Computing Optimal Schedules of Battery Usage in Embedded Systems

Marijn Jongerden, Alexandru Mereacre, Henrik Bohnenkamp, *Member, IEEE*, Boudewijn Haverkort, *Fellow, IEEE*, and Joost-Pieter Katoen

***Abstract*—The use of mobile devices is often limited by the battery lifetime. Some devices have the option to connect an extra battery, or to use smart battery-packs with multiple cells to extend the lifetime. In these cases, scheduling the batteries or battery cells over the load to exploit the recovery properties of the batteries helps to extend the overall systems lifetime. Straightforward scheduling schemes, like round-robin or choosing the best battery available, already provide a big improvement compared to a sequential discharge of the batteries. In this paper, we compare these scheduling schemes with the optimal scheduling scheme produced with two different modeling approaches: an approach based on a priced-timed automaton model (implemented and evaluated in Uppaal Cora), as well as an analytical approach (partly formulated as nonlinear optimization problem) for a slightly adapted scheduling problem. We show that in some cases the results of the simple scheduling schemes (round-robin, and best-first) are close to optimal. However, the optimal schedules, computed according to both methods, also clearly show that in a variety of scenarios, the simple schedules are far from optimal.**

***Index Terms*—Batteries, embedded systems, lifetime optimization, scheduling.**

I. Introduction

MANY autonomous devices mostly rely on batteries as power supply. The capacity of batteries is finite and, together with the current drawn from them, determines the time during which the device can be used. The battery is at the end of its lifetime when it can not deliver the desired current when it is needed.

To extend the usability of such devices, frequently several batteries are provided. Mostly, these batteries are used in sequential order, the next one when the previous one has reached the end of its lifetime. Whereas this clearly is an effective approach to prolong device lifetimes, it is not an efficient one. Efficiency, however, is very desirable: batteries are usually heavy, compared to the devices they power, and their number should thus be kept small. Furthermore, as we will see below, using multiple batteries in a nonsequential way, the available energy as stored in the battery is used more effectively, and less "unused battery capacity" is thrown away.

The question to be answered is whether it is possible to use batteries in a more clever way than just sequentially, i.e., whether there are better *schedules* for battery use. A reason why this might be the case is given by the two inherent properties of batteries, the so-called *rate-capacity effect* and the *recovery effect*. The first property is the phenomenon that a battery delivers in total a smaller charge during its lifetime when it is discharged with a high current, compared to a lower one. The second property is the phenomenon that a battery, when not used or only with a low current, can recover and procure some additional charge. These two effects are captured quite precisely in a simple mathematical model, the *Kinetic Battery Model* (KiBaM) of Manwell and McGowan [10]–[12]. In particular, if the recovery effect can be exploited to draw more charge from a battery, compared to the purely sequential case, the lifetime of the powered device is increased.

This paper, which is an extended version of [9], investigates methods to find schedules for the use of batteries for a given load function which maximize the lifetime of the device. Two different approaches are followed, which both are based on the KiBaM. First, a discretized version of the KiBaM, together with a predefined load is modelled as *linearly priced timed automata* (LPTA) [3], [4]. Then, the model-checker Uppaal Cora [1] is used to derive schedules that maximize the system lifetime. This approach is described in more detail in [9]. Here, some extra results as a first step towards random loads are given. In the second approach, which is new, the KiBaM is investigated analytically, using, among others, nonlinear optimization techniques, to solve the scheduling problem.

Both approaches yield excellent results. The LPTA approach, on one hand, provides schedules in a fully algorithmic way which come very close to the optimal lifetime, when scheduling moments are predefined (e.g., at load changes). The analytic approach, on the other hand, shows that battery scheduling is actually trivial when the moments when batteries are switched can be chosen freely. The system lifetime is not influenced by the schedule, but actually by the number of switching points that are allowed: the more, the better. A further result is that the optimal system lifetime for M batteries and a given load function can be

Manuscript received November 10, 2009; revised February 26, 2010 and April 16, 2010; accepted May 18, 2010. Date of publication June 21, 2010; date of current version August 06, 2010. The work of A. Mereacre, H. Bohnenkamp, and J.-P. Katoen was supported by the DFG Research Training Group 1295 AlgoSyn and the EU FP7 Project ICT-FP7-STREP-214755 (QUASIMODO). The work of M. Jongerden and B. Haverkort was supported by the ITEA2 GEODES Project 07013 (grant: PNEI081011). All were partially supported by DFG/NWO Bilateral Research Program ROCKS. Paper no. TII-09-11-0320.

M. Jongerden is with the CTIT, University of Twente, 7500 AE Enschede, The Netherlands (e-mail: jongerdenmr@ewi.utwente.nl).

A. Mereacre, H. Bohnenkamp, and J.-P. Katoen are with Software Modeling and Verification, RWTH Aachen University, 52056 Aachen, Germany (e-mail: mereacre@cs.rwth-aachen.de; henrik@cs.rwth-aachen.de; katoen@cs.rwth-aachen.de).

B. Haverkort is with the CTIT, University of Twente, 7500 AE Enschede, The Netherlands, and also with the Embedded Systems Institute, 5600 MB Eindhoven, The Netherlands (e-mail: brh@ewi.utwente.nl; boudewijn.haverkort@esi.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2010.2051813

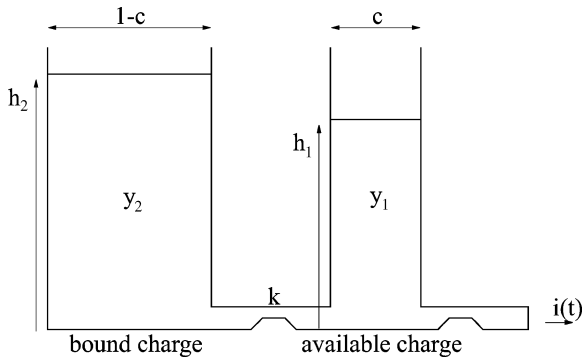

Fig. 1. KiBaM.

expressed analytically, i.e., there is a natural upper bound on the system lifetime which can not be exceeded by even more clever scheduling.

Structure of the Paper: In Section II, we introduce the necessary preliminaries of the kinetic battery model and of linearly-priced timed automata. In Section III, we describe how we derive battery schedules from an LPTA model. In Section IV, we compare the obtained optimal with some deterministic schedules. In Section V, we show optimal schedules can be obtained analytically. In Section VI, we discuss the LPTA and the analytic approach. In Section VII, we discuss related work and conclude in Section VIII.

II. Preliminaries

A. Kinetic Battery Model

The battery model we use is the Kinetic Battery Model (KiBaM) [10]–[12], which describes two essential nonlinear properties, the rate-capacity effect and the recovery effect [8]. The former is the effect that less charge can be drawn from a battery when the discharge current is increased. The latter is the effect that a nearly discharged battery can recover in a period with no or low current and make some more charge available again.

In the KiBaM the battery charge is distributed over the *available-charge well* (ACW) and the *bound-charge well* (BCW) (cf. Fig. 1). A fraction $y_1(0) = c \cdot C$ of the initial total capacity C is put in the ACW, and a fraction $y_2(0) = (1-c) \cdot C$ in the BCW. The ACW supplies electrons directly to the load $i(t)$, whereas the BCW supplies electrons only to the ACW. The charge flows from the BCW to the ACW through a "valve," where the flow rate depends on a *conductance parameter*, k, and the *height difference* $h_2(t) - h_1(t)$ between the two wells, where the heights of the two wells are given by: $h_1(t) = y_1(t)/c$ and $h_2(t) = y_2(t)/(1-c)$. The change of the charge in both wells is then described by the following system of differential equations:

$$\begin{cases} \dot{y}_1(t) = -i(t) + k\left(h_2(t) - h_1(t)\right) \\ \dot{y}_2(t) = -k\left(h_2(t) - h_1(t)\right). \end{cases} \tag{1}$$

The battery is considered empty at time $t \in \mathbb{R}_{>0}$ when there is no charge left in the available charge well, i.e., $y_1(t) = 0$.

A coordinate transform, applied to (1), makes the handling of the KiBaM easier. Obviously, the height difference $\delta(t) = h_2(t) - h_1(t)$ plays a major role, which becomes one of the coordinates after the transformation. The other is the total charge $\gamma(t) = y_1(t) + y_2(t)$. The transformed differential equations become

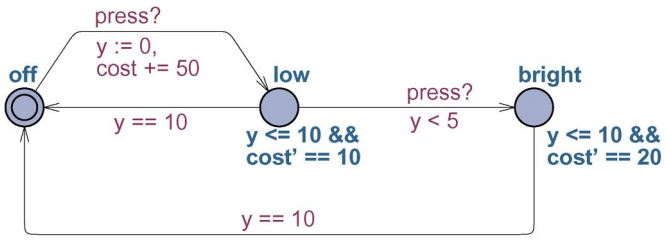

Fig. 2. Model of lamp.

$$\begin{cases} \dot{\delta}(t) = \frac{i(t)}{c} - k'\delta(t) \\ \dot{\gamma}(t) = -i(t) \end{cases} \tag{2}$$

where $k' = k/(1-c)c$. The initial conditions change to $\delta(0) = 0$ and $\gamma(0) = C$. The condition for the battery to be empty is $\gamma(t) = (1-c)\delta(t)$.

B. Discretization of the KiBaM

In this section a discretized version of the KiBaM is introduced, called dKiBaM. In the dKiBaM, the discharge and recovery processes governing the KiBaM are separated. Time is discretized in time steps of size T. Within a time step the discharge current is assumed to be constant. For a constant current I, the total charge decreases linearly. The total charge is discretized in N parts of size $\Gamma = C/N$. It takes $\Gamma/(I \cdot T)$ time units to decrease the total charge with one charge unit at rate I. At the same time, the discharge with current I will increase the height difference with Γ/c. This will be the step size of the discretization of the height difference. Once some charge is drawn from the ACW, charge will start to flow from the BCW to the ACW. This is a nonlinear process, described by the second part of the first equation in (2), $\dot{\delta}(t) = -k'\delta(t)$. The solution to this differential equation is $\delta(t) = \delta(t_0)e^{-k't}$. If at time point t_0, $\delta(t_0) = m \cdot \Gamma/c$, the time t needed to decrease the height difference by one unit is

$$t = -\frac{1}{k'} \ln\left(\frac{m-1}{m}\right). \tag{3}$$

The number of time steps needed to decrease the height difference by one unit is obtained by dividing this time by T and rounding to the nearest integer.

C. Linearly Priced Timed Automata

The basic ingredients of Linearly Priced Timed Automata (LPTA) [3], [4] are *locations*, *switches*, *clocks*, *guards*, and *invariants*. Fig. 2 depicts a simple LPTA, which models the behavior of a lamp. Location off is the starting location. While the switch from low to low (abbreviated low → low) is executed (triggered by the environment over channel press), clock y is reset to 0. Clocks are real-valued variables which are used to measure time: clock values increase linearly with rate 1 as time progresses. Clocks are used to express enabling- and urgency-conditions for transitions depending on time. The switch low → bright is executed when triggered by the environment,

again over channel press. This transition is *guarded* by expression y < 5, i.e., only if clock $y < 5$, this switch can be executed. Thus, if the environment triggers channel press a second time *within* 5 time units since the first, the lamp switches to brighter light. Locations low and bright are both annotated with the invariant $y \leq 10$, which means that both locations can only be occupied as long as $y \leq 10$ holds. During this time, the location must be vacated via low $\rightarrow$ off or bright $\rightarrow$ off. Since both switches have guard y $== 10$, this will happen precisely at the time when clock y reaches value 10.

Switching on a lamp and letting it burn uses energy, which defines costs. To account for these costs, LPTA have a global cost variable cost which can be explicitly increased while executing a switch (as is done with switch off $\rightarrow$ low), or which is increased implicitly as time progresses. For the latter, locations can be annotated with cost *rates*. In locations low and bright, we have the extra "invariants" cost$' == 10$ and cost$' == 20$, respectively, which indicate that the energy consumption is 10 and 20 units per time unit in the respective locations. When staying in these locations, cost is implicitly increasing with time, with rate 10 and 20, respectively.

D. Schedule Generation Using LPTA

Uppaal Cora [1] is a model checker for LPTA, i.e., a tool to check whether the modeled LPTA has certain properties which are expressed as logic formulae in a fragment of the *timed computation tree logic* [2]. An important problem to solve for LPTA to make model checking feasible is *minimum-cost reachability*, i.e., given an LPTA and a target state (where a *state* is a combination of a location and additional information about the clocks), determine the minimal cost of all paths leading from the initial location to the target state [4], while meeting a certain time bound. In [3] and [4], it has been shown independently that this problem is effectively computable. This very important result allows to use LPTA for schedule generation as follows. LPTA models can be nondeterministic. A model checker like Uppaal Cora can find paths—starting in the initial location—through the state space of a timed automaton to target states and compute the minimal cost to do so. These paths resolve nondeterministic choices on the way to the target. The idea of schedule generation with model checkers is to model the system to be scheduled (which is thus a combination of resources and load), but to leave the scheduling decisions open, i.e., nondeterministic. If a certain scheduling objective can be formulated as a state property of this system, then the model checker can be employed to find such a state and the path leading to it; *the determined path is a schedule*. Finding the *cheapest* path to the target state yields an *optimal* schedule.

III. LPTA Battery Scheduling Model

The behavior of batteries, as described by the dKiBaM, can be modeled using LPTA. The problem to generate schedules which maximize the lifetime of a system using more than one battery can also be solved using the cost-minimal reachability algorithms as implemented in Uppaal Cora. The cost to be minimized is the total charge left in the batteries at the time when none of them can be used anymore, i.e., when all ACW are empty.

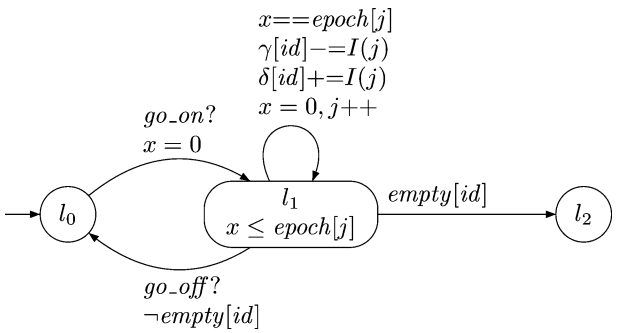

Fig. 3. Discharge automaton DC.

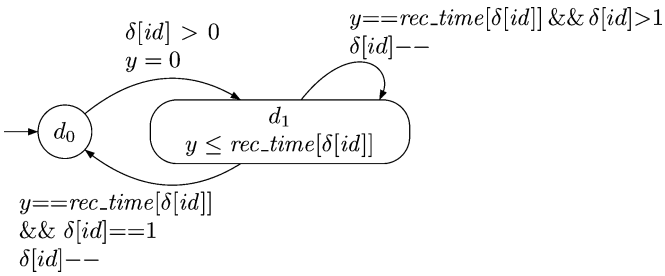

Fig. 4. Recovery automaton RC.

A. Basic Battery Model

The discharge and recovery behavior of the dKiBaM is modeled essentially by two LPTA, one, DC, modeling the discharge process, the other one, RC, the recovery process. A simplified, abstract version of these automata is shown in Figs. 3 and 4. Every battery is identified by the respective local variable id. Both automata operate on global integer arrays. The array $\gamma[id]$ keeps track of the total charge in battery id, and $\delta[id]$ of the height difference of the wells.

The discharge automaton in Fig. 3 starts in location l_0 and waits for a signal on channel go_on. Once this arrives (i.e., the battery is scheduled for use), with the switch from l_0 to l_1, clock x is set to 0. In a globally available, precomputed array $epoch[j]$, which is part of the load description, the time required to deplete the battery by one charge unit is stored. Once clock x has reached value $epoch[j]$, the self-loop transition of l_1 is executed, which decreases $\gamma[id]$ by $I[j]$, the number of charge units drawn in epoch j, and increases the height-difference $\delta[id]$ by the same amount. The global epoch counter j is increased, and x reset to 0.

The recovery automaton in Fig. 4 works autonomously, and depends on the value of $\delta[id]$. Starting in location d_0, the switch to d_1 is taken once $\delta[id] > 0$. Then the self-loop is executed until $\delta[id]$ is equal to 1. With the transition back to d_0 the height difference is set to 0 again. Important in this automaton is the timing. The time it takes to decrease the height difference by one unit depends on the height difference itself. Those times are precomputed and kept in the array rec_time: the value $rec\_time[\delta[id]]$ is the time it takes to decrease $\delta[id]$ by one. The self-loop of d_1 is, hence, executed every $rec\_time[\delta[id]]$ time-units, upon which $\delta[id]$ is decreased by one.

Two more automata, which are not described here, have to be defined to complete the model: one to describe the load applied

to the battery, the other to describe the scheduler, which makes the nondeterministic choice between the still-available batteries. The total system S with M batteries is then the parallel composition of the automata

$$S = (DC_1||RC_1)||\cdots||(DC_B||RC_B)||Load||Scheduler$$

where the indices $1,\ldots,B$ serve also as the id parameter. A detailed description of the Uppaal Cora model can be found in [9].

B. Battery Scheduling

The scheduling decisions in the LPTA model are kept nondeterministic, in order to allow Uppaal Cora to make the optimal decisions. If several automata components are replicated to model M batteries, the nondeterminism lies in the M discharge automata (Fig. 3), which all wait for a signal on channel go_on to start. Only one can actually consume the signal, once it is sent by the *Scheduler* automaton. Which one this is, is decided nondeterministically.

The optimality criterion of a schedule is the length of the lifetime of the system: the longer, the better. This criterion has to be translated into a cost function that can be minimized by Uppaal Cora. The cost that is incurred is the charge in the bound charge well of the batteries that is still available, but can not be used anymore because the available charge has once been emptied. At the end of a run, this remaining charge is added to the cost-variable, and the run is evaluated based on that.

C. Complexity

The *complexity* of finding the optimal schedule clearly depends exponentially on the number of scheduling decisions that have to be made, where the number of batteries (M) is the base. At every scheduling point one can choose between all M batteries. The number of scheduling points depends on the battery's capacity and the load applied.

Between the scheduling points the model behaves fully deterministically. The number of states in between two scheduling points will depend on the granularity of the discretization. The maximum number of state changes that can be made due to discharging is $N = C/\Gamma$, when all charge units are drained one at a time. The maximum number of state changes due to recovery is not so easy to define. However, it will be proportional to $1/\Delta$. Since in the model $\Delta = \Gamma/c$, the maximum number of states will be proportional to $(1/\Gamma)^2$. The discretization of time will not influence the maximum number of states. Introducing smaller time steps will only increase the number of time steps it takes to change states.

IV. Scheduling Results TA-KiBaM

A. Test Loads

We use the multiple battery priced-timed automaton model to find the schedule that gives the longest lifetime for a system of two batteries on a set of test loads. We use two batteries with capacity 5.5 Amin (Ampere-minute). The battery parameters are: $c = 0.166$ and $k' = 0.122\ \mathrm{min}^{-1}$ [8], corresponding to the lithium-ion battery used in the Itsy pocket computer, which was also simulated by Rakhmatov *et al.* [13], [14]. In [9], the TA-KiBaM model has been validated for the same batteries.

The time step size is set to 0.01 min. The total charge is discretized in steps of size 0.01 Amin. This leads to the discretization step size of the height difference of $0.01/c = 0.06\ \mathrm{Amin}$.

The Itsy pocket computer operates with currents up to 700 mA. In the test loads, we used two types of jobs, a low current job (250 mA) and a high current job (500 mA). With these jobs, eight different test loads have been created:

- Three continuous loads (CL) with no idle periods between the jobs: one load with only low current jobs (CL_250), one with only high current jobs (CL_500), and one alternating between a low current job and a high current job (CL_alt).
- Three intermitted loads with short idle periods of one minute between the jobs (ILs): one with only low current jobs (ILs_250), one with only high current jobs (ILs_500), one alternating between a low current job and a high current job (ILs_alt).
- Two intermitted loads with long idle periods of two minutes between the jobs ($\mathrm{IL}\ell$): one with only low current jobs ($\mathrm{IL}\ell$_250) and one with only high current jobs ($\mathrm{IL}\ell$_500).

Next to computing the maximum lifetime, we used the TA-KiBaM to compute the lifetime using three deterministic scheduling schemes:

- *Sequential scheduling*. The batteries are used sequentially, i.e., the second battery is only chosen when the first one is empty.
- *Round-robin scheduling*. For every new job a new battery is chosen. The batteries are chosen in a fixed order.
- *Best-of-two scheduling*. At the start of a job, the status of the batteries is checked. The battery with the most charge in the ACW is chosen to supply the charge for the job. Note that this requires the possibility to "measure" the charge in the ACW, a feature that is mostly absent in practice.

The scheduling decisions for the given load functions are made at the beginning of new jobs (except at time 0), which in the case of the CL loads are equidistant points in time, and for the other loads whenever the load changes. The computed lifetimes are given in Table I, along with the relative difference to the lifetime using the round-robin scheduling. For the test loads, one can see the order in performance of the different scheduling schemes. One can easily show, using the Uppaal Cora model, that the sequential scheduling is actually the worst possible way to schedule the batteries. For the test loads the round-robin and best-of-two scheme differ only for alternating jobs. These cases are clearly very bad for the round-robin scheme, since the heavy load is always put onto the same battery. This battery will get empty very fast, and then only one battery is left to handle all of the remaining load, leaving this battery with less idle time to recover. The best-of-two scheme balances the load better over the two batteries, which leads to a longer lifetime, especially in the ILs_alt case. In the other cases the best-of-two scheme behaves exactly like the round-robin scheme. Although the round-robin and best-of-two schedulers perform close to the TA-KiBaM scheduler in most cases, for some loads the schedules lead to significantly shorter lifetimes. The TA-KiBaM scheduler yields lifetime improvements up to 32%. Of course, it has to be noted,

TABLE I
SYSTEM LIFETIME COMPUTED FOR ALL TEST LOADS ACCORDING TO THE FOUR SCHEDULING SCHEMES. NEXT TO THE VALUES OF THE LIFETIMES, THE DIFFERENCE RELATIVE TO THE ROUND-ROBIN SCHEME ARE GIVEN

test load	sequential lifetime (min)	sequential difference %	round robin lifetime (min)	best-of-two lifetime (min)	best-of-two difference %	TA-KiBaM lifetime (min)	TA-KiBaM difference %
CL_250	9.12	-21.4	11.6	11.6	0	12.04	3.79
CL_500	4.10	-9.5	4.53	4.53	0	4.58	1.1
CL_alt	5.48	-10.2	6.10	6.12	0.3	6.48	6.2
ILs_250	22.80	-41.5	38.96	38.96	0	40.80	4.7
ILs_500	8.60	-17.9	10.48	10.48	0	10.48	0
ILs_alt	12.38	-3.4	12.82	16.30	27.2	16.91	31.9
ILℓ_250	45.84	-39.7	76.00	76.00	0	78.96	3.9
ILℓ_500	12.94	-18.9	15.96	15.96	0	18.68	17.0

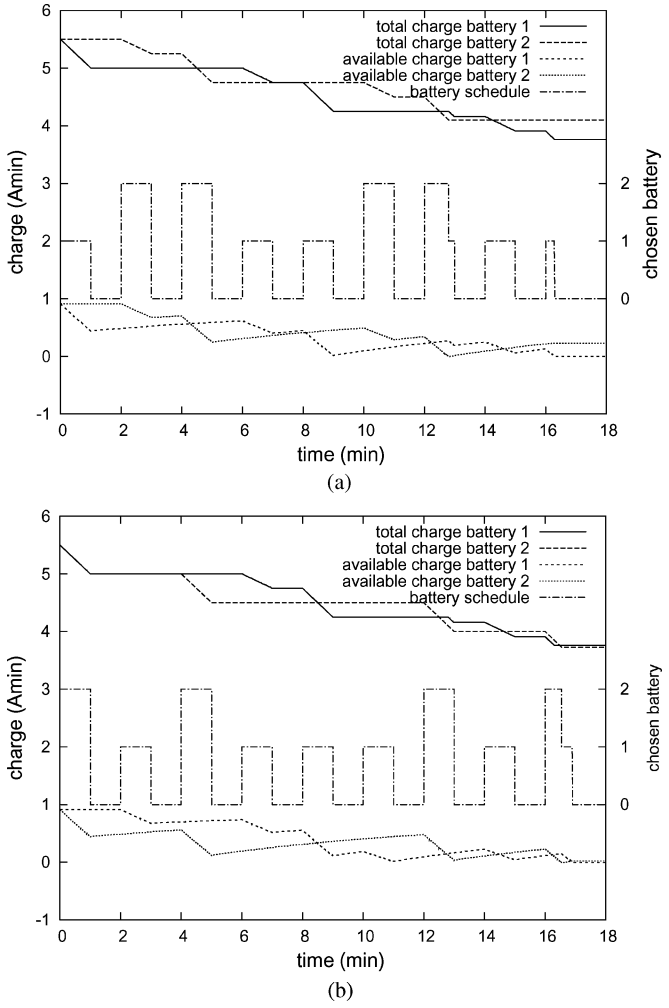

Fig. 5. The schedules and the total and available charge in the batteries for the (a) best-of-two and (b) the optimal schedule for the ILs_alt load.

that the TA-KiBaM scheduler is not practically useable, since one needs to to know the exact load function in advance.

Besides the system lifetimes, the Uppaal Cora evaluation of the TA-KiBaM also provides the actual schedules which lead to these lifetimes, as well as the evolution of the charge in the battery. Fig. 5 shows the evolution of the total and available charge in the two batteries (left y axis) for both the best-of-two scheduler [Fig. 5(a) and the TA-KiBaM scheduler (Fig. 5(b)], in the ILl_alt case. In the figure, also the two schedules (right y axis) are shown. When a battery is chosen, one can see the total and available charge decrease due to the load. The slope of the curves is proportional to the discharge current. When a battery is not used, one clearly observes the recovery effect. Note that, when the batteries are empty, still a relatively large amount of charge remains in the battery (approximately 3.9 Amin per battery). Due to the high complexity of finding the TA-KiBaM schedule, it is possible to model only a limited total battery capacity. The used discharge currents are relatively high for the battery's capacity, and will drain the available charge well fast. Therefore, there will be little time for the bound charge to become available, and a large fraction will remain unused. When the battery capacity is increased this fraction will be smaller. Using the deterministic scheduling scheme, we can compute the results for larger capacities. For example, with a ten times larger capacity, the fraction of charge left behind in the batteries will be less than 10% in the case of best-of-two scheduling.

When we look at the two schedules in Fig. 5, we see that the best-of-two schedule acts like a round-robin scheduler that switches batteries after the high current jobs. The TA-KiBaM schedule seems to behave randomly, no direct relation between the state of the batteries and the schedule can be made. The TA-KiBaM schedule does depend strongly on the size of the batteries and their parameters, as well as on the load that is applied.

B. Towards Random Loads

The test loads presented in the previous section are very regular. The discharge current switches after fixed time periods. However, most realistic loads are not regular and have discharge and idle periods of random length. As a first step towards more realistic loads, we introduce discharge periods of random length. The load profile we use is still an on-off load, where the discharge current switches between 250 and 0 mA. All off-periods last 1 min. However, the lengths of the on-periods are chosen from a uniform distribution on [1/2 min, 3/2 min]. The scheduling decision is made at the start of each on-period. To see how the different scheduling schemes perform for this random load, 500 load traces have been generated. For each of these traces, the system lifetime under the four scheduling schemes has been computed for two batteries with 5 Amin capacity. Since the loads are random, the result is now an empirical distribution. The results are shown in Fig. 6. Like with the test loads, the sequential scheduler results in much shorter lifetimes than the other schedulers, and is far from optimal. On

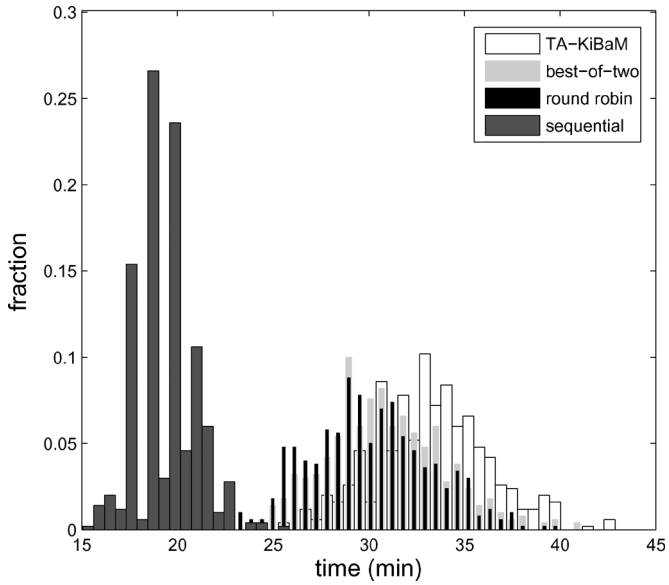

Fig. 6. Lifetime distributions of the four scheduling schemes for 500 random loads.

average the TA-KiBaM scheduler outperforms the sequential scheduling scheme by 70%. The results of the round-robin scheduler and the best-of-two scheduler lie close to each other, although the latter performs slightly better. The schedules of TA-KiBaM outperform round-robin and best-of-two by 10% and 8%, respectively.

V. Nonuniform Scheduler Generation

The LPTA approach described in Section III was based on the assumption that a scheduling decision must be made at predefined points in time. Even with this restriction, schedule generation for realistic battery capacities is not feasible due to an exponential growth of the time needed to assess all possible schedules. In this section, we investigate a slightly more general scheduling problem, which differs in two points. First, the question is not only to choose the optimal among available batteries, but also *when* to do this; second, a battery can be reused after its ACW has been completely drained and some recover period afterwards. The chosen approach to tackle this scheduling problem is *analytic*.

A. Identical Batteries

Consider M identical batteries with parameters c, k, and C. A switching mechanism is a collection of M functions $[u^{(1)},\ldots,u^{(M)}]$ with $u^{(j)}(t):\mathbb{R}_{\geqslant 0}\rightarrow\{0,1\}, j\in\{1,\ldots,M\}$. $u^{(j)}(t)=1$ iff battery j is selected at time t. Note that $\sum_{j=1}^{M}u^{(j)}(t)=1$, for $t\geq 0$.

When a battery is selected, a charge is drawn from it with current $i(t)$. Given the switching mechanism $[u^{(1)}\ldots u^{(M)}]$, the available and bound charges of all M batteries can be expressed using the system of ODEs in (1)

$$\begin{cases}\dot{y}_1^{(j)}(t)=-i(t)u^{(j)}(t)+k\left(h_2^{(j)}(t)-h_1^{(j)}(t)\right)\\ \dot{y}_2^{(j)}(t)=-k\left(h_2^{(j)}(t)-h_1^{(j)}(t)\right)\end{cases}\tag{4}$$

where $y_1^{(j)}$ is the available charge and $y_2^{(j)}$ is the bound charge in battery j. Here, we take again the initial conditions $y_1^{(j)}(0)=c\cdot C$ and $y_2^{(j)}(0)=(1-c)\cdot C$, where C is the total initial battery charge. Similar as before, $h_1^{(j)}(t)=y_1^{(j)}(t)/c$ and $h_2^{(j)}(t)=y_2^{(j)}(t)/(1-c)$.

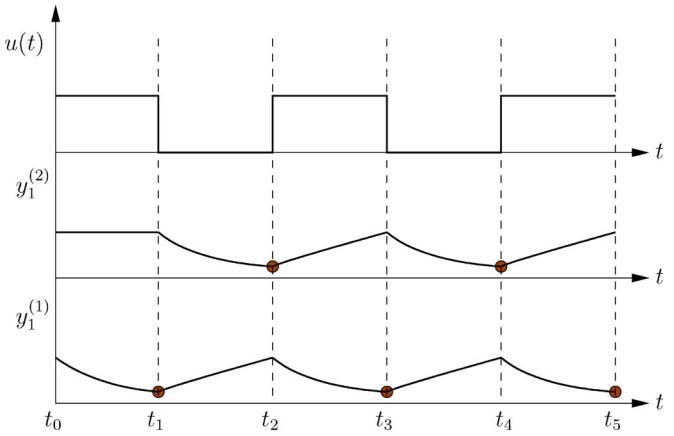

Fig. 7. Control function $u(t)$ and the available charges $y_1^{(1)}(t)$ and $y_1^{(2)}(t)$.

Example 1: Fig. 7 depicts the available charge of two batteries, where the switching functions $u^{(1)}(t):=u(t)$ and $u^{(2)}(t):=1-u(t)$ are defined in terms of the piecewise constant control function u. Note that in the interval of time $[t_0,t_1)$, the first battery is selected and therefore the available charge decreases, while the available charge of the second battery remains constant. When in the interval of time $[t_1,t_2)$ the second battery is selected, the available charge $y_1^{(2)}(t)$ decreases, while $y_1^{(1)}(t)$ increases due to the recovery effect.

Given M batteries with identical parameters k, c and C, and a switching mechanism $[u^{(1)}\ldots u^{(M)}]$, the total available charge $\hat{y}_1(t)=\sum_{j=1}^{M}y_1^{(j)}(t)$ and bound charge $\hat{y}_2(t)=\sum_{j=1}^{M}y_2^{(j)}(t)$ at time t is given by the following theorem.

Theorem 1: The total available charge $\hat{y}_1(t)$ and the total bound charge $\hat{y}_2(t)$ is given by the following system of ODEs:

$$\begin{cases}\dot{\hat{y}}_1(t)=-i(t)+k\left(\hat{h}_2(t)-\hat{h}_1(t)\right)\\ \dot{\hat{y}}_2(t)=-k\left(\hat{h}_2(t)-\hat{h}_1(t)\right)\end{cases}\tag{5}$$

where $\hat{h}_1(t)=\hat{y}_1(t)/c$, $\hat{h}_2(t)=\hat{y}_2(t)/(1-c)$ and the initial conditions are $\hat{y}_1(0)=M\cdot c\cdot C$ and $\hat{y}_2(0)=M\cdot(1-c)\cdot C$.

Proof: By adding up (4) for all $j=1,\ldots,M$, and the fact that $\sum_{j=1}^{M}u^{(j)}(t)=1$ for all $t\geq 0$.

Theorem 1 shows that the total remaining available charge of M batteries at any time t is equal to the total charge of a single battery $\hat{B}$ with initial capacity $M\cdot C$, and parameters k and c. More remarkable is the fact that the total remaining charge of all M batteries does *not* depend on the control functions $[u^{(1)}\ldots u^{(M)}]$ *at all*. This surprising result will be pivotal in the following investigation.

For a constant load function $i(t)=L$ with $L>0$, the solution to (5) can be derived as

$$\hat{y}_1(t)=-cLt+cMC-\frac{L(1-c)}{k'}\left(1-e^{-k't}\right)$$

and

$$\hat{y}_2(t)=-(1-c)Lt+(1-c)MC+\frac{L(1-c)}{k'}\left(1-e^{-k't}\right)\tag{6}$$

where $k' = k/c(1-c)$. The available charge $\hat{y}_1(t)$ from (6) is monotonously decreasing due to the fact that $d\hat{y}_1(t)/dt < 0$. Also notice that $\lim_{t\to\infty} \hat{y}_1(t) = -\infty$. There exists therefore a time point $t_f \geq 0$, where $\hat{y}_1(t_f) = 0$, and for all $\tau > t_f$, $\hat{y}_1(t_f+\tau) < 0$. One could say that t_f is the maximum total lifetime achieved by M batteries under any switching mechanism $[u^{(1)} \dots u^{(M)}]$. The maximum total lifetime t_f can be expressed analytically as follows:

$$t_f = \frac{MC}{L} + \frac{1}{k'}\left(1 - \frac{1}{c} + W\left(\frac{1-c}{c} e^{-\frac{MCk'}{L} + \frac{1-c}{c}}\right)\right) \quad (7)$$

where W is the Lambert W function.

Theorem 1 proves that it is rather simple to obtain the optimal lifetime of M batteries: the problem can be reduced to finding the maximum total lifetime of a single battery $\hat{B}$ with M-fold initial capacity, and otherwise identical parameters k and c.

Unbalanced Case: Eq. (6) describes the evolution of $\hat{y}_1$ and $\hat{y}_2$ for the special case of a initial height difference of 0 between ACW and BCW, and initial charge MC. The equations can be generalized to the unbalanced case, where the total charge is distributed over ACW and BCW with a nonnegative height difference. This is defined by the following equations:

$$\begin{aligned} y_1(\bar{y}_1, \bar{y}_2, L, t) = &- cLt + c(\bar{y}_1 + \bar{y}_2) \\ &+ ((1-c)\bar{y}_1 - c\cdot\bar{y}_2)\, e^{-k't} \\ &- \frac{(1-c)L}{k'}(1 - e^{-k't}) \\ y_2(\bar{y}_1, \bar{y}_2, L, t) = &- (1-c)Lt + (1-c)(\bar{y}_1 + \bar{y}_2) \\ &+ (c\bar{y}_2 - (1-c)\bar{y}_1)\, e^{-k't} \\ &+ \frac{(1-c)L}{k'}(1 - e^{-k't}) \end{aligned} \quad (8)$$

where $y_1(\bar{y}_1, \bar{y}_2, L, t)$ is the charge in the ACW after t time units have passed, $\bar{y}_1$ is the charge in the ACW, $\bar{y}_2$ is the charge in the BCW (both at time 0), L is the constant current applied, t the time and $k' = k/c(1-c)$. The initial total charge is thus $\bar{y}_1 + \bar{y}_2$, and the initial height difference $(\bar{y}_2/(1-c)) - (\bar{y}_1/c)$, which, for the equations to be valid, we assume to be nonnegative. It is straightforward to verify that $\hat{y}_i(t)$ [(6)] equals $y_i(cMC, (1-c)MC, L, t)$ for $i = 1, 2$, and that $y_i(\bar{y}_1, \bar{y}_2, L, t+t') = y_i(\bar{y}'_1, \bar{y}'_2, L, t')$ with $\bar{y}'_i = y_i(\bar{y}_1, \bar{y}_2, L, t)$, for $i = 1, 2$.

For the unbalanced case, the total remaining lifetime for a constant load can be expressed as

$$\begin{aligned} t_f(\bar{y}_1, \bar{y}_2, L) = &\frac{\bar{y}_1 + \bar{y}_2}{L} - \frac{1-c}{ck'} \\ &+ \frac{1}{k'} W\left(\frac{(1-c)(\bar{y}_1 k' + L) - c\bar{y}_2 k'}{cL} e^{-k'\left(\frac{\bar{y}_1+\bar{y}_2}{L} - \frac{1-c}{ck'}\right)}\right) \end{aligned} \quad (9)$$

where again W is the Lambert W function, and other parameters as before.

B. Greedy Scheduler Algorithm

Theorem 1 suggests that to reason about the scheduling of batteries is unnecessary, since any switching mechanism would do. However, this is only true under certain unrealistic assumptions. Again, we consider M batteries controlled by a piecewise constant switching mechanism $[u^{(1)} \dots u^{(M)}]$. The theorem does not take into account that for all batteries $y_1^{(j)}(t) \geq 0$ must hold at all time. Furthermore, the condition $\hat{y}_1(t_f) = 0$ with t_f being the maximum lifetime of $\hat{B}$ for given load $i(t)$ indicates that for each individual battery j, $y_1^{(j)}(t_f) = 0$, i.e., the ACW of the M batteries must all be empty at this precise point in time. The total charge left in the batteries is $MC - Lt_f$, i.e., there must be several batteries $i' \in I \subseteq \{1, \dots, M\}$ such that $y_2^{(i')}(t) > 0$. The batteries $i' \in I$ thus still have potential for recovery at time t_f. The discrete switching functions $u^{(i')}$ must then have the property to switch faster and faster between batteries as t approaches t_f, in order to ensure that $\sum_{i' \in I} y_1^{(i')}(t_f) = 0$ and no battery has time to recover from its BCW: in the time interval $[0, t_f]$ thus an infinite number of switching points must occur.

In practice, it is unrealistic to assume an infinite number of switching points. In the following, we thus consider only a finite number N of switches and investigate how to place them in the interval $[0, t_f]$. The time line is thus divided into $N+1$ pieces, such that for all $t \in [t_\ell, t_{\ell+1})$, $0 \leq \ell \leq N$, $u^{(j)}(t) = 1$, if the jth battery is selected in the interval $[t_\ell, t_{\ell+1})$. $t_0 = 0$ is not considered a switching point. The total lifetime in this setting is defined as the time t_{N+1}, which is the time when the available charge of the battery chosen last at time t_N reaches 0. Note that t_{N+1} is also not a switch, and the $u^{(j)}(t)$ are constant for $t \geq t_N$.

Constant Load: Initially, we consider that the load function $i(t)$ is constant, i.e., $i(t) = L$, for all $t \in \mathbb{R}_{\geqslant 0}$.

Therefore, for the jth battery and $\ell \geq 1$ the available and bound charge can be computed recursively as follows, using (8):

$$\begin{aligned} y_1^{(j)}(t_\ell) &= y_1\left(\bar{y}_1, \bar{y}_2, \bar{u}_\ell^{(j)} L, t_\ell - t_{\ell-1}\right) \\ y_2^{(j)}(t_\ell) &= y_1\left(\bar{y}_1, \bar{y}_2, \bar{u}_\ell^{(j)} L, t_\ell - t_{\ell-1}\right) \end{aligned} \quad (10)$$

where $\bar{y}_1 = y_1^{(j)}(t_{\ell-1})$ and $\bar{y}_2 = y_2^{(j)}(t_{\ell-1})$, $y_1^{(j)}(t_0) = c \cdot C$, $y_2^{(j)}(t_0) = (1-c)\cdot C$, $t_0 = 0$ and $u^{(j)}(t) = \bar{u}_\ell^{(j)} \in \{0, 1\}$, for all $t \in [t_{\ell-1}, t_\ell)$.

Scheduling problem: Given M batteries with parameters k, c and $N+1$ switches, the scheduling problem is to find all switching time points t_ℓ and $\bar{u}_\ell^{(j)}$, $0 \leq \ell \leq N$, $j \in \{1, \dots, M\}$, such that $y_1^{(j)}(t_\ell) \geq 0$, and the *total lifetime* t_{N+1} is *maximized*.

The problem is thus now to find a good schedule for a given number of switching points $N+1$. However, Theorem 1 already indicates that it is not the schedule that is relevant. As it turns out, more important is the number of switching points that is allowed, which is clearly indicated by Fig. 8. There, the x axis represents the number of switching points, the y axis the optimal lifetime that can be achieved with the number of switches. We see three curves for different constant load functions. The number of batteries is $M = 2$. We see clearly that for $L = 0.1$ A, the total lifetime is nearing asymptotically a constant, which is actually the optimal lifetime t_f. For the higher loads, the same phenomenon can be observed.

We are now in a position to define a family of schedules for any number of identical batteries M.

Definition 1 (Greedy Schedule Generation):

- Find all time points t_ℓ, $\ell \in \mathbb{N}$, such that $t_0 \leq t_1 \leq \dots \leq t_\ell \leq t_{\ell+1} \leq \dots$ such that for $n \in \mathbb{N}$ it holds: $y_1^{(j)}(t_{n\cdot M+j}) = 0$.

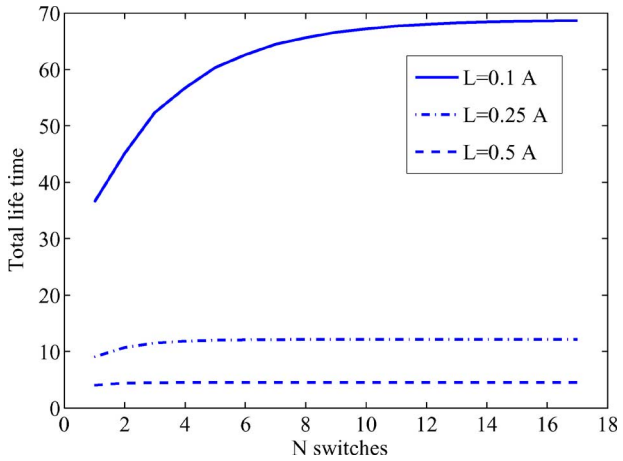

Fig. 8. Dependence of the total lifetime regarding the number of switches for three constant load functions and two batteries.

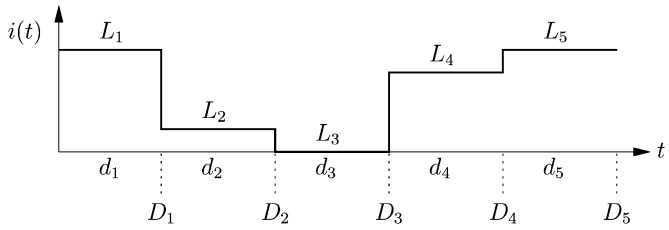

Fig. 9. Example of a piecewise constant load function $i(t)$.

- $u^{(j)}(t) = \bar{u}^{(j)}_{n\cdot M+j} = 1$, for all $t \in [t_{n\cdot M+j-1}, t_{n\cdot M+j})$.

Informally. the *greedy scheduler* (GS) works as follows: initially the first battery is drained until $y_1^{(1)}(t_1) = 0$, then we switch to the second battery and so on. When the M'th battery is drained at time point t_M, i.e., $y_1^{(j)}(t_M) = 0$ we switch back to the first battery and the scheduler algorithm continues from there on. This scheduler behaves like the sequential scheduler in Section IV with the improvement that the batteries are allowed to be reused after being emptied. Note that in general the number of switching points t_ℓ is still infinite. This can be mended by just using the first N points of the schedule, if N is given in advance, or stop switching once the condition $t_{\ell+1} - t_\ell \leq \varepsilon, \varepsilon \in \mathbb{R}_{\geqslant 0}$ for some $\ell \in \mathbb{N}$, holds. Then, we set $N = \ell + 1$. In this case, the total lifetime will be t_{N+1}.

Piecewise Constant Loads: Assume a piecewise constant load function which is defined by right-open, left-closed intervals $I_1 = [0, D_1), I_2 = [D_1, D_2), \ldots$ with interval width $d_1, d_2, \ldots$ (see Fig. 9 for an initial piece) and constant currents $L_1, L_2, \ldots$ on the respective intervals. The switching points t_ℓ, $\ell \in \mathbb{N}$, are defined precisely as in the constant case (cf. Definition 1). However, the question is, how to compute these points effectively? This can be done in an iterative manner, using (8) and (9). Assume switching points $t_0, \ldots, t_{\ell-1}$ already given and battery j chosen as the next battery at time $t_{\ell-1}$. Assume also that $t_{\ell-1} \in I_m = [D_{m-1}, D_m)$. Let $\bar{y}_1, \bar{y}_2$ be the available charge of battery j at time $t_{\ell-1}$.

1) If $t_f(\bar{y}_1, \bar{y}_2, L_m) > D_m$, set $\bar{y}_n^{(0)} := y_n(\bar{y}_1, \bar{y}_2, L_m, D_m - t_{\ell-1})$ for $n = 1, 2$. Otherwise, set $t_\ell = t_f(\bar{y}_1, \bar{y}_2, L_m)$ and terminate.
2) Compute the values $\bar{y}_n^{(1)}, \ldots, \bar{y}_n^{(v)}$ for $n = 1, 2$ and $v \geq 1$, using the boundary points of intervals $I_{m+1}, \ldots, I_{m+v}$, where v is determined by the condition $t_f(\bar{y}_1^{(v)}, \bar{y}_2^{(v)}, L_{m+v}) \leq D_{m+v}$. Then set $t_\ell = t_f(\bar{y}_1^{(v)}, \bar{y}_2^{(v)}, L_{m+v})$ and terminate.

The switching points t_ℓ, $\ell \in \mathbb{N}$ can be computed successively using this scheme, starting with $t_0 = 0$.

Note that there is in fact no proof that an infinite schedule generated by the GS algorithm would actually approach the maximal lifetime. This is an open problem, subject to further investigation. However, experimental results show that the proposed scheme comes very close to the optimal lifetime, which also for piecewise constant loads is determined by the ODE in Theorem 1.

C. Alternative Solution

Besides the schedules generated by the GS algorithm, we can generate schedules by specifying the scheduling problem as an optimization problem. To do so, we need to pick an objective function, i.e., the function which needs to be maximized, and a set of constraints.

The objective function is defined as $\sum_{\ell=0}^{N}(t_{\ell+1} - t_\ell)$, i.e., the total lifetime t_{N+1}. The set of constraints will represent the available charge at time points t_ℓ, $1 \leq \ell \leq N+1$. Using (8), we define more formally the optimization problem as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{\ell=0}^{N}(t_{\ell+1} - t_\ell) \\ \text{subject to} \quad & y_1^{(j)}(t_\ell) \geq 0, \\ & \ell \in \{0, \ldots, N+1\},\ j \in \{1, \ldots, M\}. \end{aligned}$$

Here, we assume the load function $i(t)$ is constant. Also, as for the GS algorithm, we take $u^{(j)}(t) = \bar{u}^{(j)}_{n\cdot M+j} = 1$, for all $t \in [t_{n\cdot M+j-1}, t_{n\cdot M+j})$, for $n \in \mathbb{N}$.

The above optimization problem is a nonlinear programming problem due to the fact that the set of constraints are nonlinear functions. It is important to note that by wisely choosing the set of constraints, i.e., considering only the set of constraints $y_1^{(j)}(t_\ell) \geq 0$ such that t_ℓ is the time point when the jth battery is discharging, the above nonlinear programming problem becomes a convex optimization problem. Note that this will hold as soon as the load function is constant. As the optimization problem is convex any local minimum is also a global one. Convex optimization problems can be solved efficiently, for instance by using interior-point method algorithms [6].

D. Greedy Schedule Algorithm Results

The system lifetime for the test loads introduced in Section IV-A has also been computed using the GS approach. The results are given in Table II. For most of the test loads the results are close to the lifetimes obtained by the TA-KiBaM scheduler. However, for three of the loads the schedules obtained from the GS approach result in a significantly longer lifetime. This difference is due to the fact that the TA-KiBaM scheduler can only switch batteries at fixed points in time, while in the GS schedule batteries may be switched at any time. The limitation in switching options may result in one of

TABLE II
SYSTEM LIFETIME FOR THE TEST LOADS USING THE GREEDY SCHEDULES AND THE TA-KiBaM SCHEDULES. FOR THE HIGHLIGHTED LOADS THE GREEDY SCHEDULE LEADS TO A SIGNIFICANTLY LONGER LIFETIME

test load	GS algorithm lifetime (min)	TA-KiBaM lifetime (min)
CL_250	12.16	12.04
CL_500	4.53	4.58
CL_alt	6.45	6.48
ILs_250	44.77	40.80
ILs_500	10.80	10.48
ILs_alt	16.93	16.91
ILℓ_250	84.90	78.96
ILℓ_500	21.86	18.68

the batteries being emptied sooner, and thus giving a shorter system lifetime.

Besides that the schedules of the GS approach lead to longer lifetimes, the GS approach has one major benefit compared to the TA-KiBaM. Where the TA-KiBaM cannot handle batteries with large capacity due to the exponential growth of the state space, the GS approach can cope with larger batteries.

VI. DISCUSSION

In this section, we compare the TA-KiBaM and the analytic approaches. The TA-KiBaM produces schedules for given load functions and predefined scheduling points. At these scheduling points a switch between batteries can occur, but does not have to. In the considered scenarios, decision points are predefined and coincide in the nonconstant cases with the times where the load changes. While this choice was made to keep the schedule generation within reasonable time and memory-bounds, it is of course straightforward to define more or different scheduling points. The more there are, the better the resulting schedules should approximate the results of the greedy scheduler.

In the case of batteries with identical parameters, Theorem 1 suggests that proactive scheduling as done with the TA-KiBaM is hardly necessary: as long as the internal state of a battery is accessible or reasonably well predictable, the greedy scheduler can be easily implemented, and optimal usage of a battery achieved. The situation changes however, when batteries with different parameters are considered (as is done for example in [16]). In that case Theorem 1 does not hold anymore. However, an adapted version of the TA-KiBaM can then still be used to derive schedules.

A related question is whether Theorem 1 does not in fact suggest to abandon scheduling altogether, at least if identical batteries are involved: since two batteries with identical parameters behave apparently like one battery with the same parameters and double capacity, one could conjecture that it is possible to just put two identical batteries in parallel and drain them as one big battery. However, this conjecture needs to be validated, i.e., it requires experimental evidence that two batteries indeed behave as one battery with the same KiBaM parameters when put in parallel. In particular, voltage, which is abstracted away in the KiBaM, becomes an issue. Even for two batteries of the same type a difference in the potential of the batteries can occur. Then, a current will flow between the batteries, resulting in loss of capacity and possibly damage to the batteries. Using batteries in parallel requires extra electronic circuitry which consumes some power and decreases efficiency. Scheduling could reduce the power consumption, and at the same time approach the optimal lifetime.

There are also situations where scheduling is preferable over putting batteries in parallel, for example, routing in a sensor network. Although each sensor, in general, is powered by only one battery, the entire network is powered by many. Often there are several routes from a sensor node to the data sink to send the collected data through the network. To keep all the sensors powered as long as possible, battery-aware routing has to be done, i.e., the decision on which sensor has to forward the data has to be based on the status of the sensor's batteries. Switching from one route to the other will give the batteries time to recover and thus give a longer lifetime to the sensor network as a whole. In this way, the routing problem is turned into a battery scheduling problem.

VII. RELATED WORK

Optimal scheduling of batteries has attracted quite some attention in the literature. We consider here the main approaches.

In [7], Chiasserini and Rao use a discrete-time Markov battery model to compare three different scheduling schemes in a multiple battery system. In the model, the recovery of the battery is considered as a random process. Also the load applied is stochastic. The complexity of the used model limits the battery capacity to very small batteries. The three schedulers that are considered are the round-robin and best-of-two scheduler, which are also used here, and a random scheduler. The schedulers are compared for different job arrival rates. The results show that the best-of-two scheduler outperforms the other two. However, the model does not allow for any optimization with respect to battery schedule.

Also, in [5], the analysis is limited to deterministic schedulers. Benini *et al.* consider sequential scheduling, round-robin scheduling and various types of best-of-two scheduling, where either the output voltage or the time that a battery has been unused determines which battery is to be scheduled. The different scheduling schemes are applied to several battery configurations containing up to four batteries. The loads that have been used are simple continuous and intermitted loads and two real-life example load profiles. Which scheduler performs best depends on the applied load. However, Benini *et al.* do show that for round-robin scheduling the system lifetime increases when the switching frequency is increased.

A completely different battery scheduling approach is taken in [16]. Instead of using two identical batteries, Wu *et al.* use two batteries with different discharge characteristics. The first battery has a high capacity and performs well at low discharge currents, whereas the second has a lower capacity but performs better at high discharge currents. The scheduling decision is taken with respect to the level of the discharge current. In this way both batteries are used only for currents where they perform the best. Wu *et al.* show that this way of scheduling may lead to a 30% lifetime improvement compared to using only one type of battery.

In all this work, the battery scheduling is limited to *deterministic* scheduling schemes. All show that battery scheduling gives longer system lifetime than when the batteries are used sequentially. However, they do not indicate whether longer lifetime could be possible by using smarter scheduling. In [15], Sarkar and Adamou propose an algorithm for computing an optimal scheduling scheme based on the stochastic battery model of Chiasserini and Rao. To do this they translate the problem to a stochastic shortest path problem. The optimal solution can only be computed for small batteries. However, they do show that best-of-two scheduling performs close to optimal.

VIII. Conclusion

In this paper, we provide two different approaches to maximize system lifetime by battery scheduling. The first approach models a discretized version of the KiBaM using linearly priced-timed automata. In this approach, the scheduling decision is made at predefined points in time, e.g., at load changes. With model checking techniques the best possible battery schedule is found, up to an error due to the discretization.

In the second approach, an analytical analysis of the KiBaM is made. The main result is that the actual schedule is not important when one can change between batteries at arbitrary points in time. Based on this conclusion, we propose a greedy scheduler algorithm that only switches batteries when the one used is perceived as empty.

The results show that both approaches lead to significant longer lifetimes compared to simply using the batteries sequentially. With the greedy algorithm it is possible to do computations for larger and more batteries, which is not possible with the TA-KiBaM. However, the TA-KiBaM allows for extra limitations on the time of scheduling, and the model checking techniques ensure that the best possible schedule will be found.

The modeled battery is a lithium-ion battery. This type is used in most handheld mobile devices. It would be interesting to see whether battery scheduling would also beneficial for other battery types, e.g., those that are used in big systems like electric cars.

References

[1] UPPAAL Cora webpage. [Online]. Available: www.cs.aau.dk/~behrmann/cora/index.html

[2] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," *Inf. Comput.*, vol. 104, no. 2, pp. 2–34, 1993.

[3] R. Alur, S. La Torre, and G. J. Pappas, "Optimal paths in weighted timed automata," in *Proc. Hybrid Syst.: Comput. Control (HSCC '01)*, M. D. Di Benedetto and A. Sangiovanni-Vincentelli, Eds., 2001, vol. 2034, LNCS, pp. 49–62, Springer-Verlag.

[4] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager, "Minimum-cost reachability for priced time automata," in *Proc. Hybrid Syst.: Comput. Control (HSCC '01)*, M. D. Di Benedetto and A. Sangiovanni-Vincentelli, Eds., 2001, vol. 2034, LNCS, pp. 147–161.

[5] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Extending lifetime of portable systems by battery scheduling," in *Design, Automation and Test in Europe (DATE)*, Los Alamitos, CA, 2001, pp. 197–203.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[7] C. F. Chiasserini and R. R. Rao, "Energy efficient battery management," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1235–1245, 2001.

[8] M. R. Jongerden and B. R. Haverkort, "Which battery model to use?," *IET Software*, vol. 3, no. 6, pp. 445–457, 2010.

[9] M. R. Jongerden, B. R. H. M. Haverkort, H. C. Bohnenkamp, and J. P. Katoen, "Maximizing system lifetime by battery scheduling," in *Proc. 39th Annu. IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN 2009)*, 2009, pp. 63–72.

[10] J. F. Manwell and J. G. McGowan, "Lead acid battery storage model for hybrid energy systems," *Solar Energy*, vol. 50, pp. 399–405, 1993.

[11] J. F. Manwell and J. G. McGowan, "Extension of the kinetic battery model for wind/hybrid power systems," in *Proc. 5th Eur. Wind Energy Assoc. Conf. (EWEC '94)*, 1994, pp. 284–289.

[12] J. F. Manwell, J. G. McGowan, E. I. Baring-Gould, W. Stein, and A. Leotta, "Evaluation of battery models for wind/hybrid power system simulation," in *Proc. 5th Eur. Wind Energy Assoc. Conf. (EWEC '94)*, 1994, pp. 1182–1187.

[13] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime predictions for energy-aware computing," in *Proc. 2002 Int. Symp. Low Power Electron. Design (ISLPED'02)*, 2002, pp. 154–159.

[14] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "A model for battery lifetime analysis for organizing applications on a pocket computer," *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 1019–1030, 2003.

[15] S. Sarkar and M. Adamou, "A framework for optimal battery management for wireless nodes," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 179–188, 2003.

[16] Q. Wu, Q. Qiu, and M. Pedram, "An interleaved dual-battery power supply for battery-operated electronics," in *Proc. Conf. Asia South Pacific Design Autom. (ASP-DAC'00)*, 2000, pp. 387–390.

Marijn Jongerden received the B.S. degree in mathematics and the M.S. degree in physics from the VU University Amsterdam, Amsterdam, The Netherlands, in 2000 and 2005, respectively. He is currently working towards the Ph.D. degree at the Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, in the Group on Design and Analysis of Communication Systems. His Ph.D. research is on maximizing lifetime of battery powered embedded systems.

Alexandru Mereacre received the B.S. degree in computer science from the Technical University of Moldova, Moldova, in 2005 and the M.S. degree in computer science from RWTH Aachen University, Aachen, Germany, and the University of Trento, Trento, Italy, in 2007. He is currently working towards the Ph.D. degree with the Group on Software Modeling and Verification, Technical University Aachen.

His research interests include modeling of probabilistic and stochastic systems, state space minimization and verification of time-inhomogeneous Markov chains.

Henrik Bohnenkamp (M'03) received the Diploma degree in computer science from the University Erlangen-Nürnberg, Erlangen, Germany, in 1995 and the Ph.D. degree in computer science from the RWTH Aachen University, Aachen, Germany, in 2002.

He is a Researcher with the Group on Software Modeling and Verification, Technical University Aachen. He held positions at the Computer Science Department, University of Twente, The Netherlands. His research interests include modeling of probabilistic and stochastic systems, semantics, and specification-based testing.

Boudewijn Haverkort (F'07) received the M.Sc. and Ph.D. degree in computer science from the University of Twente, Enschede, The Netherlands, in 1986 and 1991, respectively.

He has been a Lecturer at the University of Twente (1991–1995), before he was appointed Professor of Distributed Systems at the RWTH Aachen University, Aachen, Germany (1995–2002). In 2003, he was appointed as Full Professor of Design and Analysis of Communication Systems at the University of Twente. From 2008 to 2009, he was Chairman of the Department of Computer Science. Since March 2009, he is also Scientific Director and Chairman of the Embedded Systems Institute, Eindhoven, The Netherlands, a public-private partnership for applied research on embedded systems engineering. He has written over 100 papers on performance and dependability modeling and evaluation in computer and communication systems. He also published a textbook on this topic, as well as a number of conference proceedings. He has been host (general chair and PC chair) of a large number of international conferences.

Prof. Haverkort is a member of the ACM and GI.

Joost-Pieter Katoen is a Full Professor at the RWTH Aachen University, Aachen, Germany, and is associated part-time with the University of Twente. His research interests include concurrency theory, model checking, timed and probabilistic systems, and semantics. He coauthored more than 130 journals and conferences papers, and coauthored a comprehensive book (with C. Baier) on *Principles of Model Checking*.

Prof. Katoen is a Steering Committee Member of QEST, ETAPS, and FORMATS, and is Senior Member of the ACM.