Dual Balancing of SoC/SoT in Smart Batteries using Reinforcement Learning in Uppaal Stratego

Kristjansen, Martin; Kulkarni, Abhijit; Jensen, Peter Gjøl; Teodorescu, Remus; Larsen, Kim Guldstrand

Published in:
IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society

DOI (link to publication from Publisher):
10.1109/IECON51785.2023.10311828

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

Dual Balancing of SoC/SoT in Smart Batteries using Reinforcement Learning in Uppaal Stratego

Martin Kristjansen[1], Abhijit Kulkarni[2], Peter Gjøl Jensen[1], Remus Teodorescu[2], Kim Guldstrand Larsen[1]
mk@cs.aau.dk, eku@energy.aau.dk, pgj@cs.aau.dk, ret@energy.aau.dk, kgl@cs.aau.dk
[1]*Department of Computer Science, Aalborg University, Denmark*
[2]*Department of Energy, Aalborg University, Denmark*

***Abstract*—Battery packs in electric vehicles are managed by battery management systems that influence the state of charge among the cells in the pack, where such systems have received much attention in research. More recently, balancing the temperature among the cells has become a research topic. In our work, we consider a dual-balancing problem where we aim to balance both the parameters of the state of charge and temperature. We consider a Smart Battery Pack, where individual cells can be bypassed, meaning that no current is going to or from the cell, which allows the cell to cool off while the cell does not charge or discharge. Moreover, a smart battery pack can estimate each cell's characteristics, which, in turn, can be used to define a model of cell and battery pack behavior. We conduct experiments using the model of a battery pack where each cell differs in its configuration as an effect of aging. For such a pack with heterogeneous cells, we use Q-Learning in Uppaal Stratego to synthesize a controller that maximizes the time spent in a balanced state, meaning that all cells' states are within a specific range of each other. We show significant improvements in two aspects compared with two threshold-based controllers that balance either state of charge or temperature. The synthesized controllers are only unbalanced with the state of charge between 1-4% of the time and for temperature between 15-20% of the time. The threshold-based controllers are either unbalanced for the state of charge for as much as 37% of the time or for temperature for as much as 44% of the time. Finally, the maximum variations of state of charge and temperature among the cells are decreased.**

***Index Terms*—Start Battery Pack, Digital Twin, SoC & SoT, Dual-Balancing, Reinforcement Learning**

I. Introduction

The amount of electric vehicles is only growing, which requires an ever-increasing use of Li-ion batteries. These batteries are managed by a Battery Management System (BMS) with the responsibility to balance the use of the batteries in the battery pack and to provide protection in case of undesirable events such as thermal runaway. A conventional BMS aims to balance the State of Charge (SoC) and relies on an active cooling system to protect against thermal runaway. Moreover, the BMS typical for electric vehicles only balances the SoC during charging and not during discharging.

A Smart Battery (SB) Pack [1] has the potential to balance both SoC and State of Temperature (SoT) with a so-called bypass. Moreover, an SB Pack has local computation power, so sensor data can be used to estimate individual cells' characteristics [2]. With the help of the bypass, a cell can decide to either carry the load current or not, thus making it possible to balance SoC and SoT during both charging and discharging. Other research has investigated balancing an unbalanced SB Pack with homogeneous cells without a cooling system using bypasses [3]. However, the cells in that paper do not transfer heat to the environment or other cells, meaning that they reach significantly higher temperatures than they would in a real system.

Other research focuses on how to make a digital twin of a battery, which then can be used to evaluate how aspects such as SoC and State of Health (SoH) are affected by different control algorithms [4], where they use Simulink as the modeling framework. Other work also makes attempts to balance both SoC and temperature in electric vehicles by using Model Predictive Control (MPC) [5]. Here, the model is implemented and simulated in Matlab, where the control algorithms can manipulate the voltage of the cells by predicting how the behavior will be in the near future. Another approach with MPC uses the modeling tool UPPAAL STRATEGO to model the battery life-cycles of a nano-satellite in order to synthesize a controller to optimize its lifetime [6]. Other areas where UPPAAL STRATEGO has been used to synthesize an optimal controller via reinforcement learning are, e.g., controllers for floor heating in private homes [7] and for managing water ponds [8]. In all cases using UPPAAL STRATEGO, the models are defined as non-linear systems where a controller optimizing a given criterion is hard to find.

In this work, we investigate how reinforcement learning in UPPAAL STRATEGO can synthesize a control strategy for a BMS to balance both SoC and SoT in a battery pack consisting of heterogeneous cells, meaning that they all have slightly different characteristics resulting in slightly different behavior. At the same time, we consider that the cooling of the cells can affect the individual cells differently, but we cannot measure the cooling temperature for any given cell. Making actions to balance either SoC or SoT can have a negative impact on the other [9], and this problem becomes even more challenging to solve when the exact cooling temperatures of the cells are unknown, giving the system a stochastic nature. It is a complicated problem to solve optimally, so we investigate how reinforcement learning can aid in synthesizing a controller that helps balance both SoC and SoT given a system with non-

This work was supported by the S40S Villum Investigator Grant (37819) from VILLUM FONDEN, the Smart Battery Villum Investigator Grant (222860) from VILLUM FONDEN, and the OPENSRUM grant (223286) from MSCA-IF European Commission.

linear and stochastic behavior. Using reinforcement learning also helps us avoid designing the controllers by hand, which might be far from the ideal balancing algorithm.

Given that a smart battery pack has the computation power available to estimate the individual cells' states, we imagine how an electric vehicle overnight could synthesize a new balancing controller for its battery pack based on the newly estimated characteristics. This would help the battery pack as a whole in terms of uniform aging, where no cells would reach their end-of-life much faster than the other cells, which has the potential to extend the lifetime of all cells and, therefore, the entire battery pack.

In our balancing experiments during discharge, we contribute by showing how our approach based on reinforcement learning significantly outperforms two hand-made controllers designed to balance either SoC or SoT. We show that we can significantly improve the time spent in a balanced state and reduce the maximum difference in SoC and SoT.

II. Battery Model

A smart battery pack consists of n-cells connected in series as shown in fig. 1. Each i^{th} cell has a half-bridge topology across it with two complementary switches S_i and S_i'. The bypass operation in a cell in the pack is when the left switch (e.g., S_i') is turned on, resulting in bypassing the cell without affecting the load current. Thus, if a cell has low SoC in the smart battery pack during discharge mode, it can be bypassed to prevent its rapid discharge and allow other cells to catch up. Similarly, if a cell has a higher temperature than the average, it can be bypassed to cool by transmitting its excess heat to the surroundings.

The bypass operation naturally leads to a pulsed current operation of the cells. It is shown in earlier works [10] that pulsed charging can increase the cells' lifetime. Thus, the smart battery architecture is used to improve the cycle life of the Li-ion battery pack. In this work, we extend its use to perform SoC and SoT balancing under practical conditions of mismatch in the cell characteristics.

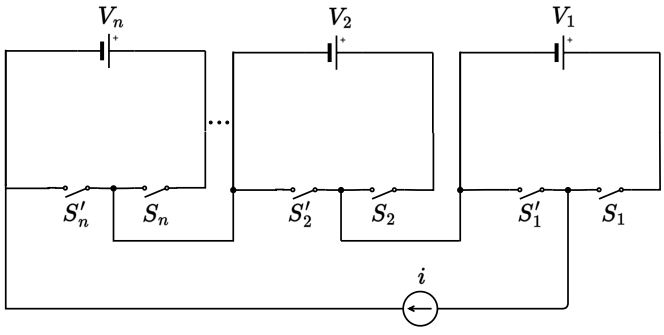

Fig. 1. Smart Battery Pack

The individual cells we model as a circuit with a single Resistor-Capacitor (RC) in series with the internal resistance, as shown in fig. 2, whose behavior can be described with ordinary differential equations (ODEs). This model is the simplest form of the equivalent circuit models (ECM) used for Li-ion cells. The key parameters of the model, R_0, R_1, and C_1, can be different for different cells, affecting their behavior in a pack. For example, the parameters R_1 and C_1 change with aging. The cell's Open Circuit Voltage (OCV) is a function of the SoC, which is modeled as a polynomial function in SoC based on the laboratory test data. Note that the SoC is just an integration of the applied current to the cell divided by the cell's capacity.

To properly model the temperature of the cell, we define the terminal voltage (V_t) of fig. 2 with the following equation:

$$V_t = OCV - iR_0 - v_{c1} \tag{1}$$

where i is the current and v_{c1} is the voltage drop across the parallel combination of R_1 and C_1. The current in discharge mode is considered positive. Thus, eq. (1) describes the cell operation during discharge. v_{c1} is then further defined with the following differential equation:

$$\frac{dv_{c1}}{dt} = \frac{-v_{c1}}{R_1 C_1} + \frac{i}{C_1} \tag{2}$$

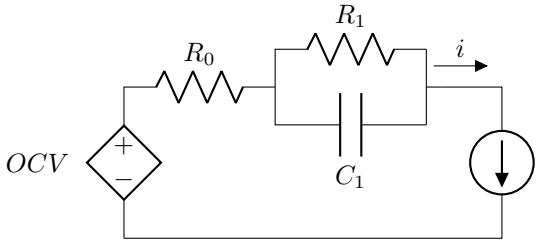

Fig. 2. A Resistor-Capacitor circuit.

We use a thermal model to model the temperature rise at the cell level. The current i in the cell produces heat due to losses, which can also be represented via the ohmic components R_0 and R_1 in fig. 2. The rate of temperature rise in the cell dT/dt depends on the heat generated (Q), convective heat coefficient (h), mass (m), and heat capacity (C_p). The following thermal equations represent this:

$$\frac{dT}{dt} = \frac{Q - hA(T_{sur} - T_{amb})}{mC_p} \tag{3}$$

The heat generated Q is a function of the power losses defined as:

$$Q = i(V_t - OCV) + i \times T\frac{\delta OCV}{\delta T} \tag{4}$$

The latter term in eq. (4) can be ignored as its effect is negligible compared to the first term as the OCV has a strong correlation with SoC than the temperature [11].

Given all this, we can define the state of a cell as the following:

Definition 1 (Battery State)**.** *The state of a cell is given by the tuple $\mathcal{B} = (C, S, T, R_0, R_1, C_1)$, where C is the cell's capacity, S is the current SoC, T is the current SoT, and where R_0, R_1, and C_1 are those shown in fig. 2.*

We then use the definition of the individual cells to define the state of a battery pack formally:

Definition 2 (Battery Pack State)**.** *The state of a smart battery pack is given as $\mathcal{BP} = \{\mathcal{B}_1, \ldots, \mathcal{B}_N\}$, which is a multi-set of N cells.*

The set $\mathcal{BP}$ then consists of cells that can be identical or different in their characteristics, such as the parameters used in the fig. 2, the cell's capacity, or the cell's current SoC and SoT. We write $\mathcal{B}_i$ to denote the pack's i$^{\text{th}}$ cell. As a shorthand, we write S_i and T_i for the i$^{\text{th}}$ cell's current SoC and SoT, respectively.

III. Optimization Criteria

We aim to balance the SoC and SoT with the bypass action. For a system to be in SoC and thermal balance, the span of SoCs and temperatures must not exceed a given threshold value. We can write this aim formally as:

Definition 3 (SoC Balance)**.** *Given a battery pack $\mathcal{BP}$ of size N and a threshold value $\theta_{SoC} \in \mathbb{R}$, the battery pack is in SoC balance iff $\forall i, j \in \{1, 2, .., N\}.|S_i - S_j| \leq \theta_{SoC}$, meaning that the maximum difference of SoC is θ_{SoC} percentage points or lower.*

In our case, we work with a threshold value $\theta_{SoC} = 2$, which is an appropriate choice since the state-of-the-art often offers a balance between 2-3 percentage points [12]. However, the threshold value can be chosen to whichever value fits in the context. Moreover, we have a similar definition for being in balance with SoT:

Definition 4 (Thermal Balance)**.** *Given a battery pack $\mathcal{BP}$ of size N and a threshold value $\theta_{SoT} \in \mathbb{R}_{\geq 0}$, the battery pack is in thermal balance iff $\forall i, j \in \{1, 2, .., N\}.|T_i - T_j| \leq \theta_{SoT}$, meaning that the maximum difference of temperature is θ_{SoT} degrees or lower.*

For the remainder of the paper, we use the threshold value $\theta_{SoT} = 1$ implying a tolerated divergence of up to $1°C$ in order to be in balance.

Ideally, the battery should always be balanced, but that might only sometimes be possible. It is possible to have situations where the bypass action helps one of the definitions of balance but not the other. We, therefore, have the primary goal to be in balance, followed by being as close as possible to a balanced state where the spans of SoC and SoT are limited as much as possible.

To later synthesize controllers, our definitions 3 and 4 are not well suited to use directly given their binary nature of whether a system is in balance or not; however, we want to distinguish between unbalanced states since some a clearly closer to be in balance than others. To accommodate a distinguishing of unbalanced states, we define what is known as cost functions, which help determine how much it "costs" to be in that state. These cost functions are then used as the optimization criteria for the synthesis algorithm, as the goal will be to minimize the expected value of a given fitness over time.

We define the cost function δ_{SoC} for an individual cell's SoC as:

$$\delta_{SoC}(S, S_\sigma) = \begin{cases} 0 & \text{if } |S - S_\sigma| \leq \frac{\theta_{SoC}}{2}, \\ (|S - S_\sigma| - \frac{\theta_{SoC}}{2})^2 & \text{otherwise.} \end{cases} \tag{5}$$

Here, δ_{SoC} takes two arguments: the cell's current SoC (S) and the mean of the current SoC (S_σ). The function also depends on the a priori given threshold θ_{SoC} that defines when the system is unbalanced. In eq. (5), we compare the given cell's current SoC with the mean value of all the cells' SoCs. If the cell is within half the current threshold's value, we define that as a good fit and return a cost of 0. Otherwise, we calculate the squared error of the difference to the balancing threshold. This gives a proportionally larger penalty for more significant deviations for cells far from the global mean. It is worth noting that the function does not follow our earlier definition of balance since it focuses on a cell being close to the mean rather than based on the entire system's state of balance. The assumption is that optimizing the individual cells' distance to the mean also optimizes the system's time in balance. Finally, if the system is unbalanced, one or more cells are guaranteed a non-zero cost value.

To calculate the total cost of SoC at any given time, we define the cost function Δ_{SoC} for all cells' SoCs as:

$$\Delta_{SoC}(\vec{S}) = \sum_{i=1}^{\|S\|} \delta_{SoC}\left(S_i, \frac{\sum_{j=1}^{\|S\|} S_j}{\|S\|}\right) \tag{6}$$

Here, the input $\vec{S}$ is a vector of SoCs. The result is a summation of the individual cell's fitness function.

We have two equivalent cost functions for the temperature, $\delta_{SoT}(T, T_\sigma)$ and $\Delta_{SoT}(\vec{T})$, where the only difference is that the inputs to the functions are the current temperatures and that the threshold value used is θ_{SoT}.

We then combine them into one cost function for a smarts battery pack's state at any given time:

$$\Delta(\vec{S}, \vec{T}) = \Delta_{SoC}(\vec{S}) * (1 - \alpha) + \Delta_{SoT}(\vec{T}) * \alpha \tag{7}$$

where α is a constant within in range $[0, 1]$. This allows us to give different weights to the two different fitness functions, as different weights are likely to affect the controller synthesis differently, ending up with controllers that might take actions that favor one type of balancing more than the other. Of course, with an α of 0 or 1, the synthesis only considers the fitness of SoC or SoT, respectively.

Given $\Delta(\vec{S}, \vec{T})$, the function can be integrated to get the fitness of taking a series of actions (bypass/no bypass). The rationale is that the lower value a synthesized controller produces, the better the controller fits to solve the optimization problem. Then, the optimization criterion of the synthesis algorithm is to minimize the expected value produced by the integral of $\Delta(\vec{S}, \vec{T})$.

IV. Controller Synthesis

In this work, we leverage Model Predictive Controller (MPC) via the modeling and synthesis tool Uppaal Stratego [13]. Given a (stochastic) model of the system to control, Uppaal Stratego can synthesize a (near-)optimal controller using a combination of partition-refinement and Q-learning [14], which are techniques designed particularly for synthesis on models with complex stochastic and non-linear dynamics. In this paper, we model the battery pack and effects of a bypass in Uppaal Stratego using Networks of Hybrid Stochastic Priced Timed Games Automata, a high-level modeling formalism supporting, e.g., behavioral descriptions as (networks of) automata and C-style code. As part of the model, we define what actions are available for any synthesized controller, leaving out the component deciding when and which cell to bypass. Instead, we rely on the before-mentioned synthesis techniques for constructing a controller. We refer the interested reader to [13] for details on Uppaal Stratego and to [14] for a thorough description of the learning algorithm.

To synthesize a controller, the template used is shown in fig. 3 as an automaton (in this case, a state machine with clocks and helper functions), where the red location is the initial location from which it makes a bypass-decisions. In contrast, the yellow location is a waiting location. The choice in the red location is between two different bypass actions and the choice not to bypass. Rather than allowing the controller freedom of which cell to bypass, we delimit the controller to the cells with the most deviating SoC and SoT. The assumption is that if the controller should make a bypass, a (near-)optimal choice would be any of those two. This delimitation reduces the action space and the required training (i.e., computational effort) to synthesize well-performing controllers. We leave for future work the study of the impact of this delimitation and to validate our assumption. As mentioned, the final yellow location is a waiting location, where a clock must reach the given frequency before the automaton returns to its deciding location, which in this case is the red location. The dotted line represents actions the synthesized controller does not control, so the synthesized controller does not include the transition from waiting to deciding. Note that time can only pass in the yellow location as the red location is called a "committed" location, implying that time cannot pass when the automaton is there. This emulates a "zero-time" action, which models a discrete change in the mode of operation.

To synthesize a controller, we need to not only define the possible actions but also what information to base the decision on. The controller observes two parameters for each battery: its *current temperature* and its SoC *deviation from the mean* SoC. The use of the deviation from the mean is called symmetry reduction, and we used this since the controller does not need to know the exact SoC of each battery since the effect of a bypass is the same at 80% and 20% SoC according to our model. However, the same is not valid for the temperature; thus, decisions are based on the absolute temperature value.

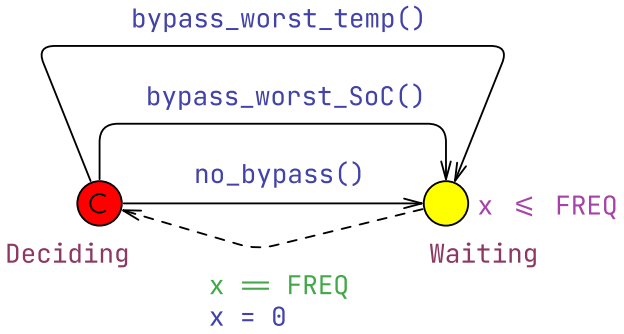

Fig. 3. An automata model of the template used to synthesize a controller that either makes no bypass, bypasses the cell with the worst SoC, or bypasses the cell with the worst temperature. The controller has a predefined frequency of when to decide whether or not to make a bypass.

TABLE I
Battery configurations of the 5-cell battery pack used in learning and simulations

	Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
Capacity in Ah	50.32	49.75	49.48	49.7	50.6
R_0 in $m\Omega$	1.0	1.0	1.0	1.0	1.0
R_1 in $m\Omega$	1.01	1.0	1.03	1.08	1.1
C_1 in kF	29.7	30	29.1	27.8	27.3

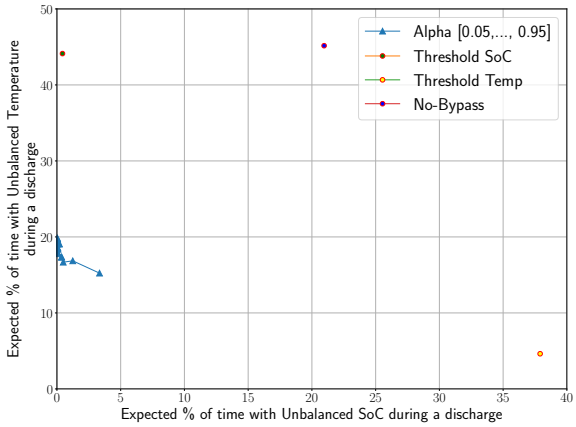

Fig. 4. Expected percentage of time in an unbalanced state given the use of one of the synthesized controllers, one of the threshold-based controllers, or not using any bypasses.

V. Experimental Setup

We have run simulations of a battery pack consisting of 5 different cells, where all differ slightly in capacities and the 1RC values R_1 and C_1. Even though a vehicle consists of battery packs of more than 5 cells, one could imagine a controller for every pack of 5 cells, such that the balancing among those cells is optimized. The small but noticeable variations in the batteries' configurations result from having aged a little. The batteries we base the simulations on are inspired by 50 Ah NMC Lithium-Ion prismatic cells with R_0 of 1.0 $m\Omega$, R_1 of 1.0 $m\Omega$, and C_1 of 30 kF. In table I,

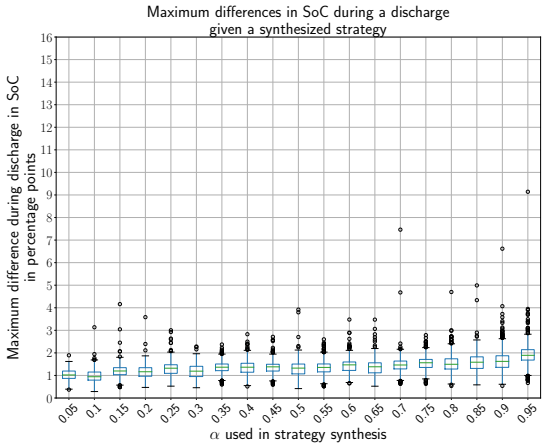

Fig. 5. Max different in SoC during discharge under the different synthesized strategies.

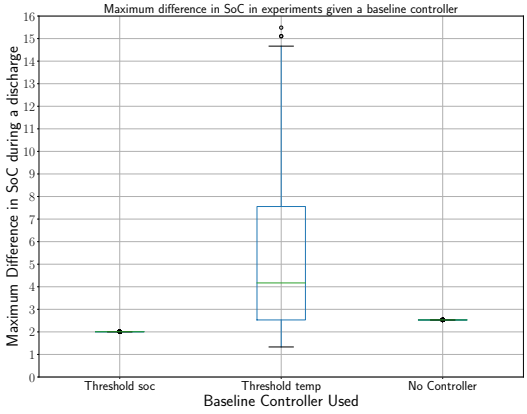

Fig. 6. Max different in SoC during discharge under the threshold controllers and no bypasses.

we have a battery pack based on those specifications. R_0 is the same for all cells as this parameter changes slowly over time, while R_1 has random variation within a 10% range. C_1 and R_1 have a relationship of $C_1R_1 = 30$, which is still there even if the batteries have aged slightly. As mentioned in the introduction, we assume that the parameters of each cell can be estimated and, therefore, be used to configure the model used in training when synthesizing a controller. The one parameter we cannot measure is the ambient temperature of the individual cells of the battery pack. However, we know there can be a slight variance between the batteries' ambient temperatures as the air heats up when flowing past the cells. In every simulation, each cell is randomly assigned an ambient temperature from a uniform distribution in the range $[24.25°C; 25.75°C]$, so there is a maximum difference of $1.5°C$. All cells start with 90% as the initial SoC and $25°C$ as the initial temperature. We simulated with 1C discharge for a duration of 50 minutes. This avoids the SoC ranges from 0-10% and 90-100% as aging is accelerated within these ranges.

To synthesize control strategies, we use a training budget of 30k simulations. The learning takes less than 6 hours to complete on an AMD EPYC 7551 with 32 cores, where each simulation is restricted to a single core. The timeframe is applicable if we imagine an electric vehicle's smart battery pack synthesizing a new controller during night-time. We synthesize controllers with several α from the series $0.05, 0.1, \ldots, 0.95$ to see how the balancing is affected by different emphasis on SoC or SoT. As references, we run simulations with two different baseline controllers based upon the threshold value θ_{SoC} and θ_{SoT}. One only balances SoC while the other balance SoT. They check the span of either SoC or SoT with a given frequency. If the span exceeds the given threshold, the SoC controller bypasses the cell with the lowest SoC while the temperature controller bypasses the cell with the highest temperature. These approaches follow the naïve framework that action should only be taken if there is a problem and not do anything preemptively.

VI. Experimental Results

We collect two metrics for the two threshold controllers and our synthesized controllers: the time spent in balanced states and the maximum difference between SoC and SoT during discharge. We conduct 1000 simulations with each synthesized controller, where each cell in each simulation is assigned a random ambient temperature from the range $[24.25°C; 25.75°C]$. In fig. 4, we see the expected time that the synthesized controllers and the two threshold controllers are unbalanced with SoC and SoT, and we also see how the system is expected to behave if no bypasses are used. As a reference, if no bypasses are performed, the system is out of balance with SoC 21% of the time while being out of balance with temperature 45% of the time. The threshold controller for SoC is only out of balance for SoC 1% of the time while having almost the same expected time of unbalanced temperature as if not using bypasses. The temperature threshold controller is only in an unbalanced state for temperature 5% of the time but is in an unbalanced state for SoC almost 38% of the time[1].

Looking at the synthesized controllers plotted in fig. 4, none of them are expected to have an unbalanced temperature of more than 20% of the time, while the worst is expected to be unbalanced 4% of the time with SoC. In terms of balancing both SoC and SoT, this is a significant improvement, but it does not tell us anything about how much the SoCs or temperatures can differ from each other when the system is in unbalance. Figures 5 and 7 show boxplots of the maximum differences between SoCs and temperatures, respectively.

The difference in temperature in fig. 7 is not highly affected by α. For many strategies, the median value is around $1°C$ while the maximum difference is $1.7°C$. Even though the ambient temperature spans $1.5°C$, all strategies can experience a higher difference than that due to the generated heat in cells with a high ambient temperature, different capacitance

[1]A reproducibility package for the findings can be found here: https://zenodo.org/record/8272060

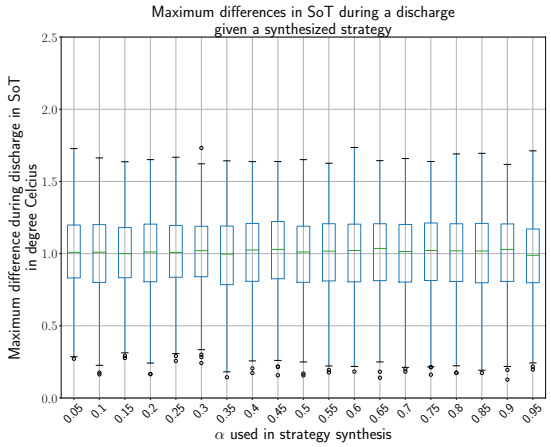

Fig. 7. Max difference in SoT during discharge under the different synthesized strategies.

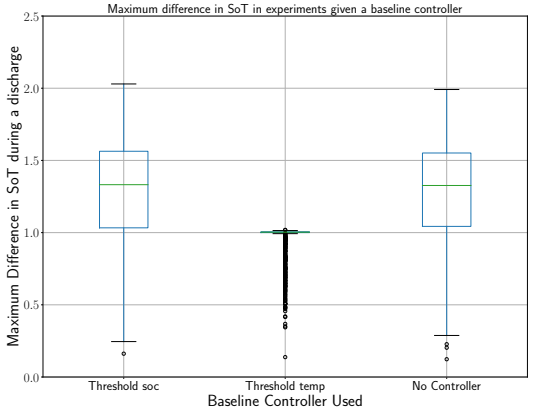

Fig. 8. Max difference in SoT during discharge under the threshold controllers and no bypasses.

among the cells affecting the voltage, and that cells with a low ambient temperature have a slower increase in temperature.

Figures 6 and 8 show the maximum spans under our threshold controllers and using no bypasses, where we see that the maximum difference in SoC and SoT can be higher than those occurring under any synthesized controller. When balancing only SoC or not using bypasses, the maximum span of SoC is consistently at 2% and 2.5%, respectively. Additionally, the temperature can span as much as $2°C$, which is $0.2°C$ higher than any strategy. Moreover, when balancing only for temperature, the difference in SoC can be as much as 15 percentage points, which is 6 percentage points higher than the worst outlier under a synthesized strategy shown in fig. 5. Only 3 simulations of the synthesized controllers had a maximum difference in SoC of more than 5 percentage points.

Our synthesized controllers do not spend more time in a thermal balance than the temperature threshold controller. However, some of our synthesized controllers balance SoC just as well as the SoC-based threshold controller and still make improvements for SoT, even when that controller is in SoC balance more than 99% of the time. This can be because the synthesized controller can take preemptive actions to avoid imbalance. In the end, all synthesized controllers are better than the baseline controllers in having a better proportion between the imbalance of SoC and SoT, and, in addition, have lesser spans in SoC and SoT.

VII. Conclusion & Future Work

In this paper, we show that we can utilize reinforcement learning with Uppaal Stratego and Q-learning to synthesize controllers for a smart battery pack consisting of 5 heterogeneous cells. We show through simulations that we make significant improvements compared to two threshold-based controllers in terms of being balanced and reducing the maximum differences between SoC and SoT.

Part of the future work is to increase the battery pack's size and verify the results on a physical system.

References

[1] R. Teodorescu, X. Sui, S. B. Vilsen, P. Bharadwaj, A. Kulkarni, and D.-I. Stroe, "Smart battery technology for lifetime improvement," *Batteries*, vol. 8, no. 10, p. 169, 2022.

[2] A. Kulkarni, H. Sorouri, Y. Zheng, X. Sui, A. Oshnoei, and R. Teodorescu, "Li-ion battery digital twin based on online impedance estimation," *CPE-POWERENG*, 2023, Accepted for publication.

[3] R. D. Fonso, X. Sui, A. B. Acharya, R. Teodorescu, and C. Cecati, "Multidimensional machine learning balancing in smart battery packs," in *IECON*. IEEE, 2021, pp. 1–6.

[4] R. Di Fonso, P. Bharadwaj, R. Teodorescu, and C. Cecati, "A battery digital twin based on neural network for testing soc/soh algorithms," in *PEMC*. IEEE, 2022, pp. 655–660.

[5] F. Altaf, *On modeling and optimal control of modular batteries: Thermal and state-of-charge balancing*. Chalmers Tekniska Hogskola (Sweden), 2016.

[6] E. R. Wognsen, B. R. Haverkort, M. Jongerden, R. R. Hansen, and K. G. Larsen, "A score function for optimizing the cycle-life of battery-powered embedded systems," in *FORMATS*. Springer, 2015, pp. 305–320.

[7] K. G. Larsen, M. Mikučionis, M. Muniz, J. Srba, and J. H. Taankvist, "Online and compositional learning of controllers with application to floor heating," in *TACAS*. Springer, 2016, pp. 244–259.

[8] M. A. Goorden, K. G. Larsen, J. E. Nielsen, T. D. Nielsen, M. R. Rasmussen, and J. Srba, "Learning safe and optimal control strategies for storm water detention ponds," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 13–18, 2021.

[9] D. J. Docimo and H. K. Fathy, "Analysis and control of charge and temperature imbalance within a lithium-ion battery pack," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1622–1635, 2018.

[10] X. Huang, J. Meng, W. Liu, F. Ru, C. Duan, X. Xu, D.-I. Stroe, and R. Teodorescu, "Lithium-ion battery lifetime extension with positive pulsed current charging," *IEEE Transactions on Industrial Electronics*, 2023.

[11] A. Farmann and D. U. Sauer, "A study on the dependency of the open-circuit voltage on temperature and actual aging state of lithium-ion batteries," *Journal of Power Sources*, vol. 347, pp. 1–13, 2017.

[12] H. Ren, Y. Zhao, S. Chen, and T. Wang, "Design and implementation of a battery management system with active charge balance based on the soc and soh online estimation," *Energy*, vol. 166, pp. 908–917, 2019.

[13] A. David, P. G. Jensen, K. G. Larsen, M. Mikucionis, and J. H. Taankvist, "Uppaal stratego," in *TACAS*. Springer, 2015, pp. 206–211.

[14] M. Jaeger, P. G. Jensen, K. G. Larsen, A. Legay, S. Sedwards, and J. H. Taankvist, "Teaching stratego to play ball: Optimal synthesis for continuous space mdps," in *ATVA*. Springer, 2019, pp. 81–97.