solutions and dealing with non-integer reward values, we also present an approximation scheme, in which the longer running time usually leads to a solution closer to the optimal solution. Our work is supported by simulation results which are based on long-term measurements by real solar cells. In our simulations, we also present how to decide a sufficient battery capacity and a prediction horizon to reach high performance.

Acknowledgements

The presented work was 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. This research has also been supported by the European Network of Excellence on Embedded System Design ARTISTDesign.

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] 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.

[3] T. H. Cormem, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[4] 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, 2005.

[5] 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.

[6] D. Li and P. H. Chou. Application/architecture power co-optimization for embedded systems powered by renewable sources. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 618–623, New York, NY, USA, 2005. ACM.

[7] S. Liu, Q. Qiu, and Q. Wu. Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting. In *Design, Automation and Test in Europe (DATE 08)*, 2008.

[8] 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. Kluwer Academic Publishers, 2007.

[9] C. Moser, J.-J. Chen, and L. Thiele. Reward maximization for embedded systems with renewable energies. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA08)*, pages 247–256, 2008.

[10] 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)*, 2008.

[11] 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.

[12] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *4th International Conference on Information Processing in Sensor Networks (IPSN05)*, pages pp. 364–369, April 2005.

[13] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, pages 457–462, April 25-27 2005.

[14] 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*. Springer.

[15] 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.

[16] F. Simjee and P. H. Chou. Everlast: long-life, supercapacitor-operated wireless sensor node. In *International symposium on Low power electronics and design (ISLPED)*, pages 197–202, 2006.