Pack Sizing and Reconfiguration for Management of Large-scale Batteries

Fangjian Jin and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2121
Email: {askme,kgshin}@umich.edu

***Abstract*—Battery systems for electric vehicles (EVs) or uninterruptible micro-grids—prototypical cyber-physical systems (CPSs)—are usually built with several hundreds/thousands of battery cells. How to deal with the inevitable failure of cells quickly and cost-effectively for vehicle warranty or uninterruptible service, for instance, is key to the development of large-scale battery systems. Use of extra (redundant/backup) cells to cope with cell failures must be minimized so as to make the target systems cheaper and lighter, while meeting the reliability requirement that is directly related to, for example, the vehicle warranty. Existing reconfigurable battery systems do not scale well because they incur a long delay in properly setting a large number of switches to bypass faulty cells or adapting to dynamically changing power demands in large battery systems for such applications as EVs.**

In this paper, we propose a scalable solution, not only to reduce the required number of backup cells and the total cost of a battery system, but also to facilitate recovery from cell failures and adapt to changing power demands while increasing battery utilization. Specifically, we optimize the pack-size by striking a balance between various types of cost in order to reduce the overall cost. We also configure battery packs and optimize their connection topology, reducing delays in failure recovery and power reallocation. Our in-depth evaluation has shown that the time to recover from cell failures remains constant irrespective of the number of cells involved, which is important to scalability. The proposed pack-sizing also reduces the cost and the size of battery systems. Moreover, fast power reallocation is achieved by utilizing prior knowledge of power usage patterns.

I. Introduction

Seen as a promising way to reduce use of fossil fuel, electric vehicles (EVs) are gaining popularity as a solution to the global warming problem. A recent sharp increase in fuel price has also accelerated the demand for EVs worldwide. Respondents to a recent survey [15] stated their strong interest in EVs, with 44% of them expressing their strong desire to purchase an EV. Twenty-nine million EVs were sold in 2010 alone and the number is predicted to rise 69% to 49 million in the next decade [14]. An EV is often powered by hundreds or thousands of battery cells.

Facing the battery performance challenges in EVs, Balch *et al.* [3] proposed a design of battery packs and a choice of appropriate chemical material according to vehicle type and zero-emission range in order to lower the costs and maximize the vehicle performance. However, this solution did not account for fault-tolerance, another key battery management issue.

The non-negligible cell failure rate in large-scale battery systems has become the roadblock in the development of cost-effective EVs. For example, the failure and replacement rate of batteries in Honda hybrids is reported to be as high as 1% [5], [12].

Vehicle manufacturers usually duplicate every battery pack to avoid deep-discharge of battery cells designed for starting engines, since repeated deep-discharges will cause capacity loss and, eventually, premature failure as a result of electrode disintegration due to mechanical stresses arising from cycling [20]. Although this solution can reduce the battery failure rate, it makes batteries both bulky and expensive. According to the report in [19], a battery with a 100-mile range is as bulky as a large gas tank. A much larger battery, such as the one in the Tesla Motors S-model, with a 300-mile range costs, $43,500 in 2011.

To reduce battery failures caused by cell voltage imbalance, Stuart *et al.* [17] proposed a modular battery management system with equalizers, which balance the State-of-Charge (SoC) of cells in order to prevent excessively high/low voltages that can virtually damage all types of batteries. This solution, however, does not work when cells connected in series become faulty since it cannot bypass faulty cells.

To tolerate the failure of battery cells connected in series, researchers proposed addition of programmable switches around battery cells to bypass faulty cells [1], [6], [18]. While these methods work well for small-scale batteries via static configuration, they are not designed to reconfigure large-scale batteries in real time.

Another popular battery-management function is to provide dynamic power in real time. For example, on a steep up-slope road, the vehicle's motor must have a higher torque and fairly high power output, while for flat or down-slope terrains, the motor is required to supply lower power [9]. While transformers only work for AC [8], the conventional solution is to use a DC-DC converter, which is not very power-efficient unless the difference between input and output voltages is small [13]. Besides such a limited range of voltage, the leakage of power is also a major problem in using a DC-DC converter [7].

To achieve fault-tolerance and deliver dynamic power, Kim and Shin [10] proposed a framework to reconfigure and control a large number of battery packs, achieving resilience to cell failures and outputting diverse levels of power. By placing programmable switches around each cell, this reconfigurable framework can bypass faulty cells. While this reconfiguration framework may extend battery life, it has not accounted for the following important issues/problems.

1) *Inflexible connectivity under-utilizes battery cells*. Exist-

ing reconfiguration frameworks, including the one in [10], restrict the way in which battery cells and packs are connected via programmable switches. For example, they require homogeneous configuration of cells and packs. All parallel branches have the same number of packs, all of which provide the same level of voltage and current. In addition, they only support at most two-level hierarchy of connections. As a result, even if there are a sufficient number of battery cells in the system, they could not be utilized to provide the required power. Such a restriction results in using more battery cells than necessary and an imbalanced SoC. Thus, we need more flexible battery connectivity that enables full utilization of cells to meet the dynamic power requirements.

2) *Long reconfiguration time prolongs failure recovery and power reallocation.* Due to the hazard and risk in switch operations, one may have to turn on/off switches sequentially, making the reconfiguration time proportional to the number of switches used in the system. In a large-scale battery system, this can become a serious problem. Long delays in bypassing faulty cells may cause not only overheat but also high self-discharge, rendering healthy cells to be overcharged [4]. Time-consuming reconfiguration also fails to meet the dynamically changing power demands in a timely manner. To overcome these undesirable effects of reconfiguration, we need speedy failure recovery and power reallocation.

3) *Inadequate pack-sizing incurs high cost*. Pack-sizing determines not only the number of cells in a pack but also the total number of cells. Existing solutions [1], [18] do not account for the effect of pack size on the cost. They also fail to compute the number of necessary backup battery cells, thereby either compromising the ability to provide the required power reliably, or requiring more backups than needed in order to maintain the required level of power in the presence of cell failures. Our goal is to reduce the number of cells while providing the required power reliably. It is also important to optimally distribute backup cells over different packs and determine the optimal number of cells in each pack.

In this paper, we address the above problems as follows.

- The connections of battery cells and packs are modeled using *Power Trees* (PTs), which enable flexible connectivity of cells and packs, and are used to optimize battery connections in order to fully utilize battery cells in the system and provide the required power.
- The reconfiguration strategy is optimized to adapt the battery system to load changes and cell failures in a timely manner. Specifically, our solution sets up battery connections in order to minimize the number of switch operations in coping with load changes and/or cell failures.
- The pack size is optimized by considering the design and the connections of packs as a whole. Thus, the total required number of backup cells is minimized at the meantime meeting the required reliability. The optimal pack size and the number of packs are also calculated efficiently by using a heuristic to lower the total system costs.

The rest of the paper is organized as follows. Section II covers the background of reconfigurable battery management, and motivates our work. Section III describes the model for battery connection topology, and states the problem that we are trying to solve. Section IV describes how to configure battery packs to recover from cell failures, reallocate power and lower the overall system costs. Section V evaluates the performance of our pack sizing and configuration. Finally, we conclude the paper in Section VI.

II. System Architecture and Motivation

A. Reconfigurable Battery System

The battery system under consideration is based on the reconfigurable battery system proposed in [10]. Each battery pack consists of a group of control units under a local controller and an array of battery cells. These control units are designed for turning on/off switches around each battery cell so that the connections of these cells can be reconfigured online in series, parallel, or a combination of both. For example, in Fig. 1, when we want to connect cells in parallel, switches (1) and (4) will be turned on by the associated control units. Meanwhile, turning on switches (5) and (6) allows more cells to be connected in parallel. When only switch (3) is turned on, the cells will be connected in series. We can also make multiple parallel groups by turning on switches (1) and (3) around the first cell, and switch (4) around the following cell. We can add more cells in between. Terminal switches (5) and (6) are separating this group from other cells. Terminals are connected/disconnected to the load. With the switches in packs, we can organize cells in series, parallel or many parallel branches of series. The design details of switches, controllers and sensors of SoC can be found in [10], [11]. We can treat a battery pack as a module, just like a single cell, and to use switches around packs to reconfigure them. For flexible reconfiguration, we can use switches to connect all pairs of battery packs' terminals, the number of which is much smaller than that of the cells. Such configuration enables hierarchical connection of the packs. The number of switches around the cells is linear in that of the cells, while the number of those around packs is the square of that of packs.

This architecture suffers from hazards and risks that limit the ability to parallelize the switch operations. For example, when a cell is bypassed, switch (2) in Fig. 1 is turned on. If the system is changed to connect this bypassed cell in series, we have to turn off switch (2) and turn on switch (3). If we operate on these two switches at the same time, switch (3) can possibly be turned on before turning off switch (2) due to random delays and an uncertain order of operations, thus short-circuiting this cell. In case of dangerous connection of batteries, we have to impose restrictions on the order of turning on/off switches. One can simply operate switches *sequentially* in order to avoid hazards and risks. Complex scheduling of switch (on/off) operations can be done but is left as our future.

B. Motivation

1) Fully Utilize Battery Cells: Existing reconfiguration schemes restrict battery connections and the number of cells that can be used in each pack. For example, in Fig. 2, each of two packs contains three cells. One of the packs contains

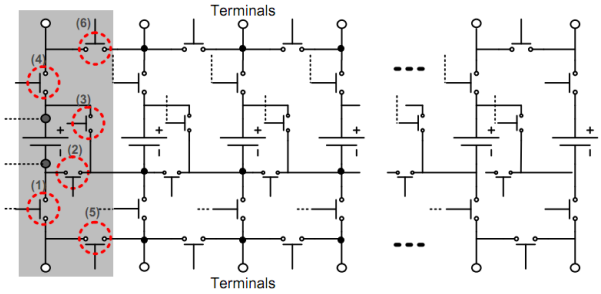

Fig. 1. Schematic diagram of a reconfigurable battery system [10]

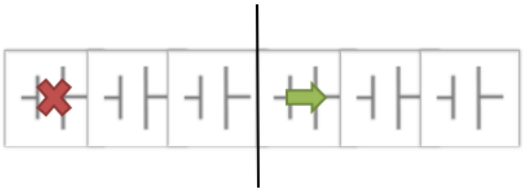

Fig. 2. Existing reconfiguration schemes (e.g., [10]) cannot fully utilize available healthy cells

a faulty cell. In the existing schemes, all packs are required to use the same number of cells and bypass the rest. Thus, the other pack, although containing three healthy cells, only uses two cells and bypasses one. In this configuration, we cannot provide the cumulative power of the remaining five healthy cells. Moreover, the scheme in [10] only allows a simple combination of connections, such as connections of parallel branches, each of which is a series chain.

To enhance battery utilization, we need to improve the existing reconfiguration schemes in the following two aspects:

1) Allow an arbitrary number of cells to be used in each pack; and
2) Allow sophisticated hierarchical connections of batteries, which are more flexible than the existing schemes.

Heterogeneous solutions are more cost-effective than the conventional homogeneous solutions. In Fig. 2, assuming 6V output voltage, the homogeneous solutions require two more cells, while the heterogeneous ones only need one more cell. This flexibility usually results in high complexity in arranging the connection of battery cells/packs. As a result, the main challenge lies in searching for a solution to the problem of dynamically connecting/disconnecting cells/packs. We meet this challenge by developing a new model of battery connections, called *Power Tree* (PT).

2) Fast Reconfiguration: When a cell fails, existing schemes first compute the number of cells that will be used in each pack, and bypass the rest of the cells. Based on the number of cells per pack, the number of packs in series and the number of parallel branches are then calculated. Finally, switches are turned on/off to change battery connections. The resulting system operates all switches around all cells and all packs, which would be time-consuming in the case of large-scale battery systems.

To reduce the reconfiguration time, we improve the reconfiguration strategy as follows.

1) Limit the reconfiguration to a small area in order to reduce switch operations that affect the performance of the entire system.
2) Optimize battery connections for reconfiguration based on the knowledge of power patterns.

We make use of the hierarchy of PT to select an appropriate part of the battery system to be reconfigured.

3) Reduce Cost via Pack-Sizing: Existing schemes do not take into account the number of cells necessary for the required power level over a given period of operation. The resulting system also fails to consider how to distribute battery cells to different packs. However, all of these factors are crucial to the reliability and the costs of a battery system.

To minimize costs without compromising reliability, we analyze the major cost components of a battery system by considering the following factors.

- Cell cost depends on the total number of battery cells.
- Fixed pack cost, such as the cost of controllers, is independent of the number of cells in a pack.
- Variable pack cost, such as the cost of sensors, depends on the number of cells in each pack.

The main challenge in pack-sizing lies in the inter-dependencies of pack size, reliability, and battery connections. We treat these factors as a whole to strike a balance between them.

III. Modeling and Problem Formulation

When load changes or some cells in a battery system fail, it must be reconfigured to provide the required level of power. To achieve this goal, we must answer the following three important questions.

Q1. How can we configure battery connections to meet the power requirements?
Q2. When cell failures occur or load changes, can we reconfigure battery connections as inexpensively and quickly as possible?
Q3. Over a given period of time, while providing the required power reliably, can we build such battery systems inexpensively?

In this section, we use a new model called, the *Power Tree* (PT), to represent the connection topology of battery packs. Based on the PT model, we formally define the problems mapped from the above questions.

A. Model

Suppose the underlying load requires voltage V_L, allowing over-voltage of no more than V_o, and current C_L, allowing over-current of no more than C_o. A single battery cell is assumed to be able to provide over-voltage V_o and over-current C_o. We can thus calculate n_s, the number of cells connected in series, and n_p, the number of parallel branches.

$$\begin{cases} n_s = \lceil \frac{V_L}{V_o} \rceil \\ n_p = \lceil \frac{C_L}{C_o} \rceil \end{cases}$$

The notation $(n_s \times n_p)$ is used to represent the fact that a battery pack contains n_p parallel branches each of which consists of n_s cells in series. We can also divide a battery pack into sub-packs which can be connected in series or in parallel. For example, a load is powered by a (3×2) pack consisting

of 2 parallel branches, each with 3 cells connected in series. This pack can be divided into series or parallel connections of cells as follows.

- In the case of series connection, the (3×2) pack can be divided into two sub-packs, one of which is (2×2) and the other sub-pack is (1×2). We get (3×2) by connecting (2×2) and (1×2) in series.

$$(3 \times 2) = (2 \times 2) + (1 \times 2).$$

- In the case of parallel connection, the (3×2) pack can be divided into two sub-packs, both of which are (3×1). We get (3×2) by connecting (3×1) and (3×1) in parallel.

$$(3 \times 2) = (3 \times 1) + (3 \times 1).$$

Formally, we can define the division of an $(n_s \times n_p)$ pack into a set of N sub-packs $\{(n_{si} \times n_{pi}), i = 1, 2, \cdots, N\}$:

- In the case of series connection, we obtain the $(n_s \times n_p)$ pack by connecting N sub-packs $\{(n_{si} \times n_{pi}), i = 1, 2, \cdots, N\}$ in series.

$$\begin{cases} n_s = \sum_{i=1}^{N} n_{si} \\ n_p = n_{p1} = n_{p2} = \cdots = n_{pN} \\ (n_s \times n_p) = \sum_{i=1}^{N} (n_{si} \times n_{pi}). \end{cases} \quad (1)$$

- In the case of parallel connection, we get the $(n_s \times n_p)$ pack by connecting N sub-packs $\{(n_{si} \times n_{pi}), i = 1, 2, \cdots, N\}$ in parallel.

$$\begin{cases} n_s = n_{s1} = n_{s2} = \cdots = n_{sN} \\ n_p = \sum_{i=1}^{N} n_{pi} \\ (n_s \times n_p) = \sum_{i=1}^{N} (n_{si} \times n_{pi}). \end{cases} \quad (2)$$

A pack can also be divided further recursively. For example, a (4×2) pack can be divided into two sub-packs (3×2) and (1×2) connected in series. One can continue the division of the first sub-pack in parallel into two sub-packs (3×1) and (3×1). Likewise, we can divide the second sub-pack into two sub-packs (1×1) and (1×1) in parallel.

$$\begin{cases} (4 \times 2) &= (3 \times 2) + (1 \times 2) \\ (3 \times 2) &= (3 \times 1) + (3 \times 1) \\ (1 \times 2) &= (1 \times 1) + (1 \times 1). \end{cases}$$

These divisions can be organized into a tree structure, which we call a *Power Tree* (PT). In a PT, children are the sub-packs of their parent node. For example, Fig. 3 shows the PT of recursive division of a (4×2) pack. According to Eq. (III-A), a (4×2) pack is divided into two serially connected sub-packs, (3×2) and (1×2). Thus, the node (4×2) has two children, (3×2) and (1×2). Similarly, the node (3×2) has two children, (3×1) and (3×1), and the node (1×2) has two children, (1×1) and (1×1). A circle in the figure represents division in series while a box represents division in parallel and an oval represents atomic packs that are not divisible.

Formally, we can describe the PT structure as follows.

- A node represents a pack and its children represent its sub-packs.
- A node belongs to one of the following three types.

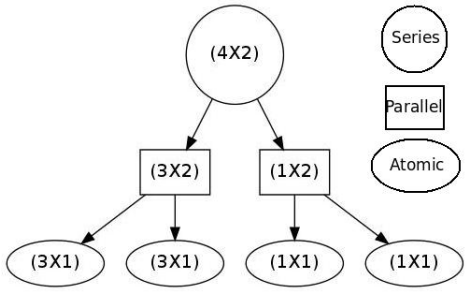

Fig. 3. PT of (4×2)

1) *Series*: children are connected in series to inherit their parent's power. Their quantitative relation satisfies Eq. (1).
2) *Parallel*: children are connected in parallel to inherit their parent's power. Their quantitative relation meets Eq. (2).
3) *Atomic*: an atomic node is a leaf node in the PT and is not divisible.

- The root of PT is the total power requirement of a given load.
- An atomic node is powered by a single battery pack.
- All but atomic nodes have each no less than two children.

B. Problems and Approaches

Using the PT structure, we would like to (i) organize batteries to provide enough power, (ii) reduce the battery reconfiguration time, and (iii) determine pack size by minimizing the total cost of a battery system.

1) Suitable Connection of Batteries: In the first question in Section III, the problem is how to connect batteries to meet the power requirements. PT is used to check if a certain connection of battery packs can meet the load requirement. For example, in Fig. 3, a battery pack of 3 cells can be used to support the power of the node (3×1) by connecting the 3 cells in series inside the battery pack. Similarly, we can use a pack of 1 cell to support the power of the node (1×1). Thus, if we have two packs of 3 cells and two packs of 1 cell, we can connect these four packs as shown in Fig. 3 to support the power (4×2).

Formally, given the load requirement $(n_s \times n_p)$, we need to construct a PT rooted at node $(n_s \times n_p)$. In this PT, there are *M* leaf nodes $\{(n_{si} \times n_{pi}),\ i = 1, 2, \cdots, M\}$. We have a set of *N* battery packs in which the number of cells is $\{b_i, i = 1, 2, \cdots, N\}$. This set of battery packs can meet the load requirement if we can construct a set of *M* pairs $\{(b_i, (n_{sj} \times n_{pj})),\ i \in 1, 2, \cdots, N;\ j = 1, 2, \cdots, M\}$, meeting the following battery-PT conditions.

(a) $M \leq N$.
(b) The first item in a pair is a battery pack and the second item in the pair is a leaf node in the PT.
(c) In a pair, battery pack b_i can support the power of the leaf node $(n_{sj} \times n_{pj})$, i.e., $b_i \geq n_{sj} \times n_{pj}$.
(d) Each leaf node $(n_{si} \times n_{pi})$ appears exactly once in the set of M pairs.
(e) Each battery pack b_i appears at most once in the set of M pairs.

2) Fast Reconfiguration: In the second question in Section III, the problem is how to make battery systems responsive to failures and load changes. As mentioned earlier, we must be able to change the connection of batteries to tolerate failures and meet the power requirements. When a cell becomes faulty, we need to reconfigure battery packs. It can be done quickly if we only need to reconfigure a small part of the battery system. Likewise, to reconfigure batteries to meet new power requirements, we can build from scratch with individual battery packs, but this can be done more efficiently by reusing parts of the existing connection arrangement. To quantify this efficiency, we define the *reconfiguration effort* as the number of battery packs to be reconfigured. Assume B_o is the packs used in the original arrangement and B_n the packs used in the new arrangement. If the reconfiguration algorithm does not reuse parts of the old arrangement, $B_o = \varnothing$.

There are three types of pack reconfiguration:

1) A battery pack, which exists in the old arrangement, has been removed in the new arrangement.
$$Set_1 = \{b_i | b_i \in B_o, b_i \notin B_n\}.$$
2) A battery pack, which is not in B_o, has been added to B_n.
$$Set_2 = \{b_i | b_i \notin B_o, b_i \in B_n\}.$$
3) A battery pack, which exists in both B_o and B_n, has been updated its connections inside or outside the pack.
$$Set_3 = \{b_i | b_i \in B_o, b_i \in B_n, b_i \ updated\}.$$

The reconfiguration effort is measured with the number of packs in these sets: $R = |Set_1| + |Set_2| + |Set_3|$. Thus, for fast reconfiguration of a battery system, we would like to minimize R subject to the constraint of valid connections.

3) Pack-Sizing: In the third question in Section III, the problem is how to reduce the cost of building a battery system. Pack-sizing determines the number of cells in a pack and the number of packs in the system to lower the overall cost while guaranteeing the required reliability. Based on battery-PT conditions, we can derive the conditions of achieving the required reliability. Suppose *N* battery packs uniformly contain *b* cells per pack. Given the battery system lifetime *T*, we can estimate the number of faulty cells $N_f(T)$ up to the lifetime*T* according to the failure distribution of battery cells.

Assuming the faulty cells are distributed uniformly among battery packs, we can construct an N-ary vector $\bar{\tau} = (\tau_1, \tau_2, \cdots, \tau_N)$, where τ_i is the number of faulty cells in the i-th battery pack. This vector meets the following constraints:
$$\begin{cases} b \geq \tau_i \geq 0, i = 1, 2, \cdots, N \\ \sum_{i=1}^{N} \tau_i = N_f \end{cases}$$

Then, we can get the N-ary vector of the remaining healthy cells $\bar{b_\tau} = (b - \tau_1, b - \tau_2, \cdots, b - \tau_N)$. Given the load requirement $(n_s \times n_p)$, the reliability condition of this battery system is

$\forall \bar{\tau}, \exists$ PT rooted with $(n_s \times n_p)$, s.t. $\bar{b_\tau}$ and PT meet the battery-PT conditions in Section III-A.

If our battery packs meet this reliability condition when the number of faulty cells is N_f, the packs can also meet this condition when the number of faulty cells is less than N_f. Thus, the reliability is $P(x \leq N_f) = F(N_f)$, where $P(x \leq N_f)$ is the probability that the number of faulty cells is no more than N_f. $F(N_f)$ is the cumulative distribution of the number of faulty cells.

If the reliability $F(N_f)$ is higher than the required value, *N* packs containing *b* cells per pack can guarantee the required reliability during the lifetime *T*. Otherwise, we need to adjust *N* and *b*. Under the constraint that the pair of *N* and *b* values should meet the required reliability, our goal is to minimize the total cost C (i.e., the cost of cells, sensors and controllers):
$$C = N \times b \times c_c + N \times (c_f + c_v) \tag{3}$$
where

- c_c is the cost of a single cell, and $N \times b \times c_c$ is the total cell cost.
- c_f is the fixed pack cost, independent of the number of cells in a pack.
- c_v is the variable pack cost, dependent on the number of cells in a pack.

IV. Solution Algorithms

A. Construction of PT

We need to arrange the connection of battery packs to meet the load requirement. From the perspective of PT that represents connections of battery packs, to meet the load requirement $(n_s \times n_p)$, we try all possible PTs rooted at $(n_s \times n_p)$ until we find a valid PT that represents valid connections. To try as many PTs as possible, we can recursively divide leaf nodes in PTs into series or parallel groups. To divide nodes in PT into series groups, the parent node and its children must meet Eq. (1). Lines 10–16 in Algorithm 1 are used to divide nodes in series. The function partition(n) is the number of all possible integer partitions of n. To divide nodes in parallel, the parent node and its children must satisfy Eq. (2). Lines 17–23 in Algorithm 1 are used to divide nodes in parallel.

However, the construction of PT is computationally expensive. The total number of partitions for an integer *n* is approximately $\frac{e^{\pi\sqrt{\frac{2n}{3}}}}{4n\sqrt{3}}$ [2]. For each node in PT, its number of partitions is $\frac{e^{\pi\sqrt{\frac{2n_s}{3}}}}{4n_s\sqrt{3}} + \frac{e^{\pi\sqrt{\frac{2n_p}{3}}}}{4n_p\sqrt{3}}$, since nodes can be divided in parallel or series. The recursive division of nodes results in the complexity being the product of the numbers of partitions. To reduce this complexity, we propose *Fast Failure Recovery* (FFR) and *Fast Power Reallocation* (FPR), both of which also reduce the number of switch operations.

Algorithm 2 can be used to check the validity of PTs. To check a PT's validity, we first sort all leaf nodes in descending order of the number of cells required. To meet the power requirement at a leaf node, we select the minimum battery pack that is not used and has no less cells than this leaf node requires. This process is applied to each leaf node sequentially in descending order of the number of cells required.

We can use a battery-leaf pair (B, L) to represent that battery *B* is used to power the leaf node *L*. Algorithm 2 is "optimal" in the sense that if there exists a connection of batteries to provide the required powers (specified by the PT), Algorithm 2 can always find such a connection.

Algorithm 1: Generate PT

Input: $(N_s \times N_p)$, the load requirement
Input: $BatteryArray$, an array of battery packs
Output: PT, a valid PT to meet the load requirement

$Q \leftarrow Queue(\varnothing)$
$PT \leftarrow InitTree(root \leftarrow (N_s \times N_p))$
$Q.Enqueue(PT)$
while *Q not empty* **do**
  $PT \leftarrow Dequeue(Q)$
  if *CheckValid(PT,BatteryArray).Validity=True* **then**
    return PT
  end
  for $\forall Node \in$ *PT's leaf nodes* **do**
    for $\forall\{n_{s1}, n_{s2}, \cdots, n_{sm}\} \in$ *Partition*$(Node.n_s)$ **do**
      $NewPT \leftarrow PT$
      for $i \leftarrow 1 : m$ **do**
        $NewPT.Node.addChild(n_{si}, n_p)$
      end
      $Q.Enqueue(NewPT)$
    end
    for $\{n_{p1}, n_{p2}, \cdots, n_{pm}\} \in$ *Partition*$(Node.n_p)$ **do**
      $NewPT \leftarrow PT$
      for $i \leftarrow 1 : m$ **do**
        $NewPT.Node.addChild(n_s, n_{pi})$
      end
      $Q.Enqueue(NewPT)$
    end
  end
end

Algorithm 2: Check the validity of a PT

Input: $PowerTree$, the PT under test
Input: $BatteryArray$, an array of battery packs
Output: $Pair$, battery-leaf pairs.

$Used \leftarrow \varnothing$
$Pair \leftarrow \varnothing$
$LeafArray \leftarrow$ leaf nodes of PT
Sort $LeafArray$ in descending order
for $\forall Node \in LeafArray$ **do**
  $ValidBattery \leftarrow \{\forall Battery > Node \text{ and } Battery \notin Used\}$
  if $ValidBattery = \varnothing$ **then**
    return $\varnothing$
  end
  else
    $min \leftarrow Min(ValidBattery)$
    $Used.add(min)$
    $Pair.add(Node, min)$
  end
end
return Pair

Theorem 1. *Algorithm 2 is optimal in that given PT and a set of battery packs, if there exists a set of battery-leaf pairs meeting the battery-PT conditions in Section III-A, Algorithm 2 can always find a set of pairs.*

Proof: Equivalently, we can prove that if Algorithm 2 cannot find a set of pairs, there is no solution for this set of battery packs to meet the power requirement. Given a set of *N* battery packs, $\{b_1, b_2, \cdots, b_N\}$, b_i is the number of cells in the i-th battery pack. Arrange leaf nodes of the PT in descending order:

$$(n_{s1} \times n_{p1}) \geq (n_{s2} \times n_{p2}) \geq \cdots \geq (n_{sM} \times n_{pM}).$$

We sequentially pair a leaf node with a battery pack. A battery pack can support a leaf node's power if the number of cells in the pack is no less than the number of required cells in that node. If the k-th leaf node cannot be supported by any available battery packs, there are only two cases to consider:

1) If $\forall b_i < n_{sk} \times n_{pk}$, no solution exists.
2) Suppose $\{b_i | b_i \geq n_{sk} \times n_{pk}\}$ are all used to support other leaf nodes, and b_i is used to support node $j, (j < k)$. If b_i is to cover node k, we need to find an available battery pack for node j. However, because $\{b_i | b_i \geq n_{sk} \times n_{pk}\}$ are unavailable and $n_{sk} \times n_{pk} \leq n_{sj} \times n_{pj}$, $\{b_i | b_i \geq n_{sj} \times n_{pj}\}$ are also unavailable. Thus, there does not exist any solution.

That is, if we cannot obtain a solution using Algorithm 2, there is no solution at all. Thus, Theorem 1 follows. ■

B. Fast Failure Recovery

The core idea of FFR is to limit the impact of faulty cells, and hence reduce the number of packs to be reconfigured.

In Algorithm 3, we try to reconfigure from the faulty pack's inside to its neighbors, and finally, to the whole battery system. In a PT, we first try the faulty leaf node, and then the last ancestor, then second to the last ancestor, and so on, eventually the root of PT. To reconfigure a node in the PT, we generate a new PT rooted at the same power as that of the reconfiguring node. Based on this new PT, we can connect battery cells, which are idle/unused or have already been used in the leaf nodes, to meet the power requirement. In Algorithm 3, "GenPt" is the function to use Algorithm 1.

C. Fast Power Reallocation

1) Basic Algorithm: We assume that, when the required power increases, both voltage and current must increase. If the circuit to provide low power is a sub-circuit of the circuit that provides high power, we can easily switch to provide these two power-levels without modifying the circuit topology. Thus, to make quicker adaptation to dynamic loads, we try to organize smaller circuits to be sub-circuits of a larger one. In general, we can extend a smaller circuit to support higher power in two steps. Assuming $n_{sm} < n_{sM}, n_{pn} < n_{pN}$, we can change the battery's output power from $(n_{sm} \times n_{pn})$ to $(n_{sM} \times n_{pN})$ by:

1) connect $(n_{sm} \times n_{pn})$ with $((n_{sM} - n_{sm}) \times n_{pn})$ in series to get $(n_{sM} \times n_{pn})$;
2) connect $(n_{sM} \times n_{pn})$ with $(n_{sM} \times (n_{pN} - n_{pn}))$ in parallel to get $(n_{sM} \times n_{pN})$.

Algorithm 3: Adjust PT with faulty cells

Input: PT, the original PT
Input: $FaultyLeaf$, the faulty leaf node
Output: $NewPT$, the new PT

$Current \leftarrow FaultyLeaf$
while $Current \neq \varnothing$ **do**
  $BatteryArray$ is the array of packs that cover the leaf nodes of $Current$ as well as idle packs.
  $(n_s \times n_p) \leftarrow Current$
  $NewPT \leftarrow GenPT((n_s \times n_p), BatteryArray)$
  if $NewPT \neq \varnothing$ **then**
    Merge NewPT into old PT.
    Return PT
  end
  else
    $Current \leftarrow Current.Parent$
  end
end

When we need the higher power $(n_{sM} \times n_{pN})$, batteries are connected based on the entire PT; when we need the lower power $(n_{sm} \times n_{pn})$, we only use the node $(n_{sm} \times n_{pn})$.

If more than two power levels are required, we sort the power levels in ascending order and apply the above method to every two adjacent power-levels.

2) Dealing with Uncertain Power in FPR: Power is not always known *a priori* and hence, we also have to deal with unexpected power. In this case, we have to consider two transitions:

1) For transiting to an unknown power-level, our topology is not optimized for this unknown power. Thus, we have to reconfigure the entire circuit to provide this unexpected power. We can only optimize the circuit for expected power-levels.
2) For transitioning from an unknown power-level to a known one, our topology for the unknown power-levels may be unsuitable for the known power-levels (e.g., too small). Thus, we have to reconfigure the entire circuit to meet the known power requirements. We have two reconfiguration choices:
   a) **Aggressive approach** reconfigures the topology to meet the maximum known power-level, so that we may not have to do more reconfiguration when loads change to lower known power-levels.
   b) **Lazy approach** only reconfigures the topology to meet the current known power-level and optimizes this topology according to the power-levels that are not higher than the current known power-level. So, we do not have to do more reconfiguration when loads change to lower known power-levels, but we have to extend reconfiguration when loads increase to higher power-levels.

Theorem 2. *The lazy approach requires no more reconfiguration effort than the aggressive one.*

Proof: Power levels are measured by the number of cells in use. Suppose $n_1, n_2, \cdots, n_N$ are the sequence of known power-levels between two unknown power-levels. $n_0 = 0$. n_{max} is the maximum known power-level. An extension from n_i to n_j can be represented as $Extend(n_i, n_j)$. If $n_i < n_j < n_k$, $Extend(n_i, n_k) = Extend(n_i, n_j) + Extend(n_j, n_k)$

- The aggressive approach reconfigures the entire circuit once, and directly uses its sub-circuits to support lower power-levels. So, its reconfiguration effort is $Extend(0, n_{max}) + 0 \times N$.
- The lazy approach reconfigures the circuit to meet the current power-level, and gradually extends it to meet higher power-levels. We extract a subsequence n_{k1}, n_{k2}, n_{km} from $n_1, n_2, \cdots, n_N$. Each power-level in this subsequence is higher than all the power-levels before this in the sequence. Thus, each element between n_{ki} and $n_{k(i+1)}$ is not higher than n_{ki} so that we can directly use a sub-circuit of the circuit for n_{ki} without reconfiguring the whole circuit. The lazy approach's reconfiguration effort is, therefore,

$$\begin{aligned} & \sum_{i=1}^{m} Extend(n_{k(i-1)}, n_{ki}) + 0 \times (N - m) \\ = \quad & Extend(0, n_{km}). \end{aligned}$$

Since $n_{km} \leq n_{max}$, the lazy approach's reconfiguration effort is not larger than the aggressive one's. We can apply this calculation to all sequences between two unknown power-levels so that the lazy approach's total reconfiguration effort is also no more than than the aggressive one's. ■

3) Estimation of Reconfiguration Effort: Suppose power-levels are independent of each other. We can then estimate the reconfiguration effort using Markov chains. States in the Markov chains represent the highest known power-level that the circuit can afford and be optimized. Assume there are *n*+1 states: the first *n* states are for known power-levels (in ascending order) and group all unknown power-levels into the last state, *n*+1.

- State transition matrix

$$P^{(n+1)\times(n+1)} = \begin{bmatrix} P_1^{n\times n} & P_2^{n\times 1} \\ P_3^{1\times n} & P_4^{1\times 1} \end{bmatrix}$$

  $P_1^{n\times n}$ is the transition matrix, each element of which represents the probability of transiting from one known power-level to another known power-level. It is an upper triangular matrix, because the circuit for low power can be extended to support high power, but the circuit for high power cannot be reduced to only support low power. $P_2^{n\times 1}$, $P_3^{1\times n}$ and $P_4^{1\times 1}$ represent, respectively, the probabilities of transiting from (i) known power-levels to unknown ones, (ii) unknown power-levels to known ones, and (iii) unknown power-levels to unknown ones.
- The reconfiguration effort of transitions is

$$R^{(n+1)\times(n+1)} = \begin{bmatrix} R_1^{n\times n} & R_2^{n\times 1} \\ R_3^{1\times n} & R_4^{1\times 1} \end{bmatrix}$$

  where $R_1^{n\times n}$ represents the reconfiguration effort of transiting from a known power-level to another known one. It is an upper triangular matrix, because the circuit for high power does not incur any additional reconfiguration effort

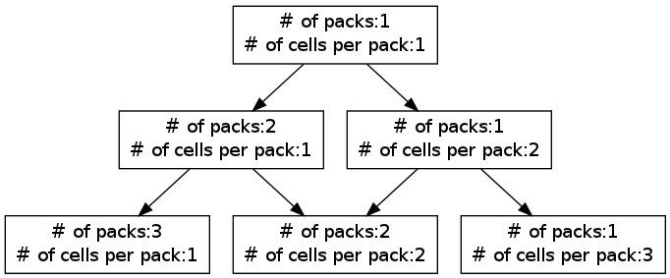

Fig. 4. An example pack-sizing graph

to support low power. $R_2^{n\times 1}$, $R_3^{1\times n}$, and $R_4^{1\times 1}$ represent, respectively, the reconfiguration efforts of transiting from (i) a known power-level to an unknown one, (ii) an unknown power-level to a known one, and (iii) an unknown power-level to another unknown one.

- The stationary distribution $\bar{\pi}$:

$$\lim_{m\to\infty} \bar{\pi}(P^{(n+1)\times(n+1)})^m = \bar{\pi} = [\pi_1, \pi_2, \cdots, \pi_{n+1}]$$

where $P^{(n+1)\times(n+1)}$ is the power transition matrix.

With these components, we can estimate the reconfiguration effort of FPR as:

$$E(R) = \sum_i \sum_j \pi_i p_{ij} r_{ij}$$

where π_i is the stationary probability of state *i*, p_{ij} is the probability of transiting from state *i* to state *j*, and r_{ij} is the (reconfiguration) effort of transiting from state *i* to *j*.

D. Pack-Sizing

Packs are usually homogeneous for ease of manufacturing, containing an identical number of cells. If there are not enough battery cells to meet the requirement, we add more packs to the system or more cells in each pack. If we treat (the number of packs, the number of cells in each pack) as a state, the transition from one state to another can be achieved by adding packs/cells. For example, in Fig.4, we begin with state (1 pack, 1 cell). If there is not enough power, we can transit to state (2 packs, 1 cell) by adding 1 pack, or to state (1 pack, 2 cells) by adding 1 cell per pack. If the power is still not enough, we can transit to state (3 packs, 1 cell), (2 packs, 2 cells) or (1 pack, 3 cells) by adding more packs/cells.

We can formally define the Pack-Sizing Graph (PSG) as follows.

- It is a directed graph.
- Each node in this graph represents a state (the number of packs, the number of cells per pack).
- Each node (n_p, n_c) has two edges pointing to nodes (n_p+1, n_c) and (n_p, n_c+1).
- There is only one node with 0 indegree. This node is the starting node (1, 1).

We can search the PSG for the best state to reduce the cost in Eq. (3). This is computationally expensive since we have to enumerate all pack-sizes and the numbers of packs. To reduce this complexity, we can design a heuristic by using a relaxed problem with fewer restrictions on actions to take. We relax the constraint that packs have to be homogeneous. Thus, we need not add one more cell in every pack or add a pack of the same size to meet the power requirement; instead, we only have to add packs of a single cell each. Then, the cost of a state is $f(s) = g(s) + h(s)$ where s is a state in PSG, $g(s)$ is the actual cost calculated using Eq. (3), and $h(s)$ is the heuristic function. It is the number of extra single-cell packs necessary to meet the power requirement. This cost function is both admissible and consistent, and hence, we can use the A* algorithm [16] to find an optimal solution.

V. Evaluation

We first describe our method for evaluating the proposed reconfiguration and pack-sizing, and then present the evaluation results in comparison with the simple scheme in [10] that only configures the battery-cell connectivity without optimization.

A. Evaluation Method

We simulate both the simple and the PT-based optimization schemes. Battery utilization—the ratio of the number cells in use to the total number of cells—is used to compare the battery connections of the simple scheme and those optimized via PT. We also evaluate the effectiveness of failure recovery and power reallocation using the reconfiguration effort, measured in number of battery packs to be reconfigured, which is proportional to the number of switch operations. The proposed FFR (Fast Failure Recovery) and FPR (Fast Power Relocation) are evaluated under the assumption that the required power and faulty cells in a system are uniformly distributed. We compute the reconfiguration effort upon failure of a single cell and also upon change of the required power. Moreover, the pack-sizing is assessed with the total cost of a battery system that consists of cell cost, fixed and variable pack costs. We set the required power and reliability for a given period of operation, and then compare the costs of the optimized and the other pack-sizes.

B. Battery Utilization

Since the simple reconfiguring scheme restricts the configuration of batteries (e.g., the number of cells in use in each pack and connections of packs), it cannot fully utilize cells in a pack, and thus, often cannot meet the power requirements even if there are enough cells in the pack. In contrast, the PT-based optimization provides more flexible intra- and inter-pack connections so that the battery system can provide the required power while the simple scheme cannot. Fig. 5 plots the results of comparing the battery utilization of the simple and PT schemes.

In Fig. 5, the utilization of both the simple and PT schemes decreases as the pack-size increases, since a larger pack-size bypasses more healthy cells. The utilization of the simple scheme is lower than that of the PT scheme because the PT scheme attempts to exploit more possible connections, unlike the simple scheme that considers only one type of connection. Specifically, in Fig. 5, the utilization of PT scheme remains high when the pack-size is not greater than 11, while that of the simple scheme decreases monotonically as the pack-size increases.

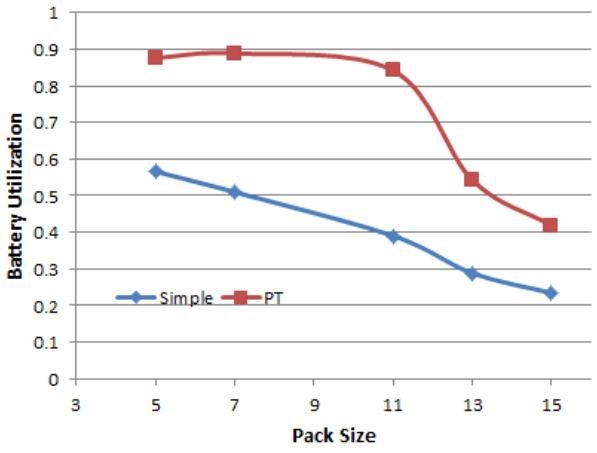

Fig. 5. Utilization of simple and PT schemes

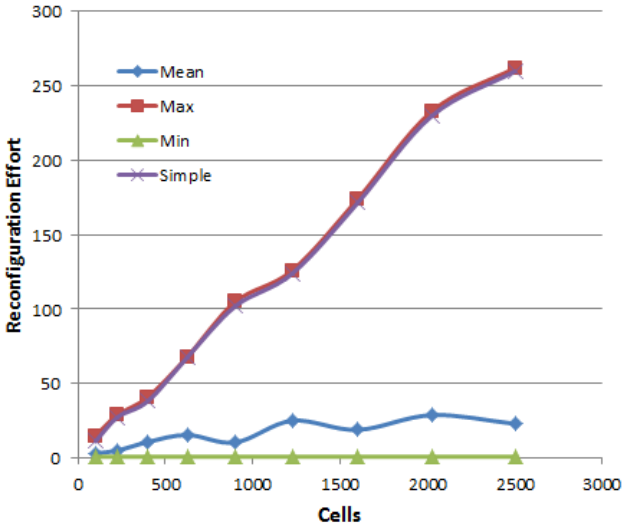

Fig. 6. Reconfiguration efforts of simple and FFR schemes

C. Fast Failure Recovery

While the simple scheme reconfigures the entire battery system upon failure of a cell, FFR effectively limits the impact of failed packs to a small area, thus reducing the reconfiguration effort. Clearly, a larger reconfiguration area requires more reconfiguration effort and time. Fig. 6 plots the results of comparing the reconfiguration efforts of the simple and FFR schemes, where "Mean," "Min," and "Max" represent the mean, minimum and maximum reconfiguration efforts of the PT scheme.

The reconfiguration effort of the simple scheme increases linearly with the power requirement since it has to reconfigure the entire battery system even upon failure of a single cell, and the number of packs in the battery system is proportional to the power requirement. By contrast, FFR's reconfiguration effort nearly remains constant irrespective of the increase of power requirement. The reason for this is that FFR usually reconfigures cell connectivity only in a small area around the failed packs (e.g., inside or in the vicinity of the failed pack) and keeps the other parts intact. The number of packs in this limited reconfiguration area nearly remains constant regardless of the size of the battery system, and the reconfiguration effort is nearly constant even if the load increases. This characteristic makes FFR *scalable*. In Fig. 6, in the worst case, FFR has to reconfigure all packs, so that the maximum reconfiguration effort is close to the total number of packs. However, in most cases, FFR only reconfigures a small part of the battery system so that the mean of reconfiguration effort is much lower than that in the worst case. Moreover, the minimum reconfiguration effort is always 1 because we can reconfigure inside a pack if the pack has enough backup cells. There is a wavy pattern on the mean line, resulting from the randomness of cell failures. In the experiment, we randomly choose a cell to fail and repeat the process many times, but the number of cells is so large that the randomness may become obvious. In addition, the amplitude of the wave is so negligible that one may treat the wave as a line.

D. Fast Power Reallocation

Based on prior knowledge of power patterns, FPR optimizes the topology of a battery system, reducing the number of packs to be reconfigured when the load changes, while the simple scheme simply reconfigures the entire battery system to meet the various power requirements. Obviously, the better the battery packs are organized, the less the reconfiguration effort is required when the load changes. Fig. 7 plots the reconfiguration efforts when the load changes, where "Lazy" and "Aggressive" represent the lazy and aggressive strategies of our PT scheme, and "Estimate" is the estimated reconfiguration of PT scheme using a Markov chain.

The reconfiguration effort of the simple scheme nearly remains constant irrespective of prior knowledge of power patterns, because the simple scheme does not exploit this prior knowledge to improve its efficiency. By contrast, prior knowledge about power patterns, if available, is exploited to reduce the reconfiguration effort of lazy FPR. Without any prior knowledge of power patterns, lazy FPR works in the same way as the simple scheme. In Fig. 7, the portion of unknown power represents the knowledge about power patterns. The simple scheme's reconfiguration effort remains high irrespective of the knowledge of power pattern. However, when the unknown portion is 0% (i.e., perfect knowledge), lazy FPR optimizes perfectly the battery system topology, making only a very small reconfiguration effort. When the unknown portion is 100% (i.e., no knowledge), lazy FPR works as poorly as the simple scheme and any other schemes. The last point of four lines coincide in Fig. 7 . The aggressive FPR performs well when the unknown portion is small, but worse than the simple scheme when the unknown portion is large because it takes time to reconfigure the circuit to support the largest power requirement, which does not always appears. Our estimation of lazy FPR with Markov chains is close to the actual lazy FPR. We also found that FPR only requires less than 0.2% more cells than the simple scheme, making it an efficient and low-cost solution.

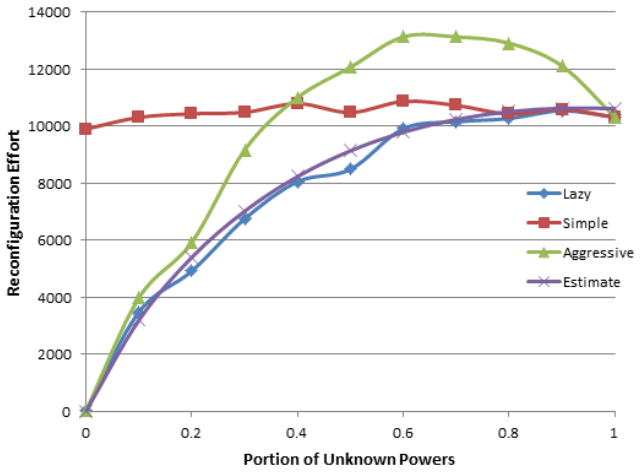

Fig. 7. Reconfiguration effort of the simple scheme and the fast power reallocation

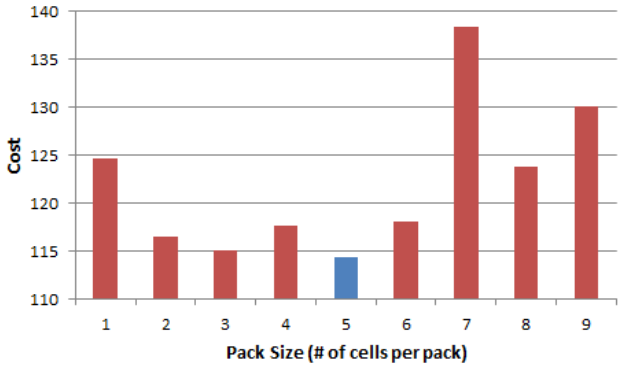

Fig. 8. Costs of different pack-sizes

E. Pack-Sizing

Pack-sizing covers the total number of cells and the number of cells per pack. One must determine the best pack size to minimize the total cost by balancing the various sources of cost, including cell cost, fixed and variable costs in a pack. The cost is modeled in Eq. (3) and its coefficients are calculated according to the costs of components. Since it is intractable to search all possible pack-sizes in order to find the optimal size, we use a heuristic algorithm that selects pack-sizes based on the sum of current cost and expected additional cost. Fig. 8 shows the costs of different pack-sizes under the uniformly-distributed failure rate of 3%. The medium pack-size is attractive, because a large pack-size incurs a high variable pack cost, while a small pack-size requires many packs and thus incurs a high fixed pack cost. In Fig. 8, the pack-size of 5 has the minimum cost while the pack-sizes of 1, 7, 8, and 9 have the highest cost, because the pack-size of 1 results in too many packs, thus too many controllers, while the pack-sizes of 7, 8, 9 result in very complex packs and high per-pack costs. In Fig. 8, the curve does not resemble a U-shaped bowl, as the pack size must be an integer value. Thus, the curve cannot be a smooth U-shaped bowl.

VI. Conclusion

It is important to recover from cell failures inexpensively and quickly since the cost and the recovery speed have become the main roadblock in the development of large-scale battery systems for such applications as EVs and uninterruptible power grids. We propose a new dynamic reconfiguration framework in which we develop algorithms for pack-sizing, failure recovery, and power reallocation to (1) improve battery utilization, (2) facilitate recovery from battery cell/pack failures and power reallocation, and (3) reduce the total cost of battery systems. We evaluated the performance of this framework in comparison with a popular existing scheme in terms of the utilization of available battery cells, the required cost and the reconfiguration effort. While enhancing the cell utilization, our algorithms for failure recovery and power reallocation are shown to dramatically reduce the reconfiguration effort that nearly remains constant irrespective of battery system size. Our optimal pack-sizing also reduces the cost and the size of a battery system. These solutions are expected to play key roles in (1) lowering the cost of large-scale battery systems, and (2) reducing the delays in operating a large number of switches necessary for reconfigurable (and hence fault-tolerant) battery systems.

Acknowledgment

The work reported in this paper was supported in part by the National Science Foundation under Grants CNS-0930813 and CNS-113820.

References

[1] Mahmoud Alahmad, Herb Hess, Mohammad Mojarradi, William West, and Jay Whitacre. Battery switch array system with application for jpl's rechargeable micro-scale batteries. *Journal of Power Sources*, 177(2):566 – 578, 2008.

[2] Sci-tech dictionary: asymptotic formula. http://www.answers.com/topic/asymptotic-formula, May 2010.

[3] R. C. Balch, A. Burke, and A. A. Frank. The affect of battery pack technology and size choices on hybrid electric vehicle performance and fuel economy. In *Proceeding 16th Annual Battery Conference*, 2001.

[4] L. Benini, A. Macii, E. Macii, and M. Poncino. Discharge current steering for battery lifetime optimization. In *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, pages 118 – 123, 2002.

[5] Hybrid Cars. First numbers on hybrid battery failure, May 2008.

[6] Song Ci, Jiucai Zhang, H. Sharif, and M. Alahmad. A novel design of adaptive reconfigurable multicell battery for power-aware embedded networked sensing systems. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 1043 –1047, Nov. 2007.

[7] Robert W. Erickson. Dc-dc power converters. Technical report, Department of Electrical and Computer Engineering, University of Colorado.

[8] William Flanagan. *Handbook of Transformer Design and Applications*. McGraw-Hill, 1993.

[9] J.W.K.K. Jayasundara and R. Munasinghe. Electric vehicle simulator to determine motor and battery specifications. In *Industrial and Information Systems (ICIIS), 2009 International Conference on*, pages 540 –545, Dec. 2009.

[10] Hahnsang Kim and Kang G. Shin. On dynamic reconfiguration of a large-scale battery system. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 87–96, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[11] Hahnsang Kim and Kang G. Shin. Dependable, efficient, scalable architecture for management of large-scale batteries. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, April 2010.

[12] Hybrid car battery problems. www.livestrong.com.

[13] Maxim. Source resistance: The efficiency killer in dc-dc converter circuits. http://www.maxim-ic.com/appnotes.cfm/an pk/3166, 2004.

[14] IDTechEx Research. Electric vehicles 2010-2020 new market research study, 2010.

[15] Pike Research. Electric vehicle consumer survey, 2010.
[16] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series, 2002.
[17] T. Stuart, F. Fang, X. Wang, C. Ashtiani, and A. Pesaran. A modular battery management system for hevs. In *Future Car Congress*, June 2002.
[18] Vinesh Sukumar, Mahmoud Alahmad, Kevin Buck, Herbert Hess, Harry Li, Dave Cox, Fadi Nessir Zghoul, Jeremy Jackson, Stephen Terry, Ben Blalock, M. M. Mojarradi, W. C. West, and J. F. Whitacre. Switch array system for thin film lithium microbatteries. *Journal of Power Sources*, 136(2):401 – 407, 2004. Selected papers presented at the International Power Sources Symposium.
[19] Techpulse360. The failure of lithium ion electric car batteries.
[20] The U.S. Department of Energy. *Primer On Lead-Acid Storage Batteries*, 1995.