Faster All-Pairs Optimal Electric Car Routing

Dani Dorfman
Max Planck Institute for Informatics, Saarbrücken, Germany

Haim Kaplan
Tel Aviv University, Israel

Robert E. Tarjan
Princeton University, NJ, USA

Mikkel Thorup
University of Copenhagen, Denmark

Uri Zwick
Tel Aviv University, Israel

Abstract

We present a randomized $\tilde{O}(n^{3.5})$-time algorithm for computing *optimal energetic paths* for an electric car between all pairs of vertices in an n-vertex directed graph with positive and negative *costs*, or *gains*, which are defined to be the negatives of the costs. The optimal energetic paths are finite and well-defined even if the graph contains negative-cost, or equivalently, positive-gain, cycles. This makes the problem much more challenging than standard shortest paths problems.

More specifically, for every two vertices s and t in the graph, the algorithm computes $\alpha_B(s,t)$, the maximum amount of charge the car can reach t with, if it starts at s with full battery, i.e., with charge B, where B is the capacity of the battery. The algorithm also outputs a concise description of the optimal energetic paths that achieve these values. In the presence of positive-gain cycles, optimal paths are not necessarily simple. For dense graphs, our new $\tilde{O}(n^{3.5})$ time algorithm improves on a previous $\tilde{O}(mn^2)$-time algorithm of Dorfman et al. [ESA 2023] for the problem.

The *gain* of an arc is the amount of charge added to the battery of the car when traversing the arc. The charge in the battery can never exceed the capacity B of the battery and can never be negative. An arc of positive gain may correspond, for example, to a downhill road segment, while an arc with a negative gain may correspond to an uphill segment. A positive-gain cycle, if one exists, can be used in certain cases to charge the battery to its capacity. This makes the problem more interesting and more challenging. As mentioned, optimal energetic paths are well-defined even in the presence of positive-gain cycles. Positive-gain cycles may arise when certain road segments have magnetic charging strips, or when the electric car has solar panels.

Combined with a result of Dorfman et al. [SOSA 2024], this also provides a randomized $\tilde{O}(n^{3.5})$-time algorithm for computing *minimum-cost paths* between all pairs of vertices in an n-vertex graph when the battery can be externally recharged, at varying costs, at intermediate vertices.

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases EV routing, Shortest Paths, Shortcuts, Sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.71

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2505.00728`

Funding *Dani Dorfman*: Partially supported by grant 2854/20 of the Israeli Science Foundation.
Haim Kaplan: Supported by the Israel Science Foundation grant no. 1156/23 and the Blavatnik Family Foundation.
Robert E. Tarjan: Partially supported by a gift from Microsoft.
Mikkel Thorup: Research supported by the VILLUM Foundation grant no. 54451.
Uri Zwick: Supported by grant 2854/20 of the Israeli Science Foundation.

Acknowledgements This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

Let $G = (V, A, c)$ be a weighted directed graph, where V is the set of vertices, $A \subseteq V \times V$ is the set of arcs, and where $c : A \to \mathbb{R}$ is a real-valued *cost* function defined on the arcs. The cost $c(uv)$ of an arc $uv \in A$ [1] is the amount of energy consumed when traversing the arc. Throughout most of this paper, it is more convenient to work with a gain function $g : A \to \mathbb{R}$ rather than a cost function. The *gain* $g(uv)$ of an arc $uv \in A$ is simply $g(uv) = -c(uv)$, i.e., the amount of energy gained by traversing the arc. The gain $g(uv)$ is negative if moving from u to v requires spending energy, or positive if energy is gained by moving from u to v.

A weighted directed graph $G = (V, A, g)$, where $g : A \to \mathbb{R}$ is a gain function, may be viewed as modeling a road network on which an electric car can roam. The electric car is assumed to have a battery of *capacity* B, where $B > 0$ is a parameter, i.e., it can store up to B units of energy. The *charge*, i.e., the amount of energy in the battery, can never be negative, and can never exceed the capacity of the 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 $b + g(uv) \geq 0$. If this condition holds, and the car traverses the arc, then it reaches v with a charge of $\min\{b + g(uv), B\}$. The car can traverse uv if $b + g(uv) > B$, but the battery does not charge beyond its capacity of B. The car can traverse a path if and only if it can sequentially traverse its arcs. Throughout most of the paper we assume that no external charging of the battery is allowed. The battery is only charged by traversing arcs with positive gain. We may assume that $g(uv) \in [-B, B]$, for every $uv \in A$, as arcs with $g(uv) < -B$ can never be used, and can thus be removed, and gains $g(uv) > B$ can be changed to $g(uv) = B$ without changing the problem. We consider the following two related natural questions:

1. Given two vertices $s, t \in V$, what is the *maximum final charge*, denoted $\alpha_B(s, t)$, with which the car can reach t if it starts at s with full battery, i.e., with a charge of B? If the car cannot reach t even with an initial charge of B at s, we let $\alpha_B(s, t) = -\infty$. More generally, we let $\alpha_b(s, t)$, where $0 \leq b \leq B$, be the maximum final charge with which the car can reach t if it starts at s with a charge of b.
2. Given two vertices $s, t \in V$, what is the *minimum initial charge* at s, denoted $\beta_0(s, t)$, that enables the car to reach t? If the car cannot reach t even with an initial charge of B at s, we let $\beta_0(s, t) = \infty$. More generally, we let $\beta_b(s, t)$, where $0 \leq b \leq B$, be the minimum initial charge at s required for reaching t with a charge of at least b.

It is not difficult to see, as shown in Dorfman et al. [5, Corollary 5.2], that $\beta_0(s, t) = B - \overleftarrow{\alpha}_B(t, s)$, where $\overleftarrow{\alpha}_B(t, s)$ denotes the maximum final charge at s when starting at t with full battery in the *reverse* of the graph. Thus, the problems of computing maximal final charges and minimum initial charges are computationally equivalent. (Note, however, that due to the reverse operation used, the single-source version of the maximum final charge problem becomes equivalent to the single-target version of the minimum initial charge problem.) In this paper, we only work with maximal final charges.

If all arc costs are nonnegative, i.e., all gains are nonpositive, then it is easy to see that $\beta_0(s, t) = \delta(s, t)$, and $\alpha_B(s, t) = B - \delta(s, t)$, if $\delta(s, t) \leq B$, where $\delta(s, t)$ is the standard distance from s to t with respect to the costs of the arcs. Otherwise, $\beta_0(s, t) = \infty$ and $\alpha_B(s, t) = -\infty$. When costs and gains can be both positive and negatives, the problem becomes more complicated. If there are no positive-gain cycles in the graph, the problem can

[1] For brevity we denote an arc from u to v by uv, rather than (u, v).

be solved using fairly simple adaptations of standard shortest paths algorithms. Thus, the single-source version of the maximal final charges problem can be solved in $O(mn)$ time using an adaptation of the classical Bellman-Ford algorithm [2, 7], and the all-pairs version of the problem can be solved in $O(mn + n^2 \log n)$ time by an adaptation of the classical algorithm of Johnson [9]. For these results see, Artmeier, Haselmayr, Leucker and Sachenbacher [1], Eisner, Funke and Storandt [6], Brim and Chaloupka [3], and Dorfman, Kaplan, Tarjan and Zwick [5].

The problem becomes much harder when the graph may contain positive-gain cycles. Part of the difficulty is that optimal paths, which are still well-defined, are not necessarily simple and might have to "hop" from one positive-gain cycle to another, until gaining enough charge to head directly to the destination. Hélouët et al. [8] obtained a polynomial time algorithm for the decision problem of determining whether $\beta_0(s,t) \leq B$. Dorfman et al. [5] obtained an $O(mn + n^2 \log n)$-time algorithm for the single-source version of the problem, which of course implies an $O(mn^2 + n^3 \log n)$-time algorithm for the all-pairs version.

Our main result is a randomized $\tilde{O}(n^{3.5})$-time[2] algorithm for solving the all-pairs versions of the maximal final charge problem, and hence also the minimum initial charge problem, improving by a $\Theta(\sqrt{n})$ factor for sufficiently dense graphs on the $O(mn^2 + n^3 \log n)$ running time of the algorithm of Dorfman et al. [5]. To appreciate our result, we draw a parallel to standard shortest paths. On a graph with n nodes and m edges (and a suitable potential function), the single source shortest path problem can be solved in $O(m+n \log n)$ time, leading to an $O(mn + n^2 \log n) = O(n^3)$ all pairs algorithm for dense graphs. A breakthrough result by Williams [11] achieved an $O(\frac{n^3}{2^{c\sqrt{\log n}}})$ time all pairs algorithm, shaving a subpolynomial factor for dense graphs.

All the discussion so far assumed that that battery cannot be recharged at intermediate vertices. A natural variant is obtained when we assume that the battery can be charged at some of the vertices of the graph, with a cost per unit of charge that may vary from vertex to vertex. The goal then, is to find *minimum-cost paths* within all pairs of vertices in the graph. This problem was considered by Khuller, Malekian and Mestre [10] in the context of conventional, gas-operated, cars, i.e., when all arc costs are positive, and by Dorfman, Kaplan, Tarjan, Thorup and Zwick [4] in the context of electric cars, i.e., when the costs, or gains, can be both positive and negative, and where there might be positive-gain cycles. The main result of Dorfman et al. [4] is a reduction from the all-pairs minimum-cost paths problem to the all-pairs maximal final charges and minimum initial charges problems, and to the standard all-pairs shortest paths problem. Combined with the results of Dorfman et al. [5], this implies an $O(mn + n^2 \log n)$-time algorithm for the all-pairs minimum-cost paths in graphs with no positive-gain cycles. Combined with our result, we obtain a randomized $\tilde{O}(n^{3.5})$-time algorithm for the all-pairs minimum-cost paths in graphs that may contain positive-gain cycles.

To obtain the improved algorithm we need to introduce many new ideas. We next try to give a rough intuitive description of some of them, ignoring some technicalities that will be dealt with later.

Optimal energetic paths can be very long. (Their length cannot be bounded as a function of n alone. A bound must also take the arc gains and the capacity of the battery into account. For more details, see [5].) A natural idea to reduce the length of optimal energetic paths is to introduce *shortcuts*, i.e., add new arcs that correspond to possibly long paths in the graph. In the standard shortest paths problem, any path in the graph can be used to generate a

[2] The $\tilde{O}(\cdot)$ hides logarithmic factors.

shortcut, with the gain (or cost) of the arc equal to the sum of the gains of the arcs on the path. This is far from being the case for energetic paths. Consider, for example, a path xyz with $g(xy) = -1$ and $g(yz) = 1$. We cannot add a new arc xz with $g(xz) = 0$ to the graph since an electric car with an empty battery would be able to traverse the new arc xz, but not the original path xyz.

Ignoring some technicalities, we can add a shortcut corresponding to a traversable path in the graph (a path that can be traversed if we start with full battery) if the path is *ascending* or *descending*. (See Figure 1(a)-(b).) Given a path $u_0u_1 \dots u_k$, let $a_i = \sum_{j=0}^{i-1} g(u_ju_{j+1})$, for $0 \leq j \leq k$, be the prefix sums of the gains along the path. We say that a path is *ascending* if $0 \leq a_i \leq a_k$, for every $1 \leq i \leq k$, and descending if $a_k \leq a_i \leq 0$, for every $1 \leq i \leq k$. If $u_0u_1 \dots u_k$ is ascending or descending, then we are allowed to add a shortcut u_0u_k with $g(u_0u_k) = a_k$. For brevity, we refer to ascending or descending paths as *monotone*.[3]

Unfortunately, most paths are not monotone. Furthermore, subpaths of monotone paths are not necessarily monotone. There may also be very long paths that do not contain any monotone subpath. We refer to such paths as *funnels*. Examples of funnels are given in Figure 1(c)-(f).

Our algorithm constructs monotone paths and funnels and combines them to obtain new monotone paths and funnels until enough information is available to find the optimal energetic paths. The exact details, some of which are quite delicate, appear in the rest of the paper.

Another idea used by our new algorithm is *sampling*. It is well known that a random set of vertices of size $(cn \log n)/k$ is likely to hit any given path of length at least k. Taking advantage of this fact in our context is again much more complicated.

The rest of this extended abstract consists of a technical review of our algorithm and main techniques.

2 Technical Review

A main tool in our algorithm is *shortcutting*. In the setting of standard shortest paths, any path $P = v_1 \dots v_k$ can be shortcutted to a single arc v_1v_k of gain $g(P) = \sum_{i=1}^{k-1} g(v_iv_{i+1})$ without affecting the lengths of the shortest paths. Unfortunately, because of the upper and lower bound constraints on the battery, this technique breaks down when applied to energetic paths. That is, by shortcutting arbitrary paths, we may change the optimal energetic paths. For example, assume $B = 10$ and let G be a graph that is composed of two paths $P_1 = v_1v_2v_3$ and $P_2 = u_1u_2u_3$, where $g(v_1v_2) = g(u_2u_3) = -5$ and $g(v_2v_3) = g(u_1u_2) = 5$. Observe that $\alpha_0(v_1, v_3) = -\infty$ and $\alpha_{10}(u_1, u_3) = 5$. On the other hand, by shortcutting the paths $v_1v_2v_3$ and $u_1u_2u_3$ (to arcs of gain 0) we will be able to reach v_3 from v_1 when starting with zero charge. Moreover by using the new 0 gain arc u_1u_3 the maximum final charge at u_3 (when starting with 10 charge at u_1) becomes 10.

The above discussion encourages us to find *safe* paths that can be shortcutted without affecting the optimal energetic paths (i.e., without affecting the α values). We call these paths *monotone paths*, see Figure 1. Monotone paths are either *ascending* or *descending*. An ascending path $P = v_1 \dots v_k$ is a *traversable* path that satisfies that whenever an electric car traverses P, the car has minimum charge at v_1 and maximum charge at v_k. A traversable path $P = v_1 \dots v_k$ is a path that does not contain any subpath $v_i \dots v_j$ of gain smaller than $-B$ (this is equivalent to saying that a car that starts at v_1 with full charge can traverse P

[3] Note that this does not imply that the prefix sums $a_1, a_2, \dots, a_k$ form a monotone sequence.

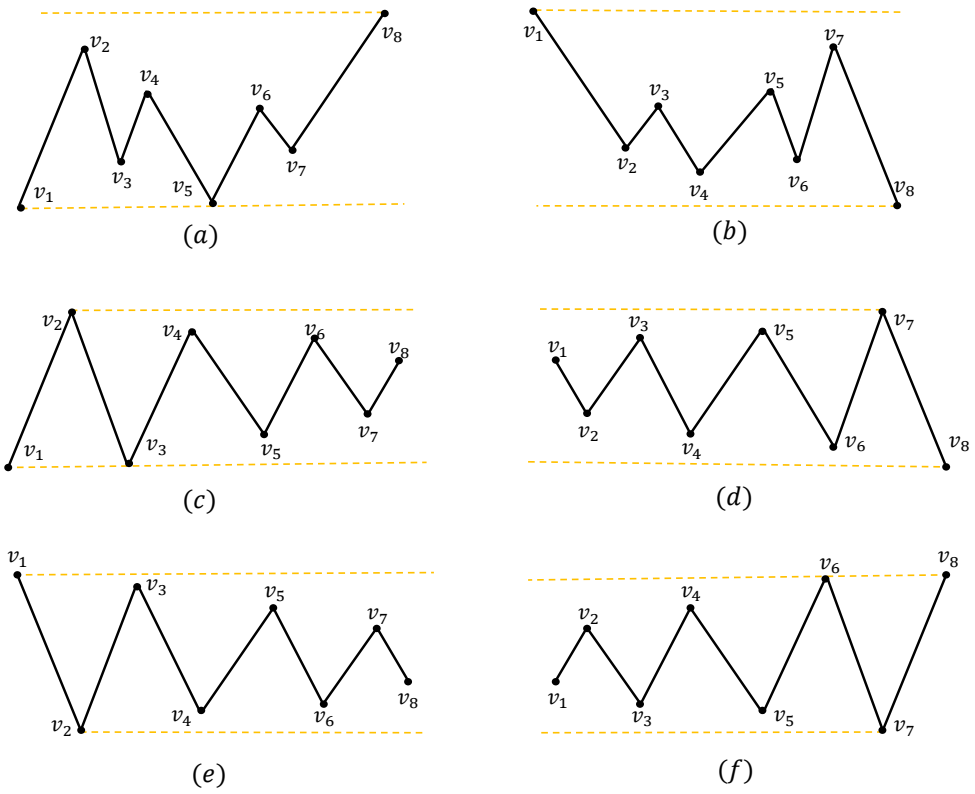

Figure 1 The graphs represent directed paths going from left to right. The vertical height of an arc e in the figure is $|g(e)|$. The vertical height of a vertex is its gain on the path (i.e. sum of arc gains). Figures (a) and (b) show an ascending path and a descending path, respectively. Figures (c)-(f) show the four possible cases for funnels. Note that in Figure (c), v_3 (which is the endpoint of the second arc of the funnel) has the same gain as v_1, this is valid.

without the charge level going below zero). Similarly, a descending path $P = v_1 \dots v_k$ is a traversable path that satisfies that whenever an electric car traverses P, the car has max charge at v_1 and minimum charge at v_k (in particular, the gain of a descending path is at least $-B$). A monotone path avoids the two problems mentioned in the previous example: The charge level of an ascending path never drops below the charge level at v_1 and therefore the path $v_1v_2v_3$ from the previous example cannot be shortcutted. Moreover, since the charge level remains below the charge level at v_k, shortcutting P does not create an alternative path from v_1 to v_k that improves the final charge at v_k, similarly to what happened with the path $u_1u_2u_3$ from the previous example.

In the full version of the paper (Theorem $F.1$), we prove that in $\tilde{O}(n^{3.5})$ time we can compute a 2-dimensional table $M[\cdot][\cdot]$ that *dominates* all simple monotone paths. That is, for every simple monotone path $P = v_1 \dots v_k$, it holds that $M[v_1][v_k] \geq g(P)$. Moreover, the table M is *sound*. That is, for every $u, v \in V$, if $M[u][v] \neq -\infty$, then there exists a monotone path P (not necessarily simple) from u to v such that $g(P) \geq M[u][v]$. Since monotone paths are traversable, it follows that if $M[u][v] \neq -\infty$, then $M[u][v] \geq -B$. Note that it is possible that $M[u][v] > B$. Once we have computed M, solving the all pairs $\alpha_B(\cdot, \cdot)$ problem is rather simple, we explain this derivation at the end of the technical review.

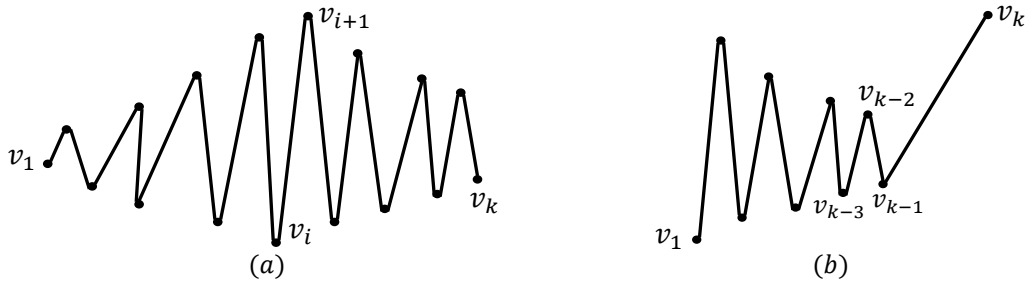

Figure 2 On the left: a double-funnel. On the right: worst case example for the simple algorithm. The depicted (directed) path $P = v_1 \dots v_k$ is monotone. Since $v_1 \dots v_{k-1}$ is a double-funnel, the only short monotone subpath of P is $v_{k-3}v_{k-2}v_{k-1}v_k$. Assume G is a path graph that contains only the path P. After the first iteration of shortcutting short monotone paths, we are left with the path $P_1 = v_1 \dots v_{v-3}v_k$ that has a similar structure to P. Thus, $\lfloor \frac{k}{2} \rfloor$ iterations are necessary in order to shortcut P into a single arc.

The following is a high level description of the computation of M. For simplicity, in this short review, we only describe how to dominate *ascending* paths. A simple observation is that every monotone path P contains a monotone subpath of edge-length 2 or 3. We call such a path a *short* monotone path. Thus, by shortcutting such a short monotone subpath into a single arc, we get an ascending path P' of smaller length than P and larger or equal gain than $g(P)$. This observation leads to a trivial $\tilde{O}(n^4)$ algorithm: Perform n iterations and generate a series of graphs $G_0 = G, G_1, \dots, G_n$. In the i'th iteration we find for every u, v the largest gain short monotone path from u to v in G_i. Once we have found all such gains, we build G_i by increasing the gains of every arc[4] (u, v) in G_{i-1} if there is a corresponding short monotone path from u to v of a better gain. We can implement each iteration in $\tilde{O}(n^3)$ time using a BST data structure. The table M stores the gains of the arcs of final graph G_n. Given a simple ascending path P in G_0, this process implicitly constructs a series of paths $P_i \in G_i$, where P_i is obtained from P_{i-1} by shortcutting as many short monotone paths as possible and P_n is a single arc.[5]

An immediate question is whether $\Theta(n)$ iterations are necessary. The answer is yes. The reason for this are *double-funnels* (see Figure 2(a)). A path $P = v_1 \dots v_k$ is a *double-funnel* if P does not contain a short monotone subpath. Double-funnels can have $\Theta(n)$ edges and an ascending monotone path which consists mainly of a long double-funnel would require $\Theta(n)$ iterations to be shortcutted into a single arc, see Figure 2(b).

As a consequence of the discussion above, in order to improve upon the simple algorithm, we need to handle double-funnels and reduce the number of iterations. A simple observation is that every ascending path can be viewed as an alternation between double-funnels (that are maximal with respect to inclusion) and short monotone paths, see Figure 3. Indeed, by the definition of a double-funnel, if we extend a double-funnel that is maximal with respect to inclusion by a single arc, the path ceases to be a double-funnel and therefore contains a short monotone path.

Let P be an ascending path such that P is not shortcutted to a single arc after $T = \sqrt{n}$ iterations of the simple algorithm. Let $P_0, \dots, P_T$ (paths in $G_0, \dots, G_T$, respectively) be the corresponding sequence of ascending paths as we defined before. For every $i = 0, \dots T$,

[4] We assume G_{i-1} is a full graph by adding arcs of gain $-\infty$.
[5] Note that P_i is not uniquely defined since short monotone paths may overlap. Moreover, we might perform shortcuts in P_{i-1} because of short monotone paths that appear in G_{i-1} and not in P_{i-1}.

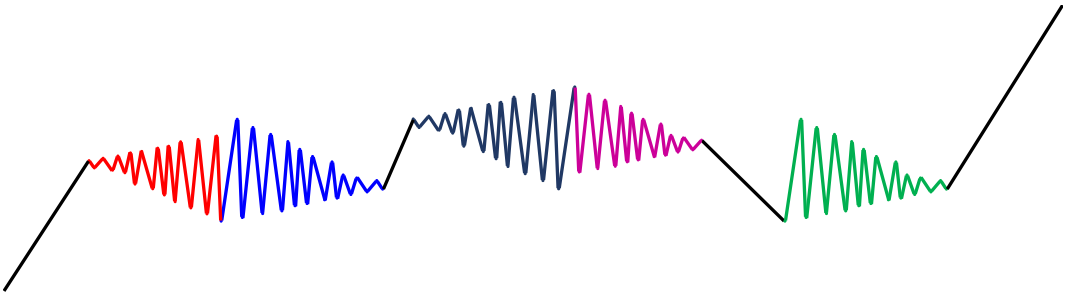

Figure 3 A decomposition of an ascending path to double-funnels that are maximal with respect to inclusion. Observe that "the gap" between two double-funnels contains a short-monotone path. The double-funnels are split into two funnels. Note that the green double-funnel is not maximal with respect to inclusion (it can be extend backwards by 2 arcs), this is done for aesthetic reasons to show "the gap" after the purple funnel.

denote by f_i the number of (maximal with respect to inclusion) double-funnels in P_i. By the interleaving property of double-funnels and short monotone paths, for every $i = 0, \ldots, T-1$, the number of short-monotone subpaths in P_i is at least f_i and therefore $|P_{i+1}| \leq |P_i| - f_i$ (where $|Q|$ denotes the number of arcs in a path Q). Since $P = P_0$ is a simple path (and thus of length at most $n-1$), and since we can uniquely charge a short monotone path that we shortcut at iteration i to each funnel in P_i it follows that $\sum_{i=1}^{T} f_i < n$, so the average number of funnels per iteration (of the T iterations that we consider) satisfies $\frac{1}{T}\sum_{i=1}^{T} f_i = \frac{1}{\sqrt{n}}\sum_{i=1}^{\sqrt{n}} f_i < \sqrt{n}$. By Markov's inequality, in at least $\frac{1}{2}T = \frac{1}{2}\sqrt{n}$ iterations, $f_i \leq 2\sqrt{n}$. Thus, in at least half of the T iterations, the paths P_i have $O(\sqrt{n})$ double-funnels. By sampling uniformly at random $\Theta(\log n)$ iterations, we are guaranteed to "hit" such an iteration w.h.p.. The final component of our algorithm is the procedure *Long-Shortcuts*(G_i), that, given a path P_i with $O(\sqrt{n})$ double-funnels, finds *long* shortcuts (i.e., shortcuts that correspond to monotone paths that could be of any length) in P_i, resulting in a path P_{i+1} that is shorter than P_i by a constant factor.

Based on the above discussion, our algorithm proceeds as follows. We perform $\tilde{\Theta}(\sqrt{n})$ iterations. In each iteration we find all short monotone path and shortcut them (this results in a modified graph with larger arc gains). Moreover, in each iteration, with probability $\tilde{\Theta}(\frac{1}{\sqrt{n}})$ we additionally call *Long-Shortcuts* which finds long monotone paths in the current graph, shortcuts them, and returns a modified graph.

We now describe the procedure *Long-Shortcuts*(G_i). We extensively use two path structures in *Long-Shortcuts*: *Arc-bounded* paths and *funnels*, see Figure 1. A path $P = v_1 \ldots v_k$ is first arc-bounded if for every $i = 2 \ldots, k$, it holds that $\sum_{j=1}^{i-1} g(v_j v_{j+1}) \leq \max\{0, g(v_1 v_2)\}$ and $\sum_{j=1}^{i-1} g(v_j v_{j+1}) \geq \min\{0, g(v_1 v_2)\}$. A last arc-bounded path is defined analogously. A path is arc-bounded if it is either first or last arc-bounded. A path P is a funnel if it is both arc-bounded and a double-funnel. Observe that any double-funnel can be decomposed to two funnels, each starts or ends at the edge of largest gain in absolute value, see Figure 3. Given the current graph G_i, *Long-Shortcuts*(G_i) stores a table $D[\cdot][\cdot]$ such that for every $u, v, w \in V$, $D[uv][w]$ stores the largest recorded gain of a first arc bounded path in G_i that starts with the arc uv and ends at w ($D[u][vw]$ is defined similarly for last arc-bounded paths). Algorithm *Long-Shortcuts* first generates arc-bounded paths (that is, stores values in the table D) and finally, finds long monotone paths based on those arc bounded paths. To ease the explanation, we begin by demonstrating the latter.

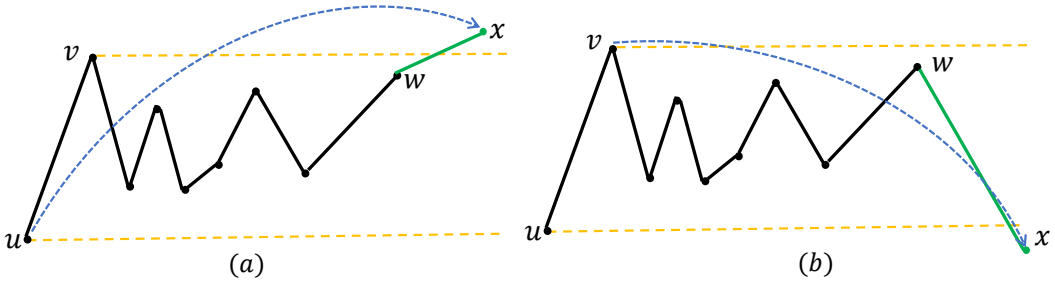

Figure 4 Finding a monotone path by extending an arc-bounded path by a single arc.

2.1 Generating monotone paths from arc-bounded paths

This part is straightforward: Given a vertex $u \in V$, we consider all arc-bounded paths that start at u and we extend each by a single arc: We scan all triplets $v, w, x \in V$, such that $D[uv][w] \neq -\infty$, and "concatenate" the arc-bounded path $P^{uv,w}$ that corresponds to $D[uv][w]$ with the arc wx, resulting in a path $P^{uv,x}$ to x that starts with uv.[6] Assume $g(uv) > 0$ (other cases are similar). If this concatenated path remains arc bounded then we did not find a monotone path. Otherwise, either $D[uv][w] + g(wx) > g(uv)$ or $D[uv][w] + g(wx) < 0$. It is easy to see that in the former case, $P^{uv,x}$ is ascending (see Figure 4(a)), and in the latter case the subpath from v to x is descending (see Figure 4(b)). It is easy to see that the running time of this process is $O(n^3)$.

2.2 Finding arc-bounded paths

As already discusses, any path can be viewed as an alternation between double-funnels (which are just two funnels that are concatenated) and short monotone paths. Thus, handling funnels has a crucial role.

We compute arc bounded paths using two building blocks.

1. A procedure *Compute-Funnels*(H) to compute funnels. Given a graph H, *Compute-Funnels*(H) returns a table $D[\cdot][\cdot]$ that dominates every funnel (which is a simple path) in H. That is, for every funnel $P = v_1 \dots v_k$ that is first arc-bounded, it holds that $D[v_1v_2][v_k] \geq g(P)$. Similarly, for every funnel $P = v_1 \dots v_k$ that is last arc-bounded, it holds that $D[v_1][v_{k-1}v_k] \geq g(P)$. Moreover, the table D is *sound*. That is, for every $u, v, w \in V$, if $D[uv][w] \neq -\infty$, then there exists a first arc-bounded path $Q = v_1 \dots v_k$ (not necessarily a funnel) such that $g(Q) \geq D[uv][w]$. The full details appear in the full version of the paper.
2. A concatenation procedure *Concatenate*(H, D, v). Given a graph H, a table $D[\cdot][\cdot]$ and a vertex $v \in V$. The procedure, in a brute force manner, scans all 4-tuples (w, x, y, z) of vertices and then tries to concatenate a first arc-bounded path in D that start with the arc vw and end at x with first arc-bounded path that start with the arc xy and end at z.[7] Note that this procedure only generates arc-bounded paths that start at v. For the full details, we refer the reader to the full version of the paper. A naive implementation of this procedure takes $O(n^4)$ time. Using a balanced binary search tree, we get a running time of $\tilde{O}(n^3)$. Since our claimed running time for the entire algorithm is $\tilde{O}(n^{3.5})$, we can use the *Concatenate* procedure only $\tilde{O}(n^{0.5})$ times.

[6] We do not actually store paths. Instead we examine the quantity $D[uv][w] + g(wx)$.

[7] Formally, we look on the quantity $D[vw][x] + D[xy][z]$ and verify some inequalities to make sure that the concatenated path is indeed first arc-bounded.

We now describe *Long-Shortcuts*(H) and the intuition about it. The algorithm starts by calling to *Compute-Funnels*(H), which in $\tilde{O}(n^{3\frac{1}{3}})$ time computes a table $D[\cdot][\cdot]$ that dominates all simple funnels in H. The algorithm then samples uniformly at random sets $S_i \subseteq V$ of size $\tilde{O}\left(\frac{\sqrt{n}}{2^i}\right)$, for $i = 1, \ldots, \log(\sqrt{n})$. Then, for every $i = 1, \ldots, \log(\sqrt{n})$ and $u \in S_i$ we perform 2^i times the procedure *Concatenate*(H, D, u). Finally, we extract monotone paths by applying the procedure from Section 2.1 on every vertex in $S = \cup_i S_i$.

We now give the intuition behind the algorithm. Recall the discussion about "hitting" an iteration in which $P_i = v_1 \ldots v_k$ (an ascending path in G_i, for some $0 \leq i \leq T = \sqrt{n}$, that represents the evolution of $P = P_0$ over the iterations of shortcutting) has at most $2\sqrt{n}$ double-funnels. Assume we run *Long-Shortcuts*(G_i). Every vertex $v_j \in P_i$ defines a first arc-bounded path $P' = v_j \ldots v_t$, where $j \leq t \leq k$ is maximal such that $v_j \ldots v_t$ is first arc-bounded, see Figure 5. Note that P' may contain several double-funnels, say f. Thus, if we apply *Concatenate*(G, D, v_j), $\Theta(f)$ times, the table D will "find" P' (that is we will have $D[v_j v_{j+1}][v_t] \geq g(P')$). By the discussion in Section 2.1, if we extend $v_j \ldots v_t$ by the arc $v_t v_{t+1}$ we will find a monotone path of length $t - j + O(1)$. For this process to be efficient, we have to balance the work we do (which is proportional to the number of funnels in P' which is the number of calls to concatenate that we need to do to find P') to compute P' with the reward we achieve (which is proportional to the length of P') by shortcutting the monotone path corresponding to P'.

We are shooting for a running time of $O(n^{3.5})$, therefore as we already said we can call concatenate at most $O(\sqrt{n})$ times (recall that it works for a single particular vertex at each call). In particular, for every $i = 1, \ldots, \log(\sqrt{n})$, the product of $|S_i|$ and the number of calls of concatenate from each vertex of S_i should be $O(\sqrt{n})$. To explain why we need the $O(\log(n))$ levels of sampling, we consider the two extreme cases which our sampling interpolates between. That is, the case of $i = \log(\sqrt{n})$ where $S_i = O(1)$ and the case of $i = 1$ where $|S_i| = O(\sqrt{n})$.

These two cases are demonstrated in Figure 5 for a path P_i of length $\Theta(n)$ and $\Theta(\sqrt{n})$ funnels. The first example ($i = \log(\sqrt{n})$), depicted in Figure 5(a), considers the case in which all funnels, except for the first one, are of constant length and the rest is filled with the first funnel which is of linear size. Moreover, the arc-bounded paths that correspond (in the manner explained in the previous paragraph) to every vertex in a short funnel are of constant length and the arc-bounded paths that correspond to vertices in the long funnel are all reaching the last arc of the path. Thus, in order to achieve sufficient reward (i.e., find long enough monotone paths), we have to sample a vertex u in the long funnel and then perform $\Theta(\sqrt{n})$ times *Concatenate*(G_i, D, u). Thus, the example shows that there are cases in which we have to perform $\Theta(\sqrt{n})$ concatenations at a single vertex.

The second extreme case, depicted in Figure 5(b), is the case in which all funnels are of length $\Theta(\sqrt{n})$ and for every $v \in P_i$, the arc-bounded path that corresponds to v contains a single funnel. Thus, for every $v \in P_i$ we can apply a single concatenation and find the arc-bounded path that corresponds to v and later extend it to a monotone path of length $O(\sqrt{n})$. In this case to reduce the length of P_i by a constant factor, we have to sample $\Theta(\sqrt{n})$ vertices (that will hit a constant fraction of the funnels) and perform a constant number of concatenation on each one of them.

2.3 Solving the all-pairs problem

Finally, we briefly describe the key observations that relate monotone paths to the computation of $\alpha_B(\cdot, \cdot)$. We begin by assuming that the optimal energetic paths are simple and later show how to solve the general case in which the optimal paths use positive cycles.

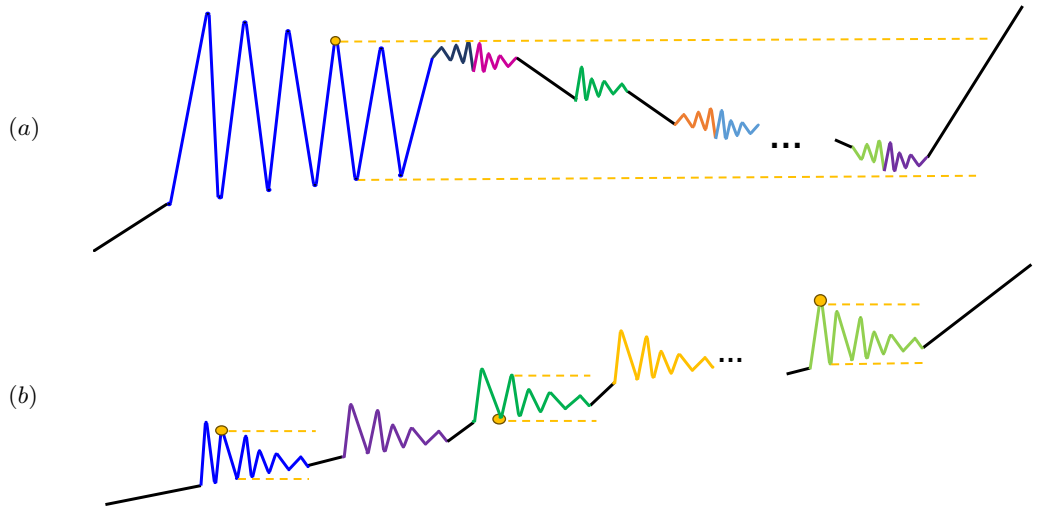

Figure 5 Two extreme cases for algorithm *Long-Shortcuts*. Black lines represent single arcs. Figure (a) shows why we need to sample $O(1)$ vertices but perform $\Theta(\sqrt{n})$ concatenations per vertex. Figure (b) shows why we need to sample $\Theta(\sqrt{n})$ vertices but perform $O(1)$ concatenations per vertex.

2.4 Simple energetic paths

Assume we have computed the table $M[\cdot][\cdot]$ that dominates every simple monotone path in G. Let $s, t \in V$ and let $P = v_1 \dots v_k$ be an optimal energetic path from $v_1 = s$ to $v_k = t$ (that is, $\alpha_B(s,t) = \alpha_B(P)$). We consider the special case in which P is simple and for every $1 < i \leq k$ it holds that $\alpha_B(v_1 \dots v_i) < B$. That is, the car starts with full charge at s and its charge level remains below B. We decompose P as follows. Let $v_{i_1} = s$ and let v_{i_2} be the vertex of lowest gain in P. We define v_{i_3} to be the vertex of highest gain in the suffix $v_{i_2} \dots v_k$ and so on, see Figure 6(a). This results in a series of vertices $s = v_{i_1}, v_{i_2}, \dots, v_{i_r} = t$. Clearly, this partitioning divides P into monotone segments that alternate between ascending and descending paths. A key observation is that these monotone paths are optimal in terms of gain. That is, for every $1 \leq j < r$, there is no monotone path Q from v_{i_j} to $v_{i_{j+1}}$ with larger gain than the subpath $v_{i_j} \dots v_{i_{j+1}}$. Otherwise, we can replace the subpath $v_{i_j} \dots v_{i_{j+1}}$ by Q and increase the final charge at t,[8] a contradiction to the optimality of P. Thus, for every $1 \leq j \leq r$, it holds that $M[v_{i_j}][v_{i_{j+1}}] = g(v_{i_j} \dots v_{i_{j+1}})$. Let G' be a directed clique whose gains are defined by $M[\cdot][\cdot]$. The final observation is that $v_{i_1} v_{i_2} \dots v_{i_r}$ is a funnel in G'. Thus, by calling *Compute-Funnels*(G') we can find this funnel.

2.5 Handling positive cycles

A simple observation is that every positive gain cycle C contains a pair of points $x, y \in C$ such that the car can start at x with zero charge, and traverse the cycle until it reaches y with a fully charged battery (i.e., B charge).[9] We say that (x, y) is an *entry-exit* pair of C, where x is the *entry* and y is the exit.

[8] We use here the fact that the battery is not full.
[9] It is possible that the car took the direct path in C from x to y, or it cycled through C several times.

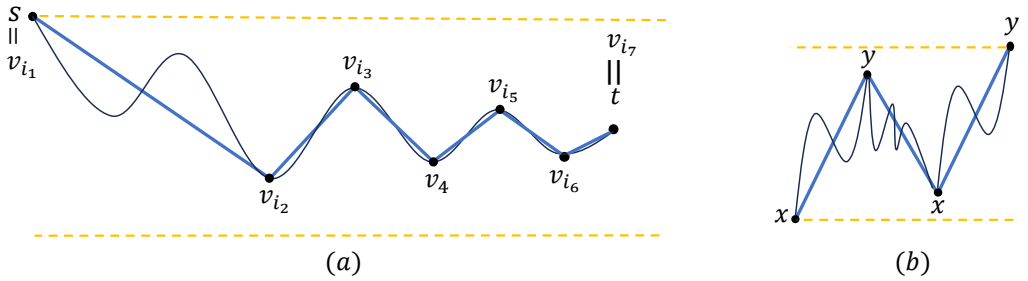

Figure 6 (a) A decomposition of an optimal path from s to t into a sequence of simple monotone paths. After shortcutting these paths, we are left with a funnel. (b) Illustration of why $M[x][y] + M[y][x] > 0$ & $M[x][y] > 0$ leads to $\alpha_0(x, y) = B$. Each blue arc represents a shortcut in M. Each such shortcut can be unwrapped into a path in G.

We prove in Lemma 6, that every traversable positive cycle C contains an entry-exit pair (x, y) such that C^{xy}, the path from x to y through C, is ascending and C^{yx}, the path from y to x through C, is descending.[10] This lemma, leads to a simple algorithm for identifying entry-exit pairs: For every $x, y \in V$, if $M[x][y] > 0$ and $M[x][y] + M[y][x] > 0$, then set $\alpha_0(x, y) = B$ (i.e., (x, y) is an entry-exit pair). The positive shortcut $M[x][y]$ indicates that there is an ascending path P^{xy} from x to y. If $M[x][y] \geq B$ then clearly we can start at x with zero charge and get to y with full charge (by using the shortcut[11] xy of gain $M[x][y]$). Otherwise, the second inequality $M[x][y] + M[y][x] > 0$ guarantees that we can start at x with zero charge and get back to x with positive charge (by using the shortcuts xy and yx). Therefore, by extending the path to y, we generate an ascending path with larger gain $M[x][y] + M[y][x] + M[x][y] > M[x][y]$, see Figure 6(*b*). By repeating this multiple times, we get an ascending path from x to y with gain larger than B justifying setting $\alpha_0(x, y) = B$.

We perform 3 additional simple inferences: For every $x, y, z \in V$

- If $M[x][y] + M[y][z] \geq 0$ and $M[x][y] \geq 0$, we deduce that the path that consists of the two shortcuts $M[x][y], M[y][z]$ is a witness that $\alpha_0(x, z) \geq 0$. That is, it is possible to start at x with zero charge and reach z: Either $M[x][y] \geq B$ and then the claim follows by the traversability of monotone paths ($M[y][z] \geq -B$) or $M[x][y] < B$ and therefore either $M[y][z] \geq 0$ or $-M[x][y] \leq M[y][z] < 0$. The former case is trivial. In the latter case, we can start with zero charge at x and reach y with $M[x][y]$ charge and then continue to z and reach it with $M[x][y] + M[y][z] \geq 0$ charge.
- If $M[x][y] + M[y][z] \geq 0$ and $M[y][z] \geq 0$, we deduce that $\alpha_B(x, z) = B$.
- If $M[x][y] \neq -\infty$ (so $M[x][y] \geq -B$), we infer that $\alpha_B(x, y) \geq 0$. That is, it is possible to reach y if we start at x with full charge.

Finally, we combine these relations into a graph H and compute its transitive closure $H^\star$. The graph H is defined as follows. $H = (V^0 \cup V^B, E(H))$, where $V^0 = \{v^0 \mid v \in V\}$ and $V^B = \{v^B \mid v \in V\}$ are two copies of V. Each vertex $v^0 \in V^0$ represents being at v with 0 charge and each vertex $v^B \in V^B$ represents being at v with full charge. An arc $u^{b_1}v^{b_2} \in E(H)$ represents that $\alpha_{b_1}(u, v) \geq b_2$.[12] We create the arcs $E(H) \subseteq \{u^{b_1}v^{b_2} \mid \alpha_{b_1}(u, v) \geq b_2\}$

[10] It is possible that $x = y$. For example in a cycle in which all arc gains are positive.
[11] Recall that using shortcuts does not change the α values since each shortcut corresponds to a monotone path in G of the same gain.
[12] Note that the other direction does not necessarily hold: It is possible that $\alpha_{b_1}(u, v) \geq b_2$ but $u^{b_1}v^{b_2} \notin E(H)$.

according to the 4 relations shown above (for example, if $M[x][y] \neq -\infty$, we add the arc $x^B y^0$ to H). In the full version of the paper (Theorem $H.12$), we claim that, w.h.p., for every $s, t \in V$, $\alpha_B(s,t) = B$ if and only if $s^B t^B \in E(H^\star)$.

Using the graph $H^\star$, our algorithm reduces the all pairs $\alpha_B(\cdot,\cdot)$ problem to the case in which the energetic paths are simple: For every $s, t \in V$, using $H^\star$, we find all vertices $x \in V$ such that $\alpha_B(s,x) = B$ and then, as in Section 2.4, we find the best energetic simple path from any such x to t.

The following is a brief review of the correctness of the algorithm. Let $s, t \in V$ and let $P = v_1 \dots v_k$ be an optimal energetic path from s to t (i.e., $\alpha_B(s,t) = \alpha_B(P)$). We argue that there is a vertex x on P such that $\alpha_B(s,x) = B$ and $\alpha_B(x,t) = \alpha_B(s,t)$. If $\alpha_B(s,t) = B$, then we are done since this relation is already recorded in $H^\star$ and we can set $x = t$. Otherwise, let $1 \leq i \leq k$ be maximal such that $\alpha_B(v_1 \dots v_i) = B$. It follows that $\alpha_B(s, v_i) = B$ and for every $i < j \leq k$ it holds that $\alpha_B(v_1 \dots v_i) < B$. This implies that $v_i \dots v_k$ must be a simple path.[13] So we conclude that the algorithm finds the optimal energetic path when inspecting $x = v_i$.

2.6 A technicality - charge drop schedules

In this section we describe *Charge drop schedules* and the technical challenge that it addresses. Before we delve into the definition, we motivate it by pinpointing several problems with our arguments.

1. Throughout this section we explained how to shortcut an ascending path to single arc via a sequence of short/long shortcut updates. A key invariant that is required for this argument to hold is the fact that given an ascending path $P = v_1 \dots v_k$, if we replace a monotone subpath $v_i \dots v_j$ of P by a monotone path Q of larger gain, then the resulting path $P' = v_1 \dots v_i \mid Q \mid v_j \dots v_k$ (The $\mid$ stands for concatenation) is ascending and $g(P') > g(P)$. Unfortunately, this argument does not hold if P is descending. For example, consider the graph G in Figure 7(a) and the descending path $P = v_1 v_2 v_3 v_4 v_5$. After performing one iteration of the simple algorithm (computing all short monotone paths and updating the gains of the graph), we are left with a graph G' with gain function g' (see Figure 7(b)) that does not contain any monotone path from v_1 to v_5. This is of course unsettling, as finding the best short shortcuts should be a good property of the algorithm and yet it destroyed some other descending paths
2. Recall the procedure *Concatenate*(G, D, v) that scans all 4-tuples (w, x, y, z) of vertices and then tries to concatenate a first arc-bounded path (stored in D) that starts with the arc vw and ends at x with first arc-bounded path that starts with the arc xy and ends at z (which is done by calculating $D[vw][x]+D[xy][z]$ and verifying some inequalities). Consider the following example: Assume $g(vw) = 5, g(xy) = 3$ and $D[vw][x] = 2, D[xy][z] = 3$. Therefore, by running *Concatenate*(G, D, v), we will concatenate the arc-bounded paths corresponding to $D[vw][x]$ and $D[xy][z]$ and get an arc bounded path that starts at vw and ends at z with gain $D[vw][z] = D[vw][x] + D[xy][z] = 5$. Unfortunately, this concatenation is not guaranteed to happen. It is possible that earlier in the run of *Concatenate*(G, D, v), the algorithm managed to improve $D[vw][x]$ to $D[vw][x] = 3$ and therefore concatenating $D[vw][x]$ to the arc-bounded path corresponding to $D[xy][z]$ does

[13] Otherwise, $v_i \dots v_k$ contains a positive cycle, so by repeating the cycle (and using the fact that no vertex on cycle, and the rest of the path, has already reached full charge) we can increase the final charge at $v_k = t$, a contradiction.

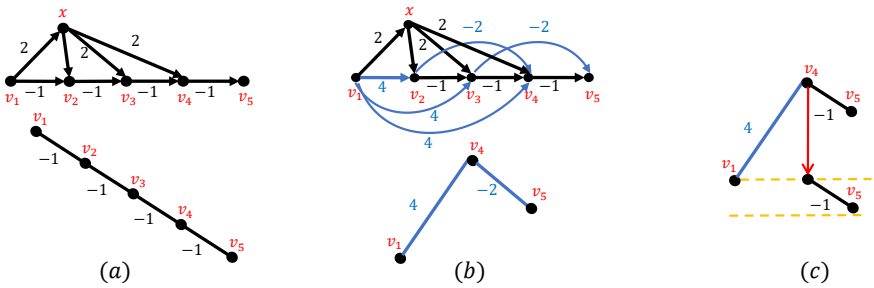

Figure 7 (a) The graph G and the descending path $P = v_1v_2v_3v_4v_5$. (b) The graph G' that we get after shortcutting all short monotone paths. Blue arcs correspond to either new arcs or arcs with increased gain. Note that there is no monotone path from v_1 to v_5 in G'. (c) By using charge drop schedule, we can transform the path $v_1v_3v_5$ into a short descending path of gain -2.

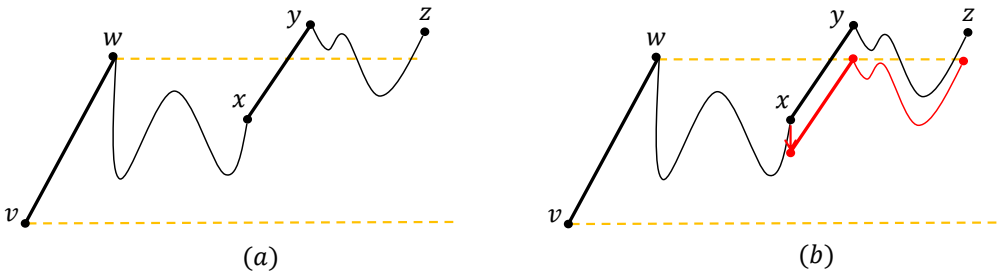

Figure 8 A use case of charge drops. (a) Two arc-bounded paths whose concatenation is not arc-bounded. (b) By applying a simple charge-drop schedule (that drops charge at x) we make the concatenated path arc-bounded.

not result anymore in an arc-bounded path, see Figure 8(a). Again, by performing an update that should be good for us (increasing $D[vw][x]$ from 2 to 3), we hurt ourself somewhere else (we did not make the update $D[vw][z] = 5$).

In both examples, we suffered from having computed values that are "too good". The simple concept that solves this problem is *charge drop*. Charge drops allow us, at any vertex along the path, to get rid of some charge, see Figure 8(b). Formally, let $P = v_1 \dots v_k$ be a path in G. A charge drop schedule is a vector $C = (d_1, d_2, \dots, d_k) \in \mathbb{R}^k_{\geq 0}$, where $d_1 = 0$. The *gain* at v_i with respect to P and C, denoted as $g^{P,C}_{v_i}$ is defined as $g^{P,C}_{v_i} = \sum_{t=1}^{i-1} g(v_tv_{t+1}) - \sum_{t=2}^{i} d_t$, for $2 \leq i \leq k$ and $g_{v_1} = 0$ otherwise. Monotone paths and arc bounded paths can be defined similarly to before by replacing the gain of an arc $g(v_iv_{i+1})$ by $g(v_iv_{i+1}) - d_{i+1}$. When P is clear from contexts, we abbreviate $g^{P,0}_{v_i}$ and write g_{v_i}.

We now show how to fix the two examples using charge drop schedules.

1. In the first example (see Figure 7) $P = v_1v_2v_3v_4v_5$ is a descending path in G, but there is no descending (or ascending) path from v_1 to v_5 in G'. Instead, G' contains the path $v_1v_4v_5$ that has positive gain. By using a simple charge drop schedule that drops 4 units of charge at v_4, we view $v_1v_4v_5$ as a short descending path of gain -2, see Figure 7(c).
2. In the second example we faced a problem when trying to concatenate an arc-bounded path corresponding to $g(vw) = 5, D[vw][x] = 3$ and an arc bounded path corresponding to $g(xy) = 3, D[xy][z] = 3$. By simply dropping a single unit of charge at x (the concatenation point), we are now able to concatenate the two paths and therefore assign $D[vw][z] = (D[vw][x] - 1) + D[xy][z] = 5$, see Figure 8.

We incorporate charge drops in our algorithm in the following places.

1. When computing all short monotone paths, if a path P (of length 2 or 3) starts by a negative gain arc, we will always apply charge drop schedule and create a descending path out of P. For example, if $P = v_1v_2v_3v_4$ and $g(v_1v_2) = -5, g(v_2v_3) = 2, g(v_3v_4) = -1$, then we record a descending path from v_1 to v_4 of gain -5 (this corresponds to dropping one unit of charge at v_4).
2. In the computation of long monotone paths. Recall that we consider tuples $u, v, w, x \in V$ and we extend the arc-bounded path that corresponds to $D[uv][w]$ by the arc wx. We incorporate charge drops in the following case: If $g(uv) < 0$ and $D[uv][w] + g(wx) \in [g(uv), 0]$ (that is the concatenated path remains arc-bounded), we record a descending path from u to x of gain $g(uv)$. This corresponds to performing a charge drop at x that drops $D[uv][w] + g(wx) - g(uv)$ charge.
3. In the concatenation procedure, whenever the concatenation of the two arc bounded paths does not yield an arc-bounded path, we perform a charge drop to force the result to be arc-bounded. That is, for every $v, w, x, y, z \in V$, if $g(vw) > g(xy) > 0$ and $D[vw][x] + D[xy][z] > g(vw)$, we set $D[vw][z] = g(vw)$. This corresponds to performing the smallest possible charge drop at x such that the concatenated path is arc-bounded, see Figure 8(b).

2.7 Main technical lemma

In this section, we prove Lemma 1, a simplified version[14] of our main lemma. Recall our algorithm: We perform $\tilde{\Theta}(\sqrt{n})$ iterations. In each iteration we find all short monotone path and shortcut them (this results in a modified graph with larger arc gains). Moreover, in each iteration, with probability $\tilde{\Theta}(\frac{1}{\sqrt{n}})$ we additionally call *Long-Shortcuts* which finds long monotone paths in the current graph, shortcuts them, and returns a modified graph.

▶ **Lemma 1.** *Let $P = v_1 \dots v_k$ be a simple ascending path in G. Let G' be the modified graph after $\sqrt{n}$ iterations of the modified algorithm and let g' be its gain function. If $|P| \le \sqrt{n}$, then $g'(v_1v_k) \ge g(P)$. If $|P| > \sqrt{n}$, then w.h.p. there is an ascending path P' in G' from v_1 to v_k in that satisfies $g'(P') \ge g(P)$ and $|P'| \le (1 - 1/\Omega(\log n)) \cdot |P|$.*

Lemma 1 is derived from Lemma 2, which is our main technical lemma. It provides guarantees about *Long-Shortcuts*, when run on a graph with an ascending path that contains few double-funnels.

▶ **Lemma 2.** *Let $P = e_1 \dots e_k$ be a simple ascending path in G from x to y. Let $t(\ge 1)$ be the number of double-funnels in P that are maximal with respect to inclusion. Let G' be the updated graph resulted from Long-Shortcuts(G).[15] If $t \le k/\sqrt{n}$, then w.h.p. there is an ascending path P' in G' from x to y that satisfies $g^{G'}(P') \ge g^G(P)$ and $|P'| \le (1 - 1/\Omega(\log n)) \cdot |P|$.*

We prove Lemma 2 at the end of this section. The derivation of Lemma 1 is now straightforward.

Proof of Lemma 1. Let $r = \sqrt{n}$ and let $G_0(= G), G_1, \dots, G_r$ be the graphs throughout the r iterations of the algorithm. Let $P_0 = P, P_1, \dots, P_r$ be a series of monotone paths, where P_i is the shortest path in G_i from v_1 to v_k that has no smaller gain (with respect to G_i) than P_{i-1} (with respect to G_{i-1}). We split the proof into cases.

[14] Lemma 1 addresses only ascending paths.
[15] Note that every non empty path contains at least one double-funnel.

Case $|P| \leq r$. Since in each of the r rounds we compute all the short monotone paths, and since every monotone path contains a short monotone path, we get that for every $1 \leq i < r$, if $|P_i| > 1$ then $|P_{i+1}| < |P_i|$. Thus, $|P_r| = 1$ and the lemma follows.

Case $|P| > r$. If $P_r \leq |P|/2$, then we are done. Otherwise $P_r > |P|/2$ and therefore for at least $r/2$ indices $0 \leq i < r$, it holds that $|P_i| - |P_{i+1}| \leq |P|/r$. This mean that, for each such index i, P_i has at most $|P|/r$ disjoint short shortcuts as subpaths. Thus, by our arguments in the previous sections (see Figure 3), P_i contains $O(|P|/r) = O(|P_i|/r)$ double-funnels that are maximal with respect to inclusion. Therefore, w.h.p. we run *Long-Shortcuts*(G_i) at an iteration i such that P_i contains $O(|P_i|/r) = O(|P_i|/\sqrt{n})$ double-funnels. Hence, the conditions of Lemma 2 are satisfied and we are done. ◀

Before proving Lemma 2, we need to introduce the following structural definitions. These definitions allow us to measure how many applications of *Concatenate* are needed in order to dominate an arc bounded path.

▶ **Definition 3.** *Let $P = e_1 \dots e_k$ be a path in G. For every $1 \leq i \leq k$ we define $s^P(i) \geq i$ to be the maximal index such that $e_i \dots e_{s^P(i)}$ is first arc-bounded. When P is clear from the context, we abbreviate and write $s(i)$.*

▶ **Definition 4.** *Let $P = e_1 \dots e_k$ be a path in G. For every i, we define $f^P(i)$ as the number of first arc-bounded funnels in $e_i \dots e_{s(i)}$ that are maximal with respect to inclusion. When P is clear from context, we abbreviate and write $f(i)$.*

The following lemma proves that for every path $P = e_1 \dots e_k$, the set of paths $\{e_i \dots e_{s(i)} \mid 1 \leq i \leq k\}$ is laminar.

▶ **Lemma 5.** *Let $P = e_1 \dots e_k$ be a path in G, then the set of intervals $\{(i, s(i)) \mid 1 \leq i \leq k\}$ is laminar.*

Proof. Let $1 \leq i \leq k$ and let $j \in (i, s(i))$. We show $(j, s(j)) \subseteq (i, s(i))$ from which the lemma follows. Denote $e_i = (u, v)$ and $e_j = (x, y)$. Since $P_i = e_i \dots v_{s(i)}$ is e_i-bounded, we have $g_w \in [\min\{g_u, g_v\}, \max\{g_u, g_v\}]$ for every $w \in P_i$. In particular $[\min\{g_x, g_y\}, \max\{g_x, g_y\}] \subseteq [\min\{g_u, g_v\}, \max\{g_u, g_v\}]$.

Since $e_j \dots e_{s(j)}$ is e_j-bounded we get that $g_w \in [\min\{g_x, g_y\}, \max\{g_x, g_y\}] \subseteq [\min\{g_u, g_v\}, \max\{g_u, g_v\}]$ for every $w \in P_j = e_j \dots e_{s(j)}$. Therefore $e_i \dots e_{s(j)}$ is e_i-bounded, so by the maximality of $s(i)$ we get that $s(i) \geq s(j)$, and therefore $(j, s(j)) \subseteq (i, s(i))$. ◀

We are now ready to prove Lemma 2.

Proof of Lemma 2. Let $F_1, \dots F_t$ be the disjoint double-funnels in P. By the discussion in Section 2, there are $O(t) = o(k)$ arcs in P that are not contained in the double-funnels (see Figure 3). Every double-funnel can be decomposed into at most 2 funnels (last arc-bounded followed by first arc-bounded). Let $F'_1, \dots, F'_{t'}$, where $t \leq t' \leq 2t$, be the corresponding funnels. We distinguish between funnels that are first-arc bounded to those which are last-arc bounded. Assume that the majority of the arcs of P belong to first-arc bounded funnels. The analysis for the other case is symmetric. Therefore, these funnels (first-arc bounded) contain at least $k/3$ arcs.[16] Among these funnels, we consider only funnels of length at least

[16] The choice of 3 and not 2 is due to the subtlety that the disjoint double-funnels do not necessarily cover all of P.

$\sqrt{n}/6$. Note that at least $k/6$ arcs belong to such funnels (if more than $k/6$ arcs belong to funnels of length at most $\sqrt{n}/6$ then we need at least $t > k/\sqrt{n}$ funnels to accommodate them, a contradiction). Denote these arcs by $e_{i_1}, \ldots e_{i_r}$ ($r \geq k/6$).

By Lemma 5, the set $A = \{(i_j, s(i_j)) \mid 1 \leq j \leq r\}$ is laminar. We refer to each item in A as an *interval*. Recall that each interval $(i_j, s(i_j))$ corresponds to a monotone path of the same length (A maximal arc bounded path extended by a single arc is monotone), see Section 2.1. Moreover, in order for *Long-Shortcuts* to shortcut the monotone path corresponding to $(i_j, s(i_j))$, *Long-Shortcuts* has to sample $v \in e_{i_j} = (v, w)$ and then perform $f(i_j)$ concatenations from v.

In the rest of the proof, we prove that *Long-Shortcuts* finds enough disjoint monotone paths of total length $\Omega(k/\log k)$. To this end, we partition A into disjoint sets $A_1, \ldots, A_{\log\sqrt{n}}$, where $A_i = \{(i_j, s(i_j)) \mid f(i_j) \in [2^i, 2^{i+1})\} \subseteq A$ correspond to all intervals/monotone paths that require $c \in [2^i, 2^{i+1})$ concatenations in order to be realized. We then prove that $A_{i^\star}$, the largest of these sets (hence of size $\Omega(k/\log n)$), contains a collection of disjoint *chains* (a chain is a set of nested intervals) $B'_1, \ldots, B'_{q'} \subseteq A_{i^\star}$ such that:

1. The chains are pairwise internally disjoint. That is, for every $1 \leq j_1 < j_2 \leq q'$ and $(\ell_1, r_1) \in B'_{j_1}$, $(\ell_2, r_2) \in B'_{j_2}$ it holds that $(\ell_1, r_1) \cap (\ell_2, r_2) = \emptyset$.
2. $|B'_j| = \Omega\left(\frac{\sqrt{n}2^{i^\star}}{\log n}\right)$, for $j = 1, \ldots, q'$. This property is crucial for the sampling to "hit" B'_j.
3. $|\bigcup_{i=1}^{q'} B'_i| = \Omega(|A_{i^\star}|) = \Omega(k/\log n)$.

Finally, by Property (2), we show that w.h.p., for every $j = 1, \ldots, q'$, *Long-Shortcuts* realizes an interval from B'_j whose length is at least $|B'_j|/2$. By combining these disjoint (Property (1)) shortcuts, we reduce the size of P by $\sum_{i=1}^{q'} |B'_i|/2 = \Omega(k/\log n)$.

We now show the lower bound on the size of A_{i^*} and prove that it contains a collection of chains $B'_1, \ldots, B'_{q'}$ that satisfy the above poperies. Since $i^\star$ is such that $|A_{i^\star}| \geq |A_i|$ for every $1 \leq i \leq \log\sqrt{n}$ and $|A| \geq k/6$ (by the laminarity of A each interval contains an edge which is not in any other interval) it follows that $|A_{i^\star}| \geq \frac{k}{6\log\sqrt{n}}$. Observe that for every $1 \leq i \leq \log\sqrt{n}$, A_i is laminar as a subset of A. Moreover, each interval in A_i cannot contain two disjoint intervals in A_i. Indeed, assume $(i_{j_1}, s(i_{j_1})), (i_{j_2}, s(i_{j_2})) \subseteq (i_{j_3}, s(i_{j_3}))$ and $(i_{j_1}, s(i_{j_1})) \cap (i_{j_2}, s(i_{j_2})) = \emptyset$, where all intervals belong to A_i. Therefore $f(i_{j_3}) \geq f(i_{j_1}) + f(i_{j_2}) \geq 2^i + 2^i = 2^{i+1}$, so $(i_{j_3}, s(i_{j_3})) \notin A_i$, a contradiction. It follows that we can decompose A_i (and in particular A_{i^*}) into a collection of internally disjoint chains.

Let $B_1, \ldots, B_q$ be the decomposition of $A_{i^\star}$ into internally disjoint chains ($A_{i^\star} = \cup_{i=1}^{q} B_i$). Since the B_i's are internally disjoint (and so are the funnels in them), $q \cdot 2^{i^\star} \leq t$. Let $A'_{i^\star}$ be the union of the B_i's that satisfy $|B_i| \geq \frac{k}{12q\log\sqrt{n}}$. It follows that

$$|A'_{i^\star}| \geq |A_{i^\star}| - q \cdot \frac{k}{12q\log\sqrt{n}} \geq \frac{k}{12\log\sqrt{n}}. \tag{1}$$

Let $B'_1, \ldots, B'_{q'}$ be the chains of $A'_{i^\star}$. Let $B'_j \subseteq A'_{i^\star}$, it holds that

$$|B'_j| \geq \frac{k}{12q\log\sqrt{n}} \overset{(1)}{\geq} \frac{k \cdot 2^{i^\star}}{12t\log\sqrt{n}} \overset{(2)}{\geq} \frac{\sqrt{n}2^{i^\star}}{12\log\sqrt{n}} = \Omega\left(\frac{\sqrt{n}2^{i^\star}}{\log n}\right),$$

where Inequality (1) follows since $q \cdot 2^{i^\star} \leq t$ and Inequality (2) follows since $t \leq k/\sqrt{n}$.

Recall that *Long-Shortcuts*(M) samples vertices to $S_{i^\star}$ i.i.d. with probability $p_{i^\star} = \Theta(\frac{\log^2 n}{2^{i^\star}\sqrt{n}})$. Since *Long-Shortcuts* performs $2^{i^\star}$ concatenations from every vertex in $S_{i^\star}$, every interval in $A_{i^\star}$ has a probability of $p_{i^\star}$ to be realized. Let $B'_j \subseteq A'_{i^\star}$. Since $|B'_j| = \Omega\left(\frac{\sqrt{n}2^{i^\star}}{\log n}\right)$, it follows by the Chernoff bound that w.h.p. we realize an interval from B'_j of length at least $0.5|B'_j|$.

Since $B'_1, \ldots, B'_{q'}$ are internally disjoint, then the above realized shortcuts (one from every B'_j) are also disjoint. Hence, by shortcutting the realized intervals we get an ascending path P' in G' of length.

$$|P'| \leq k - \sum_{j=1}^{q'} 0.5|B'_j| = k - 0.5|A'_{i^\star}| \overset{(1)}{\leq} k - 0.5\frac{k}{12\log\sqrt{n}}$$
$$= \left(1 - \Omega\left(\frac{1}{\log n}\right)\right) \cdot k = \left(1 - \Omega\left(\frac{1}{\log k}\right)\right) \cdot |P|,$$

where Inequality (1) follows from Equation (1) and the last equality holds because, according to the statement of the lemma, $\sqrt{n} \leq t\sqrt{n} \leq k < n$. ◀

3 Concluding remarks

We presented a randomized $\tilde{O}(n^{3.5})$-time algorithm for the finding optimal energetic paths between all-pairs of vertices in a weighted directed n-vertex graph with positive and negative gains that may contain positive-gain cycles. This improves upon a previous $\tilde{O}(mn^2)$-time algorithm by Dorfman et al. [5]. The new algorithm is quite involved and requires the introduction of many new ideas. Improving the running time of the algorithm is a natural open problem.

References

1 Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. *KI*, 6359:309–316, 2010. doi:10.1007/978-3-642-16111-7_35.

2 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

3 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.

4 Dani Dorfman, Haim Kaplan, Robert E. Tarjan, Mikkel Thorup, and Uri Zwick. Minimum-cost paths for electric cars. In *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 374–382. SIAM, 2024. doi:10.1137/1.9781611977936.34.

5 Dani Dorfman, Haim Kaplan, Robert Endre Tarjan, and Uri Zwick. Optimal energetic paths for electric cars. In *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, pages 42:1–42:17, 2023. doi:10.4230/LIPICS.ESA.2023.42.

6 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. doi:10.1609/AAAI.V25I1.7991.

7 Lester R. Ford. Network flow theory. Technical Report Paper P-923, RAND Corporation, Santa Monica, California, 1956.

8 Loïc Hélouët, Nicolas Markey, and Ritam Raha. Reachability games with relaxed energy constraints. *arXiv preprint arXiv:1909.07653*, 2019.

9 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.

10 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. doi:10.1145/1978782.1978791.

11 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

A Entry-Exit Pairs

In this short appendix we prove Lemma 6, which establishes an important property of traversable positive-gain cycles. Specifically, we show that any such cycle can be decomposed into an ascending path followed by a descending path.

▶ **Lemma 6.** *Let C be a traversable positive gain simple cycle in G. There exists an entry-exit pair (x, y) in C such that C^{xy}, the path from x to y through C, is ascending and C^{yx}, the path from y to x through C, is descending.*

Proof. Let (x', y') be an entry-exit of C. Consider the path P from x' to itself through C. Since C is traversable and x' is an entry, it follows that $\beta_0(P) = 0$, i.e., P can be traversed when starting at x' with zero initial charge. Let $y \in P$ be the vertex of maximum gain on P. Observe that the path from x' to y on C is ascending. Indeed the charge level cannot go below the initial charge at x' (which is zero) and the charge level at y is maximum.

If $y = x'$ then we are done. Otherwise, consider $P^{yx'}$, the simple path from y to x' through C, and let x be the vertex of minimum gain in $P^{yx'}$, see Figure 9. By the choice of x, P^{yx}, the path from y to x through C, is descending. We now show that $P^{xy} = P^{xx'}|P^{x'y}$ is ascending. Since x is of minimum gain in $P^{yx'}$, it follows that the gains of the vertices on $P^{xx'}$ are nonnegative. Moreover, since $P^{x'y}$ is ascending it follows that all gains on P^{xy} are nonnegative. We are left to show that y has maximum gain in P^{xy}. Since $P^{x'y}$ is ascending, it is enough to show that $(g_v^{P^{xx'}} =) g_v^{P^{xy}} \leq g_y^{P^{xy}}$ for every $v \in P^{xx'}$. Let $b = g_y^{P^{x'y}}$, it follows that $g_y^{P^{xy}} = g_{x'}^{P^{xx'}} + g_y^{P^{x'y}} \geq b$. We prove that $g_v^{P^{xx'}} \leq b$ for every $v \in P^{xx'}$. By contradiction, assume there is $v \in P^{xx'}$ such that $g_v^{P^{xx'}} > b$. Since all gains of vertices in P are nonnegative we get that $g_v^P = g_x^P + g_v^{P^{xx'}} > b = g_y^P$, a contradiction to the definition of y. ◀

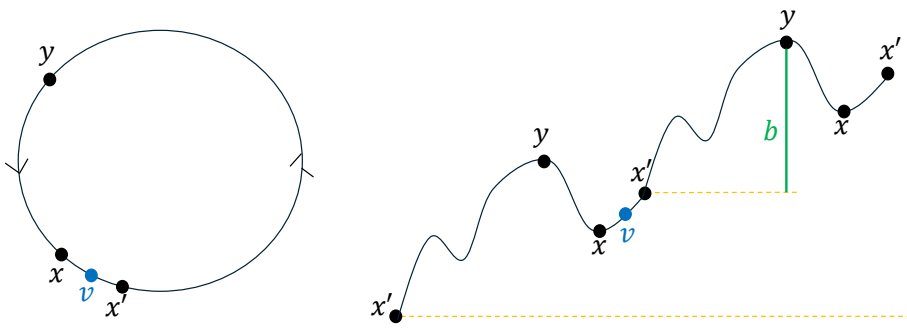

Figure 9 Illustration of Lemma 6. Note that y is of maximum gain in the path from x' to itself (through the cycle) and that x is of minimum gain on the subpath from y to x'. As shown in the proof of Lemma 6, the path from y to x is descending and the path from x to y is ascending.