Journal of Energy Storage 84 (2024) 110853

Contents lists available at ScienceDirect

Journal of Energy Storage

journal homepage: www.elsevier.com/locate/est

Research papers

AI-assisted reconfiguration of battery packs for cell balancing to extend driving runtime

Yuqin Weng *, Cristinel Ababei

Electrical and Computer Engineering Department, Marquette University, Milwaukee, WI 53233, USA

ARTICLE INFO

Keywords:
State of charge
Cell balancing
Reconfigurable battery pack
Machine learning

ABSTRACT

State of charge (SoC) cell balancing is one of the most important roles of battery management systems (BMS). The performance and lifespan of a battery pack can be significantly degraded and reduced by the presence of imbalance in cells SoC. Recently, we have shown that using a machine learning driven battery pack reconfiguration technique based on a network of controllable switches, one can periodically change the battery pack topology to effectively achieve better cell SoC equalization. As a result, the driving runtime achieved with a better balanced battery pack is increased. In this paper, we build on these promising results and investigate novel machine learning models used for prediction of the best battery pack topologies used during reconfiguration. In addition, to study the scalability of the proposed battery reconfiguration technique, we conduct our study on a battery pack with double the number of cells. For validation, we developed an in-house custom battery pack simulation tool that integrates state-of-the-art battery cell models and extended Kalman filtering (EKF) algorithms for SoC state estimation. Simulation results using several battery discharging workloads demonstrate that the machine learning algorithms can achieve better prediction accuracy compared to previous work, thereby resulting in better cell balancing, which in turn translates into up to 22.4% longer battery runtime.

1. Introduction

Adoption of electric vehicles (EVs) has increased significantly in recent years. One of the most prominent components in EVs is the battery pack. This is a crucial component because the amount of energy that it can store and its performance are directly related to the performance of the vehicle. Therefore, it is important to pursue enhancements in battery technologies and battery pack operation and optimization in order to improve EV performance, which will further speed up EV adoption. This in turn will help reduce the overall unwanted greenhouse gas (GHG) emissions and global energy consumption [1]. Environment-related departments are promoting appropriate modifications of traditional automobiles for a sustainable and renewable ecosystem [2]. In this context, different countries proposed different plans to embrace the adoption of EVs. For example, United States aims at reaching zeros emissions for 50% of new vehicles [3–5]. Likewise, China plans to continuously reduce GHG emissions after 2030. The highest ownership of EVs per capita in the world is Norway, and this is because generous incentives are given to EV owners, including exemption from taxes and tolls, and free charging at public charging stations [6].

Li-ion batteries are the most popular types of batteries in EVs. One of the reasons for that is that Li-ion technology offers high power densities, which helps storing larger amounts of energy in a relatively small and lightweight battery pack. This is very important because weight and space are top priorities when choosing a battery type in EVs [7]. However, one of the challenges of using Li-ion batteries is the battery cells imbalance with regard to the state of charge (SoC) values of individual cells. SoC imbalance occurs when battery cells have different values of cell voltage, internal resistance, and cell storage capacity during charging or discharging processes [8]. Such differences can be caused by the environment, inconsistent manufacturing, and cycle life (aging or degradation) [9] and can have adverse effects such as chemical explosion and permanent loss of capacity [10].

One of the best ways to improve battery pack runtime is through battery cell equalization or balancing techniques. The process of cell balancing usually involves a mechanism to equalize the charge or discharge levels of all cells in the battery pack, i.e., all cells charge or discharge uniformly with all cells having at all times the same SoC values. Cell equalization algorithms can maximize the battery pack performance, the longevity or lifetime of the battery pack, and the runtime of the vehicle per charge. In general, most of the methods for battery cell equalization can be categorized into two classes, passive balancing and activate balancing. The difference between these two

* Corresponding author.
E-mail addresses: yuqin.weng@marquette.edu (Y. Weng), cristinel.ababei@marquette.edu (C. Ababei).

https://doi.org/10.1016/j.est.2024.110853
Received 10 October 2023; Received in revised form 7 January 2024; Accepted 5 February 2024

classes is that active balancing allows energy transported from cells that have higher SoC values to cells that have lower SoC values, while passive balancing enables the battery cell which has more energy to dissipate energy as heat [11]. Pros and cons of these two categories of methods are discussed in [11] as well as in recent review studies. For example, the study in [12] shows that active cell balancing is usually classified into three categories, depending on the type of hardware components used: capacitors, converters, or inductors and transformers. In the category of cell balancing by using capacitors, [13,14] propose that by shifting energy between nearby cells, cell equalization can be realized with the help of capacitors. In the category of cell balancing by using converters, [15,16] show that a standard/modified DC–DC converter or a PWM converter can be used for achieving cell equalization. Lastly, [17,18] show that cell balancing can be achieved very quickly by using inductors and transformers, but the disadvantage is the high cost and high frequency of the transformer. Passive balancing methods have also been proposed and studied in past literature. The study in [19] provides a detailed survey of existing passive balancing methods in various areas and compares them with each other. [20] shows that passive balancing techniques are commonly implemented in BMSs that support grid storage applications, and network stability is the goal for those applications. Moreover, there are previous studies that use deterministic control strategies. For example, the work in [21] formulated cell balancing as an optimal control problem and it proposed to solve it with a network modeling and dynamic programming approach — to arrive at global optimal energy equalization. [22] presents a general nonlinear model predictive control (NMPC) strategy on series-connected cells for obtaining a balancing-aware optimal charging. In [23], a real-time model-predictive-controller was proposed to balance battery SoC, which in turn extended the driving range, Moreover, there have been previous studies that use machine learning (ML) and artificial intelligence (AI) algorithms as control strategies for cell balancing. For example, the study in [24] proposed an ML based control algorithm to achieve balancing of both SoC and temperature. The study in [25] also demonstrated that smarter controls and longer lifetimes can be achieved with emerging ML techniques. Nevertheless, the use of AI-based cell balancing techniques is in its early beginning. It is not clear yet how far such techniques can push cell balancing in particular and battery pack optimization in general.

Therefore, in this paper, we propose and study a novel ML-based cell balancing technique for reconfigurable battery pack systems. The proposed battery pack system is a smart system in line with recent developments in reconfigurable battery packs as a special form of future smart batteries [26]. The proposed reconfigurable battery pack system and AI-based reconfiguration technique are verified in simulations conducted with a simulation tool that we developed and reported in our recent conference paper [27]. In this paper, we extend upon this preliminary work by investigating new machine learning models for prediction of best pack topologies and of larger battery packs, and for additional discharging workloads. One of the main advantages of using an ML/AI approach to determine the reconfiguration over deterministic control strategies is that training can regularly be repeated based on data gathered from the battery pack without the need to update other models, such as equivalent circuit models, in order to improve balancing. This is also a reason for which an ML/AI based approach is proposed in work over other more conventional approaches.

2. Batteries can become smart through reconfiguration

Smart batteries usually rely on some form of reconfiguration of the cells arrangement. Reconfiguration of the battery pack provides a mechanism to achieve cell SoC equalization, which becomes very challenging in the case of static battery packs. Reconfiguration means that different combinations of battery cells connections (in series, or in parallel, or combinations of series and parallel) can be setup by means of a network of reconfigurable switches that connects the cells.

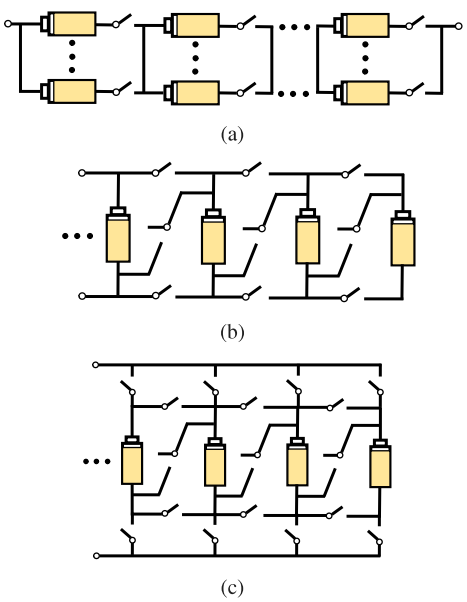


Fig. 1. Reconfigurable battery pack designs with 1, 3, and 5 switches per cell.

In this paper, a given combination of battery cells connections is called a *pack topology*. The main idea of the balancing (i.e., equalization of cells SoC) technique presented in this paper is to periodically switch between different pack topologies in a way that achieves better balance compared to the case when such switching is not done. The challenging aspect of this switching is how to identify in real-time the *optimal* topology to switch to for the next control period. Another challenge is to decide what type of network of reconfigurable switches to use. In previous literature, there have been studied several different reconfigurable battery pack designs using from 1 up to 6 switches per battery cell, which may become costly when this number is large [28]. Examples of reconfigurable pack designs that use 1, 3, and 5 switches per cell are depicted in Fig. 1.

The design with only 1–2 switches per cell is quite limited in the ability to provide a variety of connection combinations; however they allow access, exit, and bypass of each cell [29]. The advantages include low cost and easy assembly. Designs with 3–4 switches per cell are much more flexible in their ability to facilitate a variety of connections. Combinations of series, parallel, and parallel-series topologies are possible with such pack designs. Furthermore, designs with 5 switches per cell can deal with any cell fault in any pack topology, while designs with 6 switches per cell can reliably supply loads with different voltage requirements [29]. While the flexibility of pack designs with 5–6 switches per cell is the highest, the cost of manufacturing and maintenance is also high, in addition to the more complex assembly processes. In summary, the battery pack complexity and flexibility heavily depend on the number of switches. In this work, we use 3 switches per cell — as a reasonable trade-off between pack cost and flexibility.

3. Simulation framework: Cell modeling, SoC estimation, and pack topology

In this section, we describe the custom simulation framework that we developed for validation of the proposed ideas. We focus on the most important elements of the simulation framework: 1) the equivalent circuit model for the battery cell and SoC estimation techniques; and 2) the battery pack topology with a network of reconfigurable switches connecting all sixteen cells studied in this paper.

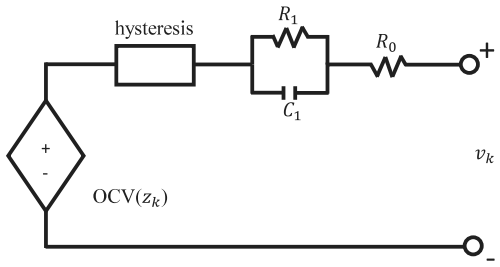


Fig. 2. Equivalent electric circuit for a battery cell used by the ESC model [31].

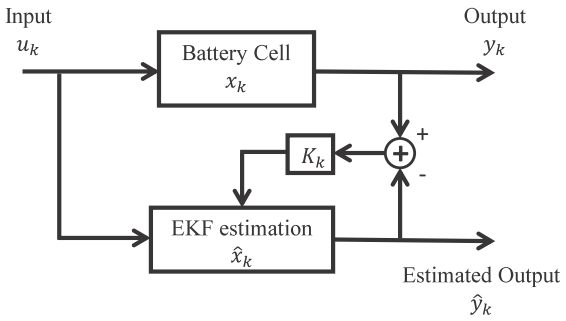


Fig. 3. Illustration of the ESC cell model in parallel with the true hardware cell. The EKF algorithm solves the model at each time step k by using the Kalman gain K_k during the state correction step or measurement update. The input u_k is the instantaneous current at each time k. The battery cell state x_k and output y_k are described by Eqs. (1) and (2). EKF estimated state and output are $\hat{x}_k$ and $\hat{y}_k$.

3.1. Cell modeling

There are several popular equivalent circuit models for modeling battery cells in the literature [30]. The most prominent one is the enhanced self-correcting (ESC) model. Besides capturing cell SoC, the ESC model also captures the hysteresis voltage and diffusion processes from the battery cell. The equivalent circuit model for the ESC model is illustrated in Fig. 2, and the state space representation is described by the following equations [31]:

$$\begin{bmatrix} z_{k+1} \\ i_{R,k+1} \\ h_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & A_{RC} & 0 \\ 0 & 0 & A_{H,k} \end{bmatrix} \begin{bmatrix} z_k \\ i_{R,k} \\ h_k \end{bmatrix} + \begin{bmatrix} (\frac{-\eta_k \Delta t}{Q}) & 0 \\ B_{RC} & 0 \\ 0 & (A_{H,k}-1) \end{bmatrix} \begin{bmatrix} i_k \\ sgn(i_k) \end{bmatrix} \tag{1}$$

$$y_k = OCV(z_k) + Mh_k - R_1 i_{R,k} - R_0 i_k \tag{2}$$

$$x_k = \begin{bmatrix} z_k \\ i_{R,k} \\ h_k \end{bmatrix} \tag{3}$$

Where Eq. (1) represents the state equation. The output equation is defined by Eq. (2). In these equations, the output of the system which is also the terminal voltage of the battery cell is denoted as y_k, and the input of the system is i_k, which is called the cell instantaneous current. The internal resistance of the battery cell is defined as R_0. There are three states captured by this model and they are SoC: z_k, diffusion process current: $i_{R,k}$, and hysteresis voltage: h_k. They are kept in a state vector in Eq. (3). Some entries in Eq. (1) are described by the following equations:

$$A_{RC} = exp(\frac{-\Delta t}{R_1 C_1}) \tag{4}$$

$$B_{RC} = 1 - exp(\frac{-\Delta t}{R_1 C_1}) \tag{5}$$

$$A_{H,k} = exp(-|\frac{\eta_k i_k \gamma \Delta t}{Q}|) \tag{6}$$

When the cell is fully charged and fully discharged, the SoC z_k is defined as 100% and 0%. While in theory, SoC could not be larger than 100% or smaller than 0%, during the process of charging or discharging in practice these limits are tighter; for example, it is typical for the lowest SoC limit to be 5%–30%, during discharging because that helps the battery reliability. The Coulombic efficiency, denoted as η_k, is usually equal to one when the cell is discharging and less than one when it is charging. To convert the continuous-time system to a discrete-time system, a small sampling interval Δt is used. The total capacity of the cell is denoted as Q, which is the total amount of charge when charging the cell from 0% to 100% SoC. The open circuit voltage of the cell, denoted as $OCV(z_k)$, is an ideal voltage source and is dependent on SoC under a certain temperature. A lookup table is used to record the cell testing data for $OCV(z_k)$ in practical applications. The diffusion voltage is represented by $R_1 i_{R,k}$ [32]. Similar to previous studies [31], the ESC model uses one $R_1 C_1$ pair in this work. The hysteresis state which is denoted as h_k, is the third state in the state equation. It also has a connection with the input i_k in Eq. (1). Finally, a small constant γ is used to model the voltage decay.

3.2. SoC estimation

Because the SoC of a battery cell is not a quantity that can be directly measured, estimation methods are typically employed. One of the most common used methods to estimate SoC of a battery cell is Kalman filtering. Kalman filter theory is a classic technique developed in the 1960's [33]. Kalman filters have been used in many application areas including control systems, signal processing and image processing due to its accuracy and robustness [34]. As a variation of Kalman filter theory, the extended Kalman filtering (EKF) is used for the estimation of non-linear state-space models. The idea of EKF is that a linearization can be used at around the current estimate by using partial derivatives of the state and output equations to compute estimates for each time step in discrete time [30]. Thus, EKF is a perfect choice for estimating battery cell SoC because the battery cell state-space model is a non-linear representation at time k. Thus, by applying EKF to the battery cell equivalent circuit model (Eqs. (1) and (2)), the estimated state and output can be generated at each time k. An illustration of how EKF is used for SoC estimation is shown in Fig. 3. The simulation tool, which we developed to be able to conduct the simulation experiments from this paper, implements the EKF approach for estimating SoC for all cells in a battery pack, which are then used to estimate the SoC of the entire pack for the best possible accuracy of results.

3.3. Battery pack topologies

The cell balancing technique proposed in this paper relies on a re-configurable battery pack topology. The inspiration for using different pack topologies to equalize cell SoCs comes from [35]. In that study, a network of programmable switches is controlled to change the connections inside a photovoltaic (PV) array. We adopt the same network of reconfigurable switches, which uses 3 switches per battery cell as shown in Fig. 4. In this structure, a set of three switches for every cell includes switches denoted as $S_{PT,i}$, $S_{PB,i}$, and $S_{S,i}$ (shown on Fig. 4(a)). These switches must be operated according to a simple rule: switches on the top and bottom levels should remain in a consistent state, meaning they should both be on or both be off; while the switch located on the middle level must be in a state that is opposite to the other two switches. Fig. 4(b) illustrates a battery pack topology changing from [2,2,2,2,2,2,2,2] to [2,4,2,2,2,2,2,2] and then to [2,2,2,4,2,2,2,2], which can

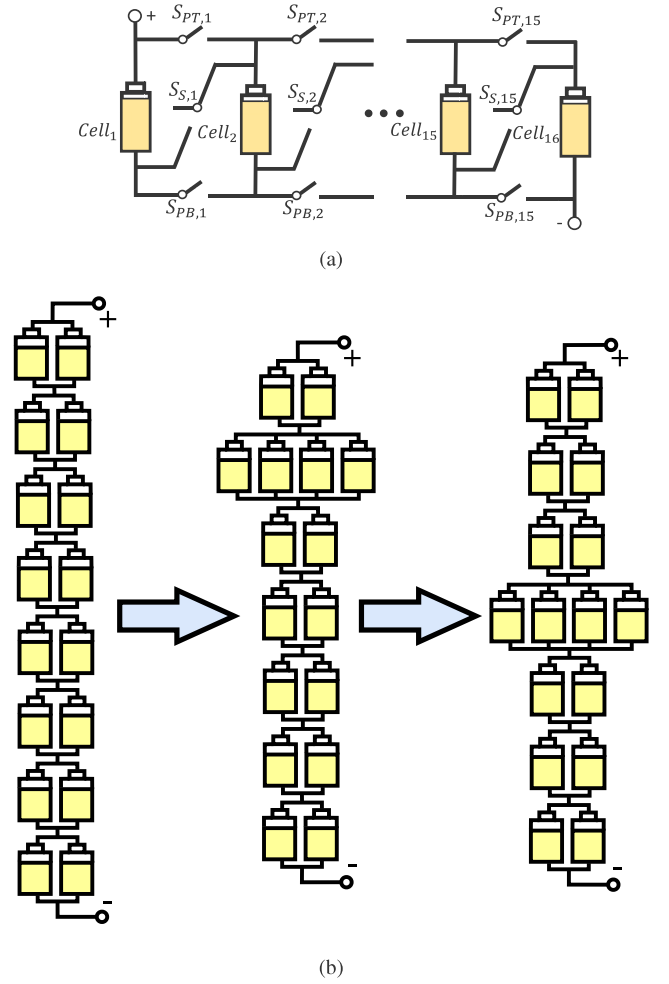


Fig. 4. (a) Reconfigurable battery pack with 16 cells. (b) Example of a pack topology changing from [2,2,2,2,2,2,2] to [2,4,2,2,2,2,2] and then to [2,2,2,4,2,2,2].

easily be achieved by appropriately closing or opening switches shown in Fig. 4(a).

Please note that our work assumes that the battery pack is connected to the load through a power converter, which will make sure that the voltage level delivered to the load is kept constant — the role of the converter is to regulate the output voltage to the necessary level. This will ensure the stability of the voltage level supplied to the load and will address the issue of potential changes in the output voltage of the battery pack due to switching between different pack topologies during the reconfiguration process. Therefore, the amount of power provided to the load should remain the same. Nevertheless, it is the power converter's efficiency and performance that may however introduce small difference in the output power supplied by different pack configurations; but, in this work we consider that to be negligible.

4. Collection of datasets and input features of ML model

4.1. Collection of datasets

The SoC balancing algorithm proposed in this paper and discussed later relies on machine learning models, which require training. Extensive SoC simulations showed that different initial SoC conditions and different cell parameter values can lead to widely different SoC variation profiles when different battery pack topologies are used. In other words, different pack topologies lead to different imbalance levels starting from the same initial conditions and during the same

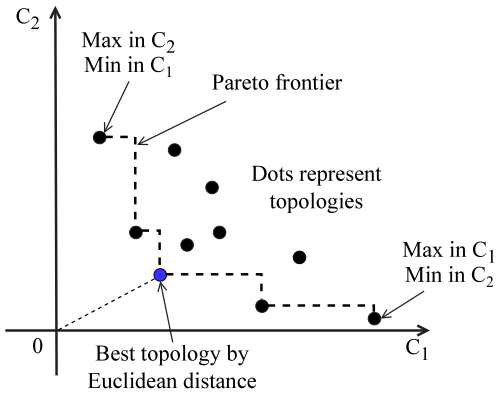


Fig. 5. Illustration of the Pareto optimality. C_1 and C_2 are calculated with Eqs. (7) and (8).

simulation time or use. In addition, it was also observed that the level of imbalance generally increases with time. This motivated us to develop a topology-switching technique that allows the battery pack to switch between different topologies during different control or operation periods. This technique helps reduce the imbalance in the next control period. For instance, during pack discharging, it is desired to maintain all SoC values high and as equal as possible among each other. In other words, we want to reduce the difference between different cells SoC as much as possible at any given time. Consequently, cells SoC in the battery pack will be kept as uniform as possible. Results of our preliminary work showed that the proposed topology switching algorithm can improve battery cell equalization even more, beyond what one can achieve by continuously using just one best topology. To identify which topology is the best for the next control period, several ML models will be investigated. To train these models, rich datasets with inputs and labels need to be created. In this section, we discuss how we generate such datasets using our simulation tool.

During dataset collection, we need to define what inputs will be used and what the output labels are. For our problem, the labels must represent identifiers of the topologies that are included in the set to work with for the purpose of switching among them. When a switch from a topology to another happens, we are interested in two objectives: 1) Maximize the summation of all SoC values of battery cells which is equivalent to minimizing the inverse of the summation of all SoC values; and 2) Minimize the range of SoC values, which is calculated as the difference between the maximum and minimum SoC values among all cells. This can be regarded as a multi-objective optimization problem. Consequently, two objective functions are defined to determine the next best topology to switch to. All possible solutions of this optimization problem represent a solution space bordered by a Pareto frontier [36]. Fig. 5 shows an example of such a solution space where each dot represents a possible next pack topology. The dashed line shows the Pareto frontier. Prior research [37] showed that the Pareto optimality of these solutions ensures that it is not possible to enhance one objective without compromising the other. The *cost* functions for two objectives C_1 and C_2 are defined as:

$$C_1 = SoC_{diff}^{max} = \max_{i\in[n]} SoC_i - \min_{i\in[n]} SoC_i \tag{7}$$

$$C_2 = 1/\sum_{i=1}^{n} SoC_i \tag{8}$$

Where we define $[n] = \{1, 2, \ldots, n\}$ with n representing the total number of cells in the battery pack.

Because we work with a total of N different topologies only, the optimization problem is essentially identifying the topology that should be used in the next control period. This identification is done by evaluating the following expression that combines the two individual

Algorithm: Dataset Generation based on Pareto Optimality

```
In: Reconfigurable pack architecture, workload
Out: Dataset for training and testing ML model
Assume discharging workload
for range_u :← [0.3 : 0.15 : 0.9] do
    for j ← 1 to N do // N: number of topologies
        for i ← 1 to M do // M: 400 simulations
            Configure pack to initial topology j
            Randomly select initial SoC from range_u
            Randomly initialize cell vars. by Gaussian distribution
            for k ← 1 to N do
                Configure pack to next topology k
                Run simulation for the next control period
                Record costs, eq. 7, eq. 8
                Find best next topology, eq. 9
                Record new datapoint: (Input features | Label)
                Input features: end SoC values, j, statistical feats.
                Label: next best topology index
            end for
        end for
    end for
end for
```

Fig. 6. Pseudocode of the simulation process used to generate datasets.

cost functions C_1 and C_2. The topology that minimizes this expression is the one identified as the best [38]:

$$d = \arg\min_k \sqrt{(C_1^k)^2 + (C_2^k)^2} \tag{9}$$

Where we define $[N] = \{1, 2, \ldots, N\}$ with N representing the total number of topologies that are in the solution space. The topologies which are acceptable in terms of output voltage levels can be considered in the solution space, more details about pruning existing topologies are described in section slowromancapvi@. In Eq. (9), the best topology which is recorded as topology k has the minimum Euclidean distance to the origin in Fig. 5. The minimum Euclidean distance is calculated with the help of Eq. (7), Eq. (8), and Eq. (9). In addition, C_1^k and C_2^k are the values of C_1 and C_2 for topology k in Eq. (9).

Our process of generating datasets is similar to what we used in our previous preliminary study and it relies essentially on the above equation. The difference here is that we are working with a battery pack with a double number of cells, which makes the problem more challenging due to the much larger number of battery pack topologies that one can potentially form by means of the reconfigurable switch network. The simulator is instrumented to conduct exhaustive simulations of the type *what-if* scenarios — during which dataset is generated for all possible next topologies. For each control period that is simulated and added to the dataset, we identify the Pareto optimal topology that is labeled as the next best topology. Looking again at Fig. 5, during the simulations for dataset generation, at the end of a simulated control period, there are N topologies evaluated via the cost functions C_1 and C_2; these topologies represent the dots in the figure. The best next topology will be identified as the one that results in the minimization of Eq. (9). That topology is recorded as the best one to switch the pack to and then continue the simulation for the next control period during the dataset generation process. Multiple simulations are run to collect data for what the best next topology is for different states of the battery cells. The process of conducting simulations for dataset generation is described with the pseudocode from Fig. 6.

4.2. Input features and output label for the ML model

The dataset collected as described in the previous section will be used for training of the machine learning models. Because we deal with supervised training, each entry into the recorded dataset will be a pair of the form Input features, Label. The input features represent

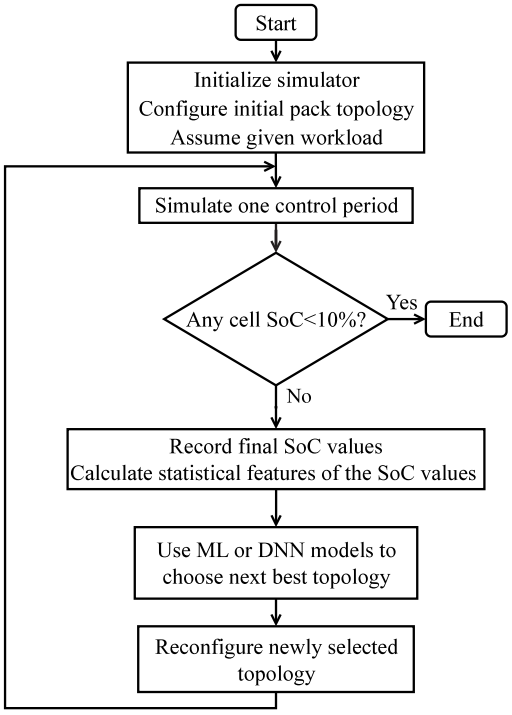


Fig. 7. Flowchart of the proposed balancing algorithm; topology switching via reconfiguration is done periodically, with a control period of 5 min.

information about the current state of the battery pack while the output label is the ID of the best topology to use next. In our implementation for the case of the battery pack with 16 cells, the final number of different topologies that we kept to work with is seven. Therefore, the output level is the ID of the next best topology as a number between 0 and 6. The input features include the following: the current state of charge (SoC) values of all 16 cells; their mean value, maximum value, minimum value, standard deviation, summation, and maximum absolute difference (i.e., distance between the largest and the smallest SoC values of all cells; in other words, this is the range or span or spread of all values of SoCs) among all 16 cells; and, the currently being used topology ID. Therefore, there is a total of 23 input values recorded as input features. This is also indicated in Fig. 8.

There are several reasons for which we chose these features. We are optimizing two objectives, C1 and C2, as described by Eqs. (7) and (8). The calculation of expressions of C1 and C2 requires most of the input features discussed above. In addition, the mean, standard deviation, and span are included as input features to capture statistical characteristics in a compressed format. In our extensive empirical experiments, we found that the accuracy of the ML models increased if we included the current SoC values of all 16 cells compared to the case when they were not included.

5. Proposed cell balancing algorithm

The idea of the proposed cell balancing algorithm is to use machine learning models to identify the next best battery pack topology to switch to for the next control period. The models are trained with the datasets generated as explained before. The simulation tool that we instrumented to generate the datasets is also used to implement and verify the proposed cell balancing algorithm. Fig. 7 shows the flowchart describing the main steps involved in the proposed algorithm. Identifying the next topology to switch to is done using a machine learning model for predictions. Once the ID of the next topology is identified, all configurable switches in the reconfigurable network

Table 1
Prediction accuracy of studied models.

Models	Testing accuracy	Training accuracy
DNN	72.0%	73.5%
SVM	66.8%	72.1%
AdaBoost	55.9%	60.7%
KNN	50.1%	53.8%
RF	40.9%	40.6%

are controlled to the correct on/off state to implement the desired topology. The information about the state of all switches for a given topology ID is easily stored and retrieved from a small loop-up table in our implementation. We investigated several models in order to find out which one would be best for our application: Adaptive Boost (AdaBoost) algorithm, support vector machine (SVM), and deep neural network (DNN).

In our preliminary work, we investigated random forest (RF) and k-nearest neighborhood (KNN) models. Here, we investigate AdaBoost as an ensemble learning technique, which is similar to RF. We selected AdaBoost because it has been shown that it is more accurate than RF for predictions generally [39]. The study in [40] showed that although RF has higher overall accuracy, AdaBoost has better class-specific accuracy. The second ML model we investigate is SVM, which is efficient and known providing good results in terms of maximizing margins between different classes. Therefore, SVM can handle high-dimensional data, and is easily applied to both binary and multi-class classification problems. The last model we investigate is DNN — because it has become very popular and widely used in many problem settings from engineering, data science, and biomedical fields. In addition, we found previous literature that used DNNs to estimate the state of health for battery cells in EVs [41,42]. Next, we describe the performance of these models.

In order to limit the amount of time put into model development and training, we restrict the analysis to only ten different topologies. During the extensive simulations used for dataset generation, it turned out that only seven were dominant and consistently used to switch between. As such, the number of classes or labels we work with is seven. All the investigated models are tested for those seven labels. The process of initially selecting a reduced number of topologies will be described in the next section; here, we focus on datasets generation and models performance.

The datasets were generated as described earlier (Fig. 6) — where each topology was simulated 400 times starting from different initial SoC values drawn randomly from ranges between 90% to 30% with a step of 15%. Hence, the total number of datapoints in datasets is $10 \times 400 \times 5 = 20000$. Datasets are split into train and test portions using a 60%/40% split. A grid search technique [43] is implemented for both AdaBoost and SVM algorithms to select suitable parameters. For AdaBoost, *n estimators* and *learning rate* parameters are set to 3000 and 1. For SVM, *C* and *gamma* parameters are set to 15000 and 0.015. The DNN model has the structure presented in Fig. 8, and it has two hidden layers — with 128 and 64 neurons, respectively. Other parameters used include: loss function as *categorical cross entropy*, optimizer as *Adam*, *learning rate* set to 0.00015 during training. The input to the DNN model from Fig. 8 is a combination of: current topology, 16 cell SoC values at the endpoint of previous control period, and several statistical features of the cells SoC. A 10% dropout regularization is applied on each hidden layer, meaning each node has a 10% chance of being dropped out at every epoch when training. To avoid overfitting, an early-stopping technique is also implemented with a patience value of 200. Both dropout regularization and early-stopping techniques are useful tools to avoid overfitting [44,45].

The training and testing accuracy for each model including the ones used in our preliminary work are reported in Table 1. In addition, for example, the loss and accuracy variations for the DNN model are

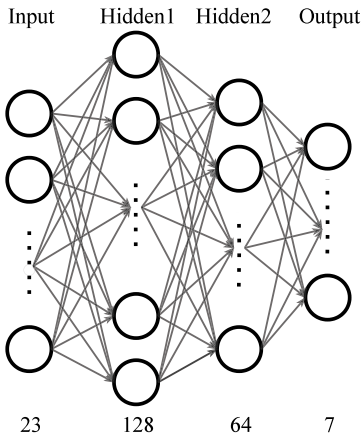
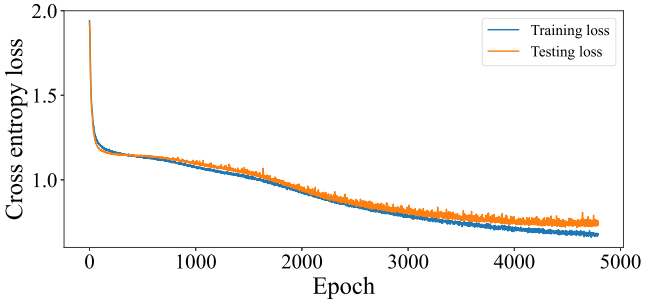
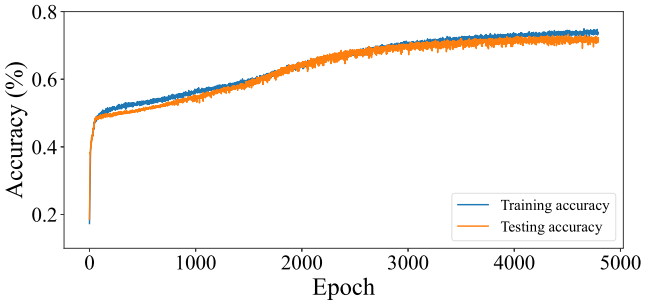


Fig. 8. Structure of the DNN model.



(a)



(b)

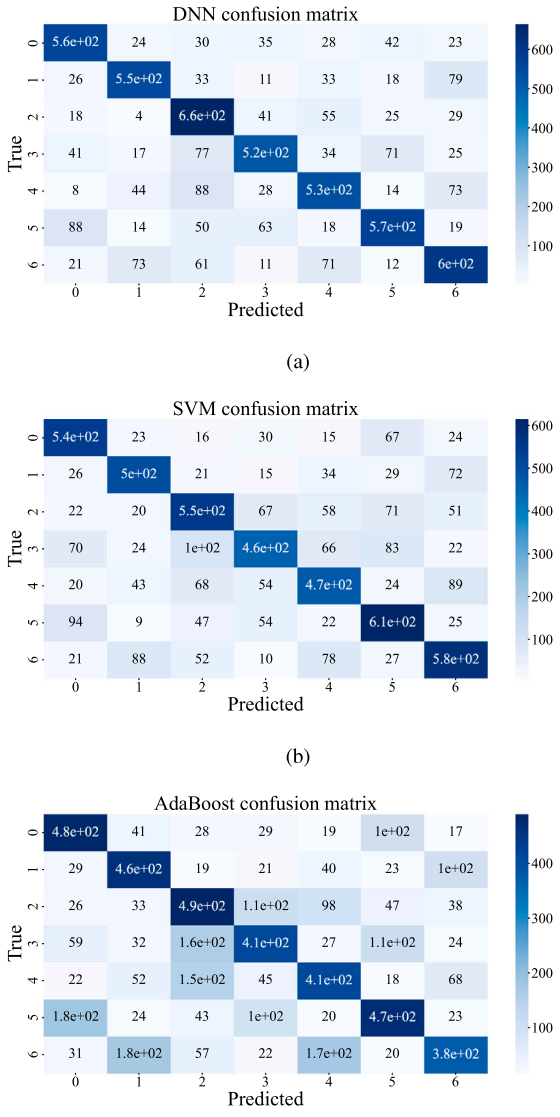
Fig. 9. Loss and accuracy plots for the DNN model.

presented in Fig. 9 — where the x-axis represents the epochs number. We find that the DNN model has the highest testing accuracy among all studied models, with the SVM model coming in close as the second. AdaBoost, KNN, and RF models are the poorest performing models. The confusion matrices for the testing portion of the DNN, SVM, and AdaBoost models are shown in Fig. 10, which again shows that the DNN model has the best performance (i.e., largest numbers of correct predictions are shown on the main diagonal).

6. Simulation results

6.1. Description of experiments

Simulations are conducted for a battery pack with 16 cells. All simulations were run on a computer with an Intel(R) Core(TM) i5-10400 CPU running at 2.90 GHz using 16MB of RAM, running on Windows 11. The GPU used for training DNN model is the NVIDIA GeForce GTX 1660 with 6 GB memory. The total simulation time for



(a)

(b)

(c)

Fig. 10. Confusion matrices for DNN, SVM, and AdaBoost models — obtained on the test portion of the datasets.

generating all datasets was approximately 30 h. The training time for DNN model was about 60 min. Numerical values of various cell parameters including z, Q, R_1, C_1, and R_0 are adopted from previous literature [31,32]. To mimic realistic variations of these parameters, their values are drawn from Gaussian probability distributions, characterized by standard deviations of 5% and mean values also reported in [31,32]. Also, the initial SoC values for all cells in the pack and generated from Gaussian distributions characterized by a mean of 0.9 (representing 90%) and 5% standard deviation — and used as starting SoC values for both the reference and proposed algorithm cases. There are thousands of topologies that can be formed for a battery pack with 16 cells; however, most of them are not realistic for practical use. For example, when all cells are connected in series or all cell are connected in parallel can have limitations in terms of required levels of current and voltage levels. Therefore, such topologies are excluded from being considered useful topologies that the proposed balancing technique can use during reconfiguration of the battery pack.

To restrict the number of different topologies that we work with – and thus to make the proposed technique realistic and efficient – we restrict our attention to topologies with 6, 7, and 8 rows. For instance, a 7-row topology could be $[2, 4, 2, 2, 2, 2, 2]$, where each number inside the list represents the number of cells is connected in parallel on that row; and, all rows are connected in series. Even when the number of rows is restricted like that, there are still more than 100 possible topologies that one can create. To further prune the topologies, a maximum cell difference between each row should be set. The maximum cell difference in the 7-row topology listed is 2. Through simulations, we found out that any topology that has a maximum cell difference larger than 3 tends to be more unbalanced than those topologies that have a maximum cell difference smaller than 3. Thus, pruning those unbalanced topologies beforehand is necessary.

However, even after such pruning of topologies, simulations for datasets generation is still computationally expensive and inefficient. The datasets generation is the most time-consuming process in this work, yet it is needed for training the machine learning models. Therefore, we keep only the top 10 topologies from this subset efficiency reasons. During the process of generating the datasets, three least-favorite topologies were chosen less than ten times in the whole datasets generating process. Thus, to further eliminate this redundancy, we have selected as final topologies only seven out of the ten topologies to continue with model training.

Simulations are carried out using three different workloads: constant workload, step workload, and a combination of multiple urban dynamo-meter drive schedule (UDDS) drive cycles (UDDS workload). For each workload, we conduct a comparison of two scenarios: discharging with one fixed topology (base case) and discharging with switching between different topologies as dictated by the proposed balancing algorithm. Only the DNN model, which turned out to be the best, is implemented in these simulations. The base case topology is selected to be $[2, 2, 2, 2, 2, 2, 2]$ because it was found through empirical simulations to be the most balanced topology when no topology switching is used. The discharging rate for UDDS workload was set to $0.1C$. The discharging rate for step workload (where the battery pack discharges for 3–5 control periods and rests for one control period) is 1.5 times as much as the UDDS workload and discharging rate for the constant workload is 2 times as much as the UDDS workload. These rates are similar to those used by previous work, in the range of 0.02-1C [22,23].

Finally, using the final selected seven topologies, the average execution time for the proposed algorithm is about 1.6 s during a full simulation (SoC from 90% down to 10%). We believed that such a small computational runtime would be similar when the proposed algorithm is deployed on real battery management systems (BMS), which have powerful enough processors. Also, we would like to note that we assumed ideal switches in our models in all simulations as a simplification. In reality, switches themselves have a non-zero resistance, which will affect the output of the battery pack. However, high-performance power MOSFET transistors (that it is envisioned to be used as switches in a future prototype, e.g., IRFP4668PBF N-channel power MOSFETs) have on resistance as small as 9.7 mΩ [46]. Hence, our simplification is acceptable.

6.2. Results and discussion

The results of the simulations are shown in Fig. 11, 12, and 13. In each of these figures, the x-axis represents the time measured in seconds. This time is the runtime of the battery pack before its overall SoC drops below a preset threshold (10%). Of course, it is desired for this time to be the longest possible because that translates into longer driving distances. The limits of 10%–90% are based on what past literature recommends typically [47]. Again, as mentioned earlier, the starting SoC values for each of the two compared simulations are the same, and generated from Gaussian distributions characterized by a mean of 0.9 (representing 90%) and 5% standard deviation. Each control period is 300 s (5 min) and is shown as the distance between two consecutive vertical lines in these figures. From these plots, we see that the proposed balancing algorithm significantly improves the

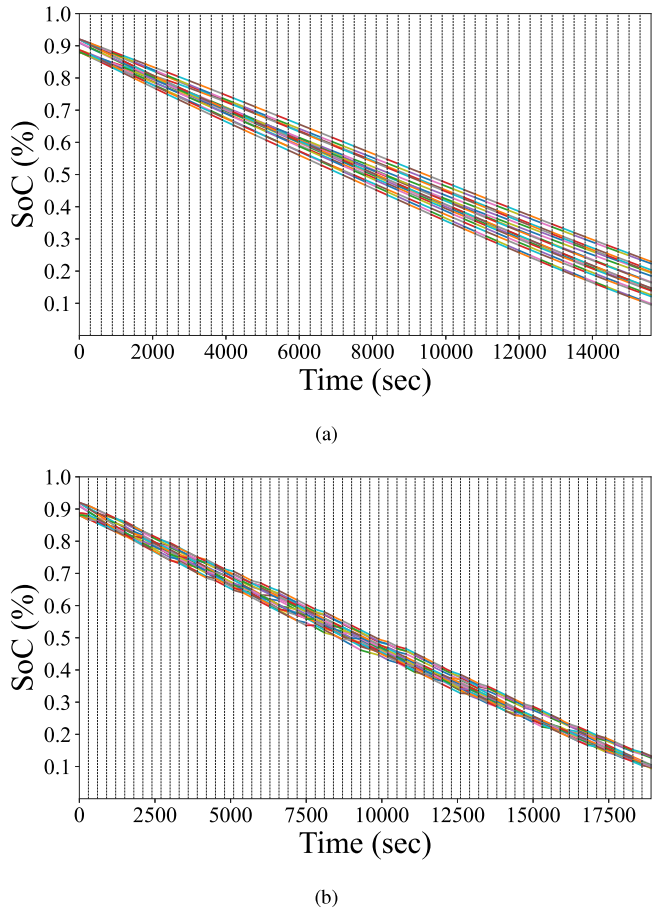


Fig. 11. (a) Variation of cells SoC for base case topology [2, 2, 2, 2, 2, 2, 2, 2], constant workload. (b) Variation of cells SoC for a pack with topology reconfigured using the proposed balancing algorithm, constant workload.

runtime of the battery pack on all workloads at different discharging rates. For constant workload, the runtime of the battery pack is improved from 15,600 to 18,900 s, i.e., 21.5%. Furthermore, compared to our previous work – in the case of constant workload – results are improved with 10.9%.

Another figure of merit that we use to assess the differences between the base case and the proposed algorithm, is the span of the range of all SoC values at the end of the simulation. We observe that also in terms of this metric, the proposed balancing algorithm improves results from a span of SoC values of 0.13 to 0.04 in the case of the constant workload. In the case of step workload, results show that the improvement in pack runtime is from 25,500 to 31,200 s, i.e., 22.4%. In terms of SoC values span, the improvement is from 0.11 to 0.04 for the step workload. Finally, in the case of UDDS workload, the pack runtime is improved from 30,300 to 36,300 s, i.e., 19.8% — which is also with 5.2% better than our previous work. The cells SoC values span is decreased from 0.07 to 0.03 for the UDDS workload. Although SoC is still not perfectly balanced (i.e., zero span of SoC values), the span of the final SoC values has significantly been reduced by the proposed technique, and as a result the runtime of the battery pack increased. A summary of all the improvements in terms of runtime length and SoC values span are listed in Table 2.

Finally, we would like to mention that while the proposed balancing algorithm provides significantly longer battery runtime, it is still not the best one could possibly achieve. The reason for that is because the DNN model's prediction accuracy is only around 73%, which is far from perfect. This results in turn to a certain fraction of reconfigurations to be done to pack topologies that are not the "perfect" or ideal ones at those times. These non-perfect topologies will make for the overall SoC values to follow a path that is possibly less optimal than the absolute best. Currently, we do not have an efficient way to estimate how far the solution achieved by the proposed algorithm is from the solution one would get with a perfect predictor. The best way to address this difference is to improve the ML model accuracy by refining further the model or by employing other, more sophisticated models — which we are currently exploring and be reporting in our future work.

Table 2
Summary of improvements in terms of runtime length and SoC values span.

Discharging Workload	Improvement in Battery Runtime Increase (%)	Improvement in SoC Values Span Reduction (×)
Constant	21.5%	3.25×
Step	22.4%	2.75×
UDDS	19.8%	2.33×

7. Conclusion and future work

In this paper, we presented a novel and enhanced cell balancing technique for reconfigurable battery packs that are integrated with networks of reconfigurable switches, which can be controlled to create different series, parallel or combinations of such connections. The objective of the balancing technique is to keep all battery cells at SoC values that are close as possible to each other during the discharging process. This is achieved by periodically switching between different battery pack topologies, which turn out to help improve cell equalization. Decisions as of what topology to switch to during each reconfiguration is done with the help of a machine learning based prediction. We studied several machine learning models among which deep neural networks seemed to offer the best prediction accuracy. Extensive simulation experiments on a 16 cell battery pack with a custom built simulation tool showed that battery runtime can be increased by to up to 22.4%.

While the results in this paper are very encouraging and demonstrate that AI based balancing approaches have great potential, there is need for more research in this direction. That is why, in our future work, we plan to first investigate scalability of the proposed technique by studying additional battery pack topologies formed with larger number of cells. More specifically, we will look into divide-and-conquer approaches that partition large numbers of cells into partitions of fewer cells and apply the technique discussed in this paper to those partitions. This means an ML model is developed only once (for a partition) and then applied during the reconfiguration algorithm to all partitions. It would also be interesting to investigate other types of workloads. Finally, we plan to develop a hardware prototype for a battery pack with 8 cells and validate in practice the benefits of battery topology reconfiguration and switching versus actual costs.

CRediT authorship contribution statement

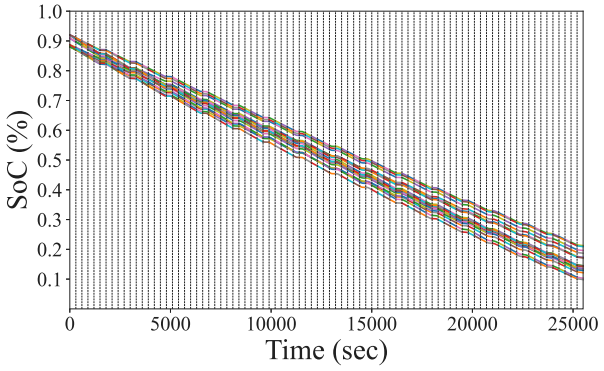
Yuqin Weng: Conceptualization, Investigation, Methodology, Software, Validation, Writing – original draft. **Cristinel Ababei:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

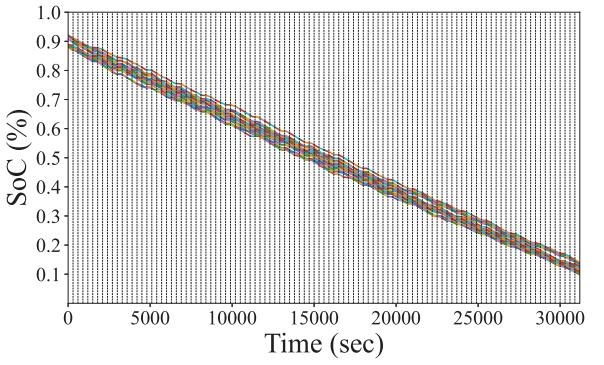
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

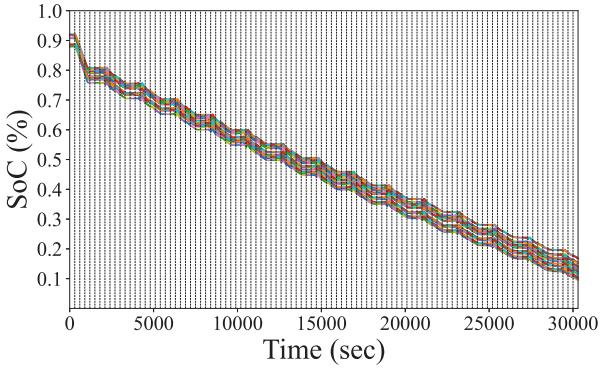


(a)

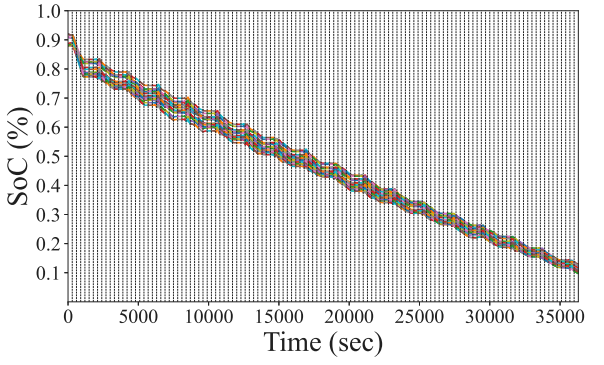


(b)

Fig. 12. (a) Variation of cells SoC for base case topology [2, 2, 2, 2, 2, 2, 2, 2], step workload. (b) Variation of cells SoC for a pack with topology reconfigured using the proposed balancing algorithm, step workload.



(a)



(b)

Fig. 13. (a) Variation of cells SoC for base case topology [2, 2, 2, 2, 2, 2, 2, 2], UDDS workload. (b) Variation of cells SoC for a pack with topology reconfigured using the proposed balancing algorithm, UDDS workload.

References

[1] M. Iqbal, A. Benmouna, M. Becherif, S. Mekhilef, Survey on battery technologies and modeling methods for electric vehicles, Batteries 9 (3) (2023) 185.

[2] M.-K. Tran, S. Panchal, T.D. Khang, K. Panchal, R. Fraser, M. Fowler, Concept review of a cloud-based smart battery management system for lithium-ion batteries: feasibility, logistics, and functionality, Batteries 8 (2) (2022) 19.

[3] K. Fang, C. Li, Y. Tang, J. He, J. Song, China's pathways to peak carbon emissions: new insights from various industrial sectors, Appl. Energy 306 (2022).

[4] N. Abhyankar, P. Mohanty, A. Phadke, Illustrative Strategies for the United States to Achieve 50% Emissions Reduction by 2030, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, 2021.

[5] P. Mock, S. Díaz, Pathways to decarbonization: the European passenger car market in the years 2021–2035, Communications 49 (2021).

[6] H. Scott, Understanding the impact of reoccurring and non-financial incentives on plug-in electric vehicle adoption – A review, Transp. Res. A 119 (2019) 1–14.

[7] S.S. Rangarajan, S.P. Sunddararaj, A. Sudhakar, C.K. Shiva, U. Subramaniam, E.R. Collins, T. Senjyu, Lithium-ion batteries—the crux of electric vehicles with opportunities and challenges, Clean Technol. (2022) 908–930.

[8] A. Samanta, S. Chowdhuri, Active cell balancing of lithium-ion battery pack using dual dc-dc converter and auxiliary lead-acid battery, J. Energy Storage 33 (2021) 102109.

[9] J. Chen, Z. Zhou, Z. Zhou, X. Wang, B. Liaw, Impact of battery cell imbalance on electric vehicle range, Green Energy Intell. Transp. 1 (3) (2022) 100025.

[10] Z.B. Omariba, L. Zhang, D. Sun, Review of battery cell balancing methodologies for optimizing battery pack performance in electric vehicles, IEEE Access 7 (2019) 129335–129352.

[11] X. Wang, S. Li, L. Wang, Y. Sun, Z. Wang, Degradation and dependence analysis of a lithium-ion battery pack in the unbalanced state, Energies 13 (22) (2020) 5934.

[12] A.K.M.A. Habib, M.K. Hasan, G.F. Issa, D. Singh, S. Islam, T.M. Ghazal, Lithium-ion battery management system for electric vehicles: constraints, challenges, and recommendations, Batteries 9 (3) (2023) 152.

[13] A.K.M.A. Habib, M.K. Hasan, M. Mahmud, S.M.A. Motakabber, M.I. Ibrahimya, S. Islam, A review: energy storage system and balancing circuits for electric vehicle application, IET Power Electron. 14 (2021) 1–13.

[14] T. Hein, A. Ziegler, D. Oeser, A. Ackva, A capacity-based equalization method for aged lithium-ion batteries in electric vehicles, Electr. Power Syst. Res 191 (2021).

[15] F. Eroglu, M. Kurto, M. Vural, Bidirectional dc–dc converter based multilevel battery storage systems for electric vehicle and large-scale grid applications: a critical review considering different topologies, state-of-charge balancing and future trends, IET Renew. Power Gener. 15 (2021) 915–938.

[16] A. Turksoy, A. Teke, A. Alkaya, A comprehensive overview of the dc-dc converter-based battery charge balancing methods in electric vehicles, Renew. Sustain. Energy Rev. 133 (2020) 110274.

[17] Y.-H. Park, R.-Y. Kim, Y.-J. Choi, An active cascaded battery voltage balancing circuit based on multi-winding transformer with small magnetizing inductance, Energies 14 (5) (2021) 1302.

[18] G. Noh, J. Lee, J.-I. Ha, Design and analysis of single-inductor power converter for both battery balancing and voltage regulation, IEEE Trans. Ind. Electron. 69 (2021) 2874–2884.

[19] A. Khalid, A. Stevenson, A.I. Sarwat, Performance analysis of commercial passive balancing battery management system operation using a hardware-in-the-loop testbed, Energies 14 (23) (2021) 8037.

[20] Z.D. Taylor, H. Akhavan-Hejazi, H. Mohsenian-Rad, Power hardware-in-loop simulation of grid-connected battery systems with reactive power control capability, in: North American Power Symposium, NAPS, 2017.

[21] N. Bouchhima, M. Schnierle, S. Schulte, K.P. Birke, Active model-based balancing strategy for self-reconfigurable batteries, J. Power Sources 322 (2016) 129–137.

[22] A. Pozzi, M. Zambelli, A. Ferrara, D.M. Raimondo, Balancing-aware charging strategy for series-connected lithium-Ion cells: a nonlinear model predictive control approach, IEEE Trans. Control Syst. Technol. 28 (2020) 1862–1877.

[23] F.S.J. Hoekstra, H.J. Bergveld, M.C.F. Donkers, Optimal control of active cell balancing: extending the range and useful lifetime of a battery pack, IEEE Trans. Control Syst. Technol. 30 (2022) 2759–2766.

[24] R.D. Fonso, X. Sui, A.B. Acharya, R. Teodorescu, C. Cecati, Multidimensional machine learning balancing in smart battery packs, in: Annual Conf. of the IEEE Industrial Electronics Society, 2021.

[25] B. Wu, W.D. Widanage, S. Yang, X. Liu, Battery digital twins: perspectives on the fusion of models, data and artificial intelligence for smart battery management systems, Energy AI 1 (2020).

[26] Z.B. Wei, J.Y. Zhao, H.W. He, G.L. Ding, H.Y. Cui, L.C. Liu, Future smart battery and management: advanced sensing from external to embedded multi-dimensional measurement, J. Power Sources 489 (2021) 229462.

[27] Y. Weng, C. Ababei, Battery pack cell balancing using topology switching and machine learning, in: IEEE Vehicle Power and Propulsion Conf., VPPC, 2022, pp. 1–6.

[28] W.J. Han, T. Wik, A. Kersten, G.Z. Dong, C.F. Zou, Next generation battery management systems: dynamic reconfiguration, IEEE Ind. Electron. Mag. 14 (4) (2020) 20–31.

[29] K. Liu, Z. Wei, C. Zhang, Y. Shang, R. Teodorescu, Q.-L. Han, Towards long lifetime battery: AI-based manufacturing and management, IEEE/CAA J. Autom. Sinica 9 (7) (2022) 1139–1165.

[30] G. Plett, Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs. Part 1. Background, J. Power Sources 134 (2) (2004) 252–261.

[31] G. Plett, ECE5720: Battery Management and Control, University of Colorado Colorado Springs, 2015, [Online]. Available: http://mocha-java.uccs.edu/ECE5720/index.html.

[32] G. Plett, ECE4710/5710: Modeling, simulation, and identification of battery dynamics, University of Colorado Colorado Springs, 2018, [Online]. Available: http://mocha-java.uccs.edu/ECE5710/index.html.

[33] W. Greg, B. Gary, An introduction to the Kalman filter, 2006, [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

[34] Y. Weng, Detection and Characterization of Actuator Attacks Using Kalman Filter Estimation (M.S. thesis), Marquette University, 2019.

[35] Y. Wang, X. Lin, Y. Kim, N. Chang, M. Pedram, Architecture and control algorithms for combating partial shading in photovoltaic systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 33 (6) (2014) 917–930.

[36] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction, KanGAL Report Number 2011003, 2011.

[37] Y. Akishita, Y. Ohsita, M. Murata, Network power saving based on Pareto optimal control with evolutionary approach, in: Int. Conf. on Computing, Networking and Communications, ICNC, 2017.

[38] C.E.A. Bundak, M.A.A. Rahman, M.K.A. Krim, N.H Osman, Fuzzy rank cluster top k Euclidean distance and triangle based algorithm for magnetic field indoor positioning system, Alex. Eng. J. 61 (5) (2022) 3645–3655.

[39] A. Kumar, Differences between random forest vs AdaBoost, 2022, [Online]. Available: https://vitalflux.com/differences-between-random-forest-vs-adaboost/.

[40] R. Saini, Integrating vegetation indices and spectral features for vegetation mapping from multispectral satellite imagery using AdaBoost and random forest machine learning classifiers, Geomat. Environ. Eng. 17 (1) (2023) 57–74.

[41] D. Yang, Y.J. Wang, R. Pan, R.Y. Chen, Z.H. Chen, A neural network based state-of-health estimation of lithium-ion battery in electric vehicles, Energy Procedia 105 (2017) 2059–2064.

[42] Y.N. Sun, J.L. Zhang, K.F. Zhang, H.H. Qi, C.J. Zhang, Battery state of health estimation method based on sparse autoencoder and backward propagation fading diversity among battery cells, Int. J. Energy Res. 45 (5) (2021) 7651–7662.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958.

[45] Y. Wei, F. Yang, M.J. Wainwright, Early stopping for kernel boosting algorithms: a general analysis with localized complexities, IEEE Trans. Inform. Theory 6 (10) (2019).

[46] Infineon, IRFP4668, 2023, [Online]. Available: https://www.infineon.com/cms/en/product/power/mosfet/n-channel/irfp4668/.

[47] O.L. Gallaga, How to take care of your electric vehicle battery, 2023, [Online]. Available: https://www.wired.com/story/how-to-take-care-electric-vehicle-battery/.