# Dynamic Effective Resistances and Approximate Schur Complement on Separable Graphs

**Gramoz Goranci**
University of Vienna, Faculty of Computer Science, Vienna, Austria
gramoz.goranci@univie.ac.at

**Monika Henzinger**
University of Vienna, Faculty of Computer Science, Vienna, Austria
monika.henzinger@univie.ac.at

**Pan Peng**[1]
Department of Computer Science, University of Sheffield, Sheffield, UK
p.peng@sheffield.ac.uk

## Abstract

We consider the problem of dynamically maintaining (approximate) all-pairs effective resistances in separable graphs, which are those that admit an $n^c$-separator theorem for some $c < 1$. We give a fully dynamic algorithm that maintains $(1+\varepsilon)$-approximations of the all-pairs effective resistances of an $n$-vertex graph $G$ undergoing edge insertions and deletions with $\tilde{O}(\sqrt{n}/\varepsilon^2)$ worst-case update time and $\tilde{O}(\sqrt{n}/\varepsilon^2)$ worst-case query time, if $G$ is guaranteed to be $\sqrt{n}$-separable (i.e., it is taken from a class satisfying a $\sqrt{n}$-separator theorem) and its separator can be computed in $\tilde{O}(n)$ time. Our algorithm is built upon a dynamic algorithm for maintaining *approximate Schur complement* that approximately preserves pairwise effective resistances among a set of terminals for separable graphs, which might be of independent interest.

We complement our result by proving that for any two fixed vertices $s$ and $t$, no incremental or decremental algorithm can maintain the $s-t$ effective resistance for $\sqrt{n}$-separable graphs with worst-case update time $O(n^{1/2-\delta})$ and query time $O(n^{1-\delta})$ for any $\delta > 0$, unless the Online Matrix Vector Multiplication (OMv) conjecture is false.

We further show that for *general* graphs, no incremental or decremental algorithm can maintain the $s-t$ effective resistance problem with worst-case update time $O(n^{1-\delta})$ and query-time $O(n^{2-\delta})$ for any $\delta > 0$, unless the OMv conjecture is false.

**2012 ACM Subject Classification** Theory of computation → Dynamic graph algorithms

**Keywords and phrases** Dynamic graph algorithms, effective resistance, separable graphs, Schur complement, conditional lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2018.40

**Related Version** A full version of the paper is available at [17], `https://arxiv.org/abs/1802.09111`.

**Funding** The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

[1] Work done in part while at the Faculty of Computer Science, University of Vienna, Austria

## 1 Introduction

Effective resistances and the closely related electrical flows are basic concepts for resistor networks [12] and were found to be very useful in the design of graph algorithms, e.g., for computing and approximating maximum flow [8, 36, 37], random spanning tree generation [38, 43], multicommodity flow [27], oblivious routing [18], and graph sparsification [44, 11]. They also have found applications in social network analysis, e.g., for measuring the similarity of vertices in social networks [33], in machine learning, e.g., for Gaussian sampling [7] and in chemistry, e.g., for measuring chemical distances [28]. Previous research has studied the problem of how to quickly compute and approximate the effective resistances (or equivalently, *energies* of electrical flows; see the full version for more discussions), as such algorithms can be used as a crucial subroutine for other graph algorithms. For example, one can $(1+\varepsilon)$-approximate the $s-t$ effective resistance in $\tilde{O}(m+n\varepsilon^{-2})$ [14] and $\tilde{O}(m\log(1/\varepsilon))$ [9] time, respectively, in any $n$-vertex $m$-edge weighted graph, for any two vertices $s,t$. (Throughout the paper, we use $\tilde{O}$ to hide polylogarithmic factors, i.e., $\tilde{O}(f(n)) = O(f(n)\cdot \text{poly}\log f(n))$.) There are also algorithms that find $(1+\varepsilon)$-approximations to the effective resistance between every pair of vertices in $\tilde{O}(n^2/\varepsilon)$ time [24]. In order to exactly compute the $s-t$ (or single-pair) and all-pairs effective resistance(s), the current fastest algorithms run in times $O(n^\omega)$ (by using the fastest matrix inversion algorithm [6, 21]) and $O(n^{2+\omega})$, respectively, where $\omega < 2.373$ is the matrix multiplication exponent [46]. In planar graphs, the algorithms for exactly computing $s-t$ and all-pairs effective resistance(s) run in times $O(n^{\omega/2})$ (by the nested dissection method for solving linear system in planar graphs [34]) and $O(n^{2+\omega/2})$, respectively.

A natural algorithmic question is how to efficiently maintain the effective resistances *dynamically*, i.e., if the graph undergoes edge insertions and/or deletions, and the goal is to support the update operations and query for the effective resistances as quickly as possible, rather than having to recompute it from scratch each time. Besides the potential applications in the design of other (dynamic) algorithms, it is also of practical interest, e.g., to quickly report the (dis)similarity between any two nodes in a social network in which its members and their relationship are constantly changing. So far our understanding towards this question is very limited: for exact maintenance, the only approach (for single-pair effective resistance) we are aware of is to invoke the dynamic matrix inversion algorithm which gives $O(n^{1.575})$ update time and $O(n^{0.575})$ query time or $O(n^{1.495})$ update time and $O(n^{1.495})$ query time [42]; for $(1+\varepsilon)$-approximate maintenance, we can maintain the spectral sparsifier of size $n\text{poly}(\log n, \varepsilon^{-1})$ with $\text{poly}(\log n, \varepsilon^{-1})$ update time [3], while answering each query will cost $\Theta(n\text{poly}(\log n, \varepsilon^{-1}))$ time. (Subsequent to the Arxiv submission [17] of this paper, Durfee et al. obtained a fully dynamic algorithm that maintains $(1+\varepsilon)$-approximations to all-pairs effective resistances of an unweighted, undirected multi-graph with $\tilde{O}(m^{4/5}\varepsilon^{-4})$ expected amortized update and query time [13].)

In this paper, we study the problem of dynamically maintaining the (approximate) effective resistances in *separable graphs*, which are those that satisfies an $n^c$-separator theorem for some $c<1$. Interesting classes of separable graphs include planar graphs, minor free graphs, bounded-genus graphs, almost planar graphs (e.g., road networks) [35], most 3-dimensional meshes [40] as well as many real-world networks (e.g., phone-call graphs, Web graphs, Internet router graphs) [5]. In the static setting, effective resistances (or electrical flows) in planar/separable graphs have been utilized by Miller and Peng [39] to obtain the first $\tilde{O}(\frac{m^{6/5}}{\varepsilon^{\Theta(1)}})$ time algorithm for approximate maximum flow in such graphs, and have also been studied by Anari and Oveis Gharan [4] in the analysis of an approximation algorithm for

Asymmetric TSP. We now give the necessary definitions to state our results.

**Effective Resistances.** Let $G = (V, E, \mathbf{w})$ be a undirected weighted graph with $\mathbf{w}(e) > 0$ for any $e \in E$. Let $\mathbf{A}$ denote its weighted adjacency matrix and $\mathbf{D}$ denote the weighted degree diagonal matrix. Let $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denote the *Laplacian* matrix of $G$. Let $\mathbf{L}^\dagger$ denote the Moore-Penrose pseudo-inverse of the Laplacian of $G$. Let $\mathbf{1}_u \in \mathbb{R}^V$ denote the indicator vector of vertex $u$ such that $\mathbf{1}_u(v) = 1$ if $v = u$ and 0 otherwise. Let $\chi_{s,t} = \mathbf{1}_s - \mathbf{1}_t$. Given any two vertices $u, v \in V$, the $s-t$ *effective resistance* is defined as $R_G(s,t) := \chi_{s,t}^T \mathbf{L}^\dagger \chi_{s,t}$.

**Separable Graphs.** Let $\mathcal{C}$ be a class of graphs that is closed under taking subgraphs. We say that $\mathcal{C}$ satisfies a $f(n)$-*separator theorem* if there are constants $\alpha < 1$ and $\beta > 0$ such that every graph in $S$ with $n$ vertices has a cut set with at most $\beta f(n)$ vertices that separates the graph into components with at most $\alpha n$ vertices each [35]. In this paper we are particularly interested in the class of graphs that satisfies an $n^{1/2}$-separator theorem, which include the class of planar graphs, $K_t$-minor free graphs and bounded-genus graphs, etc., though our approach can also be generalized to other class of graphs that satisfies a $n^c$-separator theorem, for some $c < 1$. In the following, we call a graph $f(n)$-*separable* if it is a member of a class that satisfies an $f(n)$-separator theorem.

We would like to quickly maintain the exact or a good approximation of the $s-t$ effective resistances in a $\sqrt{n}$-separable graph that undergoes edge insertions and deletions, for all pairs $s, t \in V$. We call this the *dynamic all-pairs effective resistances problem.* Our goal is to solve this problem with both small update and query times. More precisely, our data structure supports the following operations.

- Insert$(u, v, w)$: Insert the edge $(u, v)$ of weight $w$ to $G$, provided that the updated graph remains $\sqrt{n}$-separable.
- Delete$(u, v)$: Delete the edge $(u, v)$ from $G$.
- EffectiveResistance$(s, t)$: Return the exact or approximate value of the effective resistance between $s$ and $t$ in the current graph $G$.

## 1.1 Our Results

We give a fully dynamic algorithm for maintaining $(1+\varepsilon)$-approximations of all-pairs and single-pair effective resistance(s) with small update and query times for any $\sqrt{n}$-separable graph, if its separator can be computed fast. Throughout the paper, all the running times of our algorithms are measured in *worst-case* performance. All our algorithms are randomized, and the performance guarantees hold with probability at least $1 - n^{-c}$ for some $c \geq 1$.

▶ **Theorem 1.** *Let $G$ denote a dynamic $n$-vertex graph under edge insertions and deletions. Assume that $G$ is $\sqrt{n}$-separable and its separator can be computed in $s(n)$ time, throughout the updates. There exist fully dynamic algorithms that maintain $(1+\varepsilon)$-approximations of*

- *the all-pairs effective resistances with $\tilde{O}(\frac{\sqrt{n}}{\varepsilon^2} + \frac{s(n)}{\sqrt{n}})$ update time and $\tilde{O}(\frac{\sqrt{n}}{\varepsilon^2})$ query time;*
- *the $s-t$ effective resistance with $\tilde{O}(\frac{\sqrt{n}}{\varepsilon^2} + \frac{s(n)}{\sqrt{n}})$ update time and $O(1)$ query time.*

*In particular, if $s(n) = \tilde{O}(n)$, then our update times are $\tilde{O}(\frac{\sqrt{n}}{\varepsilon^2})$.*

By using the well known facts that a balanced separator of size $O(\sqrt{n})$ for planar graphs (and bounded-genus graphs) can be computed in $O(n)$ time [35], and for $K_t$-minor-free graphs (for any fixed integer $t > 0$) in $O(n^{1+\delta})$ time, for any constant $\delta > 0$ [26], we obtain dynamic algorithms for the effective resistances for planar and minor-free graphs with $\tilde{O}(\sqrt{n}/\varepsilon^2)$ and $\tilde{O}(\sqrt{n}/\varepsilon^2 + n^{1/2+\delta})$ update time, respectively.

The performance of our dynamic algorithm in planar graphs almost matches the best-known dynamic algorithm for $(1+\varepsilon)$-approximate all-pairs shortest path in planar graphs with $\tilde{O}(\sqrt{n})$ update and query time [2], though our approaches are different. This is interesting as the shortest path corresponds to flows with controlled $\ell_1$ norm while the energy of electrical flows (i.e., effective resistance) corresponds to those with minimum $\ell_2$ norm.

In order to design a dynamic algorithm for effective resistances of separable graphs (i.e., to prove Theorem 1), we give a fully dynamic algorithm that efficiently maintains an *approximate Schur complement* [30, 31, 14] of such graphs (see Section 4.1), which might be of independent interest. Approximate Schur complement can be treated as a *vertex sparsifier* that preserves pairwise effective resistances among a set of terminals (see Section 3). Therefore, our algorithm is a dynamic algorithm for *vertex effective resistance sparsifiers* with sublinear (in $n$) update time for separable graphs. The problem of dynamically maintaining graph *edge sparsifiers* has received attention very recently. For example, Abraham et al. presented fully dynamic algorithms that maintain cut and spectral sparsifiers with poly-logarithmic update times [3]. Formally, we prove the following theorem.

▶ **Theorem 2.** *For an $n$-vertex $\sqrt{n}$-separable graph $G$ whose separator can be computed in $s(n)$ time, and a terminal set $K \subseteq V$ with $|K| \leq O(\sqrt{n})$, there exists a fully dynamic algorithm that maintains a $(1+\delta)$-approximate Schur complement with respect to $K'$ such that $K \subseteq K'$ and $|K'| = O(\sqrt{n})$, while achieving $\tilde{O}(\sqrt{n}/\delta^2 + \frac{s(n)}{\sqrt{n}})$ update time. Furthermore, our algorithm supports terminal additions as long as $|K| \leq O(\sqrt{n})$.*

We complement our algorithm by giving a conditional lower bound for any *incremental* or *decremental* algorithm that maintains *single-pair* effective resistance of a $\sqrt{n}$-separable graph. Our lower bound is established from the *Online Matrix Vector Multiplication* (*OMv*) *conjecture* (see the full version).

▶ **Theorem 3.** *No incremental or decremental algorithm can maintain the (exact) $s-t$ effective resistance in $\sqrt{n}$-separable graphs on $n$ vertices with both $O(n^{\frac{1}{2}-\delta})$ worst-case update time and $O(n^{1-\delta})$ worst-case query time for any $\delta > 0$, unless the OMv conjecture is false.*

We note that there are very few conditional lower bounds for dynamic *planar/separable* graphs, as most known reductions are highly non-planar. The only recent result that we are aware of is by Abboud and Dahlgaard [1], who showed that under some popular conjecture, no algorithm for dynamic shortest paths or maximum weight bipartite matching in planar graphs has both updates and queries in amortized $O(n^{1/2-\delta})$ time, for any $\delta > 0$.

We also give a stronger conditional lower bound for the same problem in *general* graphs, which shows that it is hard to maintain effective resistances with both sublinear (in $n$) update and query times for general graphs, even for the incremental or decremental setting.

▶ **Theorem 4.** *No incremental or decremental algorithm can maintain the (exact) $s-t$ effective resistance in general graphs on $n$ vertices with both $O(n^{1-\delta})$ worst-case update time and $O(n^{2-\delta})$ worst-case query time for any $\delta > 0$, unless the OMv conjecture is false.*

We remark that both lower bounds for separable and general graphs hold for any algorithm with sufficiently high accurate approximation ratio (see Section 5 and full version).

**Comparison to [16].** In our previous work [16], we gave a fully dynamic algorithm for $(1+\varepsilon)$-approximating all-pairs effective resistances for planar graphs with $\tilde{O}(r/\varepsilon^2)$ update time and $\tilde{O}((r + n/\sqrt{r})/\varepsilon^2)$ query time, for any $r$ larger than some constant. The algorithm can also be generalized to $\sqrt{n}$-separable graphs, and we also provided a conditioned lower bound for any approximation algorithm of the $s-t$ effective resistance in general graphs in the *vertex-update* model. However, besides the apparent improvement of the performance of the dynamic algorithm (i.e., we reduce the best trade off between update time and query time from $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{2/3})$ to $\tilde{O}(n^{1/2})$ and $\tilde{O}(n^{1/2})$), our current work also improves over and differs from [16] in the following perspectives.

**I.** Our algorithm dynamically maintains the approximate Schur complement of a separable graphs by maintaining a separator tree of such graphs, rather than their *$r$-divisions* as used in [16]. In fact, we do not believe purely $r$-divisions based algorithms will achieve the performance as guaranteed by our new algorithm. This is evidenced by previous dynamic algorithms for maintaining reachability in directed planar graphs by Subramanian [45], $(1+\varepsilon)$-approximating to all-pairs shortest paths by Klein and Subramanian [29], exactly maintaining $s-t$ max-flow in planar graphs by Italiano et al. [23], all of which are based on $r$-divisions and have running times of order $n^{2/3}$ (and some of which have been improved by using other approaches).

**II.** Our current lower bound is much stronger than the previous one: the previous lower bound only holds for general graphs and the *vertex-update* model, where nodes, not edges, are turned on or off, and its proof was based on a simple relation between $s-t$ connectivity and $s-t$ effective resistance $R_G(s,t)$ (i.e., if $s,t$ is connected iff $R_G(s,t)$ is not infinity). In contrast, our new lower bounds hold for separable graphs (and also general graphs) and the edge-update model. The corresponding proofs exploit new reductions from the OMv problem to the 5-length cycle detection and triangle detection problems in separable graphs and general graphs, respectively, which might be of independent interest, and the latter problems are related to the effective resistances (see Section 5).

## 1.2 Our Techniques

Our dynamic algorithm for maintaining an Approximate Schur complement (ASC) w.r.t. a set of terminals for separable graphs is built upon maintaining a *separator tree* of such graphs and two properties (called *transitivity* and *composability*) of ASCs. Such a tree can be constructed very efficiently by recursively partitioning the subgraphs using separators. Slightly more formally, each node in the tree corresponds to a subgraph of the original graph and contains a subset of vertices as its boundary vertices which in turn are treated as terminals. For each node $H$, we will maintain an ASC $H'$ of $H$ w.r.t its terminals. We will guarantee throughout all the updates that the ASC of any node can be computed efficiently in a bottom-up fashion, by the above two properties of ASCs. This stems from the fact that we only need to recompute the ASCs of nodes that lie on a path from a *constant* number leaves to the node of interest. Since each such path has length $O(\log n)$ and the recomputation of ASC of one node takes time $\tilde{O}(\sqrt{n})$, the update time will be guaranteed to be $\tilde{O}(\sqrt{n})$. For the detailed implementation, we need to overcome the difficulty that the error in the approximation ratio might accumulate through this recursive computation and an update might require to change the set of boundary vertices of many nodes, thus resulting in a prohibitive running time. We remark that though the idea of using separator tree of planar/separable graphs is standard (e.g., [15]), the main novelty of our algorithm is to use such a tree as the backbone to dynamically maintain the approximate Schur complement.

To obtain our dynamic algorithms for all-pair effective resistance, we appropriately declare and add new terminals whenever we get a new query, and then run the above dynamic algorithm for ASC with respect to the corresponding terminal set.

To obtain our lower bound, we provide new reductions from the Online Boolean Matrix-Vector Multiplication (OMv) problem to the incremental or decremental single-source effective resistance problem. More specifically, given an OMv instance with vectors $\mathbf{u}, \mathbf{v}$ and a matrix $\mathbf{M}$, we construct a $\sqrt{n}$-separable graph $G$ such that $\mathbf{uMv} = 1$ if and only if there exists a cycle of length 5 incident to some vertex $t$ in $G$. This 5-length cycle detection problem in turn can be solved by inspecting the diagonal entry corresponding to $t$ of the inverse of a matrix that is defined from $G$. Furthermore, the diagonal entry of this matrix is inherently related to the effective resistance [41]. By appropriately dynamizing the graph $G$ and using the time bounds for the OMv problem from the conjecture, we get the conditional lower bound for separable graphs. For general graphs, the lower bound is proved in a similar way, except that the constructed graph is different and we instead use a relation between effective resistance and triangle detection problem. That is, we first reduce the OMv problem to the $t$-triangle detection problem such that the OMv instance satisfies $\mathbf{uMv} = 1$ if and only if there exists a triangle incident to some vertex $t$ in the constructed $G$. The latter problem can again be solved by checking the diagonal entry corresponding to $t$ of some matrix, which in turn encodes the effective resistance of between $t$ and a properly specified vertex $s$.

**Other Related Work.** Previous work on dynamic algorithms for planar or plane graphs include: shortest paths [29, 2, 23], $s-t$ min-cuts/max-flows [23], reachability in directed graphs [45, 22, 10], ($k$-edge) connected components [15, 20], the best swap and the minimum spanning forest [15]. There also exist work on dynamic algorithms for $\sqrt{n}$-separable graphs, e.g., on transitive closure and $(1+\varepsilon)$-approximation of all-pairs shortest paths [25].

As mentioned before, subsequent to our Arxiv submission, Durfee et al. [13] obtained a dynamic all-pairs effective resistances algorithm with $\tilde{O}(m^{4/5}\varepsilon^{-4})$ expected amortized update and query time, against an oblivious adversary. This algorithm uses ideas stemmed from this paper, in particular, one of their key ideas is to dynamically maintain an approximate Schur complement. If restricted to separable graphs, the running times of their algorithm are worse than ours. It is also interesting to note that for the (simpler) offline dynamic effective resistance problems, i.e., the sequence of updates and queries are given as an input, Li et al. [32] recently gave an incremental algorithm with $O(\frac{\operatorname{poly}\log n}{\varepsilon^2})$ amortized update and query time for general graphs.

## 2 Basic Tools

Our algorithm is built upon two tools: *separator trees* and *approximate Schur complement.*

**Separator Trees.** Let $G$ be a $\sqrt{n}$-separable graph. For an edge-induced subgraph $H$ of $G$, any vertex that is incident to vertices not in $H$ is called a *boundary vertex*. We let $\partial(H)$ denote the set of *boundary vertices* belonging to $H$. A hierarchical decomposition of $G$ is obtained by recursively partitioning the graph using separators into edge-disjoint subgraphs (called regions). This decomposition is represented by a binary (decomposition) tree $\mathcal{T}(G)$, which we refer to as a *separator tree* of $G$. For any subgraph $H$ of $G$, we use $H \in \mathcal{T}(G)$ to denote that $H$ is a node of $\mathcal{T}(G)$ (to avoid confusion with the vertices of $G$, we refer to the vertices of $\mathcal{T}(G)$ as nodes). The *height* $\eta(H)$ of a node is the number of edges in the longest path between that node and a leaf. Let $S(H)$ denote a balanced separator of the subgraph $H$. Further details on the definition and properties of $\mathcal{T}(G)$ can be found in the full version.

**(Approximate) Schur Complement (ASC).** For a given connected graph $G = (V, E)$ and a set $K \subset V$ of terminals with $1 \leq |K| \leq |V| - 1$, let $N = V \setminus K$. The partition of $V$ into $N$ and $K$ naturally induces the following partition of the Laplacian $\mathbf{L}(G)$ into blocks: $\mathbf{L}(G) = \left(\begin{smallmatrix} \mathbf{L}_N & \mathbf{L}_M \\ \mathbf{L}_M^T & \mathbf{L}_K \end{smallmatrix}\right)$. We remark that since $G$ is connected and $N$ and $K$ are non-empty, one can show that $\mathbf{L}_N$ is invertible. We have the following definition of Schur complement.

▶ **Definition 5** (Schur Complement). The (unique) *Schur complement* of a graph Laplacian $\mathbf{L}(G)$ with respect to a terminal set $K$ is $\mathbf{S}(G, K) := \mathbf{L}_K - \mathbf{L}_M^T \mathbf{L}_N^{-1} \mathbf{L}_M$.

It is known that the matrix $\mathbf{S}(G, K)$ is a Laplacian for some graph $G'$ with vertex set $K$.

▶ **Definition 6** (Approximate Schur Complement (ASC)). Given a graph $G = (V, E, \mathbf{w})$, $K \subset V$ and its Schur complement $\mathbf{S}(G, K)$, we say that a graph $H = (K, E_H, \mathbf{w}_H)$ is a $(1 \pm \varepsilon)$*-approximate Schur complement (abbr.* $(1 \pm \varepsilon)$-ASC*)* with respect to $K$ if $\forall \mathbf{x} \in \mathbb{R}^{|K|}$, $(1 - \varepsilon)\mathbf{x}^T \mathbf{S}(G, K)\mathbf{x} \leq \mathbf{x}^T \mathbf{L}(H)\mathbf{x} \leq (1 + \varepsilon)\mathbf{x}^T \mathbf{S}(G, K)\mathbf{x}$.

In particular, if $\mathbf{L}(H) = \mathbf{S}(G, K)$, then we say $H$ is a 1-ASC of $G$ w.r.t. $K$.

ASC can be computed efficiently as guaranteed in the following lemma.

▶ **Lemma 7** ([14]). *Fix $\varepsilon \in (0, 1/2)$ and $\gamma \in (0, 1)$, and let $G = (V, E, \mathbf{w})$ be a graph with $K \subset V$. There is an algorithm* ApproxSchur$(G, K, \varepsilon, \gamma)$ *that computes a $(1 \pm \varepsilon)$-ASC $H$ of $G$ with respect to $K$ such that the following statements hold with probability at least $1 - \gamma$: (1) The graph $H$ has $O(|K|\varepsilon^{-2} \log(n/\gamma))$ edges. (2) The total running time for computing $H$ is $\tilde{O}(m \log^3(n/\gamma) + n\varepsilon^{-2} \log^4(n/\gamma))$.*

## 3 Useful Properties of Approximate Schur Complement

**Approximate Schur Complement as Vertex Effective Resistance Sparsifier.** To maintain effective resistances efficiently, it will be useful to consider the following notion of *vertex sparsifier* that preserves pairwise effective resistances among a set of terminals.

▶ **Definition 8** (Vertex Resistance Sparsifier (VRS)). Given a graph $G = (V, E, \mathbf{w})$ with $K \subset V$, we say that a graph $H = (K, E_H, \mathbf{w}_H)$ is an $(1 \pm \varepsilon)$*-vertex resistance sparsifier (abbr.* $(1 \pm \varepsilon)$-VRS*)* of $G$ with respect to $K$ if $\forall s, t \in K$, $(1 - \varepsilon)R_G(s, t) \leq R_H(s, t) \leq (1 + \varepsilon)R_H(s, t)$.

The lemma below relates ASC and VRS (see the full version for the proof.)

▶ **Lemma 9.** *Let $G = (V, E, \mathbf{w})$ be a graph with $K \subset V$. If $H$ is an $(1 \pm \varepsilon)$-ASC of $G$ with respect to $K$, then $H$ is an $1/(1 \pm \varepsilon)$-VRS of $G$ with respect to $K$.*

**Transitivity and Composability of ASCs.** We will prove a *transitivity* and a *composability* property of ASCs, which will enable us to compute the ASCs of all nodes of $\mathcal{T}(G)$ in a bottom-up fashion. The corresponding proofs are deferred in the full version.

▶ **Lemma 10** (Transitivity of ASCs). *If $H'$ is an $(1 \pm \varepsilon)$-ASC of $G$ w.r.t. $K'$, and $H$ is an $(1 \pm \varepsilon)$-ASC of $H'$ w.r.t. $K$, where $K' \supseteq K$, then $H$ is an $(1 \pm \varepsilon)^2$-ASC of $G$ w.r.t. $K$.*

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be edge-disjoint graphs with terminals $K_1$ and $K_2$, respectively. Furthermore, assume that $(V_1 \cap V_2) \subset K_i$, for $i = \{1, 2\}$. The *merge* of $G_1$ and $G_2$ is the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ with terminals $K_1 \cup K_2$ formed by identifying the terminals in $V_1 \cap V_2$. We denote this operation by $G := G_1 \oplus G_2$.

▶ **Lemma 11** (Composition of ASCs). *Let $G := G_1 \oplus G_2$. If $H_1'$ is an $(1 \pm \varepsilon)$-ASC of $G_1$ with respect to $K_1$, and $H_2'$ is an $(1 \pm \varepsilon)$-ASC of $G_2$ with respect to $K_2$, then $H' := H_1' \oplus H_2'$ is an $(1 \pm \varepsilon)$-ASC of $G$ with respect to $K$.*

**Algorithm 1:** ApproxSchurNode$(H, \partial(H), \delta')$.

```
1 Set $\gamma = 1/n^3$.
2 if $H$ is a leaf then
3     Set $H' \leftarrow$ ApproxSchur$(H, \partial(H), \delta', \gamma)$.
4 if $H$ is a non-leaf then
5     Let $c_1(H), c_2(H)$ be the children of $H$.
6     Let $c_i(H)'$ be the ASC of $c_i(H)$, for $i = 1, 2$.
7     Set $R \leftarrow c_1(H)' \oplus_\phi c_2(H)'$ and $E(R) \leftarrow E(R) \cup X(H)$.
8     Set $H' \leftarrow$ ApproxSchur$(R, \partial(H), \delta', \gamma)$.
9 return $H'$.
```

## 4 Dynamic Algorithms for Effective Resistances in Separable Graphs

In this section, we first present our fully dynamic algorithm for maintaining an ASC of a $\sqrt{n}$-separable graph and then show how to extend it to dynamic effective resistances algorithm. For simplicity, we assume the separator of $G$ can be computed in $\tilde{O}(n)$ time. We defer the discussion on the general case, some implementation details and analysis to the full version.

### 4.1 Dynamic Approximate Schur Complement

Let $\delta \in (0, 1)$. Let $K \subset V$ be a set of terminals with $|K| \leq O(\sqrt{n})$. We give a data-structure for maintaining a $(1 \pm \delta)$-ASC of a $\sqrt{n}$-separable graph $G$ with respect to a set $K'$ of $\sqrt{n}$ vertices (which contains the terminal set $K$) that supports Insert and Delete operations as defined before. In addition, it supports the following operation:

- AddTerminal$(u)$: Add the vertex $u$ to the terminal set $K$, as long as $|K| \leq O(\sqrt{n})$.

**Data Structure.** We compute and maintain a balanced separator $S(G)$ of $G$ that contains $K$ and satisfies that $|S(G)| \leq O(\sqrt{n})$. We let $K' = S(G)$ and we will maintain a $(1 \pm \delta)$-ASC of $G$ w.r.t. $K'$. By definition of boundary vertices, $K' = \partial(G)$. Let $\delta' = \frac{\delta}{c \log n + 1}$ for some constant $c$. In our dynamic algorithm, we will maintain a separator tree $\mathcal{T}(G)$ (see the full version) such that for each node $H \in \mathcal{T}(G)$, we maintain its separator $S(H)$ and a set $X(H)$ of edges of $H$, which is initially empty, and an ASC $H'$ of $H$ w.r.t. $\partial(H)$. Throughout the updates, the set $X(H)$ will denote the subset of edges which are only contained in $H$ while contained in neither of its children. Let $\mathcal{D}(G, \delta)$ denote such a data-structure. We recompute $\mathcal{D}(G, \delta)$ every $\Theta(\sqrt{n})$ operations using the initialization below.

**Initialization.** We show how to efficiently compute the ASC $H'$ for each node $H$ from $\mathcal{T}(G)$. We do this in a bottom-up fashion by first calling Algorithm 1 on each leaf node and then on the non-leaf nodes, where ApproxSchur is the procedure from Lemma 7. In what follows, whenever we compute an ASC, we assume that procedure ApproxSchur from Lemma 7 is invoked on the corresponding subgraph and its boundary vertices, with approximation parameter $\delta'$ and error probability $\gamma = \frac{1}{n^3}$. We will also assume that all the calls to ApproxSchur are correct.

The following lemma shows that after invoking Algorithm 1 in a bottom-up fashion, we have computed the ASC for every node in $\mathcal{T}(G)$.

**Algorithm 2:** UPDATEAPPROXSCHUR(STACK $Q$).

```
1 while $Q \neq \emptyset$ do
2     Set $H \leftarrow Q$.POP().
3     Set $H' \leftarrow$ APPROXSCHURNODE$(H, \partial(H), \varepsilon)$.
```

▶ **Lemma 12.** *Let $H \in \mathcal{T}(G)$ be a node of height $\eta(H) \geq 0$ and $\mathrm{X}(H) = \emptyset$. Then $H' =$ APPROXSCHURNODE$(H, \partial(H), \varepsilon)$ is an $(1 \pm \delta')^{\eta(H)+1}$-ASC of $H$ with respect to $\partial(H)$.*

**Proof.** We proceed by induction on $\eta(H)$. For the base case, i.e., $\eta(H) = 0$, $H$ is a leaf node. By Lemma 7 and Algorithm 1, $H'$ is indeed a $(1 \pm \delta')$-ASC of $H$ with respect to $\partial(H)$.

Let $H$ be a non-leaf node, i.e. $\eta(H) > 0$. Let $\mathrm{c}_1(H), \mathrm{c}_2(H)$ and $\mathrm{c}_1'(H), \mathrm{c}_2'(H)$ be defined as in Algorithm 1. By properties (2), (3) and (4) of $\mathcal{T}(G)$ and the fact that $X(H) = \emptyset$, we have $H = \mathrm{c}_1(H) \oplus \mathrm{c}_2(H)$. By induction hypothesis, it follows that $\mathrm{c}_i(H)'$ is an $(1 \pm \delta')^{\eta(\mathrm{c}_i(H))+1}$-ASC of $\mathrm{c}_i(H)$, for $i = 1, 2$. Using Lemma 11 and since $\eta(\mathrm{c}_i(H)) + 1 = \eta(H)$, for $i = 1, 2$, we get that $R := \mathrm{c}_1(H)' \oplus \mathrm{c}_2(H)'$ is an $(1 \pm \delta')^{\eta(H)}$-ASC of $H$ with respect to $V(R) := \partial(\mathrm{c}_1(H)) \cup \partial(\mathrm{c}_2(H))$. Now, since $V(R) \supseteq \partial(H)$ by property (4) of $\mathcal{T}(G)$ and by Lemma 7, it follows that $H'$ is an $(1 \pm \delta')$-ASC of $R$ with respect to $\partial(H)$. Finally, applying Lemma 10 on $R$ and $H'$ we get that $H'$ is an $(1 \pm \delta')^{\eta(H)+1}$-ASC of $H$. ◀

Since $\delta' = \frac{\delta}{c \log n + 1}$ and $\eta(G) = O(\log n)$, the graph $G'$ is a $(1 \pm \delta)$-ASC of $G$ w.r.t. $\partial(G)$.

**Handling Edge Insertions.** We now describe the INSERT operation. Let us consider the insertion of an edge $e = (u, v)$ of weight $w$. We maintain a stack $Q$, which is initially set to empty. We then update the root node by adding $(u, v)$ with weight $w$ to $G$, and push $G$ onto $Q$. During the traversal of $\mathcal{T}(G)$, our procedure maintains two pointers that point to the current node $H$ (initially set to $G$) and a node $N$ (if any exists) that represents the node for which $u$ and $v$ belong to different children of $N$, respectively. As long as we have not found such a node $N$, and the current node $H$ is not a leaf, we proceed as follows.

We examine the child of $H$ that contains both $u$ and $v$ (if there is more than one, then we just pick one of them). If $u$ and $v$ belong to the same child, say $\mathrm{c}(H)$, then we add this edge to $\mathrm{c}(H)$ and update the current node $H$ to $\mathrm{c}(H)$. We then push $H$ onto $Q$. If, however, $u$ and $v$ belong to different children, then we set $N$ to be the current node $H$ and add the edge $(u, v)$ to $\mathrm{X}(N)$, since $u$ and $v$ cannot appear together in the nodes of the lower levels. At this point, this forces $u$ and $v$ to become boundary vertices in $N$ and all other nodes descending from $N$ that contain either $u$ or $v$. We handle this by making use of the ADDBOUNDARY() procedure, depicted in Algorithm 4. Finally, we recompute the ASCs of the affected nodes in a bottom-up fashion using the stack $Q$ (as shown in Algorithm 2). This procedure is summarized in Algorithm 3. We remark that for simplicity, we let $Q$.PUSH$(H)$ denote the event of pushing the pointer to $H$ to the stack $Q$, for any node $H$.

After the pre-processing step and after each insertion/deletion of an edge, our augmented separator tree $\mathcal{T}(G)$ satisfies the following invariant.

▶ **Invariant 13.** *For every edge $e$ in the current graph $G$ exactly one of the following two holds: (1) there is a leaf node $H \in \mathcal{T}(G)$ such that $e \in E(H)$, (2) there is an internal node $H \in \mathcal{T}(G)$ such that $e \in X(H)$.*

The following lemma guarantees that the updated graph $G'$ (i.e., the sparsifier of the root node $G$) is a good estimate to the Schur complement of $G$ with respect to the boundary, after the execution of INSERT$(u, v)$ in Algorithm 3, and its proof is deferred to the full version.

**Algorithm 3:** INSERT$(u, v, w)$.

```
1  Let $Q$ be an initially empty stack.
2  Set $E(G) \leftarrow E(G) \cup \{(u,v)\}$, $Q$.PUSH$(G)$, $H \leftarrow G$ and $N \leftarrow$ NIL.
3  while $N =$ NIL and $H$ is a non-leaf do
4      if there exists a child of $H$ that contains both $u$ and $v$ then
5          Let c$(H)$ denote any such a child.
6          Set $E\left(\mathrm{c}(H)\right) \leftarrow E\left(\mathrm{c}(H)\right) \cup \{(u,v)\}$.
7          Set $H \leftarrow \mathrm{c}(H)$.
8          $Q$.PUSH$(H)$.
9      else
10         Set $N \leftarrow H$.
11         Set $\mathrm{X}(N) \leftarrow \mathrm{X}(N) \cup \{(u,v)\}$.
12         ADDBOUNDARY$(u, N)$, ADDBOUNDARY$(v, N)$.
13 UPDATEAPPROXSCHUR$(Q)$.          // Update the ASCs of the nodes in $Q$
```

**Algorithm 4:** ADDBOUNDARY$(u, N)$.

```
1  Let $Q$ be an initially empty stack.
2  while $N =$ NIL do
3      if $u \notin \partial(H)$ then
4          Set $\partial(H) \leftarrow \partial(H) \cup \{u\}$.
5          $Q$.PUSH$(H)$.
6          if $H$ is a non-leaf then
7              Let $c(H)$ be the unique child that contains $u$.
8              Set $H \leftarrow \mathrm{c}(H)$.
9      if $H$ is a leaf then
10         Set $H \leftarrow$ NIL.
11 UPDATEAPPROXSCHUR$(Q)$.          // Update the ASCs of the nodes in $Q$
```

▶ **Lemma 14.** *Let $G'$ be the updated sparsifier of the root node $G$, after the insertion of edge $(u, v)$. Then $G'$ is an $(1 \pm \delta)$-ASC of $G$ with respected to $\partial(G)$.*

**Handling Terminal Additions to the Boundary.** We now describe the ADDTERMINAL$(u)$ operation. It is implemented by simply invoking ADDBOUNDARY$(u, G)$, where $G$ is the root of $\mathcal{T}(G)$. For the procedure ADDBOUNDARY$(u, H)$, we maintain a stack $Q$, which is initially set to empty. As long as the current $H$ is a node in $\mathcal{T}(G)$, we first check whether $u \in \partial(H)$. If this is the case, then we simply do nothing as the ASC $H'$ of $H$ with respect to $\partial(H)$ contains $u$. Otherwise, we add $u$ to $\partial(H)$, and push the node $H$ to $Q$. Next, if $H$ is not a leaf-node, let c$(H)$ be the *unique* child that contains $u$. We then set c$(H)$ to be our current node $H$ and perform the same steps as above, until we reach some leaf-node, in which case we set $H$ to NIL. Finally, we recompute the ASCs of the affected nodes in a bottom-up fashion using the stack $Q$. This procedure is summarized in Algorithm 4. The correctness of this procedure can be shown similarly to the correctness of INSERT().

**Handing Edge Deletions and Running Times.** The operation of deleting an edge can be handled in a symmetric way as for handling edge insertions (see the full version). For all three operations (i.e., INSERT, DELETE, ADDTERMINAL), the running times are guaranteed to be $\tilde{O}(\sqrt{n}/\delta^2)$. Their analysis are deferred to the full version.

## 4.2 Extension to Dynamic All-Pairs Effective Resistances

Our dynamic effective resistance algorithm uses the dynamic algorithm for maintaining a $(1 \pm \delta)$-ASC as a subroutine. Formally, to maintain $(1+\varepsilon)$-approximate effective resistances, we will invoke the dynamic ASC algorithm with parameter $\delta = \varepsilon/4$, to handle edge insertions/deletions, and terminal additions.

We now describe the query operation (for the case of all-pairs effective resistances). Given $s$ and $t$, we start by calling ADDTERMINAL($s$) and ADDTERMINAL($t$) from the dynamic ASC data-structure. This ensures that both $s$ and $t$ are boundary nodes at the root node $G$ (if they were not previously). Thus we obtain a $(1 \pm \delta)$-ASC, denoted as $G'$, of the root node $G$ w.r.t. $\partial(G)$ and run on $G'$ a nearly linear time algorithm for estimating the $s-t$ effective resistance (see the full version). Let $\psi$ be such an estimate. For the correctness, by Lemma 9, we have that $G'$ preserves all-pair effective resistances among vertices in $\partial(G)$ of $G$ up to an $1/(1 \pm \delta) \approx (1 \pm 2\delta)$ factor. Since we ensured that $s$ and $t$ are included in $\partial(G)$, the $s-t$ effective resistance is approximated within the same factor. By a known result (see the full version), it follows that the estimate $\psi$ approximates the effective resistance between $s$ and $t$ in $G'$, up to a $(1 \pm \delta)$ factor. Combining the above guarantees, we get $\psi$ gives an $(1 \pm 2\delta)(1 \pm \delta) \leq (1 \pm \varepsilon)$-approximation to $R_G(s,t)$, by the choice of $\delta$. The query time will be guaranteed to be $\tilde{O}(\sqrt{n}/\varepsilon^2)$. Further details are deferred to the full version.

## 5 Lower Bounds for Partially Dynamic Effective Resistances

We now give a conditional lower bound for incrementally maintaining the $s-t$ effective resistance in $O(\sqrt{n})$-separable graphs and prove Theorem 3. Our proof actually holds for any algorithm that maintains a $(1+O(\frac{1}{n^{36}}))$-approximation of $s-t$ effective resistance. The lower bounds for the decremental setting and general graphs are deferred to the full version.

**The reduction.** We reduce the **uMv** problem (see the definition in the full version) with parameters $n_1 = n_2 := n_0$ to the $s-t$ effective resistance problem as follows. Let $\mathbf{M}$ be the $n_0 \times n_0$ Boolean matrix of the **uMv** problem. Let $n = n_0^2 + 2n_0 + 2$. Let $\kappa = 3(n-1)^6$.

Given the matrix $\mathbf{M}$, we construct a graph $G_{\mathbf{M}} = (V_{\mathbf{M}}, E)$ as follows. (1) For each pair $1 \leq i,j \leq n_0$, we create two vertices $a_{ij}$ and $b_{ij}$, and add an edge $(a_{ij}, b_{ij})$ if and only if $M_{ij} = 1$. (2) For each row $i$, we create a vertex $u_i$ and add edge $(u_i, a_{ik})$ for each $1 \leq k \leq n_0$. For each column $j$, we create a vertex $v_j$ and add edge $(v_j, b_{kj})$ for each $1 \leq k \leq n_0$. This finishes the definition of $G_{\mathbf{M}}$. Note that $V_{\mathbf{M}} = \{a_{ij}, b_{ij}, 1 \leq i,j \leq n_0\} \cup \{u_i, 1 \leq i \leq n_0\} \cup \{v_j, 1 \leq j \leq n_0\}$. For any vertex $x \in V_{\mathbf{M}}$, let $\deg_{G_{\mathbf{M}}}(x)$ denote the degree of $x$ in $G_{\mathbf{M}}$.

Now we add two new vertices $t$ and $s$ to $G_{\mathbf{M}}$. For any $x \in \{a_{ij}, b_{ij}, 1 \leq i,j \leq n_0\}$, add an edge $(s,x)$ with weight $\kappa - \deg_{G_{\mathbf{M}}}(x)$. Denote the resulting graph by $G$ and note that $G$ contains $|V_{\mathbf{M}} \cup \{s,t\}| = n_0^2 + 2n_0 + 2 = n$ vertices.

Assume that $G$ is started in a dynamic effective resistance data structure. We also maintain some counters in the data structure. That is, we initialize a global counter $Y := 0$. For each vertex $x \in \{u_i, 1 \leq i \leq n_0\} \cup \{v_j, 1 \leq j \leq n_0\}$, we maintain a counter $c(x)$ which is initialized to be 0. We now explain how we use this data structure to determine **uMv**.

- Once $\mathbf{u}$ arrives, for any $i$ such that $\mathbf{u}_i = 1$, we insert an edge $(t, u_i)$ with weight 1, increase $Y$ and $c(u_i)$ by 1.
- Once $\mathbf{v}$ arrives, for any $j$ such that $\mathbf{v}_j = 1$, we insert an edge $(t, v_j)$ with weight 1, increase $Y$ and $c(v_j)$ by 1.
- Insert an edge $(s,t)$ with weight $\kappa - Y$. For each vertex $x \in \{u_i, 1 \le i \le n_0\} \cup \{v_j, 1 \le j \le n_0\}$, insert an edge $(s,x)$ with weight $\kappa - c(x) - \deg_{G_{\mathbf{M}}}(x)$.
- Perform a query EFFECTIVERESISTANCE$(s,t)$ to obtain the (approximate) $s-t$ effective resistance in the final graph. Let $\lambda =$ EFFECTIVERESISTANCE$(s,t)$. If $\lambda \le \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{1}{\kappa^6}$, then return 1; otherwise, return 0.

**Analysis.** Note that throughout the whole sequence of updates (which are only edge insertions) and queries, the dynamic graph $G$ is always $O(\sqrt{n})$-separable, with a balanced separator set $S := \{u_1, \cdots, u_{n_0}\} \cup \{v_1, \cdots, v_{n_0}\} \cup \{s,t\}$ of size $O(\sqrt{n})$.

We have the following lemma that shows an important property of our reduction. The proof of the lemma is deferred to the end of this section.

▶ **Lemma 15.** *For $\kappa = 3(n-1)^6$, assume that* EFFECTIVERESISTANCE$(s,t)$ *returns the exact value of the $s-t$ effective resistance in the final graph $G$. Then the following holds: (1) If $\mathbf{uMv} = 1$, then $\lambda \le \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{1}{\kappa^6}$; (2) If $\mathbf{uMv} = 0$, then $\lambda > \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{1}{\kappa^6}$.*

Note that by the above lemma, the $\mathbf{uMv}$ problem can be solved according to our estimator $\lambda$. Thus, the lower bound for the incremental setting in Theorem 3 follows by a reduction from OMv conjecture to the $\mathbf{uMv}$ problem (see [19] and the full version of the paper) and by noting that the total number of updates is $O(n_0) = O(\sqrt{n})$ and the total number of queries is 1.

In the following we prove Lemma 15. The proof is based on a connection between the 5-length cycle detection problem and the effective resistance problem.

**Proof of Lemma 15.** Let $G$ denote the final graph of our reduction. Let $H := G[V_{\mathbf{M}} \cup \{t\}]$ denote the subgraph induced by vertex set $V_{\mathbf{M}} \cup \{t\}$. We observe that in the graph $H$, there is a cycle of length 5 containing vertex $t$ if and only if $\mathbf{uMv} = 1$.

On the other hand, we can use our estimator $\lambda$ to distinguish if $H$ contains a 5-length cycle incident to $t$ or not. We let $\mathbf{A} \in \mathbb{R}^{(n-1)\times(n-1)}$ denote the adjacency matrix of the graph $H$. Note that all entries in $A$ are either 1 or 0.

The first claim relates the 5-length cycle detection to the trace of a matrix related to $\mathbf{A}$. Recall that we let $X_{uv}$ denote the entry of matrix $X$ with row index corresponding to vertex $u$ and column index corresponding to vertex $v$.

▶ **Claim 16.** *Let $\mathbf{B} = \kappa \cdot \mathbf{I} - \mathbf{A}$. If $H$ contains a 5-length cycle incident to $t$, then $(\mathbf{B}^{-1})_{tt} \le \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{1.1}{\kappa^6}$. If $H$ does not contain a 5-length cycle incident to $t$, then $(\mathbf{B}^{-1})_{tt} \ge \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{0.9}{\kappa^6}$.*

**Proof.** First we note that $B$ is invertible, as it is strictly symmetric diagonally dominant. Furthermore, it holds that $\kappa \cdot \mathbf{B}^{-1} = (I - \frac{1}{\kappa} \cdot \mathbf{A})^{-1}$ and thus by the Neumann series expansion, we have $\kappa \cdot \mathbf{B}^{-1} = (I - \frac{1}{\kappa} \cdot \mathbf{A})^{-1} = \sum_{i=0}^{\infty} (-\frac{1}{\kappa})^i \cdot \mathbf{A}^i$. This further implies that

$$(\kappa \cdot \mathbf{B}^{-1})_{tt} = \mathbf{1}_t^T (\sum_{i=0}^{\infty} (-\frac{1}{\kappa})^i \cdot \mathbf{A}^i) \mathbf{1}_t = \sum_{i=0}^{\infty} (-\frac{1}{\kappa})^i \cdot \mathbf{1}_t^T (\mathbf{A}^i) \mathbf{1}_t = \sum_{i=0}^{\infty} (-\frac{1}{\kappa})^i \cdot (\mathbf{A}^i)_{tt}.$$

Now observe that since $\kappa = 3(n-1)^6$, the first six terms of the above power series dominate. More precisely, note that $(\mathbf{A}^i)_{tt}$ is the number of $i$-length paths from $t$ to $t$, which is at most $(n-1)^i$. Thus $\sum_{i=6}^{\infty}|(-\frac{1}{\kappa})^i \cdot (\mathbf{A}^i)_{tt}| \leq \sum_{i=6}^{\infty} \frac{1}{\kappa^i}(\mathbf{A}^i)_{tt} \leq \sum_{i=6}^{\infty} \frac{1}{\kappa^i}(n-1)^i \leq \frac{0.9}{\kappa^5}$.

Now observe that $(\mathbf{A}^0)_{tt} = \mathbf{I}_{tt} = 1$; that $\mathbf{A}_{tt} = 0$ since $H$ is a simple graph; that $(\mathbf{A}^2)_{tt} = \deg_H(t) = Y$, where the last equation follows from the definition of $Y$; that $(\mathbf{A}^3)_{tt} = 0$ since there is no triangle containing $t$; and that $(\mathbf{A}^4)_{tt} = \sum_{w:(w,t)\in E}\sum_{x:(x,w)\in E} 1 = \sum_{w:(w,t)\in E} \deg_{G_\mathbf{M}}(w) = \det_H(t) \cdot (n_0+1) = Y(n_0+1)$. Therefore,

- If $H$ contains a 5-length cycle incident to $t$, then $(\mathbf{A}^5)_{tt} \geq 2$, and thus $(\kappa \cdot \mathbf{B}^{-1})_{tt} \leq 1 + \frac{Y}{\kappa^2} + \frac{Y(n_0+1)}{\kappa^4} - \frac{2}{\kappa^5} + \frac{0.9}{\kappa^5} = 1 + \frac{Y}{\kappa^2} + \frac{Y(n_0+1)}{\kappa^4} - \frac{1.1}{\kappa^5}$
- If $H$ has no 5-length cycle incident to $t$, then $(\mathbf{A}^5)_{tt} = 0$, and thus $(\kappa \cdot \mathbf{B}^{-1})_{tt} \geq 1 + \frac{Y}{\kappa^2} + \frac{Y(n_0+1)}{\kappa^4} - \frac{0.9}{\kappa^5}$

This completes the proof of the claim. ◀

The following claim relates $s-t$ effective resistance to $\mathbf{B}^{-1}$. The proof almost follows from Lemma 23 in [41] (see also the full version for the proof).

▶ **Claim 17.** *Let $\Lambda = \mathcal{E}_G(s,t)$ and $\mathbf{B} = \kappa \cdot \mathbf{I} - \mathbf{A}$. Then it holds that $\Lambda = (\mathbf{B}^{-1})_{tt}$.*

Finally, by the above two claims, if $\mathbf{uMv} = 1$, then $H$ contains a 5-length cycle incident to $t$, and thus $\Lambda = (\mathbf{B}^{-1})_{tt} \leq \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{1.1}{\kappa^6}$; if $\mathbf{uMv} = 0$, then $H$ does not contain any 5-length cycle incident to $t$, and thus $\Lambda = (\mathbf{B}^{-1})_{tt} \geq \frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{0.9}{\kappa^6}$. The statement of the lemma then follows from our assumption that $\lambda = \Lambda$.

Note that our lower bound actually holds if $\lambda$ is a $1 + \frac{1}{\kappa^6} = 1 + O(\frac{1}{n^{36}})$-approximation of $\Lambda$, by the above analysis and the inequality $\frac{1}{\kappa^6}(\frac{1}{\kappa} + \frac{Y}{\kappa^3} + \frac{Y(n_0+1)}{\kappa^5} - \frac{0.9}{\kappa^6}) < \frac{0.1}{\kappa^6}$. ◀

## References

1 Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *Proc. of the 57th FOCS*, pages 477–486, 2016.

2 Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proc. of the 44th STOC*, pages 1199–1218, 2012.

3 Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In *Proc. of the 57th FOCS*, pages 335–344, 2016.

4 Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *Proc. of the 56th FOCS*, pages 20–39, 2015.

5 Daniel K Blandford, Guy E Blelloch, and Ian A Kash. Compact representations of separable graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 679–688. Society for Industrial and Applied Mathematics, 2003.

6 James R Bunch and John E Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

7 Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. Efficient sampling for gaussian graphical models via spectral sparsification. In *Conference on Learning Theory*, pages 364–390, 2015.

8 Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proc. of the 43rd STOC*, pages 273–282, 2011.

9 Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m\log^{1/2}n$ time. In *Proc. of the 46th STOC*, pages 343–352, 2014.

10 Krzysztof Diks and Piotr Sankowski. Dynamic plane transitive closure. In *European Symposium on Algorithms*, pages 594–604. Springer, 2007.

11 Michael Dinitz, Robert Krauthgamer, and Tal Wagner. Towards resistance sparsifiers. In *Proc. of the 18th APPROX*, pages 738–755, 2015.

12 Peter G Doyle and J Laurie Snell. *Random Walks and Electric Networks.* Carus Mathematical Monographs. Mathematical Association of America, 1984.

13 David Durfee, Yu Gao, Gramoz Goranci, and Richard Peng. Fully Dynamic Effective Resistances. *ArXiv e-prints*, apr 2018. `arXiv:1804.04038`.

14 David Durfee, Rasmus Kyng, John Peebles, Anup B Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 730–742. ACM, 2017.

15 David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Thomas H. Spencer. Separator based sparsification. i. planary testing and minimum spanning trees. *J. Comput. Syst. Sci.*, 52(1):3–27, 1996.

16 Gramoz Goranci, Monika Henzinger, and Pan Peng. The power of vertex sparsifiers in dynamic graph algorithms. In *Proc. of the 25th ESA*, volume 87, pages 45:1–45:14, 2017.

17 Gramoz Goranci, Monika Henzinger, and Pan Peng. Dynamic effective resistances and approximate schur complement on separable graphs. *CoRR*, abs/1802.09111, 2018. `arXiv:1802.09111`.

18 Prahladh Harsha, Thomas P Hayes, Hariharan Narayanan, Harald Räcke, and Jaikumar Radhakrishnan. Minimizing average latency in oblivious routing. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 200–207. Society for Industrial and Applied Mathematics, 2008.

19 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. of the 47th STOC*, pages 21–30, 2015.

20 Jacob Holm, Giuseppe F Italiano, Adam Karczmarz, Jakub Łacki, Eva Rotenberg, and Piotr Sankowski. Contracting a planar graph efficiently. In *25th European Symposium on Algorithms, ESA 2017*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2017.

21 Oscar H Ibarra, Shlomo Moran, and Roger Hui. A generalization of the fast lup matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45–56, 1982.

22 Giuseppe F Italiano, Adam Karczmarz, Jakub Łącki, and Piotr Sankowski. Decremental single-source reachability in planar digraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1108–1121. ACM, 2017.

23 Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proc. of the 43rd STOC*, pages 313–322, 2011.

24 Arun Jambulapati and Aaron Sidford. Efficient $\tilde{O}(n/\varepsilon)$ spectral sketches for the laplacian and its pseudoinverse. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2487–2503. SIAM, 2018.

25 Adam Karczmarz. Decrementai transitive closure and shortest paths for planar digraphs and beyond. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 73–92. SIAM, 2018.

26 Ken-ichi Kawarabayashi and Bruce Reed. A separator theorem in minor-closed classes. In *Proc. of the 51st FOCS*, pages 153–162. IEEE, 2010.

27 Jonathan A Kelner, Gary L Miller, and Richard Peng. Faster approximate multicommodity flow using quadratically coupled flows. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1–18. ACM, 2012.

28 Douglas J Klein and Milan Randić. Resistance distance. *Journal of mathematical chemistry*, 12(1):81–95, 1993.

29 Philip N. Klein and Sairam Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998.

30 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proc. of the 48th STOC*, 2016.

31 Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In *Proc. of the 57th FOCS*, pages 573–582, 2016.

32 Huan Li, Stacy Patterson, Yuhao Yi, and Zhongzhi Zhang. Maximizing the Number of Spanning Trees in a Connected Graph. *ArXiv e-prints*, 2018. `arXiv:1804.02785`.

33 Huan Li and Zhongzhi Zhang. Kirchhoff index as a measure of edge centrality in weighted networks: Nearly linear time algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2377–2396. SIAM, 2018.

34 Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979.

35 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

36 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Proc. of the 54th FOCS*, pages 253–262, 2013.

37 Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *Proc. of the 57th FOCS*, pages 593–602, 2016.

38 Aleksander Mądry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 2019–2036. Society for Industrial and Applied Mathematics, 2015.

39 Gary L. Miller and Richard Peng. Approximate maximum flow on separable undirected graphs. In *Proc. of the 24th SODA*, pages 1151–1170, 2013.

40 Gary L Miller, Shang-Hua Teng, William Thurston, and Stephen A Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM (JACM)*, 44(1):1–29, 1997.

41 Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P. Woodruff. Spectrum Approximation Beyond Fast Matrix Multiplication: Algorithms and Hardness. *LIPIcs*, 94:8:1–8:21, 2018.

42 Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 509–517. IEEE Computer Society, 2004.

43 Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. *arXiv preprint arXiv:1711.06455*, 2017.

44 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

45 Sairam Subramanian. A fully dynamic data structure for reachability in planar digraphs. In *Proc. of the 1st ESA*, pages 372–383, 1993.

46 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.