Adaptive Power Management for Environmentally Powered Systems

Clemens Moser, Lothar Thiele, *Member*, *IEEE*, Davide Brunelli, and Luca Benini, *Fellow*, *IEEE*

Abstract—Recently, there has been a substantial interest in the design of systems that receive their energy from regenerative sources such as solar cells. In contrast to approaches that minimize the power consumption subject to performance constraints, we are concerned with optimizing the performance of an application while respecting the limited and time-varying amount of available power. In this paper, we address power management of, e.g., wireless sensor nodes which receive their energy from solar cells. Based on a prediction of the future available energy, we adapt parameters of the application in order to maximize the utility in a long-term perspective. The paper presents a formal model of the corresponding optimization problem including constraints concerning buffer sizes, timing, and rates. Instead of solving the optimization problem online which may be prohibitively complex in terms of running time and energy consumption, we apply multiparametric programming to precompute the application parameters offline for different environmental conditions and system states. In order to guarantee sustainable operation, we propose a hierarchical software design which comprises a worst-case prediction of the incoming energy. As a further contribution, we suggest a new method for approximate multiparametric linear programming which substantially lowers the computational demand and memory requirement of the embedded software. Our approaches are evaluated using long-term measurements of solar energy in an outdoor environment.

Index Terms—Embedded systems, energy harvesting, power management, multiparametric linear programming.

1 Introduction

Wireless sensor networks (WSNs) have opened up an exciting field of research. A WSN can be seen as a system of self-powered, wireless sensors which are able to detect and transmit events to a base station. WSNs are deployed wherever it is not possible or practical to maintain a wired network infrastructure. In recent years, a great number of prototype sensor networks have been deployed, including networks for volcano monitoring [1], habitat monitoring [2], or glacial movement monitoring [3]. For most of these deployments, unattended operation of the network for long periods is highly desirable. Furthermore, the sensor nodes are anticipated to be small and inexpensive devices which can be unobtrusively embedded in their environment [4]. Thus, a sensor node's hardware is stringently limited in terms of computation, memory, communication as well as storable energy (e.g., batteries). These resource constraints also limit the complexity of the software executed on a sensor node.

The past several years have seen an increasing interest in wireless sensor nodes which are scavenging energy from their environment. In [5], several technologies have been discussed for extracting solar, thermal, kinetic, or vibrational energy from a node's physical environment. In particular, techniques to harvest energy via photovoltaic cells have attracted the interest of the sensor network community [6]. Solar energy is certainly one of the most promising energy sources and typical environmental monitoring applications have access to solar energy. For instance, this is the case for the PermaSense network [7] deployed in the Swiss Alps to investigate Permafrost. For this deployment, unattended operation over multiple years is highly desirable. If the sensor nodes are equipped with photovoltaic cells as energy transducers, the autonomy of the network is increased substantially since frequent recharging and replacement of the batteries become unnecessary. Ideally, sensor nodes once deployed in a harsh environment benefit from a drastically increased operating time and become virtually immortal.

Recently, a number of solar powered prototype sensor nodes have been presented which perform more and more efficient energy conversion. Two of the first prototypes were Heliomote [8] and Prometheus [9]. In both systems, the solar panels are directly connected with the storage device. However, an efficient solar harvesting system should adapt the electrical operating point of the solar cell to the given light condition, using techniques called Maximum Power Point Tracking (MPPT). For solar cells, the size of a few square centimeter particular care has to be taken in order not to waste the few milliwatt generated by the solar cell. Latest MPPT circuits accurately track the maximum power point of a solar cell and can adapt to changing light conditions quickly [10]. These prototypes have successfully demonstrated that solar energy is a realistic energy source for sensor nodes and perpetual operation is indeed possible.

On the other hand, resource constraints also limit the software running on a sensor node. Recently, a number of operation systems have been presented which support resource-limited sensor nodes [11], [12]. The latter systems allow resource-aware programming and support power management. In fact, the methods presented in this paper

- *C. Moser and L. Thiele are with the Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Gloriastr.35, CH-8092 Zuerich, Switzerland. E-mail: {moser, thiele}@tik.ee.ethz.ch.*
- *D. Brunelli and L. Benini are with the Micrel Lab, Department of Electronics, Computer Science and Systems, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. E-mail: {davide.brunelli, luca.benini}@unibo.it.*

Manuscript received 19 Dec. 2008; revised 11 June 2009; accepted 19 Aug. 2009; published online 13 Oct. 2009.
Recommended for acceptance by W. Najjar.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-12-0625.
Digital Object Identifier no. 10.1109/TC.2009.158.

could be implemented in the framework of such a resource-aware operating system. Concerning power management in energy harvesting systems, the energy required for sophisticated optimization techniques may introduce a high control overhead. For sensor nodes which periodically sense and transmit data, but spend most of the time in power saving sleep modes, simple, low-complexity solutions are needed.

Clearly, the power generated by small solar cells is limited. Sensor nodes executing a given application may frequently run out of energy in times with insufficient illumination. If one strives for predictable, continuous operation of a sensor node, common power management techniques have to be reconceived. In addition to perform classical power saving techniques, the sensor node has to adapt to the stochastic nature of solar energy. Goal of this adaptation is to maximize the utility of the application in a long-term perspective. The resulting mode of operation is sometimes also called *energy neutral* operation [13]: The performance of the application is not predetermined a priori, but adjusted in a best effort manner during runtime and ultimately dictated by the power source. Therefore, storage devices like batteries are solely used as energy buffers to compensate the variations of the underlying energy source.

In this paper, we present a class of optimization problems described in terms of software and hardware of energy harvesting systems. As a main contribution, a framework for adaptive power management is proposed which builds on *multiparametric programming* techniques. In doing so, we link methods from control theory with the software design of environmentally powered systems. In addition, particular care has been taken to account for the unreliable nature of environmental energy. On the one hand, we address the design for *worst-case* situations in order to guarantee sustainable operation. On the other hand, an *approximate* multiparametric programming algorithm is presented which results in an acceptable system behavior, avoiding an overly precise computation of the application parameters. In summary, we propose a low complexity and robust software design which is well suited for resource-constrained systems like, e.g., sensor nodes.

This paper is organized as follows: In Section 2, we highlight the contributions of our work, and in Section 3, we discuss related works. In Section 4, a brief overview of the system concept is presented, followed by a detailed discussion of the models and corresponding notation in Section 5. Section 6 describes the basic principles of multiparametric linear programming and illustrates its application with practical examples. A hierarchical system design to separate decisions on energy usage and energy savings is the topic of Section 7. In Section 8, a new method for approximate multiparametric linear programming is presented. Finally, we have a detailed look at implementation issues in Section 9 before Section 10 concludes the paper.

2 Contributions

This paper is based on the results described in [14], [15], [16]. We present a set of tools and methods for adaptive power management in energy harvesting systems. We combine the different viewpoints, include corresponding simulation results, and provide a thorough discussion of implementation aspects. Specifically, the paper contains the following contributions:

- We present a formal model based on linear programming which is able to capture the performance, the parameters, and the energy model of environmentally powered systems.
- To optimize the long-term system behavior, we suggest the use of energy prediction algorithms in combination with simple model predictive controllers. Concretely, we are applying results of the well-established field of multiparametric programming to the emerging area of energy harvesting systems.
- A hierarchical software design is presented which increases the robustness toward energy prediction mistakes. By designing the upper control layer for worst-case situations, depletion of the energy storage is avoided and robustness of the overall system is increased. In addition, we show that the hierarchical design reduces the computation overhead and storage demand significantly.
- We present a new algorithm for approximative multiparametric linear programming. The algorithm computes approximated control laws which reduce the involved online overhead substantially. An experimental setup reveals that the achieved performance is not necessarily decreased compared to the optimal solution.
- We evaluate our methods by means of simulation using long-term measurements of solar energy as input data. In this way, we could extensively test the performance of our algorithms for time scales one usually wants to achieve with, e.g., solar-powered sensor networks.
- We propose practical techniques for the efficient implementation of the controllers, give a thorough analysis of the involved implementation overhead, and demonstrate the practical relevance of our approach by measurements of the controller running on a real sensor node.

3 Related Work

In [17], the authors point out how the problem of adapting the duty cycle of a solar-powered sensor can be modeled by a linear program. As objective, the *average* duty cycle shall be optimized. Instead of periodically solving this linear program online, a heuristic algorithm of reduced complexity is proposed which attempts to decrease the duty cycle in times when the scavenged energy is low (e.g., at night) and increase the duty cycle when scavenged energy is high (e.g., during the day). In contrast, the class of linear programs presented in this paper is capable of modeling a wider variety of application scenarios, constraints, and optimization objectives. We are able to handle arbitrary objectives such as maximizing the *minimum* duty cycle. For the latter objective, we are able to achieve a more balanced system behavior, i.e., we try to prevent an embedded system from shutting down during periods with little harvested energy. The work in [18] improves on the results in [17]; however, the assumed optimization objective and application remain very specific.

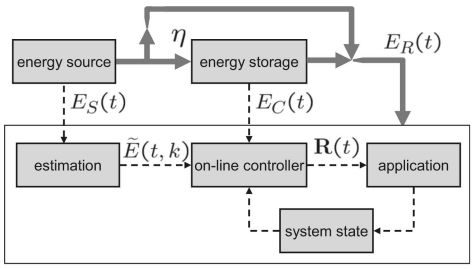


Fig. 1. Illustration of the system concept.

In [19], an optimal real-time scheduling algorithm for energy harvesting systems is presented. Taking into account the available time as well as the harvested energy, an optimal *task ordering* is determined based on the prediction of the available energy. The work in [19] is extended in [20] by taking into account a processor with different execution frequencies (DVS). In contrast to the work presented in our paper, the parameters of the application in [19], [20] are assumed to be given (in form of task arrival times, energy consumptions, and deadlines). By scheduling the different tasks, short-term decisions on the use of environmental energy are made. In our work, we are actively adapting parameters of the application to optimize the performance in a long-term perspective.

The approach in [21] is restricted to a very special scheduling problem: Periodic tasks with certain rewards are scheduled within their deadlines according to a given energy budget. The overall objective is to maximize the sum of rewards. Therefore, energy savings are achieved using Dynamic Voltage Scaling (DVS). The energy source is assumed to be solar and comprises a simplified day state as well as a night state. A similar reward maximization problem has been recently presented in [22]. In the latter work, the received reward is assumed to be a concave function over the energy consumption. Again, our approach generalizes the work in [21], [22] for a much larger variety of application scenarios and objective functions and the works in [21], [22] can be seen as special cases of our approach.

4 System Concept

The system model is depicted in Fig. 1. The whole hardware/software system is powered by an energy harvesting device that delivers in a unit time interval starting at t the energy $E_S(t)$. In the same time interval, the system uses energy $E_R(t)$. At time t, there is the stored energy $E_C(t)$ available. It is a typical feature of recent energy harvesting circuits that solar energy E_S is either used directly by the system or is stored in the energy storage for later use. In the latter case, the charging of the battery is done with some efficiency η, whereas the direct usage is free of energy losses. This possibility of bypassing the energy storage can be exploited to *save* energy by minimizing the round-trip losses in the energy storage device. For a discussion on how this system model matches real hardware implementations, the reader is referred to Section 9.

Besides the application, there are two additional software tasks running on the target architecture. The estimator predicts future energy production of the harvesting device based on measurements of the past. The controller adapts properties of the application, e.g., task activation rates, based on the estimation of future available energy, the currently stored energy, and additional information about the system state, e.g., the amount of available data memory. Parameters of the application are modified by the online controller. During execution, the system state (e.g., the amount of information stored in local memory and the stored energy) is changed. It is the duty of the controller to adjust properties of the application such that long-term objectives are optimized (for example, maximizing the sampling rate of a sensor) while respecting system constraints (for example, using not more than the available memory).

5 Basic Models and Methods

In this section, we present our model of energy harvesting systems. In Section 5.1, we start with the principal modeling of the physical power flow, followed by a brief description of our rate-based application model in Section 5.2. Next, we introduce the principles of energy prediction and receding horizon control as fundamental techniques to control the long-term behavior of the system (Section 5.3). Finally, in Section 5.4, we show how optimization problems in energy harvesting systems can be modeled by the class of linear programs.

5.1 Power Flow and Energy Storage Model

The modeling is based on the notion of discrete time $t \in \mathbb{Z}_{\geq 0}$ where the difference in physical time between two discrete time instances is denoted as T. Energy-related sensing and control may happen only at times t. In a practical setting, one may have a basic time interval T of a few minutes or even an hour.

The energy harvesting device is modeled as a power source which delivers energy $E_S(t)$ in the time interval $[t, t+1)$ of length T. Therefore, in time interval $[t_1, t_2)$ with $t_1,\ t_2 \in \mathbb{Z}_{\geq 0}$, it delivers energy $E_S(t_1, t_2) = \sum_{t_1 \leq u < t_2} E_S(u)$. The incoming power can be stored in an energy storage device, e.g., a rechargeable battery or a supercapacitor. We denote η the storage efficiency of a nonideal storage device. The energy level at time t is denoted as $E_C(t)$. In dependence on the energy $E_R(t)$ drawn from the sensor node, the increment $\Delta E_C(t)$ of the stored energy is defined at each time t according to the following statement:

$$\begin{aligned} &\textbf{if } E_S(t) > E_R(t) \\ &\qquad \textbf{then} \quad \Delta E_C(t) = \eta \cdot (E_S(t) - E_R(t)) \\ &\qquad \textbf{else} \quad \Delta E_C(t) = (E_S(t) - E_R(t)). \end{aligned}$$

If the generated energy $E_S(t)$ is higher than energy $E_R(t)$, the sensor node is powered directly by the solar cell and excess energy is used to replenish the energy storage. As in [17] and [15], we account for the efficiency η during the charging of the energy storage. As the arrangement in Fig. 1 is fully symmetrical, one could also consider η when the storage is discharged. Alternatively, efficiencies of $\frac{\eta}{2}$ for charging and discharging are thinkable; from a control point of view, all three mappings of the efficiency η are equivalent.

The energy $E_R(t)$ is used to execute tasks on various system components. For sensor nodes, tasks may be as diverse as sensing, signal processing, A/D or D/A conversion, computing, or communicating. A task $\tau_i \in \mathbf{I}$ from the

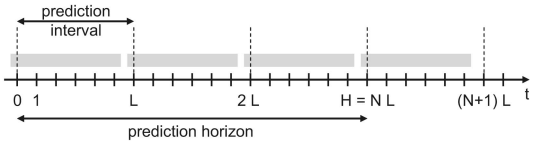


Fig. 2. Illustration of the prediction horizon.

set of tasks $\mathbf{I}$ needs energy e_i for completing a single instance. We suppose that a task is activated with a time-variant rate $s_i(t)$, i.e., during the basic time interval T starting at t, the task is executed $s_i(t)$ times. Therefore, a task needs energy $E_i(t_1, t_2) = \sum_{t_1 \le u < t_2} e_i \cdot s_i(u)$ in time interval $[t_1, t_2)$ for successful execution. Finally, we denote $\mathbf{S}(t)$ the vector of all task rates s_i at time t. The detailed application and task model will be described in the next section.

5.2 Rate-Based Application Model

As described in Section 4, parameters of the application are changed at runtime in order to optimally use the available energy in the future. In this paper, we restrict ourselves to a rate-based application model. The application consists of tasks τ_i, $i \in \mathbf{I}$. A task is instantiated $s_i(t) \ge 0$ times in the interval of length T starting at time t and the execution of each instance needs energy e_i. The activation of tasks can be modeled by a rate graph whose nodes and edges represent tasks and activation relations, respectively. In this way, various dependencies between the tasks can be modeled. The interested reader is referred to [14], where a detailed description of rate graphs for more sophisticated application models can be found.

5.3 Energy Prediction and Receding Horizon Control

According to the system model described in Section 4, the online controller receives energy estimations $\widetilde{E}(t,k)$ of the future harvested energy. Based on these predictions, the online controller computes future control parameters $\mathbf{R}$ which optimize the long-term behavior of the system. In order to keep the control problem computationally tractable, both energy prediction and calculation of future control rates are planned for a finite horizon, leading to the concept of receding horizon control (RHC) [23].

The estimation unit receives tuples $(t, E_S(t))$ for all times $t \ge 1$ and delivers N predictions on the energy production of the energy source. We assume that the prediction intervals are of equal size denoted as the number L (in units of the basic time interval T). We denote the total prediction horizon $H = N \cdot L$ (again in units of the basic time interval T), see also Fig. 2. At time t, the predictor produces estimations $\widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ for all $0 \le k < N$. We write $\widetilde{E}(t,k) = \widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ as a shorthand notation, i.e., the estimation of the incoming energy in the $(k+1)$st prediction interval after t.

The prediction algorithm should depend on the type of the energy source and the system environment. Standard techniques known from automatic control and signal processing can be applied here. In this paper, we will, e.g., provide prediction algorithms which are useful for solar cells that operate in an outdoor environment. The algorithms used for the experimental results will be presented in the respective sections.

At time t, the controller is computing the control sequence $\mathbf{R}(t + k \cdot L)$ for all prediction intervals $0 \le k < N$ based on the estimates $\widetilde{E}(t,k)$ as well as the current system state (e.g., $E_C(t)$). In other words, the rates s_i of the different tasks are planned to be constant during each prediction interval. However, *only the first* control rates $\mathbf{R}(t)$ are applied to the system during the first time step T. The rest of the control sequence is discarded. At time $t+T$, a new vector $\mathbf{R}(t)$ is computed which extends the validity of the previous vector $\mathbf{R}(t-1)$ by one time step. Again, only the first control is used, yielding an RHC strategy.

5.4 Linear Program Setup

The first step in constructing the online controller is the formulation of the optimization problem in form of a parameterized linear program (LP). Corresponding solution methods will be described in Sections 6 and 8.

One of the essential states of the system is the stored energy. Using the power model described in Section 5.1, we obtain the following state equations:

$$E_C(t + k \cdot L) = E_C(t) - k \cdot \zeta + \sum_{j=0}^{k-1} (\widetilde{E}(t,j)) - \sum_{j=0}^{k-1} \big(L \cdot \mathbf{E}^T \cdot \mathbf{S}(t + j \cdot L) + (1-\eta)\lambda(j)\big),$$

$$\forall 1 \le k \le N, \tag{1}$$

$$\lambda(j) \ge 0, \forall 0 \le j < N, \tag{2}$$

$$\lambda(j) \ge \widetilde{E}(t,j) - L \cdot \mathbf{E}^T \cdot \mathbf{S}(t + j \cdot L), \forall 0 \le j < N. \tag{3}$$

They determine upper bounds for the expected contents of the energy storage at times $t + kL$ for $1 \le k \le N$, where N denotes the number of intervals in the prediction horizon. The factor ζ accounts for energy leakage of the storage device. $\mathbf{S}(t)$ is a vector containing all activation rates $s_i(t)$ and $\mathbf{E}^T$ is a (row) vector that contains all energy requirements e_i for all $i \in \mathbf{I}$. Therefore, the total energy consumption in an interval starting at time t can be written as $L \cdot \mathbf{E}^T \cdot \mathbf{S}(t) = L \cdot \sum_{i \in \mathbf{I}} e_i \cdot s_i(t)$. The auxiliary variable $\lambda(j)$ accounts for the energy bypassing mechanism in the $(j+1)$st prediction interval as described in Section 5.1. If the predicted harvested energy is smaller than the predicted energy consumption in a certain prediction interval, $\lambda(j)$ is forced to zero. In this case, the energy difference is just drawn from the battery. The other way round, if the predicted harvested energy is greater than the predicted energy consumption in a certain interval, $\lambda(j)$ has a positive value. For this interval, the excess energy is stored with efficiency η in the battery. This corresponds exactly to the if-else-statement given in Section 5.1.

In a similar way, we can also model other system states, for example, memory. A task could produce a certain amount of data that is stored and another removes it, e.g., by means of communication to another node. In this case, we would have for $1 \le k \le N$ the state equations:

$$M(t + k \cdot L) = M(t) + L \sum_{j=0}^{k-1} \mathbf{M}^T \cdot \mathbf{S}(t + j \cdot L), \tag{4}$$

where $M(t)$ denotes the amount of stored data at time t and m_i is the amount of data produced or consumed by a task τ_i with rate s_i in a time interval of length T. $\mathbf{M}^T$ is a (row) vector that contains all data amounts m_i for all $i \in \mathbf{I}$. Of course, (1)-(4) provide only examples of possible system states and their associated changes. Moreover, there may be constraints on the feasible states, for example,

$$0 \leq E_C(t + k \cdot L) \leq E_{\max,} \tag{5}$$

$$0 \leq M(t + k \cdot L) \leq M_{\max,} \tag{6}$$

for $0 \leq k \leq N$.

One can now easily combine (1)-(6) and obtain a system of linear equalities and inequalities that contain as free variables $M(t + k \cdot L)$ (the state of the memory), $E_C(t + k \cdot L)$ (the state of the energy) for $1 \leq k \leq N$, and the rate control $\mathbf{R}(t + k \cdot L)$ for $0 \leq k < N$.

So far, no optimization goal has been formulated, and therefore, any feasible rate control $\mathbf{R}(t + k \cdot L)$ could be a solution. Any linear objective function J that makes use of the free variables given above is possible in this case. One may also define additional variables in order to model specific objectives. One possible (very simple) example would be the objective

$$\begin{aligned} &\text{maximize } J = \mu \\ &s_1(t + k \cdot L) \geq \mu \quad \forall 0 \leq k < N, \end{aligned} \tag{7}$$

which would attempt to maximize the minimal rate with which the task τ_1 is operated in the finite horizon $0 \leq k < N$. This could, for example, be a task that gathers sensor data and it is desired that the minimal rate is as large as possible. In terms of intervals, the objective translates into a minimization of the maximum interval between any two consecutive measurements. Hence, one could apply this objective in scenarios where one attempts to minimize unobserved time periods like, e.g., in environmental monitoring or intruder detection applications.

Clearly, for the objective in (7), the controller would continuously try to empty the storage at the horizon to obtain an optimal objective value. Therefore, we formulate a final-state constraint that ensures energy neutral, sustainable operation:

$$E_C(t + N \cdot L) \geq E_C(t) + \alpha(t). \tag{8}$$

A common choice for the prediction horizon is $H \cdot T = 24$ h, see, e.g., [24] or [15]. In [24], the energy offset $\alpha(t)$ is set to 0 for all times t. Following a diurnal circle, solar energy is assumed to behave similarly on consecutive days. In [15], α is manually tuned to some fixed value. However, it has become evident that the choice of $\alpha(t)$ severely influences the performance of a system: decreasing $\alpha(t)$ results in a more aggressive control behavior, running the risk to deplete energy E_C. On the other hand, increasing $\alpha(t)$ may lead to overly conservative rates $\mathbf{S}$, poor performance, and an high energy level E_C. In Section 7, a solution will be presented how α can be tuned automatically using an hierarchical control approach.

Obviously, solving at each time step t, a linear program in a resource limited system is prohibitive in general. However, we can conclude that the optimal rate control can be determined by solving a parameterized linear program, where the parameters are $\widetilde{E}(t, j)$, $E_C(t)$, and $M(t)$. In the next section, we will describe an optimal method for solving the above parameterized linear program offline and using the result for constructing an optimal online controller. In Section 8, we will show how this online controller can be approximated to obtain a less precise but much simpler controller.

6 Multiparametric Control Design

Next, we will show how to design an online controller based on multiparametric linear programming (mp-LP) which avoids solving a linear program at each time step. Thereby, we are following the ideas in [25], where the regulation of discrete-time-constrained linear systems is studied in the context of model predictive control. In [14], the application of multiparametric linear programming has been proposed for the first time for energy harvesting systems. We will briefly recall the main results in Section 6.1 and illustrate the approach with the help of simple examples in Sections 6.2 and 6.3. The last example presented in Section 6.4 shows the limits of the approach.

6.1 Controller Generation

As a first step, we define a state vector $\mathbf{X}$ consisting of the actual system state, the level of the energy storage as well as the estimation of the incoming energy over the finite prediction horizon (cp. Fig. 1). Resuming the system dynamics formulated in (1)-(8), the state vector $\mathbf{X}$ can be written as

$$\mathbf{X}(t) = (E_C(t),\ M(t),\ \widetilde{E}(t, 0), \ldots, \widetilde{E}(t, N-1))^T. \tag{9}$$

Furthermore, let us denote the vector of optimal control inputs to the system, i.e., the vector of planned rates $\mathbf{R}$ as

$$\begin{aligned} \mathbf{U}^*(\mathbf{X}, t) = (&\mathbf{R}^T(t), \mathbf{R}^T(t + L), \ldots, \\ &\mathbf{R}^T(t + (N-1) \cdot L))^T. \end{aligned} \tag{10}$$

Note that we need to compute the optimal control input $\mathbf{R}\ (t)$ for the next prediction interval as input for the application (cp. Fig. 1). The state space of $\mathbf{X}$ (in our case $\mathbb{R}^{N+2}$ bounded by possible constraints on $E_C()$, $M()$, and $\widetilde{E}()$) can now be partitioned into a number N_{CR} of polyhedrons. For each of these polyhedrons j (also called critical regions), the optimal solution $\mathbf{U}^*(\mathbf{X})$ of the control problem can be made available explicitly as

$$\mathbf{U}^*(\mathbf{X}) = \mathbf{B}_j\mathbf{X} + \mathbf{C}_j \quad \text{if } \mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, \quad j = 1, \ldots, N_{CR}, \tag{11}$$

where $\mathbf{B}_j \in \mathbb{R}^{N \times (N+2)}$, $\mathbf{C}_j \in \mathbb{R}^N$, and $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1 \ldots N_{CR}$ is a polyhedral partition of the state space of $\mathbf{X}$. In other words, every possible state vector $\mathbf{X}$ belongs to exactly one critical region j. For all state vectors within this critical region j, the vector of optimal control inputs $\mathbf{U}^*(\mathbf{X})$ can be computed as a linear function of the state vector $\mathbf{X}$. Note that for simplicity, we dropped the dependence on t of the state vector $\mathbf{X}$. The computation of the vectors and matrices of control law (11) is done offline using, e.g., the algorithm presented in [26] or other efficient solvers cited in the latter work. In principle, these solvers solve the optimization problem for *every* possible state vector $\mathbf{X}$.

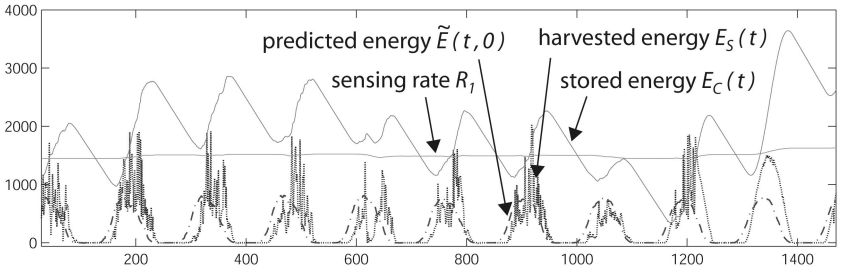


Fig. 3. Adaptation of the sensing rate R_1.

In the online case, the controller has to identify to which region j the current state vector $\mathbf{X}$ belongs. After this membership test, the optimal control moves $\mathbf{U}^*$ for the next N prediction intervals may be computed by evaluating a linear function of $\mathbf{X}$. However, according to the receding horizon policy, it is sufficient to calculate only the first rates $\mathbf{R}(t)$ for the next interval. These rates $\mathbf{R}(t)$ are identical to the rates one would obtain by solving the linear program. However, the computational demand is greatly reduced compared to solving an LP online. After having solved the mp-LP in advance, a set of N_{CR} polyhedra with associated control laws has to be stored and evaluated at each time step t. The computation demand in the online case now depends on

- the number of critical regions N_{CR} which have to be tested,
- the size of the state vector $\mathbf{X}$ (in particular, the number of prediction intervals N), and
- and finally, the number of controlled rates $\mathbf{R}$ which have to be determined.

If the number of critical regions N_{CR} gets large, the computational effort still may be large as many matrix multiplications $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j$ must be performed. Indeed, this general shortcoming of the mp-LP approach may render the calculated controllers inapplicable for resource-constrained systems. By dividing the problem in subproblems within a hierarchical framework (see Section 7) and proposing an approximate, suboptimal multiparametric solver (see Section 8), we will show how complex control problems can be mastered anyhow.

6.2 Adaptation of Sensing Rate (Example I)

We implemented online controllers for exemplary case studies using the MATLAB toolbox in [27]. Measurements of solar light intensity $[\frac{W}{m^2}]$ recorded at [28] serve as energy input $E_S(t)$. In order to simplify the presentation of the results, we normalized and scaled all physical quantities. Like that, we avoid presenting results which are conclusive for a specific system only. For instance, our results hold for different sizes of solar cells and different types of sensor nodes. The energy *prediction* algorithm that has been used is the same as in [14]. It is similar in nature to the predictor used in [24] and attempts to predict the most probable, *average* energy values for the prediction intervals.

Let us assume the following example: In a home sensing application, sensor nodes are deployed in a building for thermal comfort analysis. A sensor node is expected to measure some physical quantity like, e.g., ambient temperature and has to transmit the sampled data to a base station. We can model these requirements as a single sensing task τ_1 with rate $R_1(t)$, i.e., a task which is instantiated R_1-times in the interval $[t, t+T)$. For the sake of simplicity, the sensing task τ_1 drains at every instantiation 1 energy unit from the battery. Assume further that we want to minimize the delay which may occur due to the absence of environmental energy, such as in (7). The environmental energy may stem from solar light as well as from indoor illumination. We can formulate the linear program LP I as shown below. Note that the last inequality in LP I is used to stabilize the receding horizon controller.

$$\begin{array}{cc}
\text{maximize } J = \lambda \text{ subject to :} & \text{(LP I)} \\
R_1(t + k \cdot L) \geq \lambda & \forall 0 \leq k < N \\
E_C(t + k \cdot L) = E_C(t) + & \\
\sum_{j=0}^{k-1} (\widetilde{E}(t,j) - L \cdot R_1(t + j \cdot L)) & \forall 1 \leq k \leq N \\
E_C(t + k \cdot L) \geq 0 & \forall 1 \leq k \leq N \\
E_C(t + N \cdot L) \geq E_C(t) - 100 &
\end{array}$$

In general, the number of partitions N_{CR} of a multiparametric solution grows with the size of the state vector $\mathbf{X}$. Hence, it is of practical concern to keep the number of prediction intervals $\widetilde{E}(t,i)$ and therewith the dimension of $\mathbf{X}$ as small as possible. We chose $L = 24$ and $N = 6$ and obtain the states $\mathbf{X}(t) = (E_C(t), \widetilde{E}(t,0), \ldots, \widetilde{E}(t,5))^T$. The resulting online controller consists of $N_{CR} = 7$ partitions. Fig. 4 visualizes the partition $\mathbf{H}_i\mathbf{X}(t) \leq \mathbf{K}_i, i = 1, \ldots, 7$ for an arbitrarily chosen set of parameters $\widetilde{E}(t,2) - \widetilde{E}(t,5)$.

In Fig. 3, the generated controller is optimizing the sensing rate R_1 over a time period of seven days. We started the simulation with an energy level $E_C(0) = 500$ and found a nearly constant rate R_1 during the whole simulation time. On the other hand, the stored energy $E_C(t)$ is highly varying, since the controller successfully compensates the unstable power supply $E_S(t)$. As a consequence, the stored energy $E_C(t)$ is increasing during the day and decreasing at

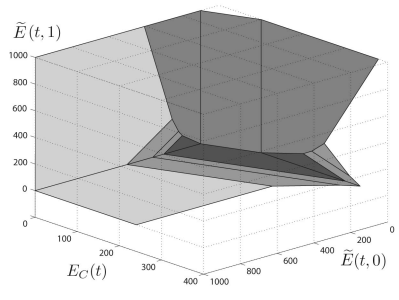


Fig. 4. Three-dimensional view of the polyhedral partition of the state space $\mathbf{X}$ for LP I. Cut through $\widetilde{E}(t,2) = 100.0$, $\widetilde{E}(t,3) = 120.0$, $\widetilde{E}(t,4) = 300.0$, and $\widetilde{E}(t,5) = 10.0$.

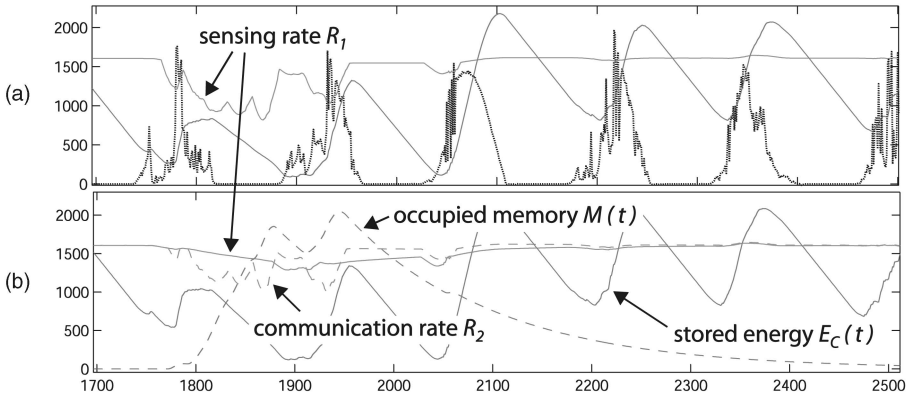


Fig. 5. (a) Scenario without local memory. (b) Scenario with optimized local memory.

night. Even the fourth displayed day with significant less sunshine is not jeopardizing the sensing rate R_1. Fig. 3 demonstrates that the online controller manages to meet the optimization goal for this simple example.

This optimization procedure could also be relevant for other applications. For instance, a camera sensor node deployed in an environmental monitoring application is expected to detect a certain event. Also, for this application, minimizing the maximum unobserved interval between two images might be a useful objective. However, it should be mentioned that there exist sensor applications for which the sensing and transmission rates are fixed. In some cases, the phenomena being sensed necessitate a certain sampling frequency. Moreover, in certain real-time applications, delaying of data transmissions may not be possible.

6.3 Local Memory Optimization (Example II)

As a second example scenario, let us consider a sensor node with an attached vision sensor, as the one presented in [29]. The application may consist of two tasks running on the sensor node: A first task τ_1 is sampling some data, e.g., capturing an image and storing the data in some local memory. A second task τ_2 is transmitting stored samples to a base station with rate R_2 and thereby frees memory from the storage device.

In the following, we derive a relation between the energy consumptions of tasks τ_1 and τ_2 for a specific sensor application. To this end, we assume the smart vision system in [29] to be attached to an MICA2 sensor node [30]. According to the measurements in [29], capturing an 8-bit monochrome image of 352×288 pixels takes approximately 4 mJ and is done in 90 ms. Meanwhile, the MICA2 mote is in idle mode consuming 9.6 mW. The recorded image is then transferred to the local memory via a serial interface with $115\,\frac{kbit}{s}$. During the data transfer, both the 50 mW power consumption of the vision sensor system and the 24 mW of the MICA2 in active mode have to be considered. Subsequently, the vision sensor is shut down and image is transmitted with 76.2 mW power consumption of the MICA2 mote. Assuming a 15 percent data overhead of the packet and a transmission rate of $15\,\frac{kbit}{s}$, the image is transmitted in 62 seconds to the base station. Finally, we get the energy ratio $\frac{0.5J}{4.7J}$ for sampling and transmitting a single image, and hence, we set the normalized energies $e_1 = 0.1$ and $e_2 = 0.9$.

$$
\begin{array}{rl}
\text{maximize } J = (\lambda - \mu) \text{ subject to :} & \text{(LP II)} \\
R_1(t + k \cdot L) \geq \lambda & \forall 0 \leq k < N \\
M(t + N \cdot L) \leq \mu & \\
E_C(t + k \cdot L) = E_C(t) + \sum_{j=0}^{k-1} \widetilde{E}(t, j) - & \\
\sum_{j=0}^{k-1} (L \cdot [0.1 \;\; 0.9] \cdot \mathbf{R}(t + j \cdot L)) & \forall 1 \leq k \leq N \\
E_C(t + k \cdot L) \geq 0 & \forall 1 \leq k \leq N \\
M(t + k \cdot L) = M(t) + & \\
L \sum_{j=0}^{k-1} [1 \;\; -1] \cdot \mathbf{R}(t + j \cdot L) & \forall 1 \leq k \leq N \\
M(t + k \cdot L) \geq 0 & \forall 1 \leq k \leq N \\
E_C(t + N \cdot L) \geq E_C(t) - 150 &
\end{array}
$$

For this application, two reasonable optimization objectives would be 1) to minimize the unobserved intervals between any two consecutive samples and 2) to minimize the amount of stored data M. The purpose of the second objective is twofold: On one hand, sensor nodes are usually small, inexpensive low power devices with constrained hardware resources such as memory. On the other hand, the objective may to some extent enforce the freshness of data arriving at the base station. The corresponding linear program is given in (LP II).

To account for the additional system state $M(t)$, we reduced the number of prediction intervals and set $L = 36$ and $N = 4$. The state space of $\mathbf{X}(t) = (E_C(t), M(t), \widetilde{E}(t, 0), \ldots, \widetilde{E}(t, 3))^T$ is divided by the control law in $N_{CR} = 39$ critical regions.

Fig. 5 displays the simulated curves of the state and control variables during six days. The figure at the top represents a scenario with no use of local memory, i.e., the control problem in LP I, whereas the bottom figure shows a comparable scenario using local memory according to LP II. Until $t = 1{,}700$, both tasks are adjusted to the same rate $R_1 = R_2$ and the memory $M(t)$ is empty. However, after two days with little harvested energy, the energy level $E_C(t)$ on the sensor node is falling and the controller starts to suspend the energy-costly communication task τ_2 by reducing rate R_2. Consequently, the number of buffered samples M is increasing starting before $t = 1{,}800$. In the following, the controller achieves to autonomously regulate the trade-off between $E_C(t)$ and $M(t)$. While a lack of $E_C(t)$ would cause the noninitiation of task τ_1, an increasing $M(t)$ directly affects the second optimization objective. After $t = 1{,}950$, the controller reduces the amount of occupied local memory by increasing the rate R_2. It becomes obvious that in terms of the minimum sampling rate R_1, the controller for LP II

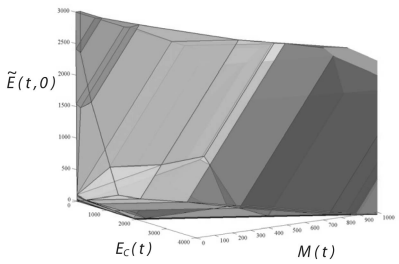


Fig. 6. Three-dimensional view of the polyhedral partition of the state space $\mathbf{X}$ for LP III. Cut through $\widetilde{E}(t,1) = 1{,}200$, $\widetilde{E}(t,2) = 1{,}000$, $\widetilde{E}(t,3) = 0$, $\widetilde{E}(t,4) = 100$, and $\widetilde{E}(t,5) = 500$.

outperforms the controller for LP I. The minimum rate $R_{1,\min} \approx 820$ for LP I is significantly lower than the minimum rate $R_{1,\min} \approx 1{,}280$ for LP II.

6.4 Optimization with Nonideal Energy Storage (Example III)

In this section, we discuss a more involved example to pinpoint the limits of a pure multiparametric control approach. First, we show that explosions of the polyhedral partition of the online controller may happen for practical examples. And second, we show that care has to be taken when choosing the energy prediction algorithm, since the robustness of the overall system severely depends on it.

Example III is similar to Example II. The main difference is that we now consider an energy storage device with a nonideal round-trip efficiency $\eta = 0.8$. Assume a video surveillance application where a first task τ_1 is recording images, performing some image compression, and stores the data in some local memory. The task is instantiated with rate $r_1(t)$ and at every instantiation, one data unit is stored. The maximum storage capacity of the sensor node is $M_{\max} = 1{,}000$. A second task τ_2 with rate $r_2(t)$ frees memory by transmitting images to a base station. We choose again $e_1 = 0.1$ and $e_2 = 0.9$ as energy demands of the tasks $r_1(t)$ and $r_2(t)$, respectively. For this application, we want to minimize the unobserved intervals between any two consecutive images (i.e., activations of task τ_1). The corresponding linear program LP III is given below.

$$
\begin{aligned}
&\text{maximize } J = \mu \text{ subject to :} && \text{(LP III)}\\
&R_1(t + k \cdot L) \geq \mu && \forall 0 \leq k < N\\
&R_1(t + k \cdot L), R_2(t + k \cdot L) \geq 0 && \forall 0 \leq k < N\\
&E_C(t + k \cdot L) = E_C(t) + \sum_{j=0}^{k-1} (\widetilde{E}(t,j) - \\
&L\ [0.1\ 0.9]\ \mathbf{R}(t + j \cdot L) - (1 - \eta)\lambda(j)) && \forall 1 \leq k \leq N\\
&\lambda(j) \geq \widetilde{E}(t,j) - L\ [0.1\ 0.9]\ \mathbf{R}(t + j \cdot L) && \forall 0 \leq j < N\\
&\lambda(j) \geq 0 && \forall 0 \leq j < N\\
&E_C(t + k \cdot L) \geq 0 && \forall 1 \leq k \leq N\\
&M(t + k \cdot L) = M(t) + \\
&\textstyle\sum_{j=0}^{k-1} L \cdot [1\ -1] \cdot \mathbf{R}(t + j \cdot L) && \forall 1 \leq k \leq N\\
&0 \leq M(t + k \cdot L) \leq M_{\max} && \forall 1 \leq k \leq N\\
&E_C(t + N \cdot L) \geq E_C(t)
\end{aligned}
$$

We performed the experiments using long-term measurements of solar light intensity measured in [31] and shorter time intervals of $T = 5$ minutes. The energy predictor used is the same as in the previous sections. We set the prediction horizon to $H \cdot T = 24$ h and subdivided the horizon in $N = 6$ intervals of length $L = 48$. For the optimization problem in LP III, we compute the explicit solution via multiparametric programming for the state vector $\mathbf{X}(t) = (E_C(t), M(t), \widetilde{E}(t,0), \ldots, \widetilde{E}(t,5))^T$. The resulting online controller consists of $N_{CR} = 1{,}049$ partitions. Fig. 6 visualizes the partition $\mathbf{H}_j\mathbf{X}(t) \leq \mathbf{K}_j, j = 1, \ldots, 1{,}049$ for an arbitrarily chosen set of parameters $\widetilde{E}(t,1) - \widetilde{E}(t,5)$.

The complete control law requires the storage of $(3N + 9) \cdot N_{CR}$ real numbers. In the worst case, the active region is the last one to be examined, and the controller will give a solution after $(3N + 6) \cdot N_{CR}$ multiplications, $(3N + 3) \cdot N_{CR}$ sums, and $N_{CR} - 1$ comparisons. In Table 1, the complexity of the control law is given in terms of storage demand and number of operations. As it turns out, even with probabilistic region checking or efficient representations of the state space, the computational demand of the problem remains considerable. This holds in particular since the control law has to be evaluated every time step T.

In Fig. 7, an exemplary situation is displayed where the generated controller is optimizing the sampling rate R_1 and the transmission rate R_2 during seven days. During the first five days, the controller achieves to optimize the rate R_1 in spite of the unstable power supply $E_S(t)$. The transmission rate R_2 is oscillating around the sampling rate R_1. Since it is favorable to use energy when available, data are stored at night and transmitted during day. At this, the finite amount of storable data $M_{\max} = 1{,}000$ is chosen large enough not to constrain the dynamic of the ratio $\frac{R_1}{R_2}$. Only if the memory is $M_{\max} < 800$, we observed that the parameter $M_{\max}$ begins to influence the overall system behavior. Note that the parameters in Fig. 7 are scaled appropriately to show them all in one diagram. The absolute values, however, are not necessary those displayed on the y-axis in Fig. 7.

However, at the end of the displayed time period, the controller is forced to suspend the sampling of data and achieves the worst objective value possible: Shortly before $t = 4{,}200$, the sampling rate R_1 is set to 0 for approximately

TABLE 1
Complexity Reduction due to Hierarchical Control Design

control design	N	$H \cdot T$	activation frequency	N_{CR}	storage (real numbers)	ops (worst case)
single controller	6	$1day$	$T = 5$ min	1049	28323	52449
hierarch., subc. 1	$N_1 = 30$	$H_1 \cdot T = 30days$	$L_1 \cdot T = 1day$	30	1920	3689
subc. 2	$N_2 = 6$	$H_2 \cdot T = 1day$	$T = 5$ min	161	2898	4829

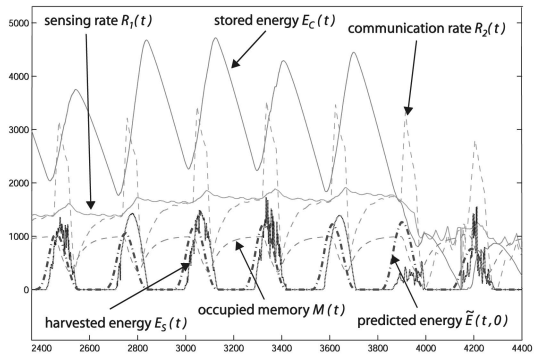


Fig. 7. Single controller for LP III, explicit solution via multiparametric programming, and average energy prediction $\widetilde{E}(t,k)$.

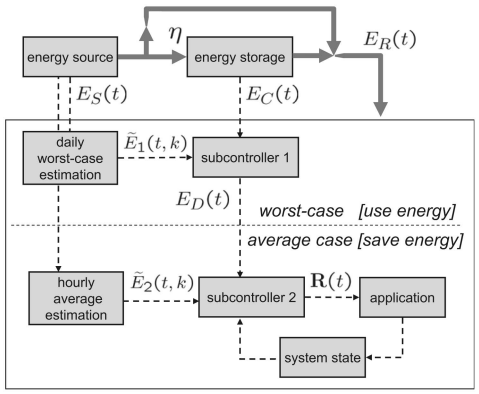


Fig. 8. Illustration of the hierarchical control model.

20 minutes. One of the main reasons for this breakdown is the averaging energy predictor $\widetilde{E}$ which—per definition—is unable to foresee extreme situations. As illustrated in Fig. 7, an overestimation of the actually incoming energy E_S may deplete the stored energy quickly, resulting in an unpredicted performance degradation. From our experiments, we know that the controller computed for LP III is unable to avoid these kind of breakdowns with the chosen predictor $\widetilde{E}$, independent of the length of the prediction horizon H, and the number of intervals N.

7 Hierarchical System Model

7.1 Design Principles

From the examples presented in the last section, we identify unexpected energy depletion as a major challenge for environmentally powered systems. In this section, we present a hierarchical system design whose primary purpose is to prevent the system from running out of energy.

The main idea is to subdivide the controller for the problem formulated in (1)-(8) into two subcontrollers, each with its dedicated energy prediction algorithm. The parameters of the prediction horizons are denoted as L_1, N_1, and H_1 for the upper control layer, and L_2, N_2, and H_2 for the lower control layer. According to Fig. 8, subcontroller 1 receives estimates of the daily energy in order to ensure long-term sustainability of the energy harvesting system. Therefore, we assume a prediction interval to be one day, i.e., $L_1 \cdot T = 24$ h, and the total prediction horizon should be chosen in the order of several weeks, e.g., $H_1 \cdot T = 30$ days. At the interface of both control layers, we define the energy allowance $E_D(t)$ as the sum of energies which shall be used by the system at the next day.

By calculating $E_D(t)$, the upper layer determines an energy budget which serves as input to the lower control layer. Receiving hourly estimates with a prediction horizon of $H_2 \cdot T = 24$ h, subcontroller 2 decides how to set the controlled parameters $\mathbf{R}(t)$ during the next day. At this, the efficiency of the energy utilization is optimized by, e.g., exploiting the sunlight directly when available.

The advantages of the hierarchical control model can be summarized as follows (see also experimental results in Section 7.2): First, by virtue of its worst-case design, the upper control layer avoids depletion of the energy storage and increases the robustness of the overall system. Second, the control formulation renders manual tuning of the final state constraint unnecessary (cp. (8)), resulting in an automatical optimization and stabilization of the system behavior. Third, by decoupling the control problem into two subproblems, the complexity of the online controller is reduced significantly. In the following, the two control layers will be described in detail.

7.1.1 Long-Term Performance Optimization

The optimization objective of subcontroller 1 is to maximize the minimal available energy $E_D(t)$ per day. In other words, we would like to guarantee sustainable operation in a worst-case sense, similar as in objective (7). For this reason, an energy prediction for a worst-case scenario is required. Unlike most of the energy estimation algorithms presented in related work, we would like to determine the *minimal accumulated energy* for the next days, which we denote as $\widetilde{E}_1(t,k)$. The respective algorithm has been presented in [15]. Note that contrary to the definition of $\widetilde{E}(t,k)$ in Section 5.3, $\widetilde{E}_1(t,k)$ denotes the *cumulated* energy for the next k intervals.

For a small number of days, the worst-case energy prediction algorithm gives pessimistic predictions $\widetilde{E}_1(t,k)$, which is reasonable if continuous operation of a sensor node has to be guaranteed. During this time, an underestimation of the actually scavenged energy may happen. In a long-term perspective of a few weeks, however, the worst-case energy prediction algorithm begins to predict the average incoming energy just as conventional estimation algorithms.

The linear optimization problem underlying subcontroller 1 is given below. With the help of the worst-case energy predictions $\widetilde{E}_1(t,k)$, it is now possible to maximize the smallest energy $E_D(t+k\cdot L_1)$ for all $0 \le k < N_1$ in the prediction horizon. However, only the first energy $E_D(t)$ is passed to subcontroller 2 and will be used during the next day.

$$\begin{aligned}
&\text{maximize } J = \mu \text{ subject to :} && \text{(subcontroller 1)}\\
&E_D(t+k\cdot L_1) \ge \mu && \forall 0 \le k < N_1\\
&E_C(t+k\cdot L_1) = E_C(t) + \widetilde{E}_1(t,k) - \sum_{j=1}^{k}(E_D(t+(j-1)\cdot L_1)) && \forall 1 \le k \le N_1\\
&0 \le E_C(t+k\cdot L_1) \le E_{\max} && \forall 1 \le k \le N_1
\end{aligned}$$

7.1.2 Short-Term Power Saving

As already indicated, the prediction horizon of the lower control layer spans $H_2 \cdot T = 24$ h. For a general control problem and a prediction $\widetilde{E}_2(t,k)$, the final state constraint of the lower control layer can be set to $\alpha(t) = E_D - \sum_{i=0}^{N_2-1} \widetilde{E}_2(t,i)$ in order to force the system to spend energy E_D during the next day. Hence, the final state constraint $\alpha(t)$ is calculated automatically by the upper layer.

In this paper, we focus on a concrete application (Example III) where energy losses due to storage efficiency η shall be minimized. Nevertheless, our methods are also applicable to other power saving techniques, like, e.g., DVS, energy-efficient packet scheduling for wireless links or applications with data compression.

For the optimization problem with nonideal energy storage in LP III, we have to ensure that the sensor node uses no more than energy E_D during the next day. The only degree of freedom to be exploited by subcontroller 2 is *when* to transmit the images with rate $R_2(t)$ in order to *save* as much energy as possible. Thus, we want to maximize the energy $E_C(t + L_2 \cdot N_2)$ at the end of the day. We formulate the linear program to be solved by subcontroller 2 as follows:

$$\begin{aligned}
&\text{maximize } J = E_C(t + L_2\ N_2) \text{ subject to :} && \text{(subcontroller 2)}\\
&\quad R_1 = \tfrac{\eta \cdot E_D(t)}{N_2 \cdot (e_1 + e_2)}\\
&\sum_{j=0}^{N_2-1} R_2(t + j \cdot L_2) = \tfrac{\eta \cdot E_D(t)}{e_1 + e_2}\\
&R_2(t + k \cdot L_2) \geq 0 && \forall 0 \leq k < N_2\\
&E_C(t + k\ L_2) = E_C(t) + \textstyle\sum_{j=0}^{k-1} \widetilde{E}_2(t,j) -\\
&e_1\ L_2\ k\ R_1 - e_2\ L_2 \sum_{j=0}^{k-1} R_2(t + j\ L_2) -\\
&\quad (1-\eta) \textstyle\sum_{j=0}^{k-1} \lambda(j) && \forall 1 \leq k \leq N_2\\
&\lambda(j) \geq \widetilde{E}_2(t,j) - e_1\ L_2\ R_1 -\\
&\quad e_2\ L_2\ R_2(t + j \cdot L_2) && \forall 0 \leq j < N_2\\
&M(t + k \cdot L_2) = M(t) +\\
&\textstyle\sum_{j=0}^{k-1}(R_1(t + j \cdot L_2) - R_2(t + j \cdot L_2)) && \forall 1 \leq k \leq N_2\\
&0 \leq M(t + k \cdot L_2) \leq M_{\max} && \forall 1 \leq k \leq N_2
\end{aligned}$$

To exploit the typical profile of solar energy during a day, an energy predictor $\widetilde{E}_2(t,k)$ is needed which predicts the most probable energy values for the next hours. In contrast to $\widetilde{E}_1(t,k)$, the hourly estimation $\widetilde{E}_2(t,k)$ should keep a history of harvested energies and use a representative, *average* value as predictor.

7.2 Optimization with Nonideal Energy Storage (Cont. Example III)

Next, we subdivide the control problem in LP III into two hierarchically structured subproblems as described in the previous section. Concerning the upper control layer, we chose a prediction horizon of 30 days, where each interval spans one day. We set the aging of old data $\gamma = 100$ for the worst-case prediction algorithm $\widetilde{E}_1$. To avoid unnecessary control overhead, subcontroller 1 is not activated every $T = 5$ minutes, but only once per day. For the lower control layer, the prediction algorithm in [14] has been used, again with the same parameters. As for the single controller, the prediction horizon of the lower control layer spans one day, with six intervals of 4 hours length.

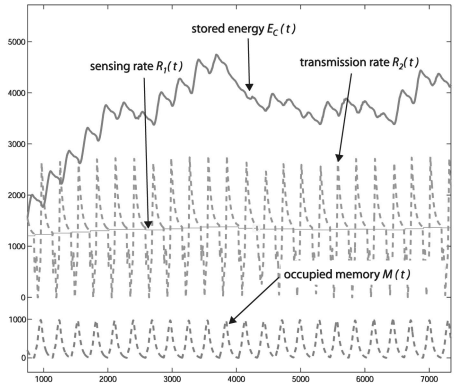


Fig. 9. Hierarchical system, multiparametric programming, worst-case prediction $\widetilde{E}_1(t,k)$, and average prediction $\widetilde{E}_2(t,k)$.

Table 1 summarizes the design parameters and the implementation overheads of both control designs. In terms of storage demand, the hierarchical approach yields a substantial reduction of the storage requirement of 83.0 percent. Since subcontroller 1 is only activated once per day, one has to divide its computation demand by L_1 to obtain the number of operations per interval T. More precisely, assuming that *all* regions have to be tested before evaluating the control law, 3,689 operations have to be executed at every 288th multiply of the basic time interval T. Hence, only 12.8 operations plus 4,829 operations of subcontroller 2 have to be executed at every time step T. This yields a reduction of the worst-case computation demand of 91.0 percent compared to a single controller.

In dependence on how the control laws are implemented in practice, the number of necessary operations will be smaller than the worst-case values in Table 1. As has been shown in [14], the *average* number of region tests can be reduced by exploiting the fact that some regions are more frequently activated then others. For the probabilistic region checking method proposed in [14], we experienced that the average number of regions tests is proportional to the total number of regions N_{CR} (and hence, also to the number of operations in the worst case). Since the worst-case number of region tests is independent of the implementation strategy used, we opted for this metric to indicate the computational effort. Nevertheless, our methods simultaneously reduce the average as well as the worst-case number of operations.

Fig. 9 depicts the performance of the hierarchical control approach. The simulation result is plotted for 22 days, covering also the seven days of simulation period of the previous experiment. This time, the controllers successfully stabilize the sampling rate at $r_1 \approx 1{,}300$. Depending on the season, we found that rate r_1 may slightly increase (in summer) or decrease (during winter). In any case, subcontroller 1 prevents the system to run out of energy and simultaneously maximizes the amount of energy E_D per day. For a small number of days, like, e.g., one-four days, the predictor $\widetilde{E}_1$ is generating rather pessimistic estimates, accounting for a couple of cloudy days. For several days or weeks, the accumulated energy is very likely converging toward an average value, like the estimate $\widetilde{E}_2$ used for subcontroller 2. Exploiting this mechanism, subcontroller 1

autonomously regulates the offset $\alpha(t)$ of the final state constraint and optimizes the long-term system behavior.

8 Approximate Control Design

In this section, we present an algorithm for approximative multiparametric linear programming. As a matter of fact, only a few such algorithms have been proposed in the literature so far (among them, e.g., the algorithms in [32] and [33]). The algorithm is presented in Sections 8.1 and 8.2, and we evaluate the algorithm for the exemplary sensor application (Example III) from the previous sections.

8.1 An Approximative MP Linear Programming Algorithm

It is well known in the control community that the size of the explicit solution obtained by multiparametric linear programming grows quickly if the complexity of the control problem increases. Indeed, already a moderate number of control variables and parameters may result in a huge number N_{CR} of critical regions. At this, adjacent regions are often characterized by almost identical control laws. This circumstance, however, has negligible impact on the resulting control profile in many cases. Rather, numerous regions N_{CR} entail an high overhead in terms of storage requirement, running time as well as energy consumption.

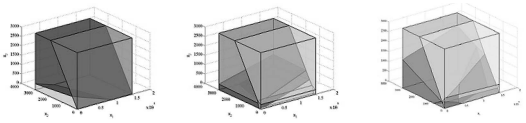
We have designed a new algorithm for approximative multiparametric linear programming. The basic idea was

- to take a large number of samples $\mathbf{X}_i$ of the state space of $\mathbf{X}$ (compare (9)),
- to solve a linear program for each sample $\mathbf{X}_i$ to obtain the respective optimal solution $\mathbf{U}_i^*$,
- to find a (preferably simple) fitting function $\hat{\mathbf{U}}^*(\mathbf{X})$ for the multidimensional data $(\mathbf{X}_i, \mathbf{U}_i^*)$,
- and finally, use $\hat{\mathbf{U}}^*(\mathbf{X})$ (which has been calculated offline) as approximation for $\mathbf{U}^*(\mathbf{X})$ in the online case.

As fitting algorithm, we opted for the algorithm proposed in [34]. As described in [16], we derive an explicit form for the approximated control rates $\hat{\mathbf{U}}^*(\mathbf{X})$ as a function of the current state $\mathbf{X}$:

$$\hat{\mathbf{U}}^*(\mathbf{X}) = \hat{\mathbf{A}}_j\mathbf{X} + \hat{\mathbf{B}}_j \quad \text{if } \hat{\mathbf{H}}_j\mathbf{X} \leq \hat{\mathbf{K}}_j, \quad j = 1, \ldots, \hat{N}_{CR}. \quad (12)$$

The approximated control law in (12) can now be used in an online controller instead of the exact solution in (11). Since the convex fitting algorithm in [34] allows to tune the number $\hat{N}_{CR}$ of critical regions, one may chose a smaller number of regions $\hat{N}_{CR} < N_{CR}$ to reduce the complexity of the control problem. The fitting algorithm then attempts to create a smaller polyhedral partition which minimizes the least-square error of the objective value function. In comparison to the optimal multiparametric solution, the fitting algorithm seems to merge smaller regions and reshapes the geometry of the partition. Albeit no performance guarantees of the heuristic algorithm are given in [34], it turns out that the algorithm performs well and produces suitable approximations, both in [34] and also in our experiments, as we will see in the next section.



(a) (b) (c)

Fig. 10. Illustration of approximated polyhedral partitions for subcontroller 2. Cut through $\widetilde{E}_2(t,1) = 1{,}200$, $\widetilde{E}_2(t,2) = 1{,}000$, $\widetilde{E}_2(t,3) = 0$, $\widetilde{E}_2(t,4) = 100$, and $\widetilde{E}_2(t,5) = 500$. (a) $\hat{N}_{CR} = 3$. (b) $\hat{N}_{CR} = 5$. (c) $\hat{N}_{CR} = 10$.

8.2 Optimization with Nonideal Energy Storage (Cont. Example III)

In Section 7, both controllers involved in the hierarchical control design have been computed using the optimal partitions of the parameter space obtained by multiparametric programming. In this section, we derive and test suboptimal, approximate control laws using the algorithm presented in previous section.

For subcontroller 1, we sampled the state space $\mathbf{X}_1 = (E_C(t), \widetilde{E}_1(t,0), \ldots, \widetilde{E}_1(t,29))$ using $N_S = 1{,}000$ samples. For subcontroller 2, $N_S = 2{,}000$ samples have been taken from the state space $\mathbf{X}_2 = (E_D(t), M(t), \widetilde{E}_2(t,0), \ldots, \widetilde{E}_2(t,5))$. In Fig. 10, some exemplary partitions of $\mathbf{X}_2$ are displayed. In dependence of the choice of the three-dimensional cut, not all $\hat{N}_{CR}$ partitions may be visible in Figs. 10a, 10b, and 10c.

We denote $\hat{R}_1$, $\hat{R}_2$, $\hat{E}_C$, and $\hat{M}$ the rate and state variables obtained if approximate control laws are applied. Let us also define the rectifier function $[\Delta E\,]^+$ as follows:

$$[\Delta E\,]^+ = \begin{cases} \Delta E, & \text{if } \Delta E \geq 0, \\ 0, & \text{if } \Delta E < 0. \end{cases} \quad (13)$$

Using this notation, we denote the average efficiency of the energy utilization

$$\eta_{avg} = 1 - \frac{(1-\eta) \cdot \sum_t [E_S(t) - e_1 R_1(t) - e_2 R_2(t)]^+}{\sum_t E_S(t)}. \quad (14)$$

In our simulations with the approximated control laws, the actual optimization objective of regulating the sensing rate is met almost as well as the exact solution. Using $\hat{N}_{CR} = 4$ for both subcontroller 1 and subcontroller 2, the maximal derivation of $\hat{R}_1$ from R_1 is 1.52 percent. Obviously, the approximated algorithm manages to save even slightly more energy then its exact counterpart. Apparently, the approximated control law even yields a gain of 0.75 percent of the average efficiency η_{avg}. The stored energy $\hat{E}_C$ is varying up to 11.57 percent from E_C. However, the peak of $\hat{E}_C$ is just 4.03 percent above the one of E_C. That is, the capacity of the energy storage is required to be approximately 5 percent higher if the system is controlled by an approximated algorithm.

Showing a comparable performance during runtime, the main advantage of the approximated control laws becomes obvious when considering their complexity. According to Table 2, the storage demand is significantly reduced by 92.44 percent compared to the optimal solution. In terms of worst-case computation demand, the reduction even amounts 98.55 percent. Table 2 also outlines the results for a second low-complexity approximation. It exhibits a slightly lower efficiency η_{avg}. On the other hand, the second approximation closely matches the optimal control rate R_1.

TABLE 2
Comparison of Multiparametric and Approximate-mp Control Design

control design	$\max_t \left\lvert \frac{\hat{R}_1(t)}{R_1(t)} - 1 \right\rvert$	$\max_t \left\lvert \frac{\hat{E}_C(t)}{E_C(t)} - 1 \right\rvert$	η_{avg}	N_{CR} (or $\hat{N}_{CR}$)	storage (real numbers)	ops (worst case)
MP, subc. 1	0%	0%	93.00%	30	1920	3689
subc. 2				161	2898	4829
approx., subc. 1	1.52%	11.57%	93.75%	4	256	308
subc. 2				4	108	69
approx., subc. 1	0.82%	5.47%	92.97%	4	256	308
subc. 2				9	243	173

In summary, we can state that the approximate solutions to the multiparametric control problem do not necessarily entail a degraded performance in terms of the controlled parameters. Indeed, in most cases, we could find simple but useful approximations for the underlying control problems. Besides the significant complexity reduction, there is another advantage of the proposed technique that should be mentioned. For highly complex control problems consisting of several thousands of regions N_{CR}, conventional solvers may be unable to find the optimal polyhedral partition. We experienced that sometimes, even no solution can be found at all. Here, our method turned out to be helpful to find a reasonable solution at all.

9 Hardware Implementation Issues

In this section, we demonstrate how the proposed control laws can be implemented efficiently and evaluate the implementation overhead on a BTnode [35]. For this purpose, we show measurements of two implementations of exemplary controllers which have been introduced and discussed in the last sections.

9.1 Average Computation Demand

In general, the identification of the active region j dominates the linear function evaluation of a control law (11) in terms of time and energy consumption. In the worst case, for all N_{CR} regions, a matrix multiplication has to be performed in order to identify the active region j at time t. However, the identification of the active region j can be simplified since usually, only a subset of all regions N_{CR} is activated and some of the regions in this subset are activated more frequently than others. This observation can be exploited by starting the search always with the region with the highest statistical occurrence, continue with the second highest, and so on. To this end, an algorithm should maintain a list of regions j ordered by their frequencies which is updated every time step t.

For the memory optimization problem in LP II in Section 6.3, we found that only 7 of 39 regions were used during the whole simulation period. Using probabilistic region testing as described above, the critical term of the form $\mathbf{H}_j \cdot \mathbf{X} \leq \mathbf{K}_j$ is only evaluated ≈ 1.45 times at time t, taking the average over the whole simulation period.

Fig. 11 displays the measured power consumption of a BTnode [35]. At first, the current energy level $E_C(t)$ of the battery and the scavenged energy $E_S(t)$ are determined via two A/D conversions, which mark the two major peaks in plot 11. Focusing on the overhead of the proposed controller, we omit predicting the future energies $\widetilde{E}(t,i)$. Instead, an average situation is displayed where the region with the second highest frequency is the active one. Subsequently, the optimal control output for this region is calculated. It becomes evident that the computations leading to the actual control actions take as long as the two A/D conversions (≈ 2 ms).

9.2 Worst-Case Computation Demand and Storage Demand

For some applications, it is also important to consider the worst-case computation demand. Here, worst-case refers to the situation where the currently active region is the *last* one to be checked. For example, for the hierarchical control design presented in Section 7, we are interested in the worst-case computation time and energy consumption for a practical implementation.

Since the daily energy estimation and subcontroller 1 are only activated once per day, their influence on the computation demand is negligible. In our test implementation on a BTnode (see Fig. 12), we measured a worst-case computation time <190 ms for subcontroller 2. The evaluation of this control law is only done every T time units, which may be every 5 or 10 minutes. For the example application introduced in Section 6.3, already one instantiation of the sensing and transmission task consumes significantly more energy than one worst-case execution of the control algorithm. Note that the architecture of the MICA2 sensor node in this example is almost identical to the one of the BTnode used in our experiments. If we assume that, e.g., the

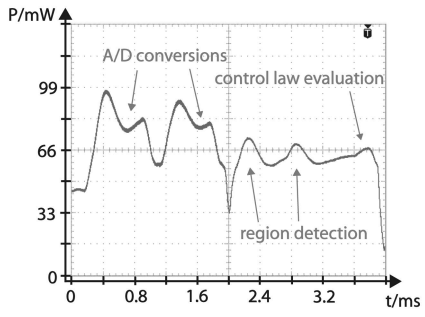


Fig. 11. Power consumption of a control law evaluation for the memory optimization problem (LP II) on a BTnode.

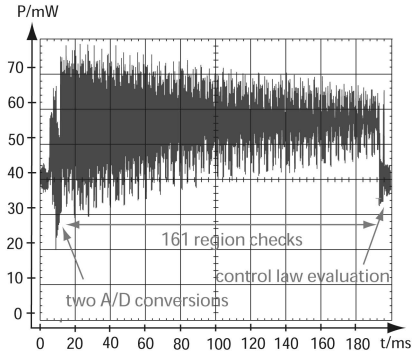


Fig. 12. Worst-case power consumption of subcontroller 2 on a BTnode.

solar panels of a sensor node are dimensioned large enough to allow for several instantiations within an interval T, the energy overhead for evaluation of the control laws can be neglected compared to the energy consumption of the actual application during the interval T.

Using multiparametric programming techniques instead of solving the optimization problem online basically means a shift from *computation* to *storage*. Thus, one has to ensure that the storage of the control law does not exceed the limits of the embedded system. From the example applications presented in this paper, all control laws could be implemented on a BTnode. For example, the controller for LP III presented in Section 6.4 requires the storage of the $N_{CR} = 1{,}049$ regions and corresponding control laws. In terms of physical memory, this amounts to 55.3 Kbyte using a 16 bit integer representation per coefficient. Using the hierarchical control approach as presented in Section 7, we could reduce the storage demand to 9.4 Kbyte. This reduction makes the control law implementable on commonly used sensor nodes like the Tmote Sky [36], which features 48 Kbyte Flash ROM or the BTnode [35], which exhibits 128 Kbyte Flash ROM. In summary, using the tools and methods described in this paper, we believe that controllers for a large class of optimization problems can be implemented on commonly used sensor network platforms.

10 Concluding Remarks

In this paper, we present a framework of tools and methods to optimize the performance of energy harvesting systems using multiparametric programming techniques. On the one hand, we have successfully designed and evaluated an hierarchical control design which is tailored to the requirements of solar-powered sensor nodes. Due to its worst-case design, the upper control layer prevents the sensor node from running out of energy. On the lower control layer, we showed how power saving techniques can work more efficiently if stability of the operation is guaranteed by the layer above. Simultaneously, the reformulation of the control problem into two subproblems yields a significant reduction in online complexity. On the other hand, we propose a new algorithm for approximative multiparametric programming. The resulting control laws are rough approximations of the optimal solution and reduce the involved online overhead substantially. An experimental setup reveals that the achieved performance may be comparable to the optimal solution. All methods are supported by extensive simulations results which are based on long-term measurements of solar energy. Measurements of our controllers running on a sensor node together with a detailed analysis of the implementation overhead show that our algorithms can be implemented efficiently.

Acknowledgments

This 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. In addition, this research has been supported by the European Network of Excellence on Embedded System Design ARTISTDesign.

References

[1] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and Yield in a Volcano Monitoring Sensor Network," *Proc. Seventh Symp. Operating Systems Design and Implementation (OSDI '06)*, pp. 381-396, 2006.

[2] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a Sensor Network Expedition," *Proc. First European Workshop Sensor Networks (EWSN)*, pp. 307-322, 2004.

[3] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A Sensor Network for Hostile Environments," *Proc. First IEEE Comm. Soc. Conf. Sensor and Ad Hoc Comm. and Networks*, 2004.

[4] K. Roemer and F. Mattern, "The Design Space of Wireless Sensor Networks," *IEEE Wireless Comm.*, vol. 11, no. 6, pp. 54-61, Dec. 2004.

[5] S. Roundy, D. Steingart, L. Frechette, P.K. Wright, and J.M. Rabaey, "Power Sources for Wireless Sensor Networks," *Proc. First European Workshop Wireless Sensor Networks (EWSN '04)*, pp. 1-17, Jan. 2004.

[6] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M.B. Srivastava, "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems," *Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN '05)*, pp. 457-462, Apr. 2005.

[7] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel, "Permadaq: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes," *Proc. Eighth ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN/SPOTS '09)*, pp. 265-276, 2009.

[8] J. Hsu, A. Kansal, J. Friedman, V. Raghunathan, and M. Srivastava, "Energy Harvesting Support for Sensor Networks," *Proc. SPOTS Track at Information Processing in Sensor Networks (IPSN '05)*, 2005.

[9] X. Jiang, J. Polastre, and D.E. Culler, "Perpetual Environmentally Powered Sensor Networks," *Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN '05)*, pp. 463-468, Apr. 2005.

[10] C. Park and P. Chou, "Ambimax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes," *Proc. Int'l Conf. Sensor and Ad Hoc Comm. and Networks (SECON '06)*, vol. 1, 2006.

[11] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh, "Lance: Optimizing High-Resolution Signal Collection in Wireless Sensor Networks," *Proc. Sixth ACM Conf. Embedded Network Sensor Systems (SenSys '08)*, pp. 169-182, 2008.

[12] K. Klues, V. Handziski, C. Lu, A. Wolisz, D. Culler, D. Gay, and P. Levis, "Integrating Concurrency Control Energy Management in Device Drivers," *Proc. ACM SIGOPS*, pp. 251-264, 2007.

[13] A. Kansal, D. Potter, and M.B. Srivastava, "Performance Aware Tasking for Environmentally Powered Sensor Networks," *Proc. SIGMETRICS '04*, pp. 223-234, June 2004.

[14] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive Power Management in Energy Harvesting Systems," *Proc. Conf. Design, Automation and Test in Europe (DATE '07)*, pp. 773-778, 2007.

[15] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Robust and Low Complexity Rate Control for Solar Powered Sensors," *Proc. Conf. Design, Automation and Test in Europe (DATE '08)*, Mar. 2008.

[16] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Approximate Control Design for Solar Driven Sensor Nodes," *Proc. 11th Int'l Conf. Hybrid Systems: Computation and Control (HSCC '08),* pp. 634-637, Apr. 2008.
[17] A. Kansal, J. Hsu, S. Zahedi, and M.B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *Trans. Embedded Computing Systems,* vol. 6, no. 4, p. 32, 2007.
[18] C.M. Vigorito, D. Ganesan, and A.G. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," *Proc. Fourth Ann. IEEE Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '07),* pp. 21-30, 2007.
[19] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-Time Scheduling for Energy Harvesting Sensor Nodes," *Real-Time Systems,* vol. 37, pp. 233-260, Kluwer Academic Publishers, 2007.
[20] S. Liu, Q. Qiu, and Q. Wu, "Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting," *Proc. Conf. Design, Automation and Test in Europe (DATE '08),* Mar. 2008.
[21] C. Rusu, R.G. Melhem, and D. Mosse, "Multi-Version Scheduling in Rechargeable Energy-Aware Real-Time Systems," *Proc. 15th Euromicro Conf. Real-Time Systems (ECRTS '03),* pp. 95-104, July 2003.
[22] C. Moser, J.-J. Chen, and L. Thiele, "Reward Maximization for Embedded Systems with Renewable Energies," *Proc. IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA '08),* pp. 247-256, 2008.
[23] M. Morari and J. Lee, "Model Predictive Control: Past, Present and Future," *Computers and Chemical Eng.,* vol. 23, no. 4, pp. 667-682, 1999.
[24] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan, "Adaptive Duty Cycling for Energy Harvesting Systems," *Proc. 2006 Int'l Symp. Low Power Electronics and Design (ISLPED '06),* pp. 180-185, 2006.
[25] A. Bemporad, F. Borrelli, and M. Morari, "Model Predictive Control Based on Linear Programming—The Explicit Solution," *IEEE Trans. Automatic Control,* vol. 47, no. 12, pp. 1974-1985, Dec. 2002.
[26] F. Borrelli, A. Bemporad, and M. Morari, "A Geometric Algorithm for Multi-Parametric Linear Programming," *J. Optimization Theory and Applications,* vol. 118, no. 3, pp. 515-540, Sept. 2003.
[27] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," http://control.ee.ethz.ch/~mpt/, 2004.
[28] Sunergy, "Recordings of Solar Light Intensity from 08/08/2005 to 04/09/2005, Datasheet," http://sunergy.dmclf.org/cms/, June 2006.
[29] M. Rahimi, R. Baer, O.I. Iroezi, J.C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In Situ Image Sensing Interpretation in Wireless Sensor Networks," *Proc. Third Int'l Conf. Embedded Networked Sensor Systems (SenSys '05),* pp. 192-204, 2005.
[30] "Crossbow Technology," http://www.xbow.com, 2009.
[31] "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, Mar. 2007.
[32] C. Filippi, "An Algorithm for Approximate Multiparametric Linear Programming," *J. Optimization Theory and Applications,* vol. 120, no. 1, pp. 73-95, Jan. 2004.
[33] C. Jones, M. Baric, and M. Morari, "Multiparametric Linear Programming with Applications to Control," *European J. Control,* vol. 13, pp. 152-170, Mar. 2007.
[34] A. Magnani and S. Boyd, "Convex Piecewise-Linear Fitting," *Optimization and Eng.,* vol. 10, no. 1, pp. 1-17, Mar. 2008.
[35] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald, "Next-Generation Prototyping of Sensor Networks," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys '04),* pp. 291-292, Nov. 2004.
[36] Moteiv Corporation, "Tmote Sky—Ultra Low Power IEEE 802.15.4 Compliant Wireless Sensor Module, Datasheet," http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf, June 2006.

Clemens Moser received the BSc and Diplom-Ingenieur degrees in electrical engineering and information technology from the Technical University of Munich, in 2003 and 2004, respectively and the PhD degree from the Swiss Federal Institute of Technology (ETH) Zurich in March 2009. He is currently a postdoctoral researcher in the Computer Engineering and Networks Laboratory of ETH Zurich. For his diploma thesis in 2004, he joined DoCoMo Euro-Labs to work on topology aspects of wireless multihop networks.

Lothar Thiele received the Diplom-Ingenieur and Dr-Ing degrees in electrical engineering from the Technical University of Munich in 1981 and 1985, respectively. He joined ETH Zurich, Switzerland, as a full Professor of computer engineering, in 1994. He is leading the Computer Engineering and Networks Laboratory of ETH Zurich. His research interests include models, methods, and software tools for the design of embedded systems, embedded software, and bioinspired optimization techniques. In 1986, he received the Dissertation Award of the Technical University of Munich; in 1987, the Outstanding Young Author Award of the IEEE Circuits and Systems Society; in 1988, the Browder J. Thompson Memorial Award of the IEEE; and in 2000-2001, the IBM Faculty Partnership Award. In 2004, he joined the German Academy of Natural Scientists Leopoldina. In 2005, he was the recipient of the Honorary Blaise Pascal Chair of University Leiden, The Netherlands. He is a member of the IEEE.

Davide Brunelli received the master's degree (summa cum laude) and the PhD degree from the University of Bologna, Italy, in 2002 and 2007, respectively, both in electrical engineering. In 2005, 2006, and 2007, he was an academic guest at the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. He is currently the holder of a postdoctoral position with the University of Bologna. His research interests concern the development of new techniques of energy scavenging for WSNs and embedded systems, optimization of low-power and low-cost WSNs, interaction and design issues in embedded personal and wearable devices, and pervasive and ubiquitous computing.

Luca Benini (S'94-M'97-SM'04-F'07) received the PhD degree in electrical engineering from Stanford University, California, in 1997. He is a full professor in the Department of Electronics, Computer Sciences and Systems (DEIS), University of Bologna, Italy. He is also the holder of a visiting faculty position with the Ecole Polytechnique Federale de Lausanne, Switzerland. He has published more than 350 papers in peer-reviewed international journals and conference proceedings, four books, and several book chapters. His research interests are in the design of system-on-chip platforms for embedded applications. He is also active in the area of energy-efficient smart sensors and WSNs. He is an associate editor of the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* and the *ACM Journal on Emerging Technologies in Computing Systems*. He is a fellow of the IEEE and the IEEE Computer Society.