Smart Reconfigurable Battery Packs with Scalable AI Based Cell Balancing

Yuqin Weng and Cristinel Ababei
Electrical and Computer Engineering Dept.
Marquette University, Milwaukee WI, USA
{yuqin.weng,cristinel.ababei}@marquette.edu

***Abstract*—Imbalance between state of charge (SoC) of cells in battery packs can cause numerous issues, including reduction of usable capacity level, degradation of performance, and shortening of lifetime. Successful approaches to mitigate such issues employ cell balancing techniques. Building on our preliminary results, in this paper, we expand our novel cell balancing technique to larger battery packs to investigate scalability aspects and to explore additional machine learning (ML) models that we employ in the proposed cell balancing algorithm. More specifically, we adopt a divide-and-conquer approach, in which the battery pack is divided into smaller partitions to which we apply the proposed ML based cell balancing. Extensive simulation experiments conducted on a 24 cells battery pack demonstrate good scalability and improved battery runtime achieved with the proposed balancing approach.**

***Index Terms*—battery cell balancing, machine learning, neural network, reconfigurable battery pack**

I. Introduction and Previous Work

One of the main challenges for battery packs, such as those used in electric vehicles (EVs), is related to the differences that exist among battery cells. These slight variations (for example, in terms of capacity, internal resistance, etc.) can result in cells imbalance that tends to increase as the pack ages, primarily due to variances in different operational conditions [1]. In turn, cells imbalance restricts the charging and discharging processes and limit the amount of usable energy per charge, thereby decreasing the total runtime/driven-distance per charge.

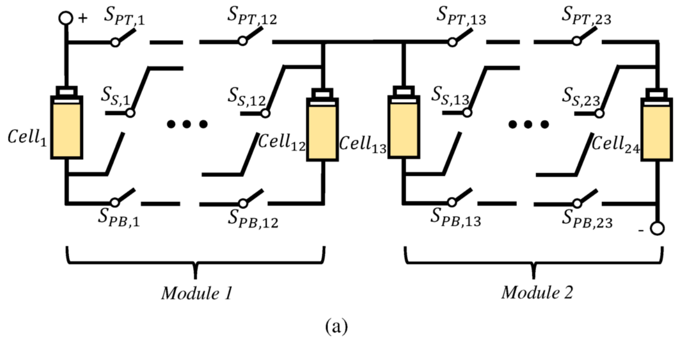
A popular approach to combat cells imbalance is to use cell balancing techniques [2], which can be generally classified into active and passive balancing as presented in [2], [3]. For example, the study in [2] proposed a model-predictive-controller (MPC) for active cell balancing. Passive balancing is employed by the study in [3], which uses an iterative algorithm that progressively reduces the imbalance in small consecutive steps by activating bleeding resistors. Machine learning has recently been used too to design balancing algorithms [4]–[6]. In [4], a machine learning control algorithm is developed in Matlab to insert/bypass cells and achieve balancing of both cell SoC and temperature. In [5], several machine learning algorithms (back propagation neural network (BPNN) and long short-term memory (LSTM)) are proposed and reported improved balancing time and optimal power loss management. The study in [6] investigated a feedforward neural network to distinguish between balanced and imbalanced Lithium-ion battery strings. In our recent work [7], we proposed a novel balancing technique that used a reconfigurable switching network to periodically change the pack topology in a way that achieves cell balancing.

In this paper, we extend our cell balancing technique from [7] to larger battery packs. We are interested in: 1) to investigate scalability aspects of the technique and 2) to explore additional ML models that we employ to predict the next best pack topologies to switch to periodically. A key benefit of deploying such an artificial intelligence (AI) approach to battery pack reconfiguration is that the AI approach has the ability to retrain the model using data collected from the battery pack operation, without the necessity to make adjustments to other models, such as equivalent circuit models used in traditional approaches.

II. Reconfigurable Battery Pack Architecture

Our proposed cell balancing algorithm is presented in the context of an assumed reconfigurable battery pack structure or architecture. The architecture is inspired by the work in [8], where a reconfigurable network of switches was used in photovoltaic (PV) arrays to reconfigure them to address partial shading issues. We adopted that reconfigurable network of switches and applied it in the context of battery packs - where we reconfigure the switches to implement different topologies and address cells imbalance issues [7]. The idea is to associate a set of three switches ($S_{PT,i}$, $S_{PB,i}$ and $S_{S,i}$) with each cell in the pack. The switches then can be configured to create combinations of series-parallel connections of all the cells to form various topologies with arbitrary number of rows and columns on each row.

In this paper, we further modify the network of switches by applying it individually to partitions of the battery pack, as illustrated in Fig 1.a, where the battery pack is split into two equal partitions indicated as *Module 1* and *Module 2*. In this way, we effectively implement a divide-and-conquer approach to deal with the exponential increase in the number of total different topologies that can be created for packs with increasing numbers of cells. In other words, instead of applying our previous balancing technique [7] to the overall pack, we first split the pack into partitions with equal number of cells and then apply the balancing technique to each partition separately. In this way, we address the challenge of scalability. An added benefit of this approach is that the



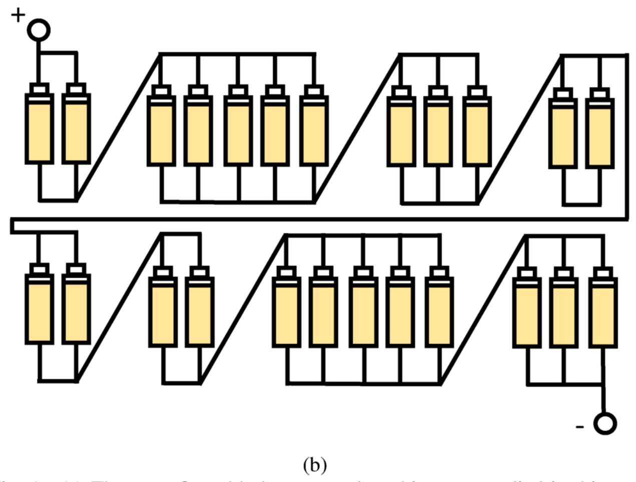
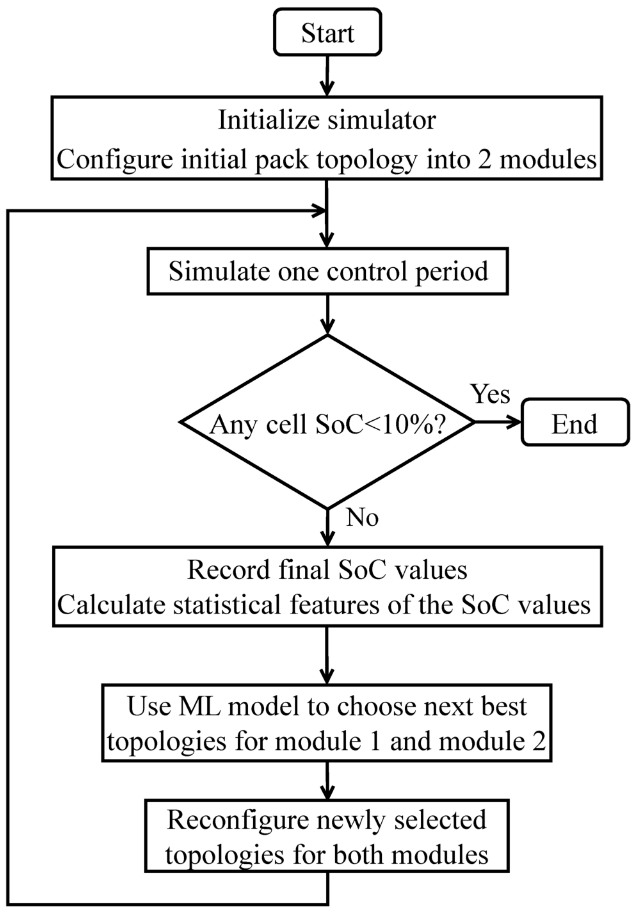


Fig. 1. (a) The reconfigurable battery pack architecture studied in this paper has 24 cells split into two separate modules, with 12 cells in each module. Modules are connected in series. (b) An example of an actual battery pack topology: module 1 with a [2,5,3,2] structure and module 2 with a [2,2,5,3] structure.



Start
Initialize simulator
Configure initial pack topology into 2 modules
Simulate one control period
Any cell SoC<10%?
Yes
End
No
Record final SoC values
Calculate statistical features of the SoC values
Use ML model to choose next best topologies for module 1 and module 2
Reconfigure newly selected topologies for both modules

Fig. 2. Flowchart of the proposed balancing algorithm; topology switching via reconfiguration is done periodically, with a control period of 5 minutes.

same AI model that is developed for one of the partitions is deployed for all other partitions, without the need to retrain the model. Hence, this scalable approach is efficient too. Further details will be provided in the next section. Fig. 1.b shows an example of a pack topology, where the first module has a structure defined as [2,5,3,2] and 2nd module has a [2,2,5,3] structure. This is achieved by configuring appropriately the on/off status of the reconfigurable switches in the battery pack. The [2,5,3,2] structure defines a topology involving 12 cells arranged as a connection of 4 rows in series. Each of the four rows has a certain number of cells connected in parallel; for example the first row has 2 cells in parallel and so on. The output voltage level of the battery pack depends on how many rows a module has. The first row in Fig. 1.b is module 1 ([2,5,3,2]) and the second row in Fig. 1.b is module 2 ([2,2,5,3]).

III. Proposed Balancing Algorithm

A. Algorithm Description

The flowchart of the proposed cell balancing algorithm is shown in Fig. 2. This is illustrated in the context of the simulation tool (discussed later), which is used as a testing means as well. The balancing algorithm is presented for a discharging process - captured by one full simulation - during

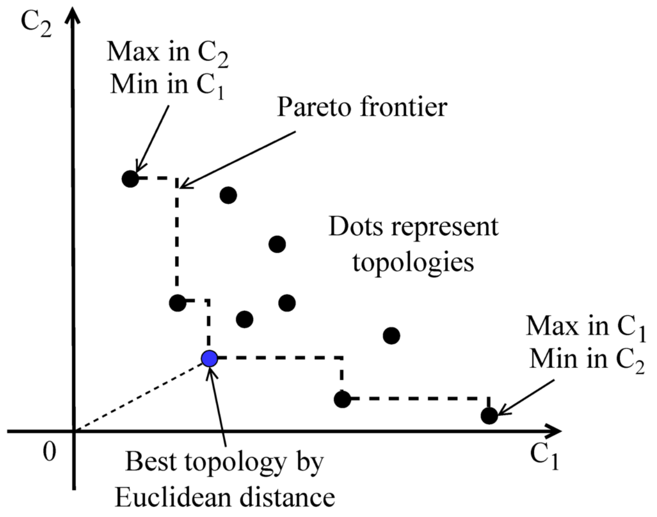


Fig. 3. Example of Pareto frontier in the space defined by the cost function components C_1 and C_2 defined by eqs. 2 and 1.

```
Algorithm: Dataset Generation based on Pareto Optimality
1: In: Reconfigurable module architecture, workload
2: Out: Dataset for training and testing ML model
3: for $range_u \leftarrow [0.3 : 0.15 : 0.9]$ do
4:     for $j \leftarrow 1$ to $N$ do // N: number of topologies
5:         for $i \leftarrow 1$ to $M$ do // M: 400 simulations
6:             Configure battery module to initial topology $j$
7:             Randomly select initial SoC from $range_u$
8:             Randomly init. cells params. // sample Gaussian distr.
9:             for $k \leftarrow 1$ to $N$ do
10:                Configure battery module to next topology $k$
11:                Run simulation for the next control period
12:                Record costs by eqs. (1) and (2)
13:                Identify Pareto Frontier (PF)
14:                Find best next topology from PF with eq. (3)
15:                Record new datapoint: (Input features, Label),
16:                Where: Input features: end SoC vals., $j$, statistics
17:                Where: Label: next best topology index
18:            end for
19:        end for
20:    end for
21: end for
```

Fig. 4. Pseudocode of the simulation process used to generate datasets.

which if any of the cells reaches an SoC of 10% or less, then the pack discharging is stopped and the simulation ends. Note that the value of 10% is chosen arbitrarily, but intentionally lower in order to push the investigation of the proposed approach to the limits. This value can be easily changed inside the simulation tool by the user to other desired values. Such values can be anywhere between 5-50% that have been discussed online [9]–[12], reported by Tesla users, or used in BMW i3 REX, Mitsubishi Outlander PHEV, or recommended in previous studies [13]–[15].

The simulation is an iterative process, where each iteration simulates the duration of a control period or epoch. The two modules of the pack are initialized first; then, the simulation progresses in control periods. At the end of each control period the developed ML model (discussed later) is used to indicate which module topology should be used in the next control period, for each of the two modules of the pack. Note that the same ML model is used for inference of next best topology in both modules. The periodic reconfiguration is such that the cells balance is preserved as best as possible and the SoC runaway is constrained within ranges or spans that will allow for the discharging process to last longer - by delaying as much as possible for any of the cells SoC to reach first 10%.

B. Generation of Datasets Used for Model Training

At the core of the proposed balancing algorithm lies the ML model used to make predictions at the end of each control period about which specific pack topology to possibly switch to (via network reconfiguration) in the next control period. These predictions are done for each of the two modules of the battery pack. To develop such ML models however we need training data, which is not readily available anywhere. This is perhaps the biggest challenge of the approach presented in this paper.

To address this challenge, we developed our simulation tool with features and running modes that allow us to generate the datasets we need for ML model development and training. More specifically, the simulation tool can be used to setup multiple simulations at the end of each control period - such that *what-if* explorations are conducted with the goal of identifying what would be the best topology to reconfigure the pack to for the next control period, given the *current status* of the pack. The quality of the possible candidate topologies (tried during these multiple simulations) is evaluated numerically by calculating a *cost function*. The cost function combines two cost components, denoted as C_1 and C_2: 1) the inverse of the summation of battery cells SoC values, and 2) the range or span of all SoC values, as the difference between the maximum and minimum SoC values among all cells.

$$C_1 = 1/\sum_{i=1}^{n} SoC_i \tag{1}$$

$$C_2 = SoC_{diff}^{max} = \max_{i\in[n]} SoC_i - \min_{i\in[n]} SoC_i \tag{2}$$

where we define $[n] = \{1, 2, .., n\}$ with n being the number of cells in the battery pack.

Note that the smaller the values of these cost components the better is the balancing among all cells in the pack. In fact, the two cost components define a two dimensional solution search space, where each possible topology (out of say a total of N topologies) that the battery pack can be reconfigured to represents a solution point. In such a space, there are usually multiple best solution points - and those form what is called the Pareto frontier [16]. An example of such a Pareto frontier is shown in Fig. 3. In our simulations, we identify the next best pack topology from the Pareto frontier - the one that is the closest to the (0,0) coordinates in the space. The closeness is calculated as the Euclidean distance:

$$d = \arg\min_{k} \sqrt{\left(C_1^k\right)^2 + \left(C_2^k\right)^2} \tag{3}$$

The total number of possible topologies N that are evaluated at the end of each control period during the simulation instrumented for dataset generation can quickly increase exponentially with the number of cells considered. That is because the total number of parallel-series connections (as that in Fig. 1.b) increases with n, the number of cells. To limit the simulation runtime for dataset generation purposes and also because many topologies are anyway not useful (e.g., all cells in series, or all cells in parallel), we restrict $N = 10$ to a small number of topologies that use 4, 5, and 6 rows only. The simulation process for dataset generation is further summarized with the help of the pseudocode in Fig. 4. Eeach topology is simulated 400 times, starting from current states given by different initial values of cell SoCs within the range 30-90% with a step of 15%. Therefore, the dataset generated will contain 400x5x10=20,000 datapoints.

TABLE I
LISTING OF ARCHITECTURAL PARAMETERS OF THE INVESTIGATED MODELS.

Model	Architectural Parameters
NN	layers input (19 neurons), hidden (128), output (10)
KNN	n = 4
SVM	C = 5000, $gamma$ = 0.0015
RF	n_estimators = 500, max_depth = 7

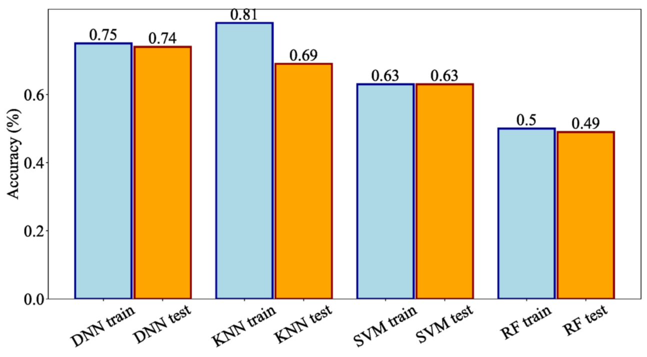


Fig. 5. Comparison of accuracy achieved with each of the four investigated models.

C. Machine Learning Models

In this paper, we investigate several machine learning models with the objective of identifying the one that provides the best performance for our application context. More specifically, we looked at: neural network (NN), K-nearest neighborhood (KNN), support vector machine (SVM), and random forest (RF) models. The dataset generated as described in the previous section is used to train and test these models using a 60/40% split. The training time of each of the models is restricted to be the same for all models for a fair comparison. The main architectural parameters of the models are listed in Table I. The results of the comparison of these models is summarized in Fig. 5. Based on this investigation, we concluded that the NN model was the best. Therefore, we selected this model to use in the proposed cell balancing algorithm. All the results presented in the remainder of this paper are obtained using the NN model. Further training was conducted using $CategoricalCrossEntropy$ as loss function, optimizer $Adam$, and $LearningRate$ of 0.00015.

IV. SIMULATION RESULTS

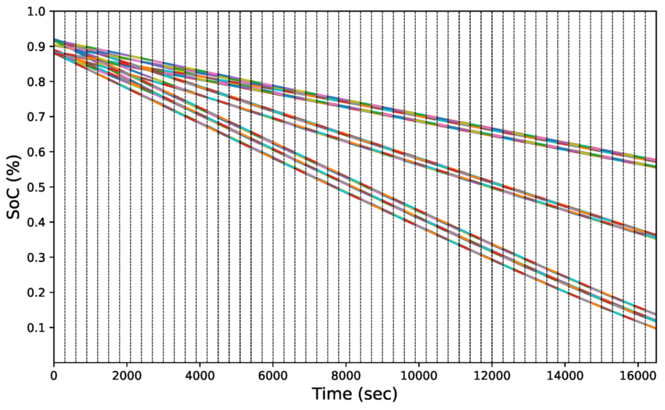
All experiments, including the dataset generation used for model training, are conducted with a custom battery pack simulation tool, which we developed for this purpose. The simulation tool integrates enhanced self-correcting (ESC) models for battery cells whose SoC is estimated with extended Kalman filtering (EKF) techniques. The tool has the ability to numerically simulate any battery pack topology that results through the reconfiguration process used by the balancing algorithm. For a more detailed description of the simulation tool, please see our recent work [7]. All simulations are started from initial states for all cells that are generated randomly from within reasonable or rational ranges - in order to mimic realistic differences between cells. For example, initial values of cells SoC are sampled from Gaussian distributions characterized by a mean of 0.9 (representing 90%) and a 5% standard deviation. The proposed balancing algorithm is tested for two types of battery discharging workloads: 1) constant workload and 2) combination of multiple urban dynamo-meter drive schedule ($UDDS$) workload. Each simulation experiment is stopped when any of the cells SoC reaches the lower limit of 10%.

The simulation results for the constant workload case are presented in Fig. 6. Fig. 6.a shows the variation of cells SoC for the battery architecture that we use as reference for comparison. The reference architecture includes the 24 cells split into two partitions or modules, where each has topologies [2,5,3,2] and [2,2,5,3]. These two fixed topologies are selected because they are the top two topologies that were found to be preferred by the ML model as the next best topology during dataset generation and model training. In other words, if the battery pack topology is to be kept fixed, then, these two topologies are the best at optimizing the two cost components from eqs. (1) and (2). Fig. 6.b shows the variation of cells SoC when the reconfigurable battery pack discussed in this paper has its topology changed periodically (every 300 sec or 5 min) as dictated by the proposed balancing algorithm. In these figures, x-axis represents time in seconds (i.e., drive time of an EV) and y-axis represents the SoC as a percentage, but normalized to a range from 0 to 1. For a correct comparison, the initial starting values of all cells SoCs (generated by sampling the distributions discussed earlier) are exactly the same in both simulations, the reference and proposed cases. Comparing the two plots from Fig. 6, we find that the battery runtime is improved from 16,500 sec (reference case) to 23,400 sec (proposed balancing); this represents an improvement of 41.8%. As expected, the SoCs span/range at the stopping time is decreased from 0.480 to 0.076 due to the balancing, which makes for all SoCs values to stay clustered together as the battery pack is discharged more uniformly.

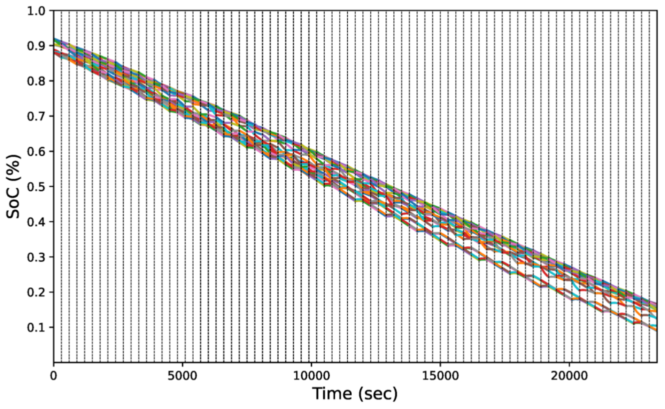
The simulation results for the $UDDS$ workload case are presented in Fig. 7. Comparing the two plots, we again find that the battery runtime is improved from 25,200 sec to 35,700 sec, which represents an improvement of 41.7%. Similarly, the SoCs span/range at the stopping time is decreased from 0.485 to 0.077 due to the balancing. Finally, Fig. 8 presents a summary of the above comparisons. It shows how battery runtime and cells SoC span/range improved in both above simulation experiments. We note that these results are better than those reported recently in [7] and they suggest that the approach presented in this paper - of divide at pack level and conquer locally - is a good and scalable strategy to deploy reconfigurable network switches and balancing algorithms.

V. CONCLUSION

We presented a new SoC balancing algorithm for reconfigurable battery packs. The main contributions of this paper include: 1) To deal with the larger number of battery cells considered, we presented a divide-and-conquer approach where the pack is divided into smaller partitions to which



(a)



(b)

Fig. 6. Constant workload case: (a) Variation of cells SoC for the *reference* battery pack fixed topology formed by two module: $[2, 5, 3, 2]$ and $[2, 2, 5, 3]$. (b) Variation of cells SoC when the proposed cell balancing algorithm is used. SoC values are normalized to range [0,1].

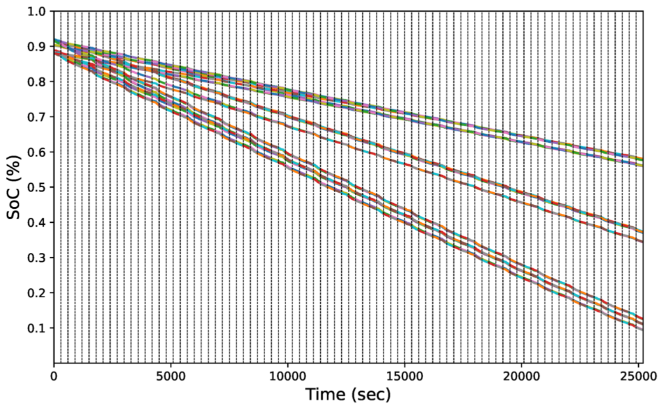
we apply the proposed neural network based cell balancing. 2) We investigated several machine learning models to identify the one that can provide the best results; we found that neural networks were the best. 3) To test the proposed balancing algorithm, we conducted simulation experiments with a custom simulation tool on a battery pack with 24 cells. Simulation results conducted with both constant and urban dynamo-meter drive schedule workloads demonstrated that the battery runtime can be improved with up to 41%.

References

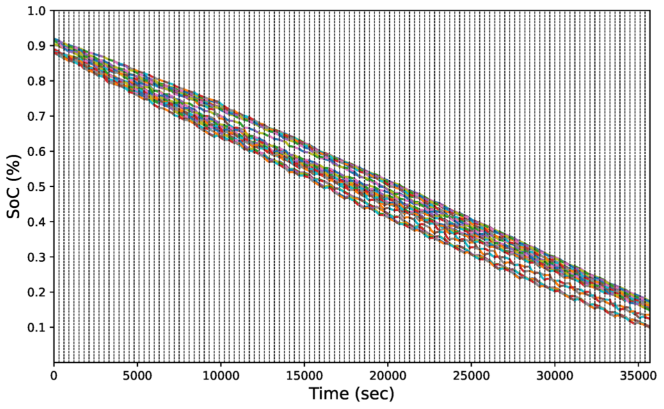
[1] M. Berecibar, F. Devriendt, M. Dubarry, I. Villarreal, N. Omar, W. Verbeke, and J.V. Mierlo, "Online state of health estimation on NMC cells based on predictive analytics." *J. of Power Sources*, vol 320, pp. 239-250, 2016.

[2] F.S.J. Hoekstra, H.J. Bergveld, and M.C.F. Donkers, "Optimal control of active cell balancing: extending the Range and useful lifetime of a battery pack," *IEEE Trans. on Control System Technology*, vol 30, no. 6, 2022.

[3] R. Di Rienzo, M. Zeni, F. Baronti, R. Roncella, and R. Saletti, "Passive balancing algorithm for charge equalization of series connected battery cells," *IEEE Int. Conf. on Industrial Electronics for Sustainable Energy Systems (IESES)*, pp. 73-79, 2022.



(a)



(b)

Fig. 7. $UDDS$ workload case: (a) Variation of cells SoC for *reference* battery pack fixed topology formed by two module: $[2, 5, 3, 2]$ and $[2, 2, 5, 3]$. (b) Variation of cells SoC when the proposed cell balancing algorithm is used. SoC values are normalized to range [0,1].

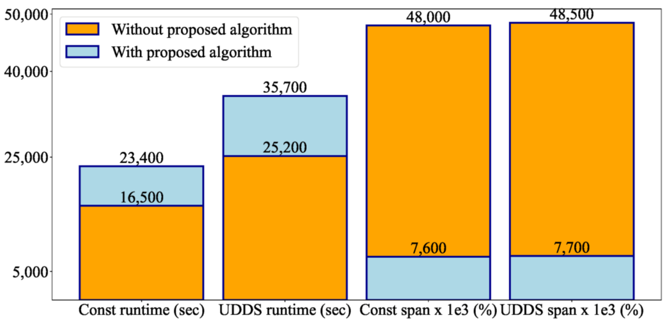


Fig. 8. Summary of simulation experiments.

[4] R. Di Fonso, X. Sui, A.B. Acharya, R. Teodorescu, and C. Cecati, " Multidimensional Machine Learning Balancing in Smart Battery Packs," *Conf. of the IEEE Industrial Electronics Society*, pp. 1-6, 2021.

[5] T. Duraisamy and D. Kaliyaperumal, "Machine learning-based optimal cell balancing mechanism for electric vehicle battery management system," *IEEE Access*, vol. 9, pp. 132846-132861, 2021.

[6] A. Gen Li and M. Preindl, "State of charge imbalance classification of lithium-ion battery strings using pulse-injection-aided machine learning," *IEEE Transportation Electrification Conference & Expo (ITEC)*, pp. 958-963, 2021.

[7] Y. Weng and C. Ababei, “Battery pack cell balancing using topology switching and machine learning,” *IEEE Conference on Vehicle Power and Propulsion (VPPC)*, 2022.

[8] Y. Wang, X. Lin, Y. Kim, N. Chang, and M. Pedram, “Architecture and control algorithms for combating partial shading in photovoltaic systems,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 6, pp. 917-930, June 2014.

[9] P. Lima, “Battery charging: Full versus Partial,” [Online]. Available: *https://pushevs.com/2018/04/27/battery-charging-full-versus-partial/*, 2018.

[10] Blackridge Research, “Top 10 Tips to Maximize EV Battery Life,” [Online]. Available: *https://www.blackridgeresearch.com/blog/top-tips-to-maximize-electric-vehicle-ev-battery-life-capacity-longevity-performance*, 2022.

[11] D. Xiang, “Is It Bad to Charge An Electric Vehicle to 100%,” [Online]. Available: *https://www.midtronics.com/blog/is-it-bad-to-charge-an-electric-vehicle-to-100/*, 2023.

[12] B. Gaton, “Debunking the 80/20 limits on EV battery charging: More FUD from fossil fuel industry,” [Online]. Available: *https://thedriven.io/2023/04/04/debunking-the-80-20-limits-on-ev-battery-charging-more-fud-from-fossil-fuel-industry/*, 2023.

[13] H. Abdi, B. Mohammadi-ivatloo, S. Javadi, A. Reza Khodaei, and E. Dehnavi, “Chapter 7 - Energy Storage Systems, Distributed Generation Systems,” *Butterworth-Heinemann, Editors: G.B. Gharehpetian, S. Mohammad Mousavi Agah*, 2017.

[14] E. Wikner and T. Thiringer, “Extending Battery Lifetime by Avoiding High SOC,” *Appl. Sci., MDPI*, 2018.

[15] Battery University, “BU-1003a: Battery Aging in an Electric Vehicle (EV),” [Online]. Available: *https://batteryuniversity.com/article/bu-1003a-battery-aging-in-an-electric-vehicle-ev*, 2023.

[16] K. Deb, “Multi-objective optimization using evolutionary algorithms: an introduction,” KanGAL Report Number 2011003, 2011.