Automatic Construction of Efficient Multiple Battery Usage Policies

Maria Fox and **Derek Long**
University of Strathclyde, UK

Daniele Magazzeni
University of Chieti-Pescara, Italy

Abstract

There is a huge and growing number of systems that depend on batteries for power supply, ranging from small mobile devices to large high-powered systems such as electrical substations. In most of these systems, there are significant user-benefits or engineering reasons to base the supply on multiple batteries, with load being switched between batteries by a control system. The key to efficient use of multiple batteries lies in the design of effective policies for the management of the switching of load between them. This paper[1] describes work in which we show that automated planning can produce much more effective policies than other approaches to multiple battery load management in the literature.

1 Introduction

We are solving the problem of extracting the maximum possible charge from a suite of battery cells that can be independently loaded. Our approach learns battery-switching policies that attempt to maximise lifetime. The novelty behind our approach is that policies are learned from high quality continuous plans generated for a large number of sampled power demand distributions. Our approach has enormous potential significance – it can extend the lifetimes of batteries in multiple battery situations, which can significantly reduce the number (and weight) of batteries required to power a device. In an environment of ubiquitous, embedded and wearable systems relying on battery power, this will support the design of lighter-weight and more flexible electrical systems. We have shown that we can achieve an efficiency improvement over other battery management techniques in the literature, of up to 20% in simulation.

There are many interesting potential applications of this technology. Efficient battery load management has the potential to significantly reduce the weight in batteries that soldiers carry to power their high-tech military equipment. A soldier carries 20-40 pounds of batteries on a typical four-day mission. The batteries power everything from soldiers' GPS systems to their night-vision goggles. Being able to extend the lifetimes of the batteries could substantially reduce the number of batteries and therefore the weight that needs to be carried. Other applications include mobile systems and battery-powered vehicles and prosthetics. The only requirement is that the system must have multiple independently schedulable batteries to benefit from our load management approach.

We restrict our attention to the situation in which the load can be serviced entirely by a single battery at a time, so the problem is distinct from the management of cells within a single battery. This is a crucial difference, because within a single battery the goal is to keep the voltage level even across the cells. This can be achieved by a simple set-point control system. In the multiple battery case the goal is to switch the load between cells in such a way that their lifetime is maximised, and this can result in very diverse load distributions. The control problem we face is much more complex because there is no single set-point that can be used to correct the behaviour of the system.

More efficient use of multiple batteries can be achieved by exploiting the phenomenon of *recovery*, which is a consequence of the chemical properties of a battery: as charge is drawn from a battery, the stored charge is released by a chemical reaction, which takes time to replenish the charge. In general, charge will be drawn from a battery faster than the reaction can replenish it and this can lead to a battery appearing to become dead when, in fact, it still contains stored charge. By allowing the battery to rest, the reaction can replenish the charge and the battery become functional once again. Thus, efficient use of multiple batteries involves carefully timing the use and rest periods. This problem can be seen as a planning problem.

The key to being able to learn intelligent policies of sufficient scope and coverage in complex situations, is to have an underlying capability in planning. This is because the problem for an automated system of deciding on the best thing to do next, is highly combinatorial even in a deterministic system. Being able to generate very good plans for deterministic cases, coupled with large-scale representative sampling, provides a strong basis for learning how to behave in complex uncertain situations. This is the heart of the approach that we have developed.

The paper is organised as follows: we first describe the battery load management problem and explain why it can be seen as a planning problem. We describe how we have exploited planning in battery load management, and we then in-

[1] A full version of this paper, including a complete account of the technical detail, was published in the Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS) [Fox *et al.*, 2011].

dicate the next steps that need to be carried out in progressing this research. Our goal is to produce usable "off-the-shelf" intelligent battery management technology, and we have made some significant progress towards achieving this.

2 The Multiple Battery Usage Problem

The multiple battery usage planning problem has been explored by several authors, from an electrical engineering perspective (*e.g.* Benini *et al.* [2003]) and also from a scheduling perspective [Jongerden *et al.*, 2009]. Due to the physical and chemical properties of batteries, it is possible to extract a greater proportion of the energy stored in a single battery of capacity C than of that stored in n batteries of capacity C/n. It therefore follows that a good measure of the efficiency of a battery load management technique is to compare the proportion of charge extracted, in the service of a load, with the maximum that could be extracted by servicing the same load from a single battery with equivalent capacity and physical properties. Benini *et al.* construct a very accurate battery model, parameterising it to capture lithium-ion, nickel-cadmium and lead-acid battery types, and show how hand constructed policies can achieve efficiency, relative to a single battery, between 70% and 97.5% (with an average around 80%) in simulation. To achieve this, the policy is constructed to select a new battery whenever the voltage of the battery currently servicing a load drops below a certain threshold. The next battery is selected according to one of four alternative policies [Benini *et al.*, 2003]:

- V_{max}: select the battery pack with highest state of charge.
- V_{min}: select the battery pack with lowest state of charge.
- T_{max}: select the battery pack that has been unused for the longest time.
- T_{min}: select the battery that has been unused for the shortest time.

The authors show that V_{max} is the best of these policies, tested on up to four batteries.

Jongerden *et al.* [2009] uses a model checking strategy, based on UPPAAL [Behrmann *et al.*, 2001], to schedule battery use given a known load profile. The approach is based on the use of a different battery model, the Kinetic Battery Model, discussed in more detail below. The model is "kinetic" because it emphasises the correct modelling of the chemical processes that govern the discharge behaviour. This is a non-linear continuous model and the authors treat it by discretisation and scheduling to a horizon. This approach allows them to find highly effective schedules, but it does not scale well because of the need to use a fine-grained discretisation of the temporal dimension. It is worth emphasising, since it contrasts with our approach, that Jongerden *et al.* work with a fixed size discretisation of time, allowing them to focus on scheduling the resources (batteries) into the load periods.

In deployed systems, the standard policies are typically static, based on rapid switching between available batteries. In fact, an optimal use of multiple batteries can be achieved theoretically by switching between them at extremely high frequency, when the behaviour converges on that of a single battery. Unfortunately, this theoretical solution is not achievable in practice because of the losses in the physical process of switching between batteries, as the frequency increases. Round-robin (which is similar to T_{max} above) or best-of-n (similar to V_{max} above) policies applied at fixed frequencies are the most commonly fielded solutions, but these often achieve less than 80% efficiency [Benini *et al.*, 2003]. The efficiency of the use of multiple batteries can be assessed both by the relative lifetime compared with a single battery (to be maximised) and by the number of switches required to achieve it (to be minimised).

2.1 The Two-Phase Approach

We show how to use planning to construct policies automatically for multiple battery problems, where load is modelled probabilistically using known distributions for load size, load duration and load frequency (or equivalently, the gaps between successive loads). Outside our approach, the best deployed solutions typically deliver less than 80% efficiency, while the best published solutions deliver less than about 95% efficiency. We show that our approach, based on construction of optimising solutions to Monte Carlo sampled problem instances and their use in the construction of appropriate policies, produces robust solutions that deliver better than 99% efficiency, while using smaller numbers of battery switches than published policies. The reduction in the number of switches required is very important in practice, because it reduces wear and tear and avoids overheating. This is a simulation result that has yet to be tested on a physical battery setup. We use the Kinetic Battery Model, a well-established continuous battery model, as the basis of our approach. The model is a first-order approximation of battery behaviour, but analysis has shown that it is one of the most accurate analytic models available [Jongerden and Haverkort, 2009].

Our approach is as follows. We define a PDDL+ model of the Kinetic Battery Model and generate a large number (thousands) of deterministic battery loads. In each case, we use a deterministic continuous planner to decide how to service the load. We use a discretise-and-validate planning approach [Della Penna *et al.*, 2009], that works by proposing a discretisation and testing to see whether the problem is rendered solvable with respect to the original continuous model. If not, a new discretisation is tried and this process continues until the problem can be solved (or the attempt is terminated). We use a sophisticated mechanism for exploring the discretisation space which exploits the heuristic that the total charge of any battery decreases monotonically (a safe assumption when batteries cannot be recharged). An important aspect of our approach is that we use *variable*, rather than *uniform*, discretisation. This yields important properties of our solutions, such as reduced switching.

This part of the process results in a training set containing thousands of high quality solutions to different fixed loads. The next step is to learn a policy from the training set. The policy will be able to determine how best to service any unseen, probabilistic, load that it encounters in the future. To acquire the policy we use the J48 classifier provided by the WEKA framework [Hall *et al.*, 2009]. This classifier learns a mapping from the "state" of the battery suite to the possible actions, and produces a decision tree that can be used to quickly determine the best action to use in any state. The work thus breaks down into two phases: building a high quality training set, which relies on a good model of battery

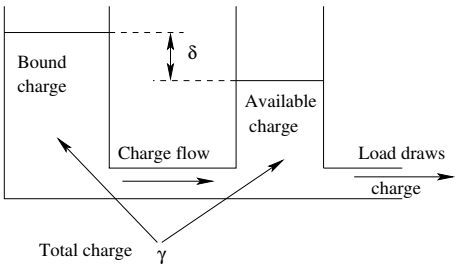

Figure 1: Kinetic Battery Model

behaviour and a sophisticated continuous planner, and constructing the policy from the training data using a classifier.

2.2 The Kinetic Battery Model

The battery model we use is the Kinetic Battery Model (KiBaM) [Manwell and McGowan, 1994], in which the battery charge is distributed over two wells: the available-charge well and the bound-charge well (see Figure 1). A fraction c of the total charge is stored in the available-charge well, and a fraction $1-c$ in the bound-charge well. The available-charge well supplies electrons directly to the load ($i(t)$), where t denotes the time, whereas the bound-charge well supplies electrons only to the available-charge well. The charge flows from the bound-charge well to the available-charge well through a "valve" with fixed conductance, k. When a load is applied to the battery, the available charge reduces, and the height difference between the two wells grows. When the load is removed, charge flows from the bound-charge well to the available-charge well until the heights are equal again.

To describe the discharge process of the battery, we follow Jongerden *et al.* [2009], adopting coordinates representing the height difference between the two wells, δ, and the total charge in the battery, γ. The change in both wells is given by the system of differential equations (1), with solutions (2):

$$\frac{d\delta}{dt} = \frac{i(t)}{c} - k'\delta \qquad \frac{d\gamma}{dt} = -i(t) \tag{1}$$

$$\delta(t) = \frac{i}{c} \cdot \frac{1-e^{-k't}}{k'} \qquad \gamma(t) = C - it. \tag{2}$$

where $k' = k/(1-c)c$, $\delta(0) = 0$ and $\gamma(0) = C$, and C is the total battery capacity.

This model is less sophisticated than that used by Benini *et al.* [2001], but a comparison of battery models by Jongerden and Haverkort [2009] concludes that the KiBaM is the best for performance modelling.

2.3 A PDDL+ Battery Model

PDDL+ [Fox and Long, 2006] is an extension of the standard planning domain modelling language, PDDL, to capture continuous processes and events. The dynamics of KiBaM can be captured very easily in PDDL+. In Figure 2 we show the two processes, consume and recover, that govern the behaviour of cells and the event triggered by attempting to load a cell once its available charge is exhausted. In addition, there is a durative action of variable duration that allows the planner to use a cell over an interval. The two processes are active whenever their preconditions are satisfied, meaning that they usually execute concurrently. Together, they model both the draining of charge and the recovery that are described in the differential equation $d\delta/dt$. An event is triggered if there is ever a positive load and no active service.

```
(:process consume
 :parameters (?c - cell)
 :precondition (switchedOn ?c)
 :effect (and (decrease (gamma ?c)
              (* #t (load)))
    (increase (delta ?c)
         (* #t (/ (load) (cParam ?c))))))

(:process recover
 :parameters (?c - cell)
 :precondition (>= (delta ?c) 0)
 :effect (and
    (decrease (delta ?c)
      (* #t (* (kprime ?c) (delta ?c))))))

(:event cellDead
 :parameters (?c - cell)
 :precondition
    (and (switchedOn ?c)
         (<= (gamma ?c)
            (* (- 1 (cParam ?c)) (delta ?c))))
 :effect (and (not (switchedOn ?c))
              (dead ?c)))
```

Figure 2: Part of PDDL+ encoding of KiBaM dynamics

The use of PDDL+ as our modelling language grants several benefits. Firstly, it allows us to use our well-established plan validation tool VAL [Howey *et al.*, 2004], to validate discretised solutions analytically against the continuous model. Secondly, it provides us with a semantics for our model in terms of a timed hybrid automaton (following Fox and Long [2006]).

3 Battery Usage Planning

In most real battery usage problems the load profile is generated by external processes, typically controlled directly or indirectly by user demands. These demands can often be modelled probabilistically, reflecting typical patterns of use. In our work we assume that the profiles are drawn from a known distribution. The consequence is that the problem is not a deterministic optimisation problem, but a probabilistic problem requiring a policy.

The problem can be cast as a continuous Markov Decision Process, in which the states are characterised by the states of charge of the batteries, the current load and the currently active battery. Battery switching actions are deterministic, but the events that cause load to change are not. The time between events is governed by a stochastic process, but the timing of switching actions is controllable. There is also a non-deterministic action, *wait(T)*, where T is a time interval, which causes a transition to a state in which time has advanced, the state of charge of battery B is updated according to the battery model and the load might be different (according to the probability distribution governing loads). The interpretation of the action is that it advances time to the next event, which will be when a battery is depleted of available charge, or when the load changes, or when T time has passed, whichever is first.

A variety of approaches have been proposed for solv-

ing continuous Markov Decision Processes. Meuleau *et al.* [2009] propose hybrid AO* search, using a dynamic programming approach to guide heuristic search for problems involving continuous resources used by stochastic actions. This approach does not handle time-dependent resource consumption, but it appears that the above MDP could be modelled for solution by this approach. The authors give empirical data for solution of problems with up to 25,000 states. Our model, with an appropriate discretisation, contains more that 10^{86} states for 8 batteries. Mausam and Weld [2008] describe a planner for concurrent MDPs, which are MDPs with temporal uncertainty. Again, these problems are similar to ours, although their planner does not manage continuous time-dependent resources, so is not directly applicable to our problem. Furthermore, the largest problems they consider contain 4,000,000 states and take more than an hour to solve.

In solving very large MDPs, researchers have identified a variety of techniques that can help to overcome the prohibitive cost of policy iteration or value iteration, the classical techniques for solving MDPs. In general, these techniques approximate the solution, often focussing on those parts of the policy that apply to states that are likely to be visited along the trajectory. Relevant techniques are discussed by Bertsekas and Tsitsiklis [1996]. We use a variant of hindsight optimisation (see *eg* Chang *et al.* [2000]), in which we solve a deterministic sampled problem, using UPMurphi [Della Penna *et al.*, 2009], modified to generate a single ideal trajectory for the problem instance. Using a collection of such samples as our base, we then learn a classifier that characterises the policy for the part of the space we have sampled. This approach is similar to other work built on the use of machine learning applied to policy rollout samples, particularly due to Fern, Yoon and Givan [2006; 2007], except that their approach uses a randomly generated starting point, whereas we start with a good quality plan, and their work addresses only propositional domains while we are concerned with continuous problems.

To extend the learned classifier to a complete policy involves ensuring that an action is assigned to every possible state. This can be achieved by adding some default behaviour to cover states that are otherwise not handled by the classifier, or else by managing run-time errors in the use of an incomplete policy in a way appropriate to the application.

3.1 Plan Search with Variable Discretisation

We now illustrate the way in which the range of differently sized duration intervals can lead to significant benefits in the size of the set of visited nodes in the search space, compared with using a fixed duration increment.

Consider the load profile shown at the top of Figure 3. The planning problem is to service the whole load profile within a temporal horizon that is equal to the duration of the profile. The set of actions is $\mathcal{A} = \{\texttt{useC1}, \texttt{useC2}, \texttt{wait}\}$ where the former actions refer to the cell being used while the latter one is applicable when there is no active service. The set of durations we use for this example is $\mathcal{D} = \{0.01, 0.4, 0.5, 1.0\}$ (measured in minutes). In practice, to define the set of durations we start with a minimum value given by the time required for the decision making process, then we add exponentially increasing values up to a maximum duration given by the longest interval between different events (i.e., load variations). In particular, the smallest duration is included in order to handle very sensitive interactions.

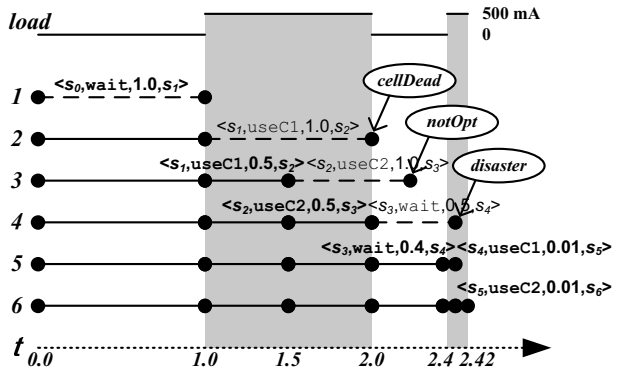

Figure 3: Example of search using variable discretisation

In the initial state s_0 there is no load and no active service and both cells have a limited initial capacity. In this setting, the plan search with variable discretisation proceeds as follows:

1. No cell is used for a period of 1 minute (when the load is idle). The corresponding transition is shown in Figure 3.
2. After 1 minute a load is applied and cell 1 is used. This corresponds to transition $\langle s_1, \texttt{useC1}, 1.0, s_2\rangle$. However, for sake of simplicity, let us assume that, due to their limited capacity, cells cannot be used continuously for 1 minute. The transition is thus not valid and a shorter duration has to be considered.
3. Cell 1 is used for 0.5 minutes. Then, since a load is still applied, the second cell is used. As before, the transition $\langle s_2, \texttt{useC2}, 1.0, s_3\rangle$ can be considered, but in this case there would be an active service and no load.
4. Cell 2 is used for 0.5 minutes. In the next period no load is applied, then no cell is used. The transition $\langle s_3, \texttt{wait}, 0.5, s_4\rangle$ is considered, but it would lead to a positive load and no active service, so the duration of action `wait` has to be reduced to 0.4.
5. To service the last load period of 0.02 minutes, cell 1 could be used. However, in this sample instance let us assume that the remaining charge in cell 1 allows it to service only 0.01 minutes. So, finally, cell 2 is used until the end of the load profile.

The validity of a transition is dynamically checked during the search since invalid transitions trigger specific events (e.g. event `cellDead` is triggered at step 2 and event `disaster` is triggered at step 4) which, in turn, violates the invariant conditions of corresponding actions (a cell must not die during use). Moreover, with variable discretisation only 6 states have to be visited in order to reach the goal, while using a uniform discretisation it is necessary to explore at least 242 states since the finest discretisation of 0.01 must be used in order to correctly handle the interactions in steps 5 and 6.

A further benefit of the use of differently sized durations in the discretisation is that favouring longer durations reduces

load profile	best-of-two lifetime		UPPAAL-KiBaM lifetime		DD-KiBaM lifetime (visited states)		8 batteries B_2 lifetime (number of switches)		
	B_1	B_2	B_1	B_2	B_1	B_2	best-of-8	DD	DD-Policy
CL_250	12.16	46.92	12.04	N/A	**12.14** (194)	**46.91** (691)	310.6 (31072)	**307.6 (485)**	**307.6 (992)**
CL_500	4.59	12.16	4.58	N/A	**4.59** (116)	**12.14** (194)	134.7 (13472)	**133.4 (266)**	**133.4 (571)**
CL_alt	7.03	21.26	6.48	N/A	**7.03** (136)	**21.20** (350)	192.8 (19280)	**190.8 (355)**	**190.8 (806)**
ILs_250	44.79	132.8	40.80	N/A	**44.76** (552)	**132.70** (1068)	660.7 (33076)	**654.1 (495)**	**654.1 (904)**
ILs_500	10.82	44.79	10.48	N/A	**10.80** (131)	**44.76** (552)	308.7 (15476)	**305.7 (293)**	**305.7 (513)**
ILs_alt	16.95	72.75	16.91	N/A	**16.92** (159)	**72.55** (599)	424.8 (21280)	**420.6 (357)**	**420.6 (614)**
ILl_250	84.91	216.9	78.96	N/A	**84.88** (488)	**216.8** (1123)	1008.9 (33692)	**998.8 (471)**	**998.8 (822)**
ILl_500	21.86	84.91	18.68	N/A	**21.85** (173)	**84.88** (488)	480.9 (16090)	**476.1 (295)**	**476.1 (597)**

Table 1: System lifetime (in minutes) for all load profiles according to different battery usages

the number of switches in the solutions we generate, leading to solutions that are better in practical terms than those based on a high frequency switching between batteries, as is shown in subsequent results.

3.2 Performance on Deterministic Load Problems

We now present a first set of experimental results to show the performance of our solver on the deterministic battery usage optimisation problem. We use the same case study proposed by Jongerden *et al.* [2009], where two types of jobs are considered, a low current job (250 mA) and a high current job (500 mA), according to the following load profiles:

- *continuous* loads: one load with only low current jobs (CL_250), one with only high current jobs (CL_500) and one alternating between the two;
- *intermittent* loads with *short idle periods* of one minute between the jobs: one with only low current jobs (ILs_250), one with only high current jobs (ILs_500), and one alternating between the two;
- *intermittent* loads with *long idle periods* of two minutes between the jobs: one with only low current jobs (ILl_250) and one with only high current jobs (ILl_500).

As a first step, we used these load profiles to validate our variable-range discretisation KiBaM model (DD-KiBaM), and to find an appropriate discretisation for the continuous variables involved in the system dynamics (*i.e.* variables γ and δ and process durations). To do this we used VAL to validate solutions for the discretised model against the continuous model, using single cell batteries. Following Jongerden *et al.* [2009], we considered two battery types, one with capacity 5.5 Ah (B_1) and one with capacity 11 Ah (B_2). Both battery types have the same parameters: $c = 0.166$ and $k' = 0.122\text{h}^{-1}$. We discretised γ and δ, rounding them to 0.00001, and, for all the load profiles above and for both battery types, we obtained the same lifetimes computed with the original KiBaM and validated by Jongerden and Haverkort [2008].

To generate the scheduling plans for multi-cell batteries, we used the approach described in Section 2.1 and the set of durations $\mathcal{D} = \{0.01, 0.02, 0.05, 0.1, 0.25, 0.5, 1.0\}$. We compared our solutions to those obtained using the UPPAAL-based approach. The resulting lifetimes are shown in Table 1 where the second column shows the theoretical upper bound given by an extremely high-frequency switching. In all load profiles considered we observe that our approach outperforms significantly the UPPAAL-based one, providing solutions that achieve more than 99% efficiency compared with the theoretical limit. The key points described in the preceding parts of this section allow the resulting search to efficiently prune the state space and quickly find the solutions. In particular, by using variable discretisation it is possible to consider a much finer discretisation for variables γ and δ than is used in [Jongerden *et al.*, 2009] and to handle very sensitive interactions. This is crucial, particularly when the available charge in the cells is almost exhausted. Jongerden *et al.* [2009] describe their plans as optimal, but it is important to note that this is only with respect to the discretisation that they use; a finer-grained discretisation offers the opportunity for a higher quality solution to be found at the cost of a much larger state space. Despite the very large state space our model creates, the solver visits a very small collection of states (as shown in the table). These problems are all solved in less than a second.

Moreover, when dealing with larger batteries of type B_2, the state space becomes so large that any exhaustive approach is infeasible. Indeed, Jongerden *et al.* [2009; 2008] were not able to handle this second case. We also found high quality solutions for batteries of type B_2, obtaining a huge improvement over the V_{max} policy.

Our results on an 8 battery system, presented in Table 1, show that we can scale effectively to much larger problems. Notice that the number of switches we use to produce the results is very significantly smaller than the best-of-8 policy (V_{max}), however the resulting solutions achieve more than 99% efficiency. The final column, labelled *DD-Policy*, shows the performance of the dynamic discretisation-based policies applied to these load profiles. These generate slightly worse performance in switches than the deterministic plans shown in the penultimate column, but this degradation is not significant when compared to the switching rate required by the best-of-8 policy.

3.3 Results from Policies

In order to use the decision tree we embedded the WEKA classes for loading the classification model into our battery simulation framework. The model for the 8 battery case is represented by a tree with 61 levels and consists of 7645 nodes, each one containing a comparison between one of the state variables and a threshold. Applying this decision tree to determine which battery to load at each decision point takes negligible time.

load	best-of-8		DD-Policy	
profile	time$_{(\sigma)}$	$sw_{(\sigma)}$	time$_{(\sigma)}$	$sw_{(\sigma)}$
R100	792.6$_{(15.50)}$	71383$_{(1379)}$	**786.2**$_{(15.40)}$	**1667**$_{(161)}$
R250	369.8$_{(1.91)}$	28952$_{(853)}$	**366.7**$_{(2.02)}$	**1518**$_{(143)}$
R500	226.7$_{(2.13)}$	14671$_{(512)}$	**224.6**$_{(2.27)}$	**987**$_{(122)}$
R750	188.3$_{(0.80)}$	11519$_{(463)}$	**186.4**$_{(0.70)}$	**302**$_{(33)}$

Table 2: Average system lifetime and number of switches for stochastic load profiles for 8 battery systems

To evaluate the performance of the policy we considered four probability distributions with different average value for the load amplitude, namely 100, 250, 500, 750 mA. For each distribution we generated 100 stochastic load profiles and we used the policy to service them. Table 2 shows the average value and standard deviation for the system lifetime and the number of switches obtained, comparing the theoretically optimal upper bound (generated by a best-of-8 policy with unlimited frequency switching) with our *Determinised-Discretised* (DD) policy.

We observe that our policy achieves more than 99% efficiency compared with the theoretical upper bound given by the best-of-8 policy executed at very high frequency (recall that very high frequency switching is infeasible in practice).

4 Conclusions and Future Work

In this paper we have presented an effective solution to an increasingly important multiple battery management problem. Our solution achieves better than 99% efficiency, compared to a maximum of 95% achieved in the literature. Although this margin is small, in many applications a small margin can be of considerable added value. We adapt several existing technologies for automated planning, to solve a problem that can be seen as an MDP. We use a form of hindsight optimisation, generating samples of determinised load profiles and solving these problems using an optimal deterministic solver, before combining the solutions to form a policy. We use a special variable-range discretisation to solve a non-linear continuous optimisation problem with very high accuracy, while exploring a very small proportion of the state space. Our policy construction approach adapts the use of machine learning to construct a classifier.

Our approach is scalable and effective. The elements of our technique that are most tailored to our problem are the selection of the discretisation range and the search heuristic. We are currently exploring techniques to automatically generate an appropriate discretisation range for a problem, based on an analysis of the problem instance characteristics and the nature of the dynamics in the domain description.

Acknowledgements

The authors wish to thank Marijn Jongerden and Boudewijn Haverkort for introducing them to this problem and for early discussions on approaches to solving it.

References

[Behrmann *et al.*, 2001] G. Behrmann, A. David, K.G. Larsen, O. Möller, P. Pettersson, and W . Yi. UPPAAL – Present and Future. In *Proc. 40th IEEE Conf. on Decision and Control*, 2001.

[Benini *et al.*, 2001] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi. Discrete-time battery models for system-level low-power design. *IEEE Trans. Very Large Scale Int. Sys.*, 9(5):630–640, 2001.

[Benini *et al.*, 2003] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi. Scheduling battery usage in mobile systems. *IEEE Trans. Very Large Scale Int. Sys.*, 11(6):1136–1143, 2003.

[Bertsekas and Tsitsiklis, 1996] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Chang *et al.*, 2000] H.S. Chang, R. Givan, and E.K. P. Chong. Online scheduling via sampling. In *Proc. AI Planning and Scheduling (AIPS)*, pages 62–71, 2000.

[Della Penna *et al.*, 2009] G. Della Penna, B. Intrigila, D. Magazzeni, and F. Mercorio. UPMurphi: a tool for universal planning on PDDL+ problems. In *Proc. 19th Int. Conf. Aut. Planning and Scheduling (ICAPS)*, pages 106–113, 2009.

[Fern *et al.*, 2006] A. Fern, S. Yoon, and R. Givan. Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. *J. AI Res. (JAIR)*, 25:75–118, 2006.

[Fox and Long, 2006] M. Fox and D. Long. Modelling Mixed Discrete-Continuous Domains for Planning. *J. AI Res. (JAIR)*, 27:235–297, 2006.

[Fox *et al.*, 2011] M. Fox, D. Long, and D. Magazzeni. Automatic Construction of Efficient Multiple Battery Usage Policies. In *Proc. Int. Conf. on Aut. Planning and Scheduling (ICAPS)*, 2011.

[Hall *et al.*, 2009] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[Howey *et al.*, 2004] R. Howey, D. Long, and M. Fox. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *Proc. Int. Conf. on Tools with AI*, pages 294–301, 2004.

[Jongerden and Haverkort, 2008] M.R. Jongerden and B.R. Haverkort. Battery Modeling. Technical Report TR-CTIT-08-01, Centre for Telematics and Inf. Tech., U. Twente, 2008.

[Jongerden and Haverkort, 2009] M.R. Jongerden and B.R. Haverkort. Which battery model to use? *IET Software (Special Issue on Performance Engineering)*, 3(6):445–457, 2009.

[Jongerden *et al.*, 2009] M. Jongerden, B. Haverkort, H. Bohnenkamp, and J.-P. Katoen. Maximizing system lifetime by battery scheduling. In *Proc. 39th IEEE/IFIP Int. Conf. on Dependable Sys. and Net. (DSN)*, pages 63–72, 2009.

[Manwell and McGowan, 1994] J.F. Manwell and J.G. McGowan. Extension of the Kinetic Battery Model for Wind/Hybrid Power Systems. In *Proc. 5th European Wind Energy Association Conf. (EWEC)*, pages 284–289, 1994.

[Mausam and Weld, 2008] Mausam and D. S. Weld. Planning with Durative Actions in Stochastic Domains. *J. AI Res. (JAIR)*, 31:33–82, 2008.

[Meuleau *et al.*, 2009] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, and Mausam. A Heuristic Search Approach to Planning with Continuous Resources in Stochastic Domains. *J. AI Res. (JAIR)*, 34:27–59, 2009.

[Yoon *et al.*, 2007] S. Yoon, A. Fern, and R. Givan. Using Learned Policies in Heuristic-Search Planning. In *Proc. Int. Joint Conf. on AI (IJCAI)*, pages 2047–2053, 2007.