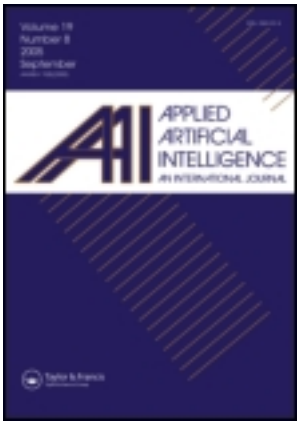

Applied Artificial Intelligence: An International Journal

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uaai20

Algorithms for Battery Utilization in Electric Vehicles

Ron Adany[a] & Tami Tamir[b]

[a] Computer Science Department, Bar-Ilan University, Ramat-Gan,, Israel

[b] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel

Published online: 14 Mar 2014.

To cite this article: Ron Adany & Tami Tamir (2014) Algorithms for Battery Utilization in Electric Vehicles, Applied Artificial Intelligence: An International Journal, 28:3, 272-291, DOI: 10.1080/08839514.2014.883906

To link to this article: http://dx.doi.org/10.1080/08839514.2014.883906

Applied Artificial Intelligence, 28:272–291, 2014

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2014.883906

Taylor & Francis
Taylor & Francis Group

ALGORITHMS FOR BATTERY UTILIZATION IN ELECTRIC VEHICLES

Ron Adany[1] and Tami Tamir[2]
[1] *Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel*
[2] *School of Computer Science, The Interdisciplinary Center, Herzliya, Israel*

□ *We consider the problem of utilizing a pack of* m *batteries serving* n *current demands in electric vehicles. When serving a demand, the current allocation might be split among the batteries in the pack. A battery's life depends on the discharge current used for supplying the requests. Any deviation from the optimal discharge-current is associated with a penalty. Thus, the problem is to serve an online sequence of current requests in a way that minimizes the total penalty associated with the service.*

We show that the offline problem, for which the sequence of current demands is known in advance, is strongly NP-hard and hard to approximate within an additive gap of $\Omega(\mathrm{m})$ *from the optimum. For the online problem, we present a competitive algorithm associated with the redundant penalty at most* m. *Finally, we provide a lower bound of 1.5 for the multiplicative competitive ratio of any online algorithm.*

INTRODUCTION

In the last few years, the idea of electrically powered cars has turned into reality. This old idea, from the early years of the automobile industry in the late 1890s, is now a real alternative to the gasoline powered vehicles (Kirsch 2000; Anderson and Anderson 2010). Electric vehicles (EVs) are currently considered the next generation of cars in the world of automobiles.

One of the most critical components of EV is the rechargeable battery (Affanni et al. 2005). The use of a battery as an energy source raises several issues such as limited driving ranges and high costs. Batteries are very expensive (Delucchi and Lipman 2001), thus, it is critically important to extend their lifetimes as much as possible. The battery's life is affected by the way the current is discharged (allocated), which is the focus of this study. It is important to distinguish between the battery's *capacity* and the battery's *life*. The battery's *capacity* is the amount of energy that can be stored in the

Address correspondence to Tami Tamir, School of Computer Science, The Interdisciplinary Center, P.O.Box 167, Herzliya, Israel. E-mail: tami@idc.ac.il

battery (and later discharged), that is, the amount of energy of one charge–discharge cycle. The battery's *life* is the total amount of energy that can be extracted from all charge–discharge cycles of the battery (MIT Electric Vehicle Team 2008). In other words, the battery's *life* is the total accumulated energy extracted from a battery during its life.

The EV battery is actually a pack of cells that are connected in serial and in parallel in order to provide the voltage and current required for propulsion. Cells are serially connected in order to provide the required voltage, and the series are connected in parallel to provide the required current. Our work considers the current allocation provided by the individual cell-series in the batteries. For simplicity, we denote each cell-series as a battery. Hence, in the EV pack there are several batteries connected in parallel.

The most dominant factor of an EV battery's life is the discharge current used. Each battery's chemistry is designed to be discharged in a specific current, whereas higher or lower currents have negative effects on it (Benini et al. 2001b; Pedram and Wu 1999; Doyle and Newman 1997). Moreover, as demonstrated in (Laman and Brandt, 1988), an optimal discharge current exists for each battery and depends on the specific chemistry. Based on these insights, we propose a penalty function that maps each discharge current to a numeric value reflecting its detrimental effect on the battery's life.

The common discharge method in EV is very simple and naïve: each current demand is supplied using all the batteries in the pack, and the load is equally divided. The rationale behind this method is simplicity of implementation; keep all batteries in the same condition (balanced) assuming that the lower the discharge current, the better. However, as described above, the behavior and performance of a real battery is more complex.

Our Results

Motivated by the possibility of extending the EV batteries' lives by a smart operation, we propose an advanced online current allocation algorithm. We also analyze the problem theoretically and provide results regarding its complexity status under common theoretical measures.

The performance of a current allocation algorithm is evaluated according to its gap from an optimal allocation, that is, the gap from an allocation that serves all current demands with the minimal possible penalty. We provide hardness results for both the offline problem, for which the current demands are known in advance, and for the online problem, for which the sequence of current demands is unknown in advance. In practice, demands should be served without a priori knowledge of forthcoming demands. Therefore, the main algorithm we suggest is for the online problem. Clearly, all hardness results we provide for the offline problem also capture the

online one. For the online problem, we also provide a lower bound on the competitive ratio of any deterministic algorithm.

A formal mathematical description of the problem is given in "Problem Definition." In "Offline Problem," we show that the problem of serving the requests with the minimal possible penalty is strongly NP-hard. Moreover, the problem is difficult to approximate within an additive gap of $\Omega(m)$ from the minimal possible penalty, where m is the number of batteries in the pack.

We then consider the online problem. In "An Almost Optimal Online Problem," we suggest a competitive algorithm in which the total penalty might be larger than the minimal possible one by at most an additive gap of m, independent of the number of requests in the sequence and of the initial capacity of the batteries. Our result is optimal, that is, it is associated with the minimal possible penalty for a significant prefix of the sequence—as long as the remaining capacity of the batteries is not below some threshold. Thus, our algorithm is optimal in the sense that it guarantees minimal damage to the batteries' lives when drivers charge their EV frequently enough and do not wait until the batteries are empty. Finally, in "A Lower Bound for the Multiplicative Competitive Ratio" we provide a lower bound of 1.5 for the competitive ratio of any online algorithm. Recall, an online algorithm has a competitive-ratio c if, for every possible instance, its objective value is within a factor of c from the optimum.

To the best of our knowledge, this work is the first to analyze the problem theoretically.

Our results can be applied to other resource allocation problems in which there are m servers (machines, batteries) that should serve n clients (jobs, current demands). Each client is associated with a request for some amount of resource. A request can be satisfied by several servers, in other words, the service of a request might split. There is a penalty associated with each allocation and the objective is to minimize the total penalty. This general problem appears in many domains. For example, in Human Resources (HR), it is reasonable to allocate tasks to workers such that each worker is assigned a specific workload, and any deviation from this workload causes some penalty, specifically, high workloads conflict with working time regulations, and low workloads might be unprofitable.

Related Work

Battery management and current allocation has been widely studied. Most of the previous work has been aimed at maximizing the battery's lifetime, that is, the time until the battery is empty and needs to be recharged.

Algorithms for extending batteries' lifetimes by various discharge allocation schemes were discussed in Benini and coauthors (2001a) and Rao,

Vrudhula, and Rakhmatov (2003). Several discharge schemes were proposed, including: (1) serial—discharging of a single battery each time until it is emptied and then discharging the next battery, (2) static switching—discharging of a single battery for a certain amount of time, (3) dynamic switching—discharging of each battery for a different amount of time depending on its physical state (e.g., remaining capacity), (4) parallel—discharging of all batteries together where the workload splits equally. In simulations and lab experiments the static and dynamic algorithms increased the lifetimes of the batteries significantly. In addition, the lifetime of batteries operated by both static and dynamic switching algorithms increased with the switching frequency.

The current demands of EVs are not known in advance and may be estimated using prediction methods based on driving profiles, driving stories, and demands' histories. In Benini and colleagues (2003), discharge methods among multiple batteries were discussed in which the current requests and their distributions over time are known. We do not deal with this type of problem because we propose an online algorithm.

PROBLEM DEFINITION

The system consists of m identical batteries, $B_1, B_2, \ldots, B_m$, all having the same initial capacity C. There are n current demands, that is, requests for current, given as an online sequence $d_1, d_2, \ldots, d_n$, where d_i is the current required to satisfy the ith request. The sequence, and in particular its length n, is unknown in advance, but it is guaranteed that the total capacity of the batteries can satisfy all requests, that is, $\sum_i d_i \leq mC$.

The ith request can be satisfied in various ways. Any allocation of d_i from the batteries is acceptable, and there are no constraints regarding the distribution of d_i among the batteries. However, there are "good" and "bad" allocations because an optimal discharge current exists.

Based on our understanding of the electrochemical properties of the individual cell-series (Benini et al. 2001b; Pedram and Wu 1999; Doyle and Newman 1997), we define a penalty function that reflects the damage to the battery's life from the discharge current. The actual penalty function depends on the specific chemistry of the battery. However, it is reasonable to assume the following basic structure. Let I_{OPT} be the optimal discharge current whose supply results in maximization of the battery life. The penalty function should fulfill the following conditions.

1. The penalty for no discharge is 0.
2. The penalty for supplying I_{OPT} is 0.
3. All other discharge currents have positive penalty values, according to their distance from I_{OPT}.

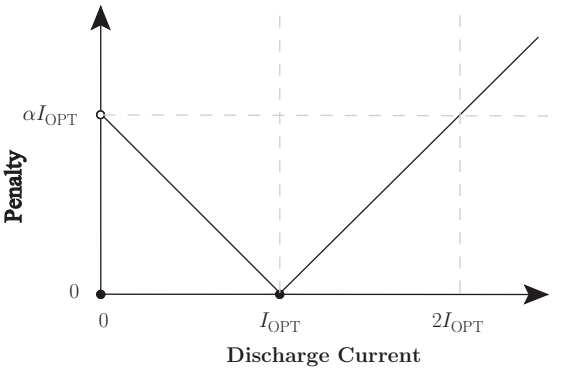

FIGURE 1 The penalty function.

We assume the following linear penalty function, described in Figure 1. Let x be the discharge current, then for some parameter $\alpha > 0$,

$$\mathrm{Penalty}(x) = \begin{cases} 0, & \text{if } x = 0 \text{ or } x = I_{\mathrm{OPT}} \\ \alpha |I_{\mathrm{OPT}} - x| & \text{otherwise} \end{cases} \tag{1}$$

By simple scaling of all demands and capacities, we assume throughout this article, without loss of generality, that $I_{\mathrm{OPT}} = 1$ and $\alpha = 1$. We denote by C_j^i the remaining capacity of battery B_j before current demand i. The goal is to determine the values x_j^i, where x_j^i is the discharge current (allocation) of battery B_j to supply the current demand i.

THE OFFLINE PROBLEM

In this section, we consider the case in which all requests are known in advance. Although this is not the case in practice, from the theoretical aspect it is interesting because any hardness result for this model also captures the online problem. The problem can be described as follows.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{Penalty}(x_j^i) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_j^i \le C && \forall j = 1 \ldots m. \\ & \sum_{j=1}^{m} x_j^i = d_i && \forall i = 1 \ldots n. \\ & 0 \le x_j^i, x_j^i \in R && \forall i = 1 \ldots n, \quad j = 1 \ldots m. \end{aligned}$$

Hardness of the Offline Problem

We first prove that the offline problem of serving all requests with the minimal possible penalty is strongly NP-hard. Next, we show that it is NP-hard to approximate the minimal possible penalty within an additive $\Omega(m)$ factor.

Theorem 3.1. *The problem of serving all requests with the minimal possible penalty is strongly NP-hard, even if all current demands are known in advance.*

Proof. We show a reduction from the *3-partition* problem. The input to 3-partition is a set of $n = 3m$ numbers, in other words, $S = \{d_1, \ldots, d_{3m}\}$, such that $\frac{1}{4} \leq d_i \leq \frac{1}{2}$ for all $1 \leq i \leq n$, and $\sum_{i=1}^{n} d_i = m$. The goal is to divide S into m subsets, $S_1, \ldots, S_m$, such that $\sum_{d_k \in S_j} d_k = 1$ for all $1 \leq j \leq m$. Each such subset must consists of exactly three numbers. The 3-partition problem is known to be strongly NP-hard (Garey and Johnson 1979).

Given S, we construct the following instance of our problem: m batteries, each with an initial capacity of $C = 1$, and $n = 3m$ current demands, where the ith request is for d_i.

Claim 3.2. *The set S has a 3-partition if and only if the minimum penalty for serving the requests is 2m.*

Proof. Given a 3-partition of S, let $S_1, \cdots, S_m$ be the required partition, in other words, $\sum_{d_k \in S_j} d_k = 1$ for all $1 \leq j \leq m$. We serve the requests as follows: for every subset S_j, assume that $S_j = \{d_{j_1}, d_{j_2}, d_{j_3}\}$, then the corresponding three requests are served from battery j. All requests are less than 1, that is, $d_i \leq 1$, thus the penalty from each is $1 - d_i$. Accordingly, the total penalty for serving requests from battery j is $(1 - d_{j_1}) + (1 - d_{j_2}) + (1 - d_{j_3})$, and the total penalty for serving all $3m$ requests, as induced by the m sets is $\sum_{j=1}^{m} [(1 - d_{j_1}) + (1 - d_{j_2}) + (1 - d_{j_3})] = 3m - \sum_{i=1}^{n} d_i = 3m - m = 2m$.

For the other direction, assume that the requests are served such that the total penalty is $2m$. Because all requests for current demand are less than 1, the penalty for serving request i is exactly $m_i - d_i$, where $m_i \geq 1$ is the number of batteries for which $x_j^i > 0$ (that allocate some current to request i). Therefore, the total penalty is $\sum_{i=1}^{n} (m_i - d_i) = \sum_{i=1}^{n} m_i - m$. In order for this sum to be $2m$, it must hold that $\sum_{i=1}^{n} m_i = 3m$. Because $n = 3m$ and m_i values are nonnegative integer numbers, $m_i = 1$ must be true for all i. In other words, each request is supplied by only a single battery. Also, the range of d_i, that is, $\frac{1}{4} \leq d_i \leq \frac{1}{2}$, implies that exactly three demands are served by each battery. Because the total current demand is exactly the total available energy, that is, $\sum_{i=1}^{n} d_i = \sum_{j=1}^{m} C$, it must be that for every $1 \leq j \leq m$, the requests that are served by the jth battery constitute a total

demand of exactly $C = 1$, and the mapping of requests to the batteries that serve them induces a 3-partition. ■

Our next hardness proof shows that the gap between the total penalty of any efficient algorithm and the penalty of an optimal algorithm is $\Omega(m)$. We use an idea suggested in a hardness proof for the problem of packing with item fragmentation (Shachnai, Tamir and Yehezkely 2008).

Theorem 3.3. *For any $m > 1$, a set of requests exists such that for some $P \geq 0$, the optimal serve of all the requests by m batteries constitute a total penalty P, whereas service of all the requests by m batteries with a total penalty less than $P + \lfloor \frac{m}{2} \rfloor$ is NP-hard.*

Proof. Recall that $m_i \geq 1$ denotes the number of batteries participating in the service of request i, in other words, $m_i = |\{j | x_i^j > 0\}|$.

The proof is based on defining a set of requests: an instance σ, with the following attributes:

(A_1) All the requests are small, that is, $d_i < 1$ for all i.
(A_2) In the optimal solution, every request for current demand d_i is provided by a single battery, or, $m_i = 1$ for all i.
(A_3) If $P \neq NP$, then in the allocation determined by any efficient algorithm $\sum_i m_i \geq n + \lfloor m/2 \rfloor$.

The penalty for any allocation of $x_i^j < 1$ is $1 - x_i^j$. Thus, the total penalty for serving request i is $m_i - d_i$. Given that σ fulfills attributes (A_1) and (A_2), we conclude that the minimal penalty achieved by an optimal allocation is $P_{\mathrm{OPT}}(\sigma) = \sum_{i=1}^{n}(m_i - d_i) = n - \sum_i d_i$. Also, by attribute ($A_3$), the total penalty of any efficient algorithm is at least $P_{\mathrm{ALG}}(\sigma) = \sum_i (m_i - d_i) \geq n + \lfloor m/2 \rfloor - \sum_i d_i$. Therefore, $P_{\mathrm{ALG}}(\sigma) - P_{\mathrm{OPT}}(\sigma) \geq \lfloor m/2 \rfloor$.

We now describe the construction of an instance σ that fulfills (A_1) through (A_3). For this, we assume that the batteries have different initial capacities and that m is even. Later we explain how these assumptions can be removed.

In order to achieve attribute (A_3), we use a reduction from the Partition problem. The input for Partition is a set $U = \{a_1, \ldots, a_h\}$ of positive integers with a total size of $2S$. The goal is to determine whether a subset $U' \subseteq U$ of a total size S exists.

For a given number of batteries m and a given instance of Partition, $U = \{a_1, \ldots, a_h\}$, construct an instance for the current allocation problem with m batteries and $n = h \cdot \frac{m}{2}$ current demands. The current demands consists of $k = \frac{m}{2}$ sets, $R_0, \ldots, R_{k-1}$ each comprising h requests. Let $M > (2S + 1)$ be an integer. The set R_0 comprises requests for current demands $a_1, a_2, \ldots, a_h$;

R_1 comprises requests for current demands $a_1M, a_2M, \ldots, a_hM$, and in general, R_ℓ comprises requests for current demands $a_1M^\ell, a_2M^\ell, \ldots, a_hM^\ell$. The battery capacities are $S, SM, SM^2, \ldots, SM^\ell, \ldots, SM^{k-1}$ and there are two batteries of each capacity. Note that the total required energy is exactly the total capacity of the batteries. Finally, let $I_{\mathrm{OPT}} = SM^k$. For simplicity of the description, the attributes are described assuming $I_{\mathrm{OPT}} = 1$; this is without loss of generality, because it is possible to scale all demands and capacities by a factor of $1/SM^k$.

If a partition of the items in U into two sets of size S exists, then a current allocation fulfilling (A_2) exists. Such an optimal allocation serves all the requests of R_ℓ by the two batteries with a capacity of SM^ℓ. Because the total demand of the requests in R_ℓ is exactly $2SM^\ell$ and a partition exists, R_ℓ can be partitioned into two sets of requests, each set with a total demand of SM^ℓ, and it is possible to serve all the requests such that every request is served by a single battery.

For the other direction, consider any service of the demands. Because the total required energy is exactly the total capacity of the batteries, for every battery j with SM^ℓ capacity, $\sum_i x_i^j = SM^\ell$ holds.

Claim 3.4. *If some battery, B_j, serves only full requests, that is, for all i either x_i^j is 0 or d_i, then a partition of U exists.*

Proof. Assume that for some $0 \le j \le k-1$, a battery B_j with capacity SM^j serves only full requests. First, note that no request from a set R_ℓ where $\ell > j$ is serviced by battery B_j. This is true because each such request is larger than SM^j (because $a_iM^{j+1} > SM^j$). Also, no request from a set R_ℓ where $\ell < j$ is serviced by battery B_j. This is true because the total demand of requests from lower sets, (i.e., $R_0, R_1, \ldots, R_{j-1}$), is $2S(1 + M + M^2 + \ldots + M^{j-1}) = 2S(M^j - 1)/(M-1)$, which is less than M^j for all $M > 2S + 1$. Therefore, any service of requests from previous sets would prevent battery B_j from supplying all its capacity. This implies that no combination of demands from sets R_ℓ where $\ell < j$, can be supplied in any allocation in which the total capacity of battery B_j is used. It follows that battery B_j services only those requests from a single set, and by scaling by $_{M^j}$, this service induces a partition of the original instance. ■

We conclude that if $P \neq NP$, then in any allocation provided by an efficient algorithm, every battery will serve at least one partial request. Thus, there are at least $m/2$ fractions of requests and attribute (A_3) holds.

In order to extend the proof to instances with identical batteries, we set the capacities of all $2k$ batteries to SM^k and add two *filler requests* of size $S(M^k - M^\ell)$ to each set of demands R_ℓ. Nonetheless, the total required energy is still exactly the total capacity of the batteries. The two smallest

filler requests constitute a total size of $2S(M^k - M^{k-1})$, which is larger than SM^k for any $M > 2$. Therefore, in any allocation of full requests, each battery serves, at most, one filler request. Furthermore, the total demand of a nonfiller request is too small to fully utilize any battery, therefore, any fully utilized battery must serve exactly one filler request. Assume that some battery serves only full requests and let $S(M^k - M^z)$ be the demand of the filler request served by the battery. The rest of the energy is allocated to requests of a total demand SM^z and the proof for the different capacity batteries can be applied.

If m is odd, we can add a single request for a current demand of SM^k, and setting $k = \lfloor m/2 \rfloor$, we attain an instance to which the above reduction can be applied. ■

AN ALMOST OPTIMAL ONLINE ALGORITHM

In this section, we describe an online current allocation scheme that is guaranteed to serve all requests in an instance σ with a penalty of at most $P_{\mathrm{OPT}}(\sigma) + m$. By Theorem 3.3, a lower bound of $P_{\mathrm{OPT}}(\sigma) + \Omega(m)$ also applies to the offline case. Therefore, our algorithm is optimal in the sense that only the constant factor in the additive term might be reduced. Another property of our algorithm is that for a significant prefix of the requests, an optimal allocation is guaranteed. This property implies that our algorithm guarantees minimal damage to the lifetimes of batteries of drivers who charge their EVs frequently enough and do not wait until the batteries are empty. The last demands (i.e., the demands when the batteries are almost empty) turn out to be the most challenging, because the amount of energy left in each battery may be negligable, and an optimal discharge current might not be possible.

The proposed solution distinguishes between two possible statuses of the batteries. As long as the status is *AboveTheLine* (to be defined later), the service of all requests is associated with the minimal possible penalty. Once the status is not *AboveTheLine*, our allocation might involve a redundant penalty. However, it is possible to bound this redundant penalty by m.

Algorithm 1 describes the current allocation scheme for a single request. Before serving request i, we sort the batteries in nonincreasing order of remaining capacity: $C_1^i \geq C_2^i \ldots \geq C_m^i$. Ties are broken arbitrarily. For simplicity, because our analysis refers to a single request, d_i, at a time, we drop the index i whenever it is clear from the context. Specifically, d is the request, C_j is the remaining capacity of the battery B_j with the jth highest remaining capacity before the request, and x_j is the current allocated to the request by battery B_j.

The algorithm considers separately different types of requests: (1) small requests—of size $d \leq 1$, (2) large requests—of size $d \geq m$, and (3) midsized

requests—of size $1 < d < m$. We state that a midsized request for current d has a *low fraction* if $d - \lfloor d \rfloor < 0.5$. Otherwise, the request has a *high fraction*. Note that a request has a low fraction if and only if $d - \lfloor d \rfloor < \lceil d \rceil - d$. For example, $d = 2.3$ has a low fraction, whereas $d = 2.8$ has a high fraction. For each type of request, the algorithm first tries to fulfill the request with the minimal possible penalty. If such an allocation is impossible, an allocation associated with some redundant penalty is determined.

In some cases, the algorithm calls the procedure *GreedyAllocation*, given in Algorithm 2. In this procedure, the allocation is done greedily from the battery with the lowest remaining capacity, moving to the next lowest battery when the first battery is emptied. Note that the order of the batteries is determined when Algorithm 1 is called and is not modified as a result of allocations done before *GreedyAllocation* is called.

In some cases, the algorithm allocates current *in a balancing way* among a selected set of batteries. That is, the current is allocated from the batteries in the set that have the highest remaining capacity, such that $x_1 \geq x_2 \geq \ldots$, while trying to balance the remaining capacities in the set after the allocation. This can be done by first allocating from battery B_1, until $C_1 = C_2$, then allocating evenly from both batteries B_1, B_2, and so on, until the whole demand is allocated.

The batteries' status before request i is defined as follows. Recall that the batteries are sorted in nonincreasing order of remaining capacity (i.e., $C_1^i \geq C_2^i \ldots \geq C_m^i$).

Definition 4.1. *If $C_1^i \geq 1.5$ and $C_j^i \geq 1$ for every $j > 1$, then the batteries' status before request i is AboveTheLine.*

The analysis of the algorithm is based on several observations. First, we show that as long as the batteries' status is AboveTheLine, then the service of all requests is associated with the minimal possible penalty. Next, we bound the total possible penalty once the batteries' status is not AboveTheLine.

Lemma 4.1. *As long as the batteries' status is AboveTheLine, the service of all requests is associated with the minimal possible penalty.*

Proof. Consider a current demand, d, and use $P_{\mathrm{OPT}}(d)$ to denote the minimal possible penalty for supplying it.

If $d \leq 1$, then $P_{\mathrm{OPT}}(d) = 1 - d$, achieved by allocating all current from a single battery—as in our algorithm (line 4). If $d \geq m$, then $P_{\mathrm{OPT}}(d) = d - m$, achieved by allocating at least 1 from each of the m batteries. Such an allocation is done in our algorithm (lines 10–11). Because the status is *AboveTheLine* (i.e., $C_j^i \geq 1$ for every $j \geq 1$), the two above allocations are possible.

Algorithm 1. Current Allocation for a Single Request

Input: Current demand, d, battery capacities, $C_1 \geq \ldots \geq C_m$.

Let $\hat{m}$ be the number of batteries with a capacity of at least 1, i.e., $\hat{m} = max_j\, C_j \geq 1$.
if $d \leq 1$ **then** {small request}
  if $d \leq C_1$ **then**
    Allocate the demand from x_1, i.e., $x_1 = d$.
  else
    CALL: *GreedyAllocation*.
  end if
else if $d \geq m$ **then** {large request}
  if $\hat{m} = m$ **then**
    Determine initial allocation of $x_j = 1$ from batteries $j = 1, \ldots, m$.
    Allocate the remaining demand in a balancing way among batteries $j = 1, \ldots, m$.
  else {$\hat{m} < m$}
    Determine initial allocation of $x_j = 1$ from batteries $j = 1, \ldots, \hat{m}$.
    Allocate the remaining demand, i.e., $d - \hat{m}$, by *GreedyAllocation*.
  end if
else if $d - \lfloor d \rfloor < 0.5$ **then** {midsized request: low-fraction request}
  if $\hat{m} \geq \lfloor d \rfloor$ and $d \leq \sum_{j=1}^{\lfloor d \rfloor} C_j$ **then**
    Determine initial allocation of $x_j = 1$ from batteries $j = 1, \ldots, \lfloor d \rfloor$.
    Allocate the remaining demand, i.e., $d - \lfloor d \rfloor$, in a balancing way from batteries $j = 1, \ldots, \lfloor d \rfloor$.
  else
    if $\hat{m} \geq \lfloor d \rfloor$ and $d > \sum_{j=1}^{\lfloor d \rfloor} C_j$ **then**
      Determine allocation of $x_j = C_j$ for all batteries $j = 1, \ldots, \lfloor d \rfloor$.
      Allocate the remaining demand, i.e., $d - \sum_{j=1}^{\lfloor d \rfloor} C_j$, by *GreedyAllocation*.
    else {$\hat{m} < \lfloor d \rfloor$}
      Determine initial allocation of $x_j = 1$ from batteries $j = 1, \ldots, \hat{m}$.
      Allocate the remaining demand, i.e., $d - \hat{m}$, by *GreedyAllocation*.
    end if
  end if
else {midsized request: high-fraction request}
  if $\hat{m} \geq \lceil d \rceil$ **then**
    Determine initial allocation of $x_j = 0.5$ from batteries $j = 1, \ldots, \lceil d \rceil$.
    Allocate the remaining demand, i.e., $d - \lceil d \rceil /2$, in a balancing way from batteries $j = 1, \ldots, \lceil d \rceil$ limiting the allocation from any single battery by 1, i.e., $x_j \leq 1$ for all j.
  else
    Determine initial allocation of $x_j = 1$ from batteries $j = 1, \ldots, \hat{m}$.
    Allocate the remaining demand, i.e., $d - \hat{m}$, by *GreedyAllocation*.
  end if
end if
Update $C_j = C_j - x_j$ for all batteries $j = 1, \ldots, m$.

If $1 < d < m$, then the minimal possible penalty is the distance of d from the nearest integer, given by $P_{\mathrm{OPT}}(d) = min\{d - \lfloor d \rfloor\, ,\, \lceil d \rceil - d\}$. In both cases, this value is at most 0.5. If the request has a low fraction, then penalty $P_{\mathrm{OPT}}(d)$ can be achieved by allocating at least 1 from $\lfloor d \rfloor$ batteries. If the

Algorithm 2. GreedyAllocation

Input: Current demand, d, battery capacities, $C_1 \geq \ldots \geq C_m$, discharge currents, $x_1, \ldots, x_m$

1: $i = m$
2: **while** $d > 0$ **do**
3: Set $x' = \min\{C_i - x_i, d\}$
4: Update allocation, $x_i = x_i + x'$
5: Update remaining demand, $d = d - x'$
6: $i = i$–1
7: **end while**

request has a high fraction, then penalty $P_{\mathrm{OPT}}(d)$ can be achieved by allocating at least 0.5 and at most 1 from $\lceil d \rceil$ batteries. Such allocations are done in our algorithm (lines 18–19 and 31–32 respectively). Note that because the status is *AboveTheLine* (i.e., $C_1^i \geq 1.5$ and $C_j^i \geq 1$ for every $j > 1$), the above allocations are possible. Therefore, if the batteries' status before the allocation is *AboveTheLine*, then it is always possible to follow the suggested allocation, which is associated with the minimal possible penalty. ■

We conclude that the algorithm might cause a redundant penalty only when the batteries' status is not *AboveTheLine*. In order to bound the total redundant penalty, we first bound the gap between the remaining capacities of any pair of batteries.

Claim 4.2. *For any request i and two batteries,* j, j', $|C_j^i - C_{j'}^i| < 1.5$.

Proof. The proof is by induction on the number of requests serviced, that is, on the value of i. Initially, before the first request ($i = 1$), all capacities equal C and the claim is clearly valid. Assume that the claim holds before the allocation of the ith request, we show it is valid also afterward. Let S denote the set of batteries allocating current to the ith request, that is, $k \in S$ if and only if $x_k^i > 0$.

Consider first the case that the batteries' status is *AboveTheLine*, where the only relevant allocations are in lines 4, 10–11, 18–19 and 31–32. If both $j, j' \notin S$, then clearly the remaining capacities do not change and the gap between the remaining capacities remains the same. If both $j, j' \in S$, because the only relevant allocations are done in a balancing way, the gap between the remaining capacities can only decrease. If $j \in S$ and $j' \notin S$, then it must hold that $C_j^i \geq C_{j'}^i$ and $d_i < m$. In this situation the maximal allocation of each battery is less than 1.5, as can be easily verified. In addition, because the allocations are done from batteries with high capacities, the (absolute) gap between the remaining capacities can only decrease.

When the batteries' status is not *AboveTheLine*, either the maximal capacity of a battery is less than 1.5, which clearly implies that the maximal gap is less than 1.5, or the minimal capacity of a battery is less than 1. It is easy to verify that, in this case, either the maximal allocation of any single battery is 1.5 and the allocation is done from batteries with high capacities, or the algorithm allocates at least 1 from any battery with capacity at least 1 before allocating (greedily) current from the batteries with the low capacity. ■

We are now ready to bound the total redundant penalty after the batteries' status is not *AboveTheLine*.

Theorem 4.3. *The total penalty of the algorithm might be larger than the minimal possible penalty by at most m.*

Proof. Let $P_{\mathrm{OPT}}(d_i), P_{\mathrm{ALG}}(d_i)$ denote the minimal possible penalty and the penalty caused by the algorithm for servicing the ith request, respectively. Let $\Delta(i) = P_{\mathrm{ALG}}(d_i) - P_{\mathrm{OPT}}(d_i)$ denote the redundant penalty, which occurred in the service of request i, and e_i denote the number of batteries emptied as a result of servicing request i. Finally, let m' denote the number of batteries with a positive remaining capacity after all requests are served, in other words, $m = m' + \sum_i e_i$. We show that $\sum_i \Delta(i) \leq m$ by showing that $\sum_i \Delta(i) \leq \sum_i e_i + m'$. By Lemma 4.1, as long as the batteries' status is *AboveTheLine*, $\Delta(i) = 0$ holds. Therefore, we need to bound the redundant penalty after the batteries' status is not *AboveTheLine*. The batteries' status is not *AboveTheLine* if one of the two following conditions is valid (1) $C_1^i < 1.5$, (2) $C_1^i \geq 1.5$ and $C_m^i < 1$.

Our analysis is based on combining the following three claims, whose proof follows.

1. Whenever condition (1) is valid, $\Delta(i) \leq e_i$ (Claim 4.4).
2. Condition (2) might be valid only for a single request for which $\Delta(i) > 0$ and for this request, $\Delta(i) \leq e_i + 1$ (Claim 4.5).
3. Either $m' > 0$, or for the last request, $e_i > 0$ and $\Delta(i) \leq e_i - 1$ (Claim 4.6).

Claim 4.4. *If* $C_1^i < 1.5$ *and* $\Delta(i) > 0$, *then* $\Delta(i) \leq e_i$.

Proof. Except for one case (detailed as follows), redundant penalty is caused only when *GreedyAllocation* is called. Assume that *GreedyAllocation* allocates current from k_1 batteries for which $C_j^i \leq 1$, and k_2 batteries for which $C_j^i > 1$ (see Figure 2a). Denote by K_1, K_2 the corresponding sets of batteries; by X_1, X_2 the total current allocated by *GreedyAllocation* from K_1, K_2, respectively; and by p_1, p_2 the total penalty for K_1, K_2, respectively.

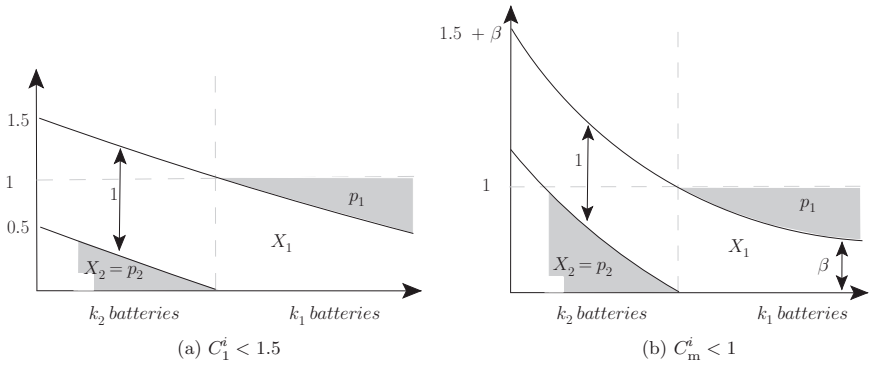

FIGURE 2 Cases when the batteries' status is not *AboveTheLine*.

Our analysis distinguishes between different possible values of d_i. If $d_i \leq 1$, then $P_{\mathrm{OPT}}(d_i) = 1 - d_i$. Note that $k_2 > 0$ is impossible in this case, because a small request will be serviced optimally if some battery has remaining capacity of at least 1. Thus, the algorithm allocates d_i greedily from K_1, of which at least $k_1 - 1$ batteries are emptied. Thus, $X_1 = d_i$ and $P_{\mathrm{ALG}}(d_i) = k_1 - d_i$. We find that $\Delta(i) = (k_1 - d_i) - (1 - d_i) = k_1 - 1$, or, $\Delta(i) \leq e_i$.

If $1 < d_i < m$ and d_i has a low fraction, let $w = d_i - \lfloor d_i \rfloor$. It holds that $P_{\mathrm{OPT}}(d_i) = w$. The algorithm might have a redundant penalty in two cases:

1. If $\hat{m} \geq \lfloor d_i \rfloor$ and $d_i > \sum_{j=1}^{\lfloor d_i \rfloor} C_j$ (line 21). In this case, $\lfloor d_i \rfloor$ batteries are emptied before *GreedyAllocation* is called (line 22) and the remaining demand of $g_i = d_i - \sum_{j=1}^{\lfloor d_i \rfloor} C_j$ is allocated greedily (line 23). The penalty for the $\lfloor d_i \rfloor$ batteries is $p_0 = \sum_{j=1}^{\lfloor d_i \rfloor} C_j - \lfloor d_i \rfloor$. Note that $P_{\mathrm{OPT}}(d_i) > p_0$ because $d_i > \sum_{j=1}^{\lfloor d_i \rfloor} C_j$ and $P_{\mathrm{OPT}}(d_i) = d_i - \lfloor d_i \rfloor$. If $k_2 = 0$, then the whole allocation is from K_1, of which at least $k_1 - 1$ batteries are emptied. It holds that $P_{\mathrm{ALG}}(d_i) = p_0 + p_1 = p_0 + k_1 - X_1 < p_0 + k_1$. Thus, $\Delta(i) \leq (p_0 + k_1) - (p_0) = k_1$. Because $d_i > 1$ and $\lfloor d_i \rfloor$ batteries are emptied before the greedy allocation, we have $e_i \geq k_1$, thus, $\Delta(i) \leq e_i$. If $k_2 = 1$, then because the remaining demand $g_i \leq 0.5$, the single battery in K_2 allocates at most 0.5, and has penalty at most 1. At least one battery is emptied before the greedy allocation and at least k_1 batteries are emptied by *GreedyAllocation*, thus $e_i > k_1$. We have $P_{\mathrm{ALG}}(d_i) = p_0 + p_1 + p_2 \leq p_0 + (k_1 - X_1) + 1 < p_0 + k_1 + 1$. Because $P_{\mathrm{OPT}}(d_i) \geq p_0$, we get $\Delta(i) \leq k_1 + 1 \leq e_i$. Note that $k_2 > 1$ is impossible in this case, because every K_2 battery has remaining capacity of at least 1 when *GreedyAllocation* is called and $g_i \leq 0.5$, thus, a single k_2 battery can supply the whole greedy request.
2. If $\hat{m} < \lfloor d_i \rfloor$, then the algorithm allocates $g_i = d_i - \hat{m}$ greedily, where g_i is the remaining current demand from line 26, and it holds that $g_i \geq$

$1 + w$. If $k_2 = 0$, then the whole allocation is from K_1, of which at least $k_1 - 1$ batteries are emptied. It holds that $P_{\mathrm{ALG}}(d_i) = k_1 - X_1 = k_1 - g_i$. We find that $\Delta(i) = (k_1 - g_i) - (w) < k_1 - 1$, that is, $\Delta(i) < e_i$ If $k_2 \geq 1$, then by the algorithm, an initial allocation of 1 is determined for each of the k_2 batteries before *GreedyAllocation* is called (line 25). Because $C_1^i < 1.5$, each such battery allocates in the greedy phase at most 0.5. Thus, the allocation is from k_1 batteries, that are all emptied, and k_2 batteries, each allocating in the greedy phase at most 0.5. Thus, $g_i = X_1 + X_2 \leq X_1 + 0.5k_2$, implying $X_1 \geq g_i - 0.5k_2$. Also, because all K_2 batteries allocate 1 before *GreedyAllocation* is called, we have $p_2 = X_2 < 0.5k_2$. Thus $P_{\mathrm{ALG}}(d_i) = p_1 + p_2 \leq (k_1 - g_i + 0.5k_2) + 0.5k_2$, and $\Delta(i) \leq (k_1 - g_i + k_2) - (w) \leq k_1 - 1 - w + k_2 - w < k_1 + k_2 - 1$, or, $\Delta(i) \leq e_i$.

If $1 < d_i < m$ and d_i has a high fraction, let $w = \lceil d_i \rceil - d_i$. It holds that $P_{\mathrm{OPT}}(d_i) = w$. If $k_2 = 0$, then the algorithm allocates $g_i = d_i - \hat{m}$ greedily from K_1, of which at least $k_1 - 1$ are emptied. Note that $g_i \geq 1 - w$. Thus, $P_{\mathrm{ALG}}(d_i) = k_1 - X_1 = k_1 - g_i$. We find that $\Delta(i) = (k_1 - g_i) - (w) < k_1 - 1$, that is, $\Delta(i) < e_i$. If $k_2 \geq 1$, then, as explained in the low-fraction case, $P_{\mathrm{ALG}}(d_i) \leq k_1 - g_i + 0.5k_2 + 0.5k_2$, thus, $\Delta(i) \leq (k_1 - g_i + k_2) - w \leq k_1 - 1 + w + k_2 - w = k_1 + k_2 - 1$, in other words, $\Delta(i) \leq e_i$.

If $d_i \geq m$, then, by definition of the penalty function, we have $p_1 = \sum_{j \in K_1}(1 - C_j^i) = k_1 - X_1 < k_1$. It must hold that $k_2 > 0$ because some battery must allocate more than 1 in order to fulfill a large request. By the algorithm, an initial allocation of 1 is determined for each of the k_2 batteries before *GreedyAllocation* is called (in line 13). Because $C_1^i < 1.5$, each such battery allocates in the greedy phase at most 0.5. The total penalty is, therefore, $P_{\mathrm{ALG}}(d_i) < k_1 + 0.5k_2$. All batteries participating in *GreedyAllocation*, except perhaps the last one, are emptied, thus, $e_i = k_1 + k_2 - 1$. If $k_2 \geq 2$, then $P_{\mathrm{ALG}}(d_i) < k_1 + 0.5k_2 \leq k_1 + k_2 - 1 \leq e_i$, and we are done.

If $k_2 = 1$, then it must be that no battery was emptied before request i is serviced, in other words, $C_m^i > 0$, as otherwise the remaining $m - 1$ (or fewer) batteries have total remaining capacity at most $m - 1 + 0.5 < m$ and cannot fulfill a large request. This implies that $\hat{m} = m - k_1$ and $m - k_1$ batteries allocates 1 to d_i before *GreedyAllocation* is called (in line 13). Thus, $d_i \geq m - k_1 + X_1 + X_2$. Also, $P_{\mathrm{OPT}}(d_i) = d_i - m = X_1 + X_2 - k_1$. Finally, because the single battery in K_2 allocates at most 0.5 in the greedy allocation, $X_2 < 0.5$. Together with the fact that $d_i \geq m$, we get $m - k_1 + X_1 + 0.5 > m$, implying $k_1 - X_1 < 0.5$. Combining the above, $P_{\mathrm{ALG}}(d_i) = k_1 - X_1 + X_2$, and $\Delta(i) = (k_1 - X_1 + X_2) - (X_1 + X_2 - k_1) = 2(k_1 - X_1) < 1 \leq k_1 \leq e_i$. Note that $k_1 \geq 1$ as otherwise $\hat{m} = m$ and the allocation is optimal. ■

Next, we consider the case in which the batteries' status is not *AboveTheLine* when $C_1^i > 1.5$ and $C_m^i < 1$. We bound the redundant penalty

in this case and also show that the algorithm might accumulate a redundant penalty at most once before the condition $C_1^i < 1.5$ holds.

Claim 4.5. *If* $C_1^i \geq 1.5, C_m^i < 1$ *and* $\Delta(i) > 0$, *then* $\Delta(i) \leq e_i + 1$ *and* $C_1^{i+1} < 1.5$.

Proof. First note that when $C_1^i \geq 1.5$, the condition in line 21 is not valid and redundant penalty might be caused only by *GreedyAllocation*. By Claim 4.2, if $C_m^i < 1$, then $C_1^i < 2.5$. Assume that *GreedyAllocation* is called. According to the algorithm, this happens only after we set to 1 the allocation of each of the $\hat{m}$ batteries with $C_j^i \geq 1$ (lines 13, 25, 34). Thus, as a result of such an allocation, the remaining capacity of all batteries is less than 1.5, and in particular $C_1^{i+1} < 1.5$.

Claim 4.2 also implies that because $C_1^i \geq 1.5$, no battery was emptied before request i is serviced, that is $C_m^i > 0$. Let $\beta = C_m^i$. By Claim 4.2, $C_1^i \leq 1.5 + \beta$ (see Figure 2b). We have $X_1 \geq \beta k_1$, thus, $p_1 \leq k_1 - X_1 \leq k_1(1 - \beta)$. Each K_2 battery has capacity at most $C_1^i \leq 1.5 + \beta$ and allocates an initial allocation of 1 before *GreedyAllocation* is called. Thus, the maximal penalty for each battery in K_2 is $(0.5 + \beta)$ and $p_2 = X_2 \leq k_2(0.5 + \beta)$. Summing up, $P_{\mathrm{ALG}}(d_i) = p_1 + p_2 \leq k_1(1 - \beta) + k_2(0.5 + \beta)$. The number of emptied batteries is at least $k_1 + k_2 - 1$. If $\beta \leq 0.5$ or $k_2 = 0$, then $\Delta(i) \leq P_{\mathrm{ALG}}(d_i) \leq k_1 + k_2 \leq e_i + 1$.

If $\beta > 0.5$ and $k_2 > 0$, we analyze separately different values of d_i. Because *GreedyAllocation* is called after allocating 1 from each of the $\hat{m}$ batteries with $C_j^i \geq 1$, we have $d_i = \hat{m} + X_1 + X_2$. Moreover, because no battery is empty and $k_2 > 0$ we have that $\hat{m} = m - k_1$. Therefore,

$$d_i = m - k_1 + X_1 + X_2. \tag{2}$$

If $d_i > m$, let $d_i = m + D$. $P_{\mathrm{OPT}}(d_i) = D$. By Equation (2), we have $D = X_1 + X_2 - k_1 = d_i - m$. Thus, $P_{\mathrm{OPT}}(d_i) = D = X_1 + X_2 - k_1$. Also, $P_{\mathrm{ALG}}(d_i) = p_1 + p_2 = k_1 - X_1 + X_2$. We conclude that $\Delta(i) \leq (k_1 - X_1 + X_2) - (X_1 + X_2 - k_1) = 2(k_1 - X_1)$. Because $\beta > 0.5$, it holds that $X_1 > k_1/2$, thus, $2(k_1 - X_1) < k_1$, implying $\Delta(i) < k_1 < e_i + 1$.

If $1 < d_i < m$, then by Equation (2), $m - k_1 + X_1 + X_2 = d_i < m$, implying $X_2 < k_1 - X_1$. We have $P_{\mathrm{ALG}}(d_i) = k_1 - X_1 + X_2 < 2(k_1 - X_1)$. Because $\beta > 0.5$, it holds that $X_1 > k_1/2$, thus, $2(k_1 - X_1) < k_1$ implying $\Delta(i) < P_{\mathrm{ALG}}(d_i) < k_1 < e_i + 1$.

Finally, note that if $d_i < 1$, then because $C_1^i \geq 1.5$, the algorithm is optimal. ■

Combining Claims 4.4 and 4.5, we find that for all requests for which $e_i > 0$, except perhaps for one, it holds that $\Delta(i) \leq e_i$. In addition, we have that for a single request, the one for which *GreedyAllocation* is called when $C_m^i < 1$, it might be that $e_i < \Delta(i) \leq e_i + 1$. To complete our analysis, we show that this gap of 1 is closed by the last request. In the proof below we assume that when the last request is supplied $C_1^n < 1.5$, because otherwise, the scenario handled in Claim 4.5 never happens and there is no gap to bridge, that is, before the last request $\sum_i \Delta(i) \leq \sum_i e_i$.

Claim 4.6. *Either $m' > 0$, or for the last request, $e_i > 0$ and $\Delta(i) \leq e_i - 1$.*

Proof. If all batteries are emptied, then the last request empties the last battery. Assume that greedy empties k_1 batteries for which $C_j^i \leq 1$, and k_2 batteries for which $C_j^i > 1$. The proof is identical to the proof of Claim 4.4—recall that $C_1^n < 1.5$. The analysis in the proof of Claim 4.4 refers to $e_i = k_1 + k_2 - 1$. Formally, it is shown that when $C_1^n < 1.5$, it holds that $\Delta(i) \leq k_1 + k_2 - 1$. When batteries are emptied, $e_i = k_1 + k_2$, therefore, for the last request, $\Delta(i) \leq k_1 + k_2 - 1 = e_i - 1$. ■

We conclude that either the last battery is not emptied, or, if it is emptied, for at least one request (the last one), at least one battery is emptied with no redundant penalty. This compensates for the additional penalty the algorithm might have accumulated when $C_1^i \geq 1.5$ (discussed in Claim 4.5). Summing up, the total redundant penalty of the algorithm is, at most, m. ■

A LOWER BOUND FOR THE MULTIPLICATIVE COMPETITIVE RATIO

A more common performance measure of the online algorithm's quality is its *competitive ratio*, defined as the maximal possible ratio between the algorithm's objective value and an optimal one. In this section, we show that, according to this measure, no algorithm can be better than 1.5 competitive. Formally,

Theorem 5.1. *For every online algorithm, ALG, a sequence of requests σ such that $P_{\mathrm{ALG}}(\sigma) \geq 1.5 P_{\mathrm{OPT}}(\sigma)$ exists.*

Proof. Consider an instance with $m = 2$ batteries with an initial capacity $C = 1$. The sequence σ is constructed by the adversary as a response to the behavior of the algorithm. The possible sequences, provided by the adversary, and the possible allocations of the algorithm are described in Figure 3. Every node corresponds to a possible configuration of the batteries and is described as a vector of the remaining capacities.

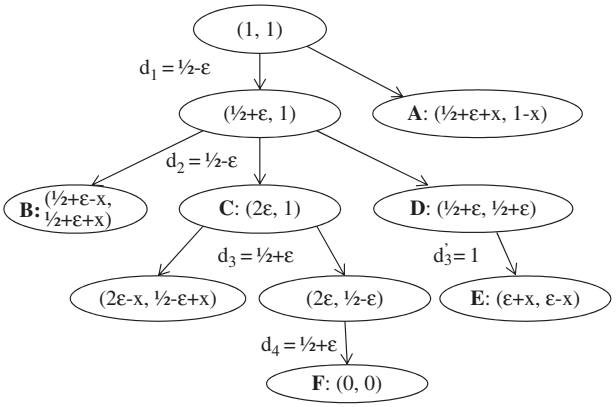

FIGURE 3 Possible behavior of an online algorithm and the adversary.

The first request presented by the adversary is for $d_1 = 0.5 - \varepsilon$. If the algorithm splits the service between the two batteries (configuration A in the figure), then $P_{\mathrm{ALG}}(\sigma) = 2 - d_1 = 1.5 + \varepsilon$, whereas $P_{\mathrm{OPT}}(\sigma) = 1 - d_1 = 0.5 + \varepsilon$. In this case, the adversary halts; the whole sequence is a single request, and the competitive ratio is arbitrarily close to 3 (by fixing $\epsilon \to 0$).

If the algorithm supplies the whole first request from a single battery, without loss of generality, from B_1, then the adversary proceeds with $d_2 = 0.5 - \varepsilon$. The algorithm can choose one of three options: split the service (configuration B), provide the whole request from battery B_1 (configuration C), or provide the whole request from battery B_2 (configuration D).

If the algorithm chooses to move to configuration B, the adversary halts. The penalty for the second request is $2 - d_2 = 1.5 + \varepsilon$. In addition to the penalty for the first request, the total penalty is $P_{\mathrm{ALG}}(\sigma) = (0.5 + \varepsilon) + (1.5 + \varepsilon) = 2 + 2\varepsilon$, whereas an optimal service has penalty $P_{\mathrm{OPT}}(\sigma) = 2(0.5 + \varepsilon) = 1 + 2\varepsilon$. The competitive ratio is arbitrarily close to 2.

If the algorithm chooses to move to configuration D, then the adversary proceeds with $d'_3 = 1$. The algorithm must serve the request d'_3 from both batteries (configuration E) with total penalty for the whole sequence $P_{\mathrm{ALG}}(\sigma) = (0.5 + \varepsilon) + (0.5 + \varepsilon) + 1 = 2(1 + \varepsilon)$. An optimal service for this sequence (reaching configuration C and providing the whole request d'_3 from battery B_2) would incur a penalty of $P_{\mathrm{OPT}}(\sigma) = 1 + 2\varepsilon$. The resulting competitive ratio is arbitrarily close to 2.

We are left with the case in which the algorithm chooses to move to configuration C. The adversary proceeds with $d_3 = 0.5 + \varepsilon$. If the algorithm splits the service (left node of C), the penalty for the last request is $1.5 - \epsilon$ and the total penalty for the whole sequence is $P_{\mathrm{ALG}}(\sigma) = 2.5 + \varepsilon$, whereas an optimal service for this sequence (providing the whole request d_3 from

battery B_2) would incur a penalty of $P_{\mathrm{OPT}}(\sigma) = 1.5 + \varepsilon$. The resulting competitive ratio is arbitrarily close to 2.5, and larger than 1.5 for any $\epsilon \leq 0.5$. Otherwise (right node of C), the adversary proceeds with a final request $d_4 = 0.5 + \varepsilon$. The algorithm must move to configuration F. For the whole sequence the penalty is $P_{\mathrm{ALG}}(\sigma) = (0.5 + \varepsilon) + (0.5 + \varepsilon) + (0.5 - \varepsilon) + (1.5 - \varepsilon) = 3$, whereas an optimal service of this sequence (through configuration D) would incur a penalty of $P_{\mathrm{OPT}}(\sigma) = 2$. Thus, the competitive ratio for this scenario is 1.5.

We conclude that for any behavior of the algorithm, the adversary can proceed in a way that guarantees $P_{\mathrm{ALG}}(\sigma) \geq 1.5 P_{\mathrm{OPT}}(\sigma)$. ■

DISCUSSION AND OPEN PROBLEMS

This article presented studied the problem of utilizing the pack of batteries serving current demands in Electric Vehicles. We formulated the problem as a combinatorial optimization problem, provided hardness results that are valid even for the offline scenario, and suggested an efficient, almost optimal online algorithm. Several important problems remain open:

1. Consider additional penalty functions. In particular, nonlinear penalty functions as well as penalty functions that are not symmetric around the optimal discharge current.
2. Our algorithm ("An Almost Optimal Online Algorithm") is guaranteed to have a redundant penalty of, at most, m. On the other hand, it is NP-hard to guarantee a redundant penalty lower than $0.5m$, even in the offline case (Theorem 3.3). Can this gap be closed?
3. Consider the combination of several different battery packs in a single EV, each having its own optimal discharge current.

Additional problems arise when the model is extended to consider additional parameters such as battery temperature and other environmental effects. A totally different direction is to study adaptive switching algorithms that are based on learning the driver's driving pattern.

REFERENCES

Affanni, A., A. Bellini, G. Franceschini, P. Guglielmi, and C. Tassoni. 2005. Battery choice and management for new-generation electric vehicles. *IEEE Transactions on Industrial Electronics* 52(5):1343–1349.

Anderson, C., and J. Anderson. 2010. *Electric and hybrid cars: A history*. Jefferson, NC, USA: McFarland.

Benini, L., D. Bruni, A. Macii, E. Macii, and M. Poncino. 2003. Discharge current steering for battery lifetime optimization. *IEEE Transactions on Computers* 52(2):985–995.

Benini, L., G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi. 2001a. Extending lifetime of portable systems by battery scheduling. In *Proceedings of the conference on design, automation and test in Europe*, 197–203. Piscataway, NJ, USA: IEEE Press.

Benini, L., G. Castelli, A. Macii, and R. Scarsi. 2001b. Battery-driven dynamic power management. *IEEE Design & Test of Computers* 18(2):53–60.

Delucchi, M., and T. Lipman. 2001. An analysis of the retail and lifecycle cost of battery-powered electric vehicles. *Transportation Research Part D: Transport and Environment* 6(6):371–404.

Doyle, M., and J. Newman 1997. Analysis of capacity–rate data for lithium batteries using simplified models of the discharge process. *Journal of Applied Electrochemistry* 27(7):846–856.

Garey, M. R., and D. S. Johnson. 1979. *Computers and intractability. A guide to the theory of np-completeness*. New York, NY, USA: W.H. Freeman.

Kirsch, D. 2000. *The electric vehicle and the burden of history*. New Brunswick, NJ, USA: Rutgers University Press.

Laman, F., and K. Brandt. 1988. Effect of discharge current on cycle life of a rechargeable lithium battery. *Journal of Power Sources* 24(3):195–206.

MIT Electric Vehicle Team 2008. A guide to understanding battery specifications. http://mit.edu/evt.

Pedram, M., and Q. Wu. 1999. Design considerations for battery-powered electronics. In *Proceedings of the 36th annual conference on design automation (DAC'99)*, 861–866. Washington, DC, USA: IEEE Computer Society.

Rao, R., S. Vrudhula, and D. Rakhmatov. 2003. Analysis of discharge techniques for multiple battery systems. In *Proceedings of the 2003 international symposium on low power electronics and design*, 44–47. New York, NY, USA: ACM.

Shachnai, H., T. Tamir, and O. Yehezkely. 2008. Approximation schemes for packing with item fragmentation. *Theory of Computing Systems* 43(1):81–98.