





Optimal energetic paths for electric cars

Dani Dorfman  



Tel Aviv University

Haim Kaplan  

Tel Aviv University

Robert E. Tarjan  

Princeton University

Uri Zwick  

Tel Aviv University

Abstract

A weighted directed graph $G = (V, A, c)$, where $A \subseteq V \times V$ and $c : A \rightarrow \mathbb{R}$, naturally describes a road network in which an electric car, or vehicle (EV), can roam. An arc $uv \in A$ models a road segment connecting the two vertices (junctions) u and v . The cost $c(uv)$ of the arc uv is the amount of *energy* the car needs to travel from u to v . This amount can be positive, zero or negative. We consider both the more realistic scenario where there are no negative cycles in the graph, as well as the more challenging scenario, which can also be motivated, where negative cycles may be present.

The electric car has a battery that can store up to B units of energy. The car can traverse an arc $uv \in A$ only if it is at u and the charge b in its battery satisfies $b \geq c(uv)$. If the car traverses the arc uv then it reaches v with a charge of $\min\{b - c(uv), B\}$ in its battery. Arcs with a positive cost deplete the battery while arcs with negative costs may charge the battery, but not above its capacity of B . If the car is at a vertex u and cannot traverse any outgoing arcs of u , then it is stuck and cannot continue traveling.

We consider the following natural problem: Given two vertices $s, t \in V$, can the car travel from s to t , starting at s with an initial charge b , where $0 \leq b \leq B$? If so, what is the maximum charge with which the car can reach t ? Equivalently, what is the smallest *depletion* $\delta_{B,b}(s, t)$ such that the car can reach t with a charge of $b - \delta_{B,b}(s, t)$ in its battery, and which path should the car follow to achieve this? We also refer to $\delta_{B,b}(s, t)$ as the *energetic cost* of traveling from s to t . We let $\delta_{B,b}(s, t) = \infty$ if the car cannot travel from s to t starting with an initial charge of b . The problem of computing energetic costs is a strict generalization of the standard shortest paths problem.

When there are no negative cycles, the single-source version of the problem can be solved using simple adaptations of the classical Bellman-Ford and Dijkstra algorithms. More involved algorithms are required when the graph may contain negative cycles.

2012 ACM Subject Classification Theory of Computation \rightarrow Design and analysis of algorithms

Keywords and phrases Electric cars, Optimal Paths, Battery depletion

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.61

Funding *Dani Dorfman*: Supported by ISF grant 2854/20.

Haim Kaplan: Supported by ISF grant 1595/19 and the Blavatnik Family Foundation.

Robert E. Tarjan: Partially supported by a gift from Microsoft.

Uri Zwick: Supported by ISF grant 2854/20.

Acknowledgements We would like to thank the anonymous reviewers for their comments, for pointing out reference [14], and for pointing out the relation to the 1-VASS problem.



© Dan Dorfman and Haim Kaplan and Robert Tarjan and Uri Zwick;
licensed under Creative Commons License CC-BY 4.0

31st Annual European Symposium on Algorithms (ESA 2023).

Editors: Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman; Article No. 61;
pp. 61:1–61:16



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A weighted directed graph $G = (V, A, c)$, where $A \subseteq V \times V$ and $c : A \rightarrow \mathbb{R}$, naturally describes a road network in which an electric car can roam. An arc $uv \in A$ models a road segment connecting the two vertices (junctions) u and v . The cost $c(uv)$ of the arc uv is the amount of *energy* the electric car needs to travel from u to v . This amount can be positive, e.g., if the road segment is uphill; zero; or negative, e.g., if the road segment is downhill. We consider both the more realistic scenario where there are no negative cycles in the graph, as well as the more challenging scenario, which can also be motivated, where the graph may contain negative cycles. (A cycle is negative if the sum of its arc costs is negative.)

An electric car is equipped with a battery that can store up to B units of energy, where $B > 0$ is a parameter. We assume that the electric car cannot be charged along the way and has to rely on the initial charge available in its battery. If the car is currently at vertex u with charge b in its battery, where $0 \leq b \leq B$, then it can traverse an arc $uv \in A$ if and only if $c(uv) \leq b$. If this condition holds, and the car traverses the arc, then it reaches v with a charge of $\min\{b - c(uv), B\}$. In particular, the charge in the battery cannot be negative and cannot exceed B . The car can traverse uv if $b - c(uv) > B$ (which can hold only if $c(uv) < 0$), but the battery will not charge beyond its capacity of B . We may assume that $c(uv) \in [-B, B]$ for every $uv \in A$, as arcs with $c(uv) > B$ can never be used, and thus can be removed, and costs $c(uv) < -B$ can be changed to $c(uv) = -B$.

We consider the following natural problem: Given two vertices $s, t \in V$, can the car travel from s to t , starting at s with an initial charge b , where $0 \leq b \leq B$? If so, what is the *maximum final charge* $\alpha_{B,b}(s, t)$ with which the car can reach t ? Equivalently, what is the *minimum depletion* $\delta_{B,b}(s, t)$ such that the car can reach t with a charge of $b - \delta_{B,b}(s, t)$ in its battery, and which path should the car follow to achieve this? (If $b < B$ then the minimum depletion $\delta_{B,b}(s, t)$ may be negative.) We also refer to $\delta_{B,b}(s, t)$ as the *energetic cost* of traveling from s to t . Note that $\delta_{B,b}(s, t) = b - \alpha_{B,b}(s, t)$. We let $\alpha_{B,b}(s, t) = -\infty$ and $\delta_{B,b}(s, t) = \infty$ if the car cannot travel from s to t starting with an initial charge of b .

We also consider the related problem of finding the *minimum initial charge* at s , if any, that will allow the car to travel to t , ending with a charge of at least b in the battery. We denote this quantity by $\beta_{B,b}(s, t)$. We show that minimum initial charges can be computed by computing maximum final charges, or minimum energetic costs, on the *reversed* graph.

If all arc costs are non-negative, then $\delta_{B,b}(s, t) = \delta(s, t)$, if $\delta(s, t) \leq b \leq B$, where $\delta(s, t)$ is the standard distance, i.e., the length of a shortest path, from s to t in the graph G , where $c(uv)$ is the length of uv . Otherwise, $\delta_{B,b}(s, t) = \infty$. If there are negative arc costs but no negative cycles, $\delta_{B,b}(s, t) = \delta(s, t)$ if and only if there exists a shortest path P from s to t such that the length of every prefix of P is in the interval $[b - B, b]$. In general, energetic costs may be larger than distances, since the charge in the battery is constrained to remain in the interval $[0, B]$, i.e., it is not allowed to go negative and it is capped at B . (For example, the electric car may not be able to traverse a mountain pass and may need to take a detour.) It is always true that $\delta_{B,b}(s, t) \geq \delta(s, t)$.

The problem of computing minimum energetic costs is thus a strict generalization of the standard shortest paths problem, even if there are no negative cycles in the graph. Interestingly, the energetic costs $\delta_{B,b}(s, t)$ are well-defined even if there are negative cycles in the graph. The problem is then an interesting variant of one-player *energy games* with a reachability objective. It is also related to the 1-VASS (Vector Addition Systems with States) problem. (See references Section 1.2.) The presence of negative cycles poses interesting algorithmic challenges. The corresponding minimum energetic paths are still finite, but

are not necessarily simple. A minimum energy path from s to t may need to go through a sequence of negative cycles, using each one of them to gain sufficient energy to reach the next negative cycle, and eventually the target t .

When there are no negative cycles, the single-source version of the energetic cost problem can be solved using simple, but subtle, adaptations of the classical Bellman-Ford [4, 15] and Dijkstra [11] algorithms. Similar algorithms, some less efficient, are explicit or implicit in previous papers (see Section 1.2). We present simpler, self-contained versions of these algorithms with simpler correctness proofs. We also present efficient algorithms for finding minimum energetic paths in the presence of negative cycles.

Unlike the standard shortest paths problem, the single-target version of the minimum energetic paths problem is *not* equivalent to the single-source version. In particular, one cannot solve the single-target problem by running a single-source algorithm backward. Although there is always a tree of minimum energetic paths from a source vertex s to all other vertices reachable from it, there are simple examples in which there is no tree of minimum energetic paths to a target vertex t from all vertices that can reach it.

1.1 Our results

We show that the single-source version of the minimum energetic paths problem with negative arc costs but no negative cycles can be solved in $O(mn)$ time using a simple adaptation of the Bellman-Ford [4, 15] algorithm, where $m = |A|$ and $n = |V|$. Furthermore, if a *valid potential function* $p : V \rightarrow \mathbb{R}$ is given, i.e., a function for which $c(uv) + p(u) - p(v) \geq 0$ for every $uv \in A$, then the single-source version can be solved in $O(m + n \log n)$ time using an adaptation of Dijkstra's [11] algorithm equivalent to the A^* search heuristic (see, e.g., Hart et al. [21]). Since a valid potential function can be found in $O(mn)$ time using the standard Bellman-Ford algorithm, the all-pairs version of the minimum energetic paths problem can be solved $O(mn + n^2 \log n)$ time.

The $O(mn)$ bound matches the time bound of the standard Bellman-Ford algorithm, which is still the fastest known algorithm for the single-source shortest paths problem in a directed graph with general (real) arc costs, and no negative cycles. The $O(mn + n^2 \log n)$ bound almost matches the best time bound of $O(mn + n^2 \log \log n)$ obtained by Pettie [31] for the standard APSP problem with general arc costs. (For a survey of other related results, see Zwick [38].)

Much faster algorithms are known for the standard single-source shortest paths problem when arc costs are integral. Bringmann et al. [8], improving a breakthrough result of Bernstein et al. [5], obtained an $O(m \log^2 n \log(nW) \log \log n)$ -time algorithm when arc costs are integers that are at least $-W$, where $W \geq 1$. By using these algorithms to find a valid potential function, and then using the energetic version of Dijkstra's algorithm, we get the same improved time bound for the single-source version of the minimum energetic paths problem with negative arc costs but no negative cycles.

We also present a more involved $O(mn + n^2 \log n)$ -time algorithm for solving the single-source minimum energetic cost problem in the presence of negative cycles. This gives, of course, an $O(mn^2 + n^3 \log n)$ -time algorithm for the all-pairs and single-target versions of the problem.

When the capacity B of the battery is sufficiently large, we show that the all-pairs version of the minimum energetic cost problem can be solved in $O(mn + n^2 \log n)$ time.

1.2 Related results

Various adaptations of the Bellman-Ford and Dijkstra algorithms for problems similar or equivalent to the minimum energetic paths problem defined here, when there are no negative cycles in the graph, were given by several authors. Eisner et al. [14], improving upon Artmeier et al. [2], and Brim and Chalupka [6] give versions of these algorithms with the same running times as ours, but using a slightly different approach. Baum et al. [3] give a version of Dijkstra's algorithm but with a much slower running time. They also show that the maximum charge with which t can be reached when starting at s with charge b is a piece-wise linear function of b with at most $O(n)$ breakpoints.

Brim and Chalupka [6] consider the related problem of solving one-player *energy games* and the more complicated problem of solving two-player *energy games*. (For more on energy games, see also Brim et al. [7] and Dorfman et al. [12].) The problem of finding minimum energy paths in the presence of negative cycles may be seen as a variant of one-player energy games with a reachability objective. The minimum energetic paths problem is also related to the 1-VASS problem. (See, e.g., Almagor et al. [1] and Künnemann et al. [26].)

Khuller et al. [23] consider a related problem in which the battery (or the fuel tank) can be recharged at intermediate vertices, with a possibly different price per unit of charge at each intermediate vertex. All arc costs are non-negative. They give various algorithms for computing a cheapest path from s to t . Among these algorithms is a $O(n^2 \Delta \log n)$ -time algorithm for the single-target version, where Δ is a bound on the number of rechargings allowed, and an $O(n^3)$ -time algorithm for the single-target version when the number of rechargings is unbounded.

Several authors, including Lehmann [27], Tarjan [33] and Mohri [30] considered generalized versions of the shortest paths problem defined by *semirings*. If (R, \oplus, \otimes) is a semiring and P is an s - t path whose arcs have costs c_i , the cost of P is defined to be $c(P) = \otimes_{i=1}^k c_i$. The goal is to find $\oplus_P c(P)$, where P ranges over all s - t paths, assuming this quantity is well defined. The standard shortest paths problem corresponds to the *tropical* semiring $(\mathbb{R}, \min, +)$. All these results assume, as part of the definition of a semiring, that \otimes is associative. Thus, as we shall see, none of these results apply to our problem, as our operation \otimes is not associative.

Generalized versions of Dijkstra's algorithm were obtained by various authors, most notably by Knuth [25]. These generalizations are of a different nature and are apparently not related to the version given here.

Other non-standard versions of the shortest paths problem were also considered. Perhaps the most famous one is the *bottleneck* shortest paths problem. See, e.g., [17, 9] for the single-pair version, and [36, 13] for the all-pairs version. Vassilevska [35] considered an interesting non-standard *non-decreasing* version of the shortest paths problem related to reading train schedules. Finally, Madani et al. [29] considered the *discounted* shortest paths problem. All these problems are quite different from the problem considered here.

2 Minimum energetic paths

To simplify the presentation, we concentrate on the computation of $\delta_B(s, t) = \delta_{B,B}(s, t)$, i.e., the energetic cost of traveling from s to t when starting with a fully charged battery of capacity B . Computing $\delta_{B,b}(s, t)$, for an arbitrary $0 \leq b \leq B$, can easily be reduced to computing $\delta_B(s, t)$: Add a new vertex s' and an arc $s's$ of cost $c(s's) = B - b$. Then, it is easy to see that $\delta_{B,b}(s, t) = \delta_B(s', t) - (B - b)$. A similar idea can be incorporated directly into the algorithms that we describe. We begin by defining the energetic cost of a path.

61:4 Optimal energetic paths for electric cars

► **Definition 2.1** (Energetic cost of a path). A path $P = u_0 u_1 \dots u_k$ is traversable if it can be traversed when starting from u_0 with a fully charged battery. If P is traversable, the final charge in the battery when reaching u_k is $B - d_B(P)$, where $d_B(P)$ is defined to be the depletion, or the energetic cost of the path. Note that $0 \leq d_B(P) \leq B$. If the path is not traversable, we let $d_B(P) = \infty$.

To obtain a simple formula for $d_B(P)$ we define the following operations:

$$x \oplus_B y = [x + y]_0^B, \quad [z]_0^B = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } 0 \leq z \leq B \\ \infty & \text{otherwise} \end{cases}$$

We assume that $x + \infty = \infty + x = \infty$ for every $x \in [-B, B] \cup \{\infty\}$. For brevity, we sometimes write $x \oplus y$ instead of $x \oplus_B y$ when B is understood from the context.¹ Note that for every $x, y \in \mathbb{R}$ and $B > 0$ we have $x + y \leq x \oplus_B y$. It is important to note that \oplus_B is *not* associative. (For example, $B \oplus (B \oplus -B) = B$ while $(B \oplus B) \oplus -B = \infty$, and $(-1 \oplus -2) \oplus 2 = 2$ while $-1 \oplus (-2 \oplus 2) = 0$, assuming $B \geq 2$.)

► **Lemma 2.2.** Let $P = u_0 u_1 \dots u_k$ be a directed path, let $P' = u_0 \dots u_{k-1}$, and let $c_i = c(u_{i-1} u_i)$, for $i = 0, 1, \dots, k$. Let B be the capacity and initial charge of battery at u_0 . If $k = 0$ then $d_B(P) = 0$. If $k > 0$ then

$$d_B(P) = d_B(P') \oplus c_k = ((\dots((0 \oplus c_1) \oplus c_2) \oplus \dots) \oplus c_{k-1}) \oplus c_k.$$

Proof. Let b_i be the charge of the battery at u_i and let $d_i = B - b_i$ be the depletion of the battery at u_i , for $i = 0, 1, \dots, k$. Clearly $d_0 = 0$. It is easy to prove by induction that $d_i = d_{i-1} \oplus_B c_i$. The lemma follows. ◀

As mentioned, the operation \oplus_B is *not* associative. Thus, in general, $d_B(P) \neq 0 \oplus (c_1 \oplus (\dots \oplus (c_{k-2} \oplus (c_{k-1} \oplus c_k)) \dots))$. In Section 7 we show, however, that $c_1 \oplus (c_2 \oplus (\dots \oplus (c_{k-1} \oplus (c_k \oplus 0)) \dots))$ also has an interesting meaning.

► **Definition 2.3** (Energetic costs, minimum energetic paths). The energetic cost $\delta_B(s, t)$ of traveling from s to t , starting from s with a fully charged battery of capacity B , is defined as

$$\delta_B(s, t) = \min\{d_B(P) \mid P \text{ is an } s\text{-}t \text{ path in } G\}.$$

If $\delta_B(s, t) < \infty$ and P is an s - t path satisfying $\delta_B(s, t) = d_B(P)$, then P is said to be a minimum energetic path from s to t .

If there are no negative cycles in the graph, then for every path P from s to t there is a simple path P' such that $d_B(P') \leq d_B(P)$. (It is in fact enough to require that there are no traversable negative cycles in the graph.) Thus, the minimum in the definition above can be taken over simple paths only. The definition of $\delta_B(s, t)$ is also meaningful in the presence of negative cycles, though minimum energetic paths are not necessarily simple.

It is not difficult to see, and it will also follow from the correctness of the algorithms that we present in the next sections, that for every source vertex $s \in V$ there is always a tree of minimum energetic paths to all other vertices that can be reached from it. The simple example given in Figure 1 shows that a tree of minimum energetic paths to a given target vertex t does not always exist.

To deal with negative cycles, we need the following definition.

¹ Note that \oplus is not related to the semiring framework mentioned in Section 1.2.

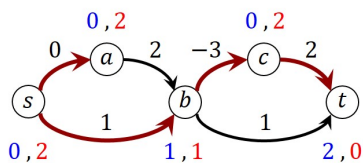


Figure 1 A simple but illustrative example. Assume $B \geq 3$. The two numbers next to each vertex u are $\delta_B(s, u)$ and $\delta_B(u, t)$. The bold arcs constitute a tree of minimum energetic paths from the source vertex s to all other vertices. Another such tree can be obtained by replacing the arc ct by bt . On the other hand, the only minimum path from a to t is $abct$, while the only minimum path from b to t is the arc bt . Thus, there is no tree of minimum paths to t from all other vertices.

Definition 2.4 (Entry-exit pairs). Let C be a negative cycle in $G = (V, A, c)$ and let B be the maximum capacity of the battery. A pair of vertices (x, y) on C is an entry-exit pair of C if the car can start at x with an empty battery and eventually reach y , possibly after going several times around the cycle, with a full battery, i.e., with a charge of B .

The following lemma, similar to the gasoline puzzle of Lovász [28, p. 31] (see also Klarner [24, p. 283] and Winkler [37, p. 2]), says that every negative cycle has an entry-exit pair.

Lemma 2.5. Any negative cycle C in $G = (V, A, c)$ contains at least one entry-exit pair. Such a pair can be found in $O(|C|)$ time.

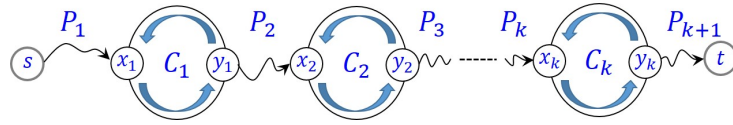
Proof. We show first that every negative cycle has an entry x , i.e., a vertex x from which the cycle can be traversed, starting with an empty battery, when the capacity of the battery is ignored. We next show that the entry x has a corresponding exit y when the capacity of the battery is not ignored.

Let $C = v_0v_1 \cdots v_{\ell-1}v_0$ be a negative cycle in G of length ℓ . Let $c_j = c(v_jv_{j+1})$, for $j = 0, 1, \dots, \ell - 1$. (We let $v_\ell = v_0$.) For $j \geq \ell$, let $c_j = c_{j \bmod \ell}$. Let $s_i = \sum_{j=0}^{i-1} c_j$ be the prefix sums of the costs around the cycle, starting from v_0 . (We allow $i > \ell$ by wrapping around the cycle.) Note that $s_\ell = \sum_{j=0}^{\ell-1} c_j < 0$ as the cycle is negative. This also implies that $s_{\ell+i} < s_i$ for every i . Vertex v_0 is an entry if and only if $s_1, \dots, s_{\ell-1} \leq 0$. If v_0 is not an entry, let k be the index for which s_k is maximized. Note that $0 \leq k < \ell$. We claim that v_k is an entry. Let $s'_i = \sum_{j=0}^{i-1} c_{k+j}$ be the prefix sums starting from v_k . Then $s'_i = s_{k+i} - s_k \leq 0$, by the definition of k , for every i .

Assume, without loss of generality, that v_0 is an entry on C . Let $s_i = \sum_{j=0}^{i-1} c_j$ as above and let $i = \min\{j \mid s_j \leq -B\} \pmod{\ell}$. Then (v_0, v_i) is an entry-exit pair. It is not difficult to find i in $O(\ell)$ time. ◀

The following lemma characterizes the structure of minimum energetic paths when the graph may contain negative cycles. It is not difficult to give a direct proof of the lemma. Its correctness also follows from the correctness of the algorithms that we present.

Lemma 2.6. If there is an s - t path P with $d_B(P) \leq B$, then there is an s - t path P' such that $d_B(P') \leq d_B(P)$ and such that P' has the following form (see Figure 2): either P' is simple, or there is a sequence C_1, C_2, \dots, C_k of simple negative cycles, where $k < n$, with entry-exit pairs $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$ on them, such that P' is composed of a simple path from s to x_1 , followed by sufficiently many traversals of C_1 that end in y_1 with a full battery, followed by a simple path from y_1 to x_2 , followed by sufficiently many traversals of C_2 that end in y_2 with a full battery, and so on, and finally a simple path from y_k to t . Furthermore, all entries x_1, x_2, \dots, x_k are distinct, and all exits y_1, y_2, \dots, y_k are distinct.



■ **Figure 2** Generic structure of minimum energetic paths in the presence of negative cycles. If $\delta_B(s, t) \leq B$, then there is a minimum energetic path from s to t of the form shown, where C_1, \dots, C_k are simple negative cycles and (x_i, y_i) is an entry-exit pair on C_i , for $i = 1, 2, \dots, k$. All entries x_1, x_2, \dots, x_k are distinct and all exits y_1, y_2, \dots, y_k are distinct. The paths P_1, P_2, \dots, P_{k+1} are simple but not necessarily disjoint from the cycles C_1, C_2, \dots, C_k .

3 An energetic version of the Bellman-Ford algorithm

Recall that for any x and y , $x \oplus_B y \geq x + y$. This implies that in a graph without negative cycles, if there is a traversable path from s to t , there is such a path that is simple and hence contains at most $n - 1$ arcs. This means that if there are no negative cycles, we can solve the single-source minimum energetic paths problem using the Ford-Bellman [4, 15] shortest path algorithm: We merely replace $+$ by \oplus_B .

We base our description of the algorithm on that in [34], which uses a queue as suggested by Gilsinn and Witzgall [18].

The algorithm maintains a tentative energetic cost $d(v)$ for each vertex v , equal to the minimum of the energetic costs of paths from s to v found so far. Initially $d(s) = 0$ and $d(v) = \infty$ for $v \neq s$, where s is the source. It also maintains a queue Q , initially containing s . The algorithm repeats the following step until Q is empty:

Scan a vertex: Delete the front vertex u on Q . For each arc uv , if $\delta(u) \oplus_B c(uv) < \delta(v)$,
relax uv: Set $\delta(v) \leftarrow \delta(u) \oplus_B c(uv)$, set $\pi(v) \leftarrow u$, and add v to the back of Q if it is not on Q .

Pseudocode of the algorithm, which we call *e-BF*, is given on the left of Figure 3. The correctness proof and analysis of the standard Bellman-Ford algorithm in the absence of negative cycles (see e.g., Tarjan [34]) translates directly to this version.

► **Theorem 3.1.** *If $G = (V, A, c)$ has no traversable negative cycles then *e-BF* finds minimum energetic paths from s to all vertices in $O(mn)$ time.*

Proof. We define *passes* over the queue. Pass 0 is the first scan step of s . Given that pass k is defined, pass $k + 1$ is the sequence of scan steps of vertices added to Q during pass k . A straightforward induction on k shows that for each vertex v that has a minimum-energy path of at most k arcs, $d(v)$ is the energetic cost of such a path after k passes. It follows that the energetic costs are correctly computed. The π values computed describe a tree of minimum-energy paths from s to all vertices reachable from s using a fully charged battery of capacity B . Since each pass takes $O(m)$ time, the total time of the algorithm is $O(mn)$. ◀

In addition to the non-existence of negative cycles, the only thing required for correctness of the algorithm is that \oplus_B is non-decreasing in its second argument: If $y \leq z$, $x \oplus y \leq x \oplus z$.

As in the standard version of the Bellman-Ford algorithm, one can add *subtree disassembly* [32, 10], which does not improve the worst-case time bound but is likely to speed up the algorithm in practice. It is also easy to modify the algorithm so that it finds a traversable negative cycle that can be reached from s , if one exists.

The correctness of *e-BF* implies the following corollary, which we use to prove the correctness of the energetic variant of Dijkstra's algorithm:


```

e-BF( $G = (V, A, c), B, s$ ):
  for  $u \in V$  do
     $d(u) \leftarrow \infty$ 
     $\pi(u) \leftarrow \text{null}$ 
   $d(s) \leftarrow 0$ 
   $Q \leftarrow \text{Queue}()$ 
   $Q.\text{insert-last}(s)$ 
  while  $Q \neq \emptyset$  :
     $u \leftarrow Q.\text{DeleteFirst}()$ 
    for  $uv \in A$  do
      if  $d(v) > d(u) \oplus_B c(uv)$  :
         $d(v) \leftarrow d(u) \oplus_B c(uv)$ 
         $\pi(v) \leftarrow u$ 
        if  $v \notin Q$  :
           $Q.\text{InsertLast}(v)$ 
  return  $d$ 

e-Dijkstra( $G = (V, A, c), p, B, s$ ):
  for  $u \in V$  do
     $d(u) \leftarrow \infty$ 
     $\pi(u) \leftarrow \text{null}$ 
   $d(s) \leftarrow 0$ 
   $H \leftarrow \text{min-heap}()$ 
   $H.\text{insert}(s, -p(s))$ 
  while  $H \neq \emptyset$  :
     $v \leftarrow H.\text{delete-min}()$ 
    for  $uv \in A$  do
      if  $d(v) > d(u) \oplus_B c(uv)$  :
         $d(v) \leftarrow d(u) \oplus_B c(uv)$ 
         $\pi(v) \leftarrow u$ 
        if  $v \notin H$  :
           $H.\text{insert}(u, d(v) - p(v))$ 
        else:
           $H.\text{decrease-key}(u, d(v) - p(v))$ 
  return  $d$ 

```

■ **Figure 3** Energetic variants of the Bellman-Ford and Dijkstra algorithms.

► **Corollary 3.2.** *If each $d(v) < \infty$ corresponds to the energetic cost of some path from s to v , and $d(v) \leq d(u) \oplus_B c(uv)$ for every $uv \in A$, then $d(v) = \delta_B(s, v)$, for every $v \in V$.*

4 An energetic version of Dijkstra's algorithm

If all arc costs are non-negative, Dijkstra's algorithm [11] with $+$ replaced by \oplus_B will solve the single-source problem. This algorithm replaces the queue Q in the Bellman-Ford algorithm by a heap H . The key of a vertex v in the heap is $d(v)$. Each scan step deletes a vertex of minimum key from the heap. When a relaxation decreases the key of a vertex in the heap, the algorithm does the appropriate *decrease-key* operation on the heap. If all arc costs are non-negative, the algorithm deletes each vertex from H at most once, and when a vertex v is deleted from H , $d(v)$ is the minimum energetic cost of a path from s to v . The proof of correctness mimics that of the standard Dijkstra algorithm. The algorithm does at most n heap insertions, at most n heap deletions, and at most m decrease-key operations. If the heap is a Fibonacci heap [16] or equally efficient data structure, e.g., [20], the total running time is $O(m + n \log n)$. In fact, the algorithm is identical to the standard algorithm with $d(v)$ values greater than B replaced by ∞ .

More interesting is that if arc costs can be negative, but there are no negative cycles, we can use a variant of the A^* search algorithm, which is a modification of Dijkstra's algorithm, to solve the single-source minimum energetic paths problem in $O(m + n \log n)$ time, provided that we have a *valid potential function* $p: V \rightarrow \mathbb{R}$. A potential p is *valid* if $c(uv) + p(u) - p(v) \geq 0$ for every arc $uv \in A$. It is well-known that a valid potential function exists if and only if the graph contains no negative cycles.

The A^* search algorithm is almost identical to Dijkstra's algorithm. The only difference

is that the *key* of vertex v in the heap is $d(v) - p(v)$, and not just $d(v)$, where p is a valid potential function. In the original setting of the A^* search heuristic, $-p(v)$ is an estimate of the distance from v to the destination t . The correctness of the algorithm only requires, however, that p is a valid potential function. If p is valid, the A^* algorithm deletes each vertex v from the heap at most once, and when v is deleted, $d(v) = \delta_B(s, v)$, the energetic cost of traveling from s to v .

An energetic version of the A^* is obtained simply by replacing $+$ by \oplus_B in relaxations. We assume the algorithm is given a potential p that is valid for $+$, not \oplus_B . The algorithm begins with $d(s) = 0$ and $d(v) = \infty$ for each vertex $v \in V \setminus \{s\}$, and H containing s . The key of a vertex v in H is $d(v) - p(v)$. The algorithm repeats the following step until H is empty:

Scan a vertex: Delete from H a vertex u with minimum key $d(u) - p(u)$. For each arc uv , if $d(u) \oplus_B c(uv) < d(v)$, *relax* uv : Set $d(v) \leftarrow d(u) \oplus_B c(uv)$; $\pi(v) \leftarrow u$; add v to H with key $d(v) - p(v)$ if $v \notin H$, or decrease the key of v to $d(v) - p(v)$ if $v \in H$.

Pseudocode of the resulting algorithm, which we call *e-Dijkstra*, is given on the right of Figure 3. The main step towards establishing the correctness of *e-Dijkstra* is the following:

► **Lemma 4.1.** *If p is a valid potential then e-Dijkstra maintains the following invariant: if u has been deleted from H while v has not been deleted from H yet, then $d(u) - p(u) \leq d(v) - p(v)$. As a consequence, each vertex u is inserted and deleted from H at most once.*

Proof. We prove the lemma by induction on the number of heap operations. The lemma is true initially, as no vertex was deleted from H yet. Suppose it is true just before u is deleted from H . Since $d(u) - p(u)$ is *minimum* among all $u \in H$, and since $d(v) = \infty$ for all vertices not yet inserted into H , the invariant holds just after u is deleted from H . By the induction hypothesis, $d(u) - p(u)$ is now *maximum* over all u' already deleted from H . Suppose the invariant holds just before the relaxation of an arc uv . Just after the relaxation, $d(v) = d(u) \oplus_B c(uv) \geq d(u) + c(uv)$. Hence

$$d(v) - p(v) \geq d(u) + c(uv) - p(v) \geq d(u) - p(u) ,$$

where the last inequality follows by the validity of p . Since the relaxation strictly decreased $d(v)$, it follows that v could not have already been deleted from H , since it would violate the claim that $d(u) + p(u)$ is maximum over all vertices already deleted from H . Thus, v is either in H or was not inserted into H yet. Decreasing the key of v to $d(v) - p(v)$, or inserting v into H with this key, does not violate the invariant. ◀

The proof of Lemma 4.1 is the same as the proof of the corresponding lemma for the standard version of A^* except for the use of the inequality $x \oplus_B y \geq x + y$. Using Lemma 4.1 we can easily prove the correctness of the algorithm.

► **Theorem 4.2.** *If $G = (V, A, c)$ has no negative cycles and p is a valid potential for G , then e-Dijkstra finds minimum energetic paths from s to all vertices in $O(m + n \log n)$ time.*

Proof. When a vertex u is removed from H , all outgoing arcs uv are scanned and all appropriate relax operations are performed. By Lemma 4.1, $d(u)$ will not be changed again. Thus, when the algorithm terminates $d(v) \leq d(u) \oplus_B c(uv)$ for every arc $uv \in A$. By Corollary 3.2, we have $d(v) = \delta_B(s, v)$, for every $v \in V$. As in the proof of Theorem 3.1 we get that the π values describe a tree of minimum energetic paths from s to all vertices that can be reached from s .

The algorithm performs at most n heap insertions, at most n heap deletions, and at most m decrease-key operations. With an efficient heap implementation the total running time is $O(m + n \log n)$. ◀

To obtain a valid potential function we can use any standard shortest path algorithm: If s is an arbitrary source from which all vertices are reachable, there are no negative cycles, and $p(v) = \delta(s, v)$, where $\delta(s, v)$ is the standard distance from s to v , then $c(uv) + p(u) - p(v) \geq 0$, for every arc uv , by the triangle inequality. (If there is no such vertex s in the graph, add a new vertex s and connect it with zero-cost arcs to all other vertices.)

Thus we can compute minimum energetic paths from k sources in $O(m + n \log n)$ time per source plus the time to solve one standard single-source shortest path problem with the given arc costs. The extra time needed for this preprocessing is $O(mn)$ if we use Bellman-Ford, or $O(m \log^2 n \log(nW) \log \log n)$ if all arc costs are integral and we use the algorithm of Bringmann et al. [8]. (The faster algorithm is helpful only if $k = \Omega(\log^2 n \log(nW) \log \log n)$, where k is the number of sources.)

► **Corollary 4.3.** *The all-pairs minimum energetic paths problem on a graph $G = (V, A, c)$ with no negative cycles can be solved in $O(mn + n^2 \log n)$ time.*

The resulting all-pairs algorithm is very similar to Johnson's [22] algorithm for the standard all-pairs shortest paths problem.

5 Finding minimum paths in the presence of negative cycles

In this section we describe an algorithm e -Negative(G, B, s) that finds minimum energetic costs from a source vertex $s \in V$ to all other vertices in a directed graph $G = (V, A, c)$ that may contain negative cycles. It is not difficult to extend the algorithm to return a succinct description of the minimum energetic paths. We assume that s has no incoming arcs. (Incoming arcs of s are not useful as we start from s with a full battery.)

Algorithm e -Negative maintains a set R of reachable vertices that were already processed, a set of reachable vertices Q that are waiting to be processed, and a set $Y \subseteq R$ of exits. Initially $R = \emptyset$, $Q = \{s\}$ and $Y = \emptyset$. Let $G_{R,Y}$ be the graph obtained from G by removing the outgoing arcs of vertices not in R , removing the incoming arcs of vertices in Y , and for every $y \in Y$, adding a 0-cost arc sy . (Note that vertices in $V \setminus R$ may have incoming arcs in $G_{R,Y}$, but no outgoing arcs.) We may assume that the vertex set of $G_{R,Y}$ is $V_{R,Y} = R \cup N(R)$, where $N(R)$ are the out-neighbors of the vertices of R . We also have $Q \subseteq N(R)$. The algorithm maintains the invariant that $G_{R,Y}$ has no negative cycles and $p : V_{R,Y} \rightarrow \mathbb{R}$ is a valid potential function for it, and that all vertices in $R \cup Q$ can be reached when starting from s with a full battery.

The algorithm is composed of rounds. In each round the algorithm removes a vertex $u \in Q$ and processes it, i.e., adds it to the set R . If $G_{R \cup \{u\}, Y}$ does not contain negative cycles, all we need to do is find a valid potential function for $G_{R \cup \{u\}, Y}$ and update the set Q of vertices reachable in $G_{R \cup \{u\}, Y}$ when starting from s with a full battery.

To check whether $G_{R \cup \{u\}, Y}$ contains a negative cycle we construct a graph $\bar{G} = \bar{G}_{R,Y,u}$ as follows. The graph is obtained by starting from $G_{R,Y}$, adding a new source vertex \bar{u} , and for every $uv \in A$, adding an arc $\bar{u}v$ with $c(\bar{u}v) = c(uv)$. We also remove s and its outgoing arcs. (Note that $u \notin R$ has no outgoing arcs in \bar{G} .) The new graph \bar{G} does not contain negative cycles. The function p is a valid potential function for \bar{G} if we let

61:10 Optimal energetic paths for electric cars

$p(\bar{u}) = \max_{uv \in A} (p(v) - c(uv))$.² All negative cycles in $G_{R \cup \{u\}, Y}$ must pass through u . Thus, $G_{R \cup \{u\}, Y}$ contains a negative cycle if and only if $\delta_{\bar{G}}(\bar{u}, u) < 0$.

We thus run Dijkstra on \bar{G} starting from the source \bar{u} using the potential function p . If $\delta_{\bar{G}}(\bar{u}, u) < 0$ then a shortest path from \bar{u} to u becomes a negative cycle C in $G_{R \cup \{u\}, Y}$ when we replace \bar{u} with u . (Any path of negative cost from \bar{u} to u will do. We can thus stop the algorithm as soon as such a path is discovered.) In $O(|C|)$ time we find an exit y on C and add it to Y . This removes the incoming arcs of y from $G_{R \cup \{u\}, Y}$ and adds a 0-cost arc sy . We update the graph \bar{G} accordingly, i.e., remove the incoming arcs of y , and run Dijkstra again. (Note that p is still a valid potential function for \bar{G} , since s is not included in it.)

When no more negative cycles are found, the graph $G_{R \cup \{u\}, Y'}$, where Y' is the new set of exits, does not contain negative cycles. We still need to find a valid potential function for it. To do this it is enough to find distances from s to all other vertices. Let $\delta(s, v)$ be the distance from s to v in $G_{R \cup \{u\}, Y'}$. Let $\delta'(s, v)$ be the distance from s to v in $G_{R, Y'}$. Finally, let $\delta''(\bar{u}, v)$ be the distance from \bar{u} to v in $\bar{G}_{R, Y', u}$. Clearly, $\delta(s, v) = \min\{\delta'(s, v), \delta'(s, u) + \delta''(\bar{u}, v)\}$, since each shortest path in $G_{R \cup \{u\}, Y'}$ either does not pass through u , in which case its length is $\delta'(s, v)$, or it does pass through u in which case its length is $\delta'(s, u) + \delta''(\bar{u}, v)$.

To find the distances $\delta'(s, v)$ we simply run Dijkstra on $G_{R, Y'}$ using the potential function p , after we set $p(s) = \max_{sv \in A} (p(v) - c(sv))$. The distances $\delta''(\bar{u}, v)$ were already computed. Thus we can easily compute the distances $\delta(s, v)$ in $G_{R \cup \{u\}, Y'}$ and then set $p(v) = \delta(s, v)$, for every $v \in V_{R \cup \{u\}, Y'}$.

Given the newly computed potential function p , we can now run *e-Dijkstra* on $G_{R \cup \{u\}, Y'}$ to compute energetic costs and the new set of vertices Q' that are reachable from s but are not in $R \cup \{u\}$. The algorithm then lets $R \leftarrow R \cup \{u\}$, $Y \leftarrow Y'$ and $Q \leftarrow Q'$. This completes the round. If $Q = \emptyset$ the algorithm terminates. Otherwise it proceeds to the next round, processing the next vertex from Q .

We claim that the energetic costs computed by the last *e-Dijkstra*, which are also the values returned by *e-Negative*, are exactly the energetic costs $\delta_B(s, v)$ in the input graph $G = (V, A, c)$. (If $v \notin R$, then $\delta_B(s, v) = \infty$.)

► **Theorem 5.1.** *Algorithm e-Negative(G, B, s) finds minimum energetic costs in a graph $G = (V, A, c)$ that may contain negative cycles when the capacity of the battery is B . Its running time is $O(mn + n^2 \log n)$.*

Proof. The correctness of the algorithm follows from the explanations above combined with a few simple observations. It follows by induction that at the beginning of each round $G_{R, Y}$ contains no negative cycles, p is a valid potential function for it, and all vertices in $R \cup Q$ can be reached when starting from s with a full battery.

In each round, the algorithm moves a vertex u from Q to R . To do this, it repeatedly finds negative cycles in the graph $G_{R \cup \{u\}, Y}$. All such negative cycles must pass through u . As explained, there is such a negative cycle in $G_{R \cup \{u\}, Y}$ if and only if $\delta_{\bar{G}}(\bar{u}, u) < 0$, where $\bar{G} = \bar{G}_{R, Y, u}$. A negative path from \bar{u} to u corresponds to a negative cycle C , since \bar{u} and u represent the same vertex. If a negative cycle C is found, an exit y on C is identified and added to Y . (Note that this removes arcs from \bar{G} since the incoming arcs of y are removed.) The arc sy is added to $G_{R \cup \{u\}, Y \cup \{y\}}$ but not to $\bar{G} = \bar{G}_{R, Y \cup \{y\}, u}$.

² This works as \bar{u} has no incoming arcs. Equivalently we can use the fact that Dijkstra's algorithm works correctly even if the reduced costs of some of the outgoing arcs of the source are negative, which follows as $p(\bar{u})$ does not really affect the running of the algorithm. (It only affects the key of \bar{u} when it is alone in the heap.)

We next argue that $\delta_B(s, y) = 0$, for every $y \in Y$. Indeed, each vertex y added to Y is an exit on a negative cycle C all whose vertices can be reached when starting from s with a full battery. In particular, the entry x on C corresponding to y can be reached, and by the definition of entry-exit pairs, y can be reached with full battery, i.e., $\delta_B(s, y) = 0$. This also justifies the addition of the 0-cost arc sy .

Let $\delta'_B(s, v)$ be the energetic costs computed by the last call to *e-Dijkstra* on the final $G_{R,Y}$. We assume that $\delta'(s, v) = \infty$ for every $v \notin R$. Let $\delta_B(s, v)$ be the energetic costs in the input graph G . It is easy to see that $\delta_B(s, v) \leq \delta'(s, v)$, for every $v \in V$. This follows as $G_{R,Y}$, without the arcs $\{sy \mid y \in Y\}$, is a subgraph of G , and the addition of the arcs $\{sy \mid y \in Y\}$, as argued, does not change the energetic costs.

We next show that when the algorithm terminates we have $\delta_B(s, v) = \delta'_B(s, v)$, for every $v \in V$. Suppose, for the sake of contradiction, that there is a vertex v for which $\delta_B(s, v) < \delta'_B(s, v)$. Let P be a minimum energetic path from s to v in G . If P passes through a vertex of Y , let y be the last vertex from Y on P and let P' be path composed of the arc sy followed by the portion of P from the last occurrence of y on P to v . Since $\delta_B(s, v) < \delta'_B(s, v)$, it follows that P' is not a path in $G_{R,Y}$. Thus, P' must contain a vertex not in R . Let u be the first vertex not in R on P' . It follows that $\delta'_B(s, u) < \infty$, since the portion of P' from s to u is also a path in $G_{R,Y}$. Since $u \notin R$, we must have $u \in Q$ and the algorithm should not have terminated.

The algorithm performs at most $2n$ calls to Dijkstra, since following each such call either an exit is added to Y or a vertex is added to R . The algorithm performs at most n calls to *e-Dijkstra*. Thus, the running time of the algorithm is $O(mn + n^2 \log n)$. ◀

6 A faster all-pairs algorithm for large batteries

In this section we obtain an $O(mn + n^2 \log n)$ -time algorithm for the all-pairs energetic cost problem, in graphs that may contain negative cycles, when the capacity of the battery B is sufficiently large relative to the arc costs in the graph. The initial charge b may be arbitrary. More specifically, we assume that $B \geq 3nM$, where $M = \max_{uv \in A} |c(uv)|$.

Recall that $\delta_{B,b}(s, t)$ is the energetic cost of getting from s to t , starting from s with a charge b , where $0 \leq b \leq B$, when the capacity of the battery is B . The energetic cost is the difference between the initial charge, in this case b , and the final charge. Recall that $\delta_{B,b}(s, t) \in [b - B, b]$. (In particular, when $b < B$, the energetic cost may be negative.)

We started Section 2 with a simple reduction from the computation of $\delta_{B,b}(s, t)$ to that of $\delta_B(s, t) = \delta_{B,B}(s, t)$. The reduction introduces an arc of cost $B - b$, which may be much larger than M , so we cannot use it when the battery capacity is large. To make our results more general we work directly with $\delta_{B,b}(s, t)$.

The improved algorithm is obtained by using a preprocessing step, described in Section 6.1, that finds sets of entries or exits that *hit* all negative cycles. When the battery is large, such sets can be used to efficiently solve the single-source problem, from any source, in $O(m + n \log n)$ time.

We first describe, in Section 6.2, an efficient algorithm when $b \geq nM$. We then use this algorithm in Section 6.3 to obtain an efficient algorithm when $B \geq 3nM$ and b is arbitrary.

6.1 Finding sets of entries or exits that hit all negative cycles

A vertex x is an entry if it is an entry on some negative cycle. Similarly, a vertex y is an exit if it is an exit on some negative cycle.

61:12 Optimal energetic paths for electric cars

A set of vertices $Z \subseteq V$ is said to hit all negative cycles in a graph G if there are no negative cycles in the graph $G \setminus Z$, or equivalently in the graphs $G \setminus in(Z)$ or $G \setminus out(Z)$, where $G \setminus Z$ is the graph obtained by removing all the vertices of Z from G and $G \setminus in(Z)$ and $G \setminus out(Z)$ are the graphs obtained just by removing the incoming or outgoing arcs, respectively, of the vertices in Z .

We are interested in hitting all negative cycles with either a set of entries, or a set of exits. We show that this can be done efficiently.

► **Lemma 6.1.** *A set $Y \subseteq V$ of exits that hit all negative cycles in G , and a valid potential function p for $G \setminus in(Y)$, can be found in $O(mn + n^2 \log n)$ time.*

Proof. Add to G an auxiliary source vertex \bar{s} and connect it with 0-cost arcs to all other vertices of G . Running *e-Negative* of Section 5 on the resulting graph will construct a set Y of exits that hit all negative cycles in G and a valid potential function p . ◀

By a slight adaptation of the algorithm, i.e., finding an entry on each negative cycle found and removing its outgoing arcs, we can also get:

► **Lemma 6.2.** *A set $X \subseteq V$ of entries that hit all negative cycles in G , and a valid potential function p for $G \setminus out(X)$, can be found in $O(mn + n^2 \log n)$ time.*

6.2 An algorithm for large initial charges

We assume that $nM \leq b \leq B$, i.e., the initial charge is sufficient to traverse any path of at most n arcs. Let $u \rightsquigarrow v$ denote that there is a directed path from u to v in the graph.

Suppose that C is a negative cycle in G and that (x, y) is an entry-exit pair on it. Suppose that s is the current source. Since x and y are on a cycle we clearly have $s \rightsquigarrow x$ if and only if $s \rightsquigarrow y$. Furthermore, if $s \rightsquigarrow y$ then there is also a simple path from s to x , i.e., a path that uses at most $n - 1$ arcs. Since we start from s with a sufficiently large initial charge, we can reach x and eventually, by the definition of entry-exit pairs, reach y with a fully charged battery. This justifies the following algorithm.

Find a set Y of exits that hit all negative cycles and find a valid potential function p for $G \setminus in(Y)$. For any given source s , find the set $Y_s \subseteq Y$ of exits reachable from s . (This can be easily done in linear time.) Next, construct a graph G_s obtained from $G \setminus in(Y)$ by adding arcs of cost $b - B$ from s to every $y \in Y_s$. (These arcs ensure that we reach y with a fully charged battery without needing to use the incoming arcs of y .) Run *e-Dijkstra* on G_s using p , after suitably adjusting the potential of s . (We need a slight modification of *e-Dijkstra* that works when starting from s with an initial charge b . Alternatively, we can add an auxiliary source \bar{s} , add an arc $\bar{s}s$ of cost $B - b$, and run *e-Dijkstra* from \bar{s} .) This takes only $O(m + n \log n)$ time per vertex, giving an $O(mn + n^2 \log n)$ -time algorithm for the all-pairs problem.

6.3 An algorithm for large batteries

We now describe an algorithm for $3nM \leq B$, and any value of b . If $nM \leq b$, we can use the algorithm of the previous section. We can thus assume that $b \leq nM$ and thus $B - b \geq 2nM$, i.e., the battery is initially far from being fully charged.

Start by using the algorithm from Section 6.1 to find a set X of entries that hit all negative cycles of G .

Let P be a minimum energetic path from s to t . We may assume that all cycles on P , if any, are negative, since otherwise they can be removed without increasing the energetic cost.

If P does not pass through any vertex of X , then it is also a path in $G \setminus \text{out}(X)$. Otherwise, let $x \in X$ be the first entry appearing on P . Suppose that x is an entry of a negative cycle C and that y is a corresponding exit on C . In general, the path P may not pass through y . This is one of the main difficulties that algorithm *e-Negative* had to deal with. However, when $B - b \geq 2nM$, we show that there must exist a minimum energetic path P' from s to t that passes through both x and y . We can further assume that during the last visit of y , the battery is fully charged.

The path P must reach the first entry $x \in X$ after traversing at most $n - 1$ arcs. If P is simple, this is obviously true. Otherwise, a cycle C' must be formed after traversing at most n arcs. By definition, there is an entry $x \in C' \cap X$. (Note that x is not necessarily an entry of C' , but it is an entry of some negative cycle C with a corresponding exit y .)

If b_x is the charge in the battery when reaching the first entry x on P , then $B - b_x \geq nM$. (The additional charge gained by traversing the portion of P from s to x is at most nM .) Since x is an entry of C , we can traverse C and get back to x , passing y along the way, with a charge that is at least b_x . (Note that it is important here that the battery is not close to being fully charged when starting from x , otherwise the claim is not necessarily true.) This gives us the path P' .

This suggests the following algorithm. Start, as mentioned, by finding a set X of entries that hit all negative cycles and a potential function p for $G \setminus \text{out}(X)$.

For each vertex $s \in V$, run *e-Dijkstra* on $G \setminus \text{out}(X)$ starting from s using the potential function p . This finds minimum energetic paths from s that do not pass through X . It also finds the set X_s of entries that can be reached when starting from s with an initial charge of b . Let Y_s be the set of exits that correspond to the entries in X_s . Construct a new graph G_s as follows. Remove from G all the outgoing arcs of s . For each $y \in Y_s$ add a 0-arc from s to y . Now run the algorithm of Section 6.2, starting from s with a fully charged battery. (Assuming that a set Y of exits that hit all negative cycles of G was already precomputed.) Subtract $B - b$ from the results obtained, to adjust for the fact that the initial charge is actually b and not B , and take the minimum with the results returned by *e-Dijkstra* call.

The computation of the hitting sets X and Y takes $O(mn + n^2 \log n)$. For every source we need only $O(m + n \log n)$ time. The total time of the algorithm is thus $O(mn + n^2 \log n)$.

7 Minimum initial charges and maximum final charges

To end the paper, we consider two problems that are closely related to the minimum energetic paths problem. Let $G = (V, A, c)$ be a graph with no (traversable) negative cycles and let B be the capacity of the battery. For two vertices $s, t \in V$, we let $\alpha_B(s, t)$ be the *maximum final charge* with which it is possible to reach t when starting at s with a full battery, or $-\infty$, if it is not possible to travel from s to t . We also let $\beta_B(s, t)$ be the *minimum initial charge* required at s for getting to t , or ∞ , if no initial charge (of at most B) is sufficient.

The maximum final charge problem is not really a new problem as $\alpha_B(s, t) = B - \delta_B(s, t)$. As noted in the introduction, $\beta_B(s, t)$ is the smallest b such that $\delta_{B,b}(s, t) \leq b$, or equivalently $\delta_{B,b}(s, t) < \infty$, if there is such a b . Thus, if B and all arc costs are integral, then we can find $\beta_B(s, t)$, for a specific pair s and t , by a binary search.

There is, however, a more interesting relation between the minimum initial-charge problem and the minimum energetic cost problem. Namely, $\beta_B(s, t)$ is equal to $\delta_B^{\overleftarrow{G}}(t, s)$ the energetic cost of traveling from t to s in the *reversed graph* \overleftarrow{G} , the graph obtained by reversing all the arcs in the graph G and retaining all arc costs. This relation follows easily from the following lemma, analogous to Lemma 2.2, whose simple proof is omitted. For a path P from s to t ,

let $b_B(P)$ be the minimum initial charge at s with which the path P can be traversed.

► **Lemma 7.1.** *Let $P = u_0u_1 \dots u_k$ be a directed path, let $P' = u_1 \dots u_k$, and let $c_i = c(u_{i-1}u_i)$, for $i = 1, \dots, k$. Let B be the capacity of the battery. If $k = 0$ then $b_B(P) = 0$. If $k > 0$ then*

$$b_B(P) = c_1 \oplus_B b_B(P') = c_1 \oplus (c_2 \oplus (\dots \oplus (c_{k-1} \oplus (c_k \oplus 0)) \dots)).$$

► **Corollary 7.2.** *For every $s, t \in V$, $\beta_B(s, t) = \delta_B^{\overleftarrow{G}}(t, s)$.*

As immediate corollaries, it follows that we can solve the single-target version of the minimum initial-charge paths problem in $O(m+n \log n)$ time, if we are given a valid potential, and the all-pairs version of the problem in $O(mn + n^2 \log n)$.

8 Concluding remarks and open problems

We have presented a clear definition of the minimum energetic paths problem, which is a strict extension of the standard shortest paths problem, and explained its relation to two other related problems: minimum initial-charge paths and maximum final-charge paths. We have also presented efficient algorithms for the minimum energetic paths problem in three different settings.

When there are no negative cycles in the graph, the minimum energetic paths problem can be solved using relatively simple adaptations of the classical Bellman-Ford and Dijkstra algorithms. We present simple descriptions and simple correctness proofs of these algorithms. In particular, we have obtained an $O(mn)$ -time algorithm for the single-source version, when arc costs may be negative but there are no negative cycles, and an $O(mn + n^2 \log n)$ -time algorithm for the all-pairs version.

An interesting feature of the minimum energetic paths problem is that it is well defined even if the graph contains negative cycles. Furthermore, minimum energetic costs are always obtained by finite length, though not necessarily simple, minimum energetic paths. (This is not the case for the standard shortest paths problem.) Using new algorithmic techniques, we have obtained an $O(mn + n^2 \log n)$ -time algorithm for the single-source version of the problem. Our best algorithm for the all-pairs version runs the single-source algorithm from each vertex, yielding a running time of $O(mn^2 + n^3 \log n)$.

We have obtained a more efficient algorithm for the all-pairs version when the capacity of battery is sufficiently large, i.e., $B \geq 3nM$, where $M = \max_{uv \in A} |c(uv)|$ is the maximum absolute value of an arc cost. The running time of the improved algorithm is $O(mn + n^2 \log n)$.

The obvious open problems are whether any of our time bounds can be improved. In particular, is it possible to get an $O(mn)$ -time algorithm for the single-source version when the graph may contain negative cycles? Is there an $O(n^3)$ -time algorithm for the all-pairs version when the graph may contain negative cycles?

Another interesting problem is whether the new techniques of Bernstein et al. [5] and Bringmann et al. [8], or the older technique of Goldberg [19] can be used to obtain an improved algorithm for the single-source version of the minimum energetic paths problem, when negative cycles may be present in the graph.

Finally, as mentioned, the single-target version of the minimum energetic paths problem is not equivalent to the single-source version. Currently, our fastest algorithms for the single-target version actually solve the all-pairs version. Is there a faster solution? The fact that there may not be a tree of minimum energetic paths to a given target may indicate that the single-target version is harder than the single-source version.

References

- 1 Shaull Almagor, Nathann Cohen, Guillermo A. Pérez, Mahsa Shirmohammadi, and James Worrell. Coverability in 1-vass with disequality tests. In *Proc. of 31st CONCUR*, volume 171 of *LIPICs*, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CONCUR.2020.38.
- 2 Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. *KI*, 6359:309–316, 2010.
- 3 Moritz Baum, Julian Dibbelt, Thomas Pajor, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Energy-optimal routes for battery electric vehicles. *Algorithmica*, 82:1490–1546, 2020.
- 4 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.
- 5 Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 600–611. IEEE, 2022. doi:10.1109/FOCS54457.2022.00063.
- 6 Lubos Brim and Jakub Chaloupka. Using strategy improvement to stay alive. *Int. J. Found. Comput. Sci.*, 23(3):585–608, 2012. doi:10.1142/S0129054112400291.
- 7 Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal methods in system design*, 38(2):97–118, 2011.
- 8 Karl Bringmann, Alejandro Cassis, and Nick Fischer. Negative-weight single-source shortest paths in near-linear time: Now faster! *CoRR*, abs/2304.05279, 2023. arXiv:2304.05279, doi:10.48550/arXiv.2304.05279.
- 9 Shiri Chechik, Haim Kaplan, Mikkel Thorup, Or Zamir, and Uri Zwick. Bottleneck Paths and Trees and Deterministic Graphical Games. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 27:1–27:13, 2016.
- 10 Boris V. Cherkassky, Loukas Georgiadis, Andrew V. Goldberg, Robert E. Tarjan, and Renato F. Werneck. Shortest-path feasibility algorithms: An experimental evaluation. *ACM J. Exp. Algorithmics*, 14, 2009. doi:10.1145/1498698.1537602.
- 11 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.
- 12 Dani Dorfman, Haim Kaplan, and Uri Zwick. A faster deterministic exponential time algorithm for energy games and mean payoff games. In *Proc. of 46th ICALP*, volume 132 of *LIPICs*, pages 114:1–114:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.114.
- 13 Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proc. of 20th SODA*, pages 384–391, 2009.
- 14 Jochen Eisner, Stefan Funke, and Sabine Storandt. Optimal route planning for electric vehicles in large networks. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3637>.
- 15 Lester R. Ford. Network flow theory. Technical Report Paper P-923, RAND Corporation, Santa Monica, California, 1956.
- 16 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.
- 17 Harold N. Gabow and Robert E. Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417, 1988.
- 18 Judith F. Gilsinn and Christoph Witzgall. A performance comparison of labeling algorithms for calculating shortest path trees. Technical Report 772, US National Bureau of Standards, 1973.

- 19 Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM J. Comput.*, 24(3):494–504, 1995. doi:10.1137/S0097539792231179.
- 20 Thomas Dueholm Hansen, Haim Kaplan, Robert E. Tarjan, and Uri Zwick. Hollow heaps. *ACM Trans. Algorithms*, 13(3):42:1–42:27, 2017. doi:10.1145/3093240.
- 21 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.
- 22 Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977. doi:10.1145/321992.321993.
- 23 Samir Khuller, Azarakhsh Malekian, and Julián Mestre. To fill or not to fill: The gas station problem. *ACM Transactions on Algorithms (TALG)*, 7(3):1–16, 2011.
- 24 David A Klarner, editor. *The mathematical gardner*. Wadsworth International, 1982.
- 25 Donald E. Knuth. A generalization of Dijkstra’s algorithm. *Information Processing Letters*, 6(1):1–5, 1977. doi:10.1016/0020-0190(77)90002-3.
- 26 Marvin Künnemann, Filip Mazowiecki, Lia Schütze, Henry Sinclair-Banks, and Karol Wegrzycki. Coverability in VASS revisited: Improving rackoff’s bound to obtain conditional optimality. *CoRR*, abs/2305.01581, 2023. arXiv:2305.01581, doi:10.48550/arXiv.2305.01581.
- 27 Daniel J. Lehmann. Algebraic structures for transitive closure. *Theor. Comput. Sci.*, 4(1):59–76, 1977. doi:10.1016/0304-3975(77)90056-1.
- 28 László Lovász. *Combinatorial problems and exercises*, volume 361. American Mathematical Soc., 2007.
- 29 Omid Madani, Mikkel Thorup, and Uri Zwick. Discounted deterministic markov decision processes and discounted all-pairs shortest paths. *ACM Trans. Algorithms*, 6(2):33:1–33:25, 2010. doi:10.1145/1721837.1721849.
- 30 Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350, 2002. doi:10.25596/jalc-2002-321.
- 31 Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004. doi:10.1016/S0304-3975(03)00402-X.
- 32 Robert E. Tarjan. Shortest paths. Technical report, AT&T Bell Laboratories, 1981.
- 33 Robert E. Tarjan. A unified approach to path problems. *Journal of the ACM*, 28(3):577–593, 1981. doi:10.1145/322261.322272.
- 34 Robert E. Tarjan. *Data structures and network algorithms*. SIAM, 1983.
- 35 Virginia Vassilevska. Nondecreasing paths in a weighted graph or: how to optimally read a train schedule. In *Proc. of 19th SODA*, pages 465–472, 2008.
- 36 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All pairs bottleneck paths and max-min matrix products in truly subcubic time. *Theory Comput.*, 5(1):173–189, 2009. doi:10.4086/toc.2009.v005a009.
- 37 Peter Winkler. *Mathematical puzzles: a connoisseur’s collection*. A K Peters, 2004.
- 38 Uri Zwick. Exact and approximate distances in graphs - A survey. In *Proc. of 9th ESA*, volume 2161 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2001. doi:10.1007/3-540-44676-1_3.