The Power of Vertex Sparsifiers in Dynamic Graph Algorithms*

Gramoz Goranci[1], Monika Henzinger[2], and Pan Peng[3]

1 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
`gramoz.goranci@univie.ac.at`
2 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
`monika.henzinger@univie.ac.at`
3 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
`pan.peng@univie.ac.at`

Abstract

We introduce a new algorithmic framework for designing dynamic graph algorithms in minor-free graphs, by exploiting the structure of such graphs and a tool called *vertex sparsification*, which is a way to compress large graphs into small ones that well preserve relevant properties among a subset of vertices and has previously mainly been used in the design of approximation algorithms.

Using this framework, we obtain a Monte Carlo randomized fully dynamic algorithm for $(1+\varepsilon)$-approximating the energy of electrical flows in n-vertex planar graphs with $\tilde{O}(r\varepsilon^{-2})$ worst-case update time and $\tilde{O}((r+\frac{n}{\sqrt{r}})\varepsilon^{-2})$ worst-case query time, for any r larger than some constant. For $r = n^{2/3}$, this gives $\tilde{O}(n^{2/3}\varepsilon^{-2})$ update time and $\tilde{O}(n^{2/3}\varepsilon^{-2})$ query time. We also extend this algorithm to work for minor-free graphs with similar approximation and running time guarantees. Furthermore, we illustrate our framework on the all-pairs max flow and shortest path problems by giving corresponding dynamic algorithms in minor-free graphs with both sublinear update and query times. To the best of our knowledge, our results are the first to systematically establish such a connection between dynamic graph algorithms and vertex sparsification.

We also present both upper bound and lower bound for maintaining the energy of electrical flows in the incremental subgraph model, where updates consist of only vertex activations, which might be of independent interest.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Dynamic graph algorithms, electrical flow, minor-free graphs, max flow

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.45

1 Introduction

A *dynamic graph* is a graph that undergoes constant changes over time. Such changes or updates may correspond to inserting/deleting an edge or activating/deactivating a vertex from the graph. The goal of a *dynamic graph algorithm* is to maintain some property of a graph and support an intermixed sequence of update and query operations that can be processed quickly. In particular, the algorithm should at least beat the trivial one that recomputes the solution from scratch after each update. The last three decades have witnessed a large body of research on dynamic graph algorithms for a number of fundamental

* 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.

properties, including connectivity, minimum spanning tree, shortest path, matching and so on. Most of these problems have been considered in both general graphs as well as planar graphs, with quite different techniques and trade-offs between update and query times. In particular, many dynamic algorithms for planar graphs heavily depend on the duality of planar graphs [41, 26, 19] and do not seem easily generalizable to a larger class of graphs, e.g., the family of minor-free graphs.

In this paper, we provide a new algorithmic framework for designing dynamic graph algorithms in *minor-free* graphs that are free of a K_t-minor for any fixed integer $t \geq 1$, by utilizing a tool called *vertex sparsification* as well as the structure of minor-free graphs. Vertex sparsification is a way of compressing large graphs into smaller ones that well preserve the relevant properties (e.g., cut, flow and distance information) among a subset of vertices (called *terminals*) (e.g., [36, 7, 29]). Besides the natural motivation of achieving more space-efficient storage and obtaining faster algorithms on the reduced graphs, it has also found applications in the design of approximation algorithms [36], network design and routing [11]. We show that good quality and efficiently constructible vertex sparsifiers can be used to give efficient dynamic graph algorithms. To the best of our knowledge, our results are the first to systematically establish such a connection between dynamic graph algorithms and vertex sparsification.

We illustrate our algorithmic framework on the *all-pairs electrical flow, all-pairs max flow* and *all-pairs shortest path* problems in minor-free graphs. Previously, there is no known dynamic algorithms for the first problem (even for special class of graphs), and for the second problem, we only know dynamic algorithms for planar graphs. Due to space constraints, we focus on electrical flow in the conference version of the paper and give the results for max flow and shortest path in the full version.

The *electrical flow* problem is one of the most fundamental problems in electrical engineering and physics [12], and recently received increasing interest in computer science due to its close relation to linear equation solvers [40, 28], graph sparsification [39, 38], maximum flows (and minimum cuts) [10, 31, 33, 22, 34]. Slightly more formally, the $s-t$ electrical flow problem asks to find the flow (current) that minimizes the energy dissipation of a weighted graph when one unit of flow is injected at the source s and extracted at the sink t.

In the dynamic *all-pairs electrical flow* problem, our objective is to minimize the update time and the query time for outputting the exact (or approximate) energy of the $s-t$ electrical flow (see Section 2 for formal definitions) in the current graph, for any two vertices s, t. In the following, we will focus on *fully dynamic* graphs, in which updates consist of both edge insertions and deletions. Specifically, we allow the following operations:

- $\textsc{Insert}(u, v, r)$: Insert the edge (u, v) with resistance r in G, provided that the new edge preserves the planarity (or minor-freeness) of G.
- $\textsc{Delete}(u, v)$: Delete the edge (u, v) from G.
- $\textsc{ElectricalFlow}(s, t)$: Return the exact (or approximate) energy of the $s-t$ electrical flow in the current graph G.

For a graph G and two vertices s, t, we let $\mathcal{E}_G(s, t)$ denote the energy of the $s-t$ electrical flow. For any $\alpha \geq 1$, we say that an algorithm is an α-approximation to $\mathcal{E}_G(s, t)$ if $\textsc{ElectricalFlow}(s, t)$ returns a positive number k such that $\mathcal{E}_G(s, t) \leq k \leq \alpha \cdot \mathcal{E}_G(s, t)$.

1.1 Our Results

We present the first non-trivial fully dynamic algorithm for maintaining a $(1+\varepsilon)$-approximation to the energy of the $s-t$ electrical flow in planar and minor-free graphs. Our algorithm achieves both sublinear worst-case query and update times. (Throughout the paper, we

use $\tilde{O}(\cdot)$ to hide polylogarithmic factors, i.e., $\tilde{O}(f(n)) = O(f(n) \cdot \operatorname{poly}\log f(n))$; "with high probability" refers to "with probability at least $1 - \frac{1}{n^c}$, for some $c > 0$".)

▶ **Theorem 1.** *Fix $\varepsilon \in (0,1)$ and integer $t > 0$. Let $r \geq c$ for some large constant $c > 0$. Given a K_t-minor-free graph $G = (V, E, \mathbf{w})$ with positive edge weights, we can maintain a $(1+\varepsilon)$-approximation to the all-pairs electrical flow problem with high probability. The worst-case update time per operation is $\tilde{O}(\frac{n^\xi \cdot r}{\varepsilon^2})$ and the worst-case query time is $\tilde{O}((r + n/\sqrt{r})\varepsilon^{-2})$, for any constant $\xi > 0$. Furthermore, if G is planar, then ξ can be chosen to be 0.*

Note that by setting $r = n^{2/3}$, we obtain a dynamic algorithm for planar graphs with worst-case $\tilde{O}(n^{2/3}\varepsilon^{-2})$ update time and $\tilde{O}(n^{2/3}\varepsilon^{-2})$ query time. One may be tempted to reduce our problem to dynamically maintaining *spectral sparsifiers*. Despite the fact that such sparsifiers approximately preserve electrical flows and that a $(1 \pm \varepsilon)$-spectral sparsifier can be maintained with amortized update time $\operatorname{poly}(\log n, \varepsilon^{-1})$ [5], performing query operations on the sparsifier about the energy of $s - t$ electrical flow still requires $\Omega(n)$ time.

We also give a dynamic algorithm for all-pairs electrical flow for minor-free graphs in the *incremental subgraph* model, where the updates in the dynamic graph are a sequence of *vertex activation* operations. Our algorithm maintains a $(1+\varepsilon)$-approximation of the energy of electrical flows in minor-free graphs with $\tilde{O}(r\varepsilon^{-2})$ *amortized* update time and $\tilde{O}((r + n/\sqrt{r})\varepsilon^{-2})$ *worst-case* query time, for any $r \geq c$. For $r = n^{2/3}$, this gives $\tilde{O}(n^{2/3}\varepsilon^{-2})$ amortized update and worst-case query time. We complement this result by showing the following conditional lower bound: there is no incremental algorithm in the subgraph model that C-approximates the energy of the electrical flows in *general* graphs with both $O(n^{1-\varepsilon})$ worst-case update time and $O(n^{2-\varepsilon})$ worst-case query time, for any $C > 0$ and constant $\varepsilon > 0$, unless the *online matrix vector multiplication (oMv)* conjecture if false. Our results show a polynomial gap of dynamic algorithms for subgraph electrical flows between minor-free graphs and general graphs, conditioned on the oMv conjecture. These results might be of independent interest and the details are deferred to the full version, due to space constraints.

Our second result from our algorithmic framework is a dynamic algorithm for all-pairs max flows in minor free graphs. In this problem, the query $\textsc{MaxFlow}(s,t)$ asks the exact (or approximate) $s - t$ max flow value in the current graph.

▶ **Theorem 2.** *Let $t > 0$ be a fixed integer. Let $r \geq c$ for some large constant $c > 0$. Given a K_t-minor-free graph $G = (V, E, \mathbf{w})$ with positive edge weights, we can maintain a $O(1)$-approximation to the all-pairs max-flows problem. The worst-case update time is $\tilde{O}(n^\xi r + r^3)$ for any constant $\xi > 0$, and the worst-case query time is $\tilde{O}(r + n/\sqrt{r})$. One can also maintain a $O(\log^4 n)$-approximation to the all-pairs max-flows problem, with worst-case update time $\tilde{O}(n^\xi r + r)$, and worst-case query time $\tilde{O}(r + n/\sqrt{r})$.*

Note that by setting $r \in \left(\operatorname{poly}\log n, o(n^{1/3})\right)$, we can maintain a $O(1)$-approximation to the all-pairs max flows problem in minor free graphs with both sublinear worst-case update and query times. By setting $r = n^{2/3}$, we can obtain $O(\log^4 n)$-approximation with worst-case $\tilde{O}(n^{\xi+2/3})$ update time and $\tilde{O}(n^{2/3})$ query time.

We remark that Italiano et al. [19] have given a fully dynamic algorithm for *exact* all-pairs max-flow in planar graphs with worst-case $\tilde{O}(n^{2/3})$ update and $\tilde{O}(n^{2/3})$ query time. Their algorithm is based on maintaining an edge decomposition (called *r-division*) of the planar graph, which is similar to ours, while there are some substantial differences. First of all, their algorithm does not seem easily generalizable to minor-free graphs since it depends on the duality of planar graphs. Second, it is required that the embedding of the graph does not change throughout the sequence of updates [19], which is not necessary in our algorithm.

Third, though their algorithm can answer the exact max flow value with the aforementioned running time guarantee, it does not provide an update/query trade-off as ours.

Our third result is a fully dynamic algorithm for all-pairs shortest paths in minor-free graphs. In this problem, the query ShortestPath(s,t) asks the exact (or approximate) shortest path length between s and t in the current graph.

▶ **Theorem 3.** *Let $t, q \geq 1$. Given a K_t-minor-free graph $G = (V, E, \mathbf{w})$ with positive edge weights, we can maintain a $(2q-1)$-approximation to the all-pair shortest path problem. The worst-case expected update time is $\tilde{O}(n^{6/7})$ and the worst-case query time is $\tilde{O}(n^{\frac{6}{7}+\frac{3}{7q}})$.*

Note that for $q \geq 4$, both update time and query time of the above algorithm will be sublinear. We remark that for the special case of planar graphs, the above running time and approximation guarantee are worse than the result of Abraham et al. [3], who gave a fully dynamic $(1+\varepsilon)$-approximation for planar shortest path problem with worst-case $\tilde{O}(\sqrt{n})$ update and query time. However, it is unclear how to generalize their algorithm to minor-free graphs. There are also works on fully dynamic all-pairs shortest path in general graphs (e.g., [9, 4]), for which there is no known algorithm with non-trivial worst-case update time that breaks $O(n)$ barrier.

We want to point it out that all the above results might be generalized to a larger class of graphs that admit efficiently constructible good separators, while our main focus is to bring up this new algorithmic framework. Due to space constraints, the proofs of Theorem 2 and 3 are deferred to the full version.

1.2 Our Techniques

Our fully dynamic algorithms in planar and minor-free graphs combine the ideas of maintaining an edge decomposition of the current graph G and approximately preserving the relevant properties or quantities by smaller "substitutes", which allow us to operate on a small piece of the graph during each update (in the amortized sense) and significantly reduce the size of the query graph that well preserves the property of G. These "substitutes" refer to the vertex sparsifiers for the corresponding properties.

Such an edge decomposition is called r-division [15]. Given some graph G and a parameter r, we partition G into a collection of $O(n/r)$ edge disjoint subgraphs (called *regions*), each contains at most $O(r)$ vertices. This induces a partitioning of the vertex set into *interior* vertices (those that are incident only to vertices within the same region) and *boundary* vertices (those that are incident to vertices in different regions). In addition, we ensure that the total number of boundary vertices is $O(n/\sqrt{r})$. Maintaining an r-division has also been used in some previous dynamic algorithms for planar graphs [41, 26, 16, 19].

Now a key observation is that for any s, t, by removing all the interior vertices from other regions that do not contain s, t and adding some edges with appropriate weights among boundary vertices, one can guarantee that the resulting graph exactly preserves the quantities between s, t (e.g., the energy of $s-t$ electrical flow, the value of $s-t$ max flow). Now let us elaborate on the electrical flow problem, for which the aforementioned reduction is called *Schur Complement*. The problem of performing such a Schur Complement on a region is that it is very time-consuming as it adds too many edges among boundary vertices. Instead, we resort to a recent tool called approximate Schur Complement ([13]; see Section 3.1), which well approximates the pairwise effective resistances among boundary vertices and also gives a sparse graph (or a substitute) induced by all boundary vertices. Now for an update, we only recompute a constant number of such substitutes (and we need to periodically rebuild the data structure); for a query, we take the small graph defined by choosing appropriate regions

and substitutes, and answer the query according to the $s-t$ effective resistance on this small graph. Since such a substitute can be computed very fast and is sparse, we are ensured to obtain sublinear amortized update time and worst-case query time. Using a global rebuilding technique, we show that one can also achieve worst-case update time.

Our approach differs from the previous dynamic planar graph algorithms in that the r-division we use does not require that the boundary of each region contains a constant number of faces or the duality of planar graphs, since we only need to maintain the r-division and fast compute the approximate Schur Complement.

Such an approximate Schur Complement can be viewed as a vertex *spectral/resistance sparsifier* by treating boundary vertices as terminals. To obtain dynamic algorithms for the all-pairs max flows (resp., shortest paths) problems, we can use vertex cut sparsifiers (resp., distance sparsifiers), which well preserve the values of minimum cut separating any subset of terminals (resp., the distances among all terminal pairs).

1.3 Related Work

In the static setting, the electrical flow problem amounts to solving a system of linear equations, where the underlying matrix is a Laplacian (see the monograph of Doyle and Snell [12]). Christiano et al. [10] used the electrical flow computation as a subroutine within the multiplicative-weights update framework [8], to obtain the breakthrough result of $(1-\varepsilon)$-approximating the undirected maximum $s-t$ flow (and the minimum $s-t$ cut) in $\tilde{O}(mn^{1/3}\varepsilon^{-11/3})$ time. This has inspired and led to further development of fast algorithms for approximating $s-t$ maximum flow, which culminated in an $\tilde{O}(m)$ time algorithm for this problem in undirected graphs [37].

Lipton, Rose and Tarjan [32] consider the problem of designing fast algorithms for *exactly* solving linear systems where the matrix is positive definite and the associated graph is planar. Their result implies an $O(n^{3/2})$ time algorithm for electrical flow in planar graphs. This was latter improved to $O(n^{\omega/2})$ by Alon and Yuster [6], where $\omega = 2.37..$ is the exponent in the running time of the fastest algorithm for matrix multiplication [42]. Miller and Koutis [27] consider parallel algorithms for approximately solving planar Laplacian systems. Their algorithm runs in $\tilde{O}(n^{1/6+c})$ parallel time and $O(n)$ work, where c is any positive constant. We refer the reader to [24] for other useful properties of Laplacians on planar graphs.

Related data structure concepts dealing with spectral properties of graphs include semi-streaming and dynamic algorithms for maintaining spectral sparsifiers. Kelner and Levin [23] give single-pass incremental streaming algorithm using near-linear space and total update time. This was extended by Kapralov et al. [20] to the dynamic semi-streaming model which allows both edge insertions and deletions. Recently, Abraham et al. [5] give a fully-dynamic algorithm for maintaining spectral sparsifiers in poly-logarithmic amortized update time.

There is a line of work on dynamic algorithms for planar graphs that maintains information about important measures like reachability, connectivity, shortest path, max-flow etc. Subramanian [41] shows a fully-dynamic algorithm for maintaining reachability in directed planar graphs in $O(n^{2/3}\log n)$ time per operation. For the connectivity measure, Eppstein et al. [14] give an algorithm with $O(\log^2 n)$ amortized update time and $O(\log n)$ query time. Dynamic all-pairs shortest path problem in planar graphs was initiated by Klein and Subramanian [26], who showed how to maintain a $(1+\varepsilon)$-approximation to shortest paths in $O(n^{2/3}\log^2 n \log D)$ amortized update time and $O(n^{2/3}\log^2 n \log D)$ worst-case query time, where D denotes the sum of edge lengths. The best known algorithm is due to Abraham, Chechik and Gavoille [3] and maintains a $(1+\varepsilon)$-approximation in $O(\sqrt{n}\log^2 n/\varepsilon)$ worst-case time per operation. Italiano et al. [19] obtain a fully-dynamic algorithm for *exact* $s-t$ max-flow in planar graphs with $O(n^{2/3}\log^{8/3})$ worst-case time per operation.

Motivated by the recent developments on proving conditional lower-bounds for dynamic problems [2, 18], Abboud and Dahlgaard [1] give conditional lower-bounds for a class of dynamic graph problems restricted to planar graphs. Specifically, under the conjecture that all-pair-shortest path problem cannot be solved in truly subcubic time, they show that no algorithm for dynamic shortest path in planar graphs can support both updates and queries in $O(n^{1/2-\varepsilon})$ amortized time, for $\varepsilon > 0$.

2 Preliminaries

We consider a weighted undirected graph G undergoing edge insertions/deletions or vertex activations/deactivations. Our dynamic algorithms are characterized by two time measures: *query time*, which denotes the time needed to answer a query, and *update time*, which denotes the time needed to perform an update operation. We say that an algorithm has $O(t(n))$ *worst-case* update time, if it takes $O(t(n))$ time to process *each* update. We say that an algorithm has $O(t(n))$ *amortized* update time if it takes $O(f \cdot t(n))$ total update time for processing f updates (edge insertions/deletions or vertex activations/deactivations).

Basic Definitions. Let $G = (V, E, \mathbf{w})$ be any undirected weighted graph with n vertices and m edges, where for any edge e, its weight $\mathbf{w}(e) > 0$. Let $\mathbf{A}$ denote the weighted adjacency matrix, let $\mathbf{D}$ denote the weighted degree diagonal matrix, and let $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denote the *Laplacian* matrix of G. We fix an arbitrary orientation of edges, that is, for any two vertices u, v connected by an edge, exactly one of $(u, v) \in E$ or $(v, u) \in E$ holds. Let $\mathbf{B} \in \mathbb{R}^{m \times n}$ denote the incidence matrix of G such that for any edge $e = (u, v)$ and vertex $w \in V$, $\mathbf{B}((u, v), w) = 1$ if $u = w$, -1 if $v = w$, and 0 otherwise. We will also think of the weight $\mathbf{w}(e)$ of any edge e as the *conductance* of e, and its reciprocal $\frac{1}{\mathbf{w}(e)}$, denoted as $\mathbf{r}(e)$, as the *resistance* of e. Let $\mathbf{R} \in \mathbb{R}^{m \times m}$ denote a diagonal matrix with $\mathbf{R}(e, e) = \mathbf{r}(e)$, for any edge e. Note that $\mathbf{L} = \mathbf{B}^T \mathbf{R}^{-1} \mathbf{B}$.

For any $\mathbf{x} \in \mathbb{R}^n$, the quadratic form associated with $\mathbf{L}$ is given by $\mathbf{x}^T \mathbf{L} \mathbf{x}$. For any two different vertices u, v, let $\chi_{u,v} \in \mathbb{R}^n$ denote the vector such that $\chi_{u,v}(w) = 1$ if $w = u$, -1 if $w = v$ and 0 otherwise. For any two vertices $s, t \in V$, an $s - t$ *flow* is a mapping $\mathbf{f} : E \to \mathbb{R}^+$ satisfying the following conservation constraint: for any $v \neq s, t$, it holds that $\sum_{e=(v,u)} \mathbf{f}(e) = \sum_{e=(u,v)} \mathbf{f}(e)$, where for any edge $e = (v, u)$, $\mathbf{f}(e) := \mathbf{f}(v, u)$ and $\mathbf{f}(u, v) := -\mathbf{f}(v, u)$.

We will let $\mathrm{val}(\mathbf{f}) = \sum_{v:(s,v)\in E} \mathbf{f}(s, v)$ denote the *value of an* $s - t$ *flow*. Note that for an $s - t$ flow with value 1, it holds that $\mathbf{B}^T \mathbf{f} = \chi_{s,t}$. Given an $s - t$ flow $\mathbf{f}$, its *energy* (with respect to the resistance vector $\mathbf{r}$) is defined as $\mathcal{E}_{\mathbf{r}}(\mathbf{f}, s, t) = \sum_e \mathbf{r}(e)\mathbf{f}(e)^2 = \mathbf{f}^T \mathbf{R} \mathbf{f}$.

We define the $s - t$ *electrical flow* in G to be the $s - t$ flow that minimizes the energy $\mathcal{E}_{\mathbf{r}}(\mathbf{f}, s, t)$ among all $s - t$ flows with *unit* flow value. It is known that such a flow is unique [12].

Any $s - t$ flow $\mathbf{f}$ in G is an $s - t$ electrical flow with respect to $\mathbf{r}$, iff there exists a vertex potential function $\phi : V \to \mathbb{R}^+$ such that for any $e = (u, v)$ that is oriented from u to v, $\mathbf{f}(e) = \frac{\phi(v) - \phi(u)}{\mathbf{r}(e)}$. It is known that such a vector ϕ satisfies that $\phi = \mathbf{L}^\dagger \chi_{s,t}$, where $\mathbf{L}^\dagger$ denotes the (Moore-Penrose) pseudo-inverse of $\mathbf{L}$. In addition, $\mathbf{f} = \mathbf{R}^{-1} \mathbf{B}^T \phi = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{L}^\dagger \chi_{s,t}$ [12].

The *effective* $s - t$ *resistance* $R_G(\mathbf{r}, s, t)$ of G with respect to the resistances $\mathbf{r}$ is the potential difference between s, t when we send one unit of electrical flow from s to t. That is, $R_G(\mathbf{r}, s, t) = \phi(s) - \phi(t) = \chi_{s,t}^T \mathbf{L}^\dagger \chi_{s,t}$, where ϕ is the vector of vertex potentials induced by the $s - t$ electrical flow of value 1. We will often denote $R_G(\mathbf{r}, s, t)$ by $R_G(s, t)$ when $\mathbf{r}$ is clear from the context. It is known that the effective $s - t$ resistance is equal to the energy of the $s - t$ electrical flow of value 1, that is $R_G(\mathbf{r}, s, t) = \mathcal{E}_{\mathbf{r}}(\mathbf{f}, s, t)$.

Graph r-Divisions. Let $G = (V, E)$ be a graph. Let $F \subset E$ be a subset of edges. We call the subgraph G_F induced by all edges in F a *region*. For a subgraph P of G, any vertex that is incident to vertices not in P is called a *boundary* vertex. The *vertex boundary* of P, denoted by $\partial_G(P)$ is the set of boundary vertices belonging to P. All other vertices in P will be called *interior vertices* of P.

▶ **Definition 4.** Let $c_1, c_2 > 0$ be some constant. For any $r \in (1, n)$, a *weak r-division* (with respect to c_1, c_2) of an n-vertex graph G is an edge partition of it into regions $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, where $\ell \leq c_1 \cdot \frac{n}{r}$ such that

- Each edge belongs to exactly one region.
- Each region P_i contains r vertices.
- The total number of all boundary vertices, i.e., $\cup_i \partial_G(P_i)$, is at most $c_2 n/\sqrt{r}$.

It is known that such an r-division (even with the stronger guarantee that each region has $O(\sqrt{r})$ boundary vertices) for planar graphs can be constructed in linear time [17, 25, 15].

▶ **Lemma 5** ([25]). *Let $c > 0$ be some constant. There is an algorithm that takes as input an n-vertex planar graph G and for any $r \geq c$, outputs an r-division of G in $O(n)$ time.*

We will need the following property on the boundary vertices of the r-division output by the above algorithm (see Section 3.3 in [25]).

▶ **Lemma 6** ([25]). *For an n-vertex planar graph G, let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, $\ell = O(n/r)$ be the r-division by the algorithm in Lemma 5. Then it holds that $\sum_{i=1}^{\ell} |\partial_G(P_i)| = O(n/\sqrt{r})$.*

Graph Sparsification. Graph Sparsification aims at compressing large graphs into smaller ones while (approximately) preserving some characteristics of the original graph. We present two notions of sparsification. The first requires that the quadratic form of the large and sparsified graph are close. The second requires that all-pairs effective resistances of the corresponding graphs are close.

▶ **Definition 7** (Spectral Sparsifier). Let $G = (V, E, \mathbf{w})$ be a weighted graph and $\varepsilon \in (0, 1)$. A $(1 \pm \varepsilon)$-*spectral sparsifier* for G is a subgraph $H = (V, E_H, \mathbf{w}_H)$ such that for all $\mathbf{x} \in \mathbb{R}^n$, $(1 - \varepsilon)\mathbf{x}^T\mathbf{L}\mathbf{x} \leq \mathbf{x}^T\widetilde{\mathbf{L}}\mathbf{x} \leq (1 + \varepsilon)\mathbf{x}^T\mathbf{L}\mathbf{x}$, where $\mathbf{L}$ and $\widetilde{\mathbf{L}}$ are the Laplacians of G and H, respectively.

▶ **Definition 8** (Resistance Sparsifier). Let $G = (V, E, \mathbf{w})$ be a weighted graph and $\varepsilon \in (0, 1)$. A $(1 \pm \varepsilon)$-*resistance sparsifier* for G is a subgraph $H = (V, E_H, \mathbf{w}_H)$ such that for all $u, v \in V$, $(1-\varepsilon)R_H(u, v) \leq R_G(u, v) \leq (1+\varepsilon)R_H(u, v)$, where $R_G(u, v)$ and $R_H(u, v)$ denote the effective $u - v$ resistance in G and H, respectively.

We remark that Definition 7 implies approximations for the pseudoinverse Laplacians, that is

$$\forall \mathbf{x} \in \mathbb{R}^n \quad \frac{1}{(1+\varepsilon)}\mathbf{x}^T\mathbf{L}^\dagger\mathbf{x} \leq \mathbf{x}^T\widetilde{\mathbf{L}}^\dagger\mathbf{x} \leq \frac{1}{(1-\varepsilon)}\mathbf{x}^T\mathbf{L}^\dagger\mathbf{x},$$

Since by definition, the effective resistance between any two nodes u and v is the quadratic form defined by the pseudo-inverse of the Laplacian computed at the vector $\chi_{u,v}$, it follows that the effective resistances between any two nodes in G and H are the same up to a $(1 \pm \varepsilon)$ factor. By our definitions for resistance and spectral sparsifiers, we have the following fact.

▶ **Fact 9.** *Let $\varepsilon \in (0, 1)$ and let G be a graph. Then every $(1 \pm \varepsilon)$-spectral sparsifier of G is a $(1 \pm \varepsilon)$-resistance sparsifier of G.*

The following lemma says that given a graph, by decomposing the graph into several pieces, and computing a good sparsifier for each piece, then one can obtain a good sparsifier for the original graph which is the union of the sparsifiers for all pieces. The proof is deferred to the full version of the paper.

▶ **Lemma 10** (Decomposability). *Let $G = (V, E, \mathbf{w})$ be a weighted graph whose set of edges is partitioned into $E_1, \ldots, E_\ell$. Let H_i be a $(1 \pm \varepsilon)$-spectral sparsifier of $G_i = (V, E_i)$, where $i = 1, \ldots, \ell$. Then $H = \bigcup_{i=1}^{\ell} H_i$ is a $(1 \pm \varepsilon)$-spectral sparsifier of G.*

3 A Dynamic Algorithm for Electrical Flow in Minor-Free Graphs

In order to present our dynamic algorithm for electrical flows, we first introduce the notion of *approximate Schur Complement*.

3.1 Schur Complement as Vertex Resistance Sparsifier

In the previous section we introduced graph sparsification for reducing the number of edges. For our application, it will be useful to define sparsifiers that apart from reducing the number of edges, they also reduce the number of vertices. More precisely, given a weighted graph $G = (V, E, \mathbf{w})$ with terminal set $K \subset V$, we are looking for a graph $H = (V_H, E_H, \mathbf{w}_H)$ with $K \subseteq V_H$ and as few vertices and edges as possible while preserving some important feature among terminals vertices. Graph H is usually referred to as a *vertex sparsifier* of G.

Exact Schur Complement. We first review a folklore result [35] on constructing vertex sparsifiers that preserve effective resistances among terminal pairs. For sake of simplicity, we first work with Laplcians of graphs. For a given connected graph G as above, let $N = V \setminus K$ be the set of non-terminal vertices in G. The partition of V into N and K naturally induces the following partition of the Laplacian $\mathbf{L}$ of G into blocks:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_N & \mathbf{L}_M \\ \mathbf{L}_M^T & \mathbf{L}_K \end{bmatrix}$$

We remark that since G is connected and N and K are non-empty, $\mathbf{L}_N$ is invertible. We next define the Schur complement of $\mathbf{L}$, which can be viewed as an equivalent to $\mathbf{L}$ only on the terminal vertices.

▶ **Definition 11** (Schur Complement). The *Schur complement* of a graph Laplacian $\mathbf{L}$ with respect to a terminal set K is $\mathbf{L}_S^K := \mathbf{L}_K - \mathbf{L}_M^T \mathbf{L}_N^{-1} \mathbf{L}_M$.

It is known that the matrix $\mathbf{L}_S^K$ is a Laplacian matrix for some graph G' [30]. We can think of Schur Complement as performing Gaussian elimination on the non-terminals $V \setminus K$. This process recursively eliminates a vertex $v \in V \setminus K$ by deleting v and adding a clique with appropriate edge weights on the neighbors of v in the current graph (see, e.g. [30]). The following lemma shows that the quadratic form of the pseudo-inverse of the Laplacian $\mathbf{L}$ will be preserved by taking the quadratic form of the pseudo-inverse of its Schur Complement, for vectors supported on the terminals. See the full version for the proof.

▶ **Lemma 12.** *Let $\mathbf{d}$ be a vector of a graph G whose vertices are partitioned into terminals K, and non-terminals N and only terminals have non-zero entries in $\mathbf{d}$. Let $\mathbf{d}_K$ be the restriction of $\mathbf{d}$ on the terminals and let $\mathbf{L}_S^K$ be the Schur complement of the Laplacian $\mathbf{L}$ of G with respect to K. Then $\mathbf{d}^T \mathbf{L}^\dagger \mathbf{d} = \mathbf{d}_K^T (\mathbf{L}_S^K)^\dagger \mathbf{d}_K$.*

Using interchangeability between graphs and their Laplacians, we can interpret the above result in terms of graphs as well. We first present the following notion of sparsification.

▶ **Definition 13** (Vertex Resistance Sparsifier)**.** Let $G = (V, E, \mathbf{w})$ be a weighted graph with $K \subset V$ and $\alpha \geq 1$. An α-vertex resistance sparsifier of G with respect to K is a graph $H = (K, E_H, \mathbf{w}_H)$ such that for all $s, t \in K$, $R_H(s,t) \leq R_G(s,t) \leq \alpha \cdot R_H(s,t)$.

The lemma below relates the Schur Complement and resistance sparsifiers.

▶ **Lemma 14.** *Let $G = (V, E, \mathbf{w})$ be a weighted graph with $K \subset V$, Laplacian matrix $\mathbf{L}$ and Schur Complement $\mathbf{L}_S^K$ (with respect to the terminal set K). Then the graph $H = (K, E_H, \mathbf{w}_H)$ associated with the Laplacian $\mathbf{L}_S^K$ is a 1-vertex resistance sparsifier of G with respect to K.*

Proof. Fix some terminal pair (s,t) and consider the vectors $\chi_{s,t}$ and $\chi'_{s,t}$ of dimension n and k, respectively. Lemma 12, the definition of effective resistance and the fact that $\chi_{s,t}$ and $\chi'_{s,t}$ are valid vectors for $\mathbf{L}$ and $\mathbf{L}_S^K$ give: $R_G(s,t) = \chi_{s,t}^T \mathbf{L}^\dagger \chi_{s,t} = \chi'^T_{s,t} \mathbf{L}_S^{K\dagger} \chi'_{s,t} = R_H(s,t)$. ◀

Approximate Schur Complement. We need the following lemma due to Durfee et al. [13].

▶ **Lemma 15** ([13])**.** *Fix $\varepsilon \in (0, 1/2)$ and $\delta \in (0,1)$. Let $G = (V, E, \mathbf{w})$ be a weighted graph with n vertices, m edges. Let $K \subset V$ with $|K| = k$. Let $\mathbf{L}$ be the Laplacian of G and $\mathbf{L}_S^K$ be the corresponding Schur complement with respect to K. Then there is an algorithm* ApproxSchur$(G, K, \varepsilon, \delta)$ *that returns a Laplacian matrix $\widetilde{\mathbf{L}}_S^K$ with associated graph $\widetilde{H}$ on the terminals K such that the following statements hold with probability at least $1 - \delta$:*

1. *The graph $\widetilde{H}$ has $O(k\varepsilon^{-2}\log(n/\delta))$ edges.*
2. *$\mathbf{L}_S^K$ and $\widetilde{\mathbf{L}}_S^K$ are spectrally close, that is*

$$\forall \mathbf{x} \in \mathbb{R}^k \quad (1-\varepsilon)\mathbf{x}^T \mathbf{L}_S^K \mathbf{x} \leq \mathbf{x}^T \widetilde{\mathbf{L}}_S^K \mathbf{x} \leq (1+\varepsilon)\mathbf{x}^T \mathbf{L}_S^K \mathbf{x}.$$

The total running time for producing $\widetilde{H}$ is $\tilde{O}((n+m)\varepsilon^{-2}\log^4(n/\delta))$.

In the following, we call the Laplacian $\widetilde{\mathbf{L}}_S^K$ (or equivalently, the graph $\widetilde{H}$) satisfying the above two conditions an *approximate Schur Complement* of G with respect to K. Note that by definition, the graph $\widetilde{H}$ is a $(1 \pm \varepsilon)$-spectral sparsifier of the graph H that is associated with graph $\mathbf{L}_S^K$, which in turn is a 1-vertex resistance sparsifier of G with respect to K. Therefore, $\widetilde{H}$ is a $(1 \pm O(\varepsilon))$-vertex resistance sparsifier of G with respect to K (see Section 2).

3.2 Proof of Theorem 1

We now present a fully dynamic algorithm for maintaining the energy of electrical flows up to a $(1+\varepsilon)$ factor in minor-free graphs and prove Theorem 1. We start with the special case of planar graphs.

Data Structure. In our dynamic algorithm, we will maintain an r-division $\mathcal{P} = \{P_1, \cdots, P_\ell\}$ of G with $\ell = O(n/r)$ and for each region P_i, we compute a graph $\widetilde{H}_i$ by invoking the algorithm ApproxSchur in Lemma 15 with parameters P_i, $K = \partial_G(P_i)$, $\varepsilon = \frac{\varepsilon}{6}$ and $\delta = 1/n^3$.

Let $\mathcal{D}(G)$ denote such a data structure for G, and let $T_{\mathcal{D}(G)}$ denote the time to compute $\mathcal{D}(G)$. Note that by Lemma 5 and 15, $T_{\mathcal{D}(G)} = \tilde{O}(n + \frac{n}{r} \cdot r\varepsilon^{-2}) = \tilde{O}(n\varepsilon^{-2})$. Furthermore, note that there are at most $O(n/r)$ regions, and for each such a region P_i, the corresponding graph $\widetilde{H}_i$ is *not* an approximate Schur Complement of P_i with respect to its boundary $\partial_G(P_i)$ with probability at most $1/n^3$. Therefore, by the union bound, with probability at least $1 - n \cdot \frac{1}{n^3} = 1 - \frac{1}{n^2}$, for any $i \leq \ell$, the graph $\widetilde{H}_i$ is an approximate Schur Complement of P_i

with respect to $\partial_G(P_i)$, and thus a $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier of H_i, where H_i denotes the exact Schur complement of P_i with respect to $\partial_G(P_i)$. In the following, we will condition on this event. This data structure $\mathcal{D}(G)$ will be recomputed every $T_{\text{div}} := \Theta(n/r)$ operations.

Handling Edge Insertions/Deletions. We now describe the INSERT operation. Whenever we compute an approximate Schur Complement, we assume that the procedure APPROXSCHUR from Lemma 15 is invoked on the corresponding region and its boundary vertex set, with $\varepsilon = \frac{\varepsilon}{6}$ and error probability $\delta = 1/n^3$. Let us consider inserting an edge $e = (x, y)$.

- If both x, y belong to the same region, say P_i, then we add the edge e to P_i, and recompute an approximate Schur Complement $\widetilde{H}_i$ of the region P_i (with respect to its boundary vertex set) from scratch.
- If x and y do not belong to the same region, we do the following.
  - If x is an interior vertex of some region P_x, then adding an edge (x, y) will make x a boundary vertex. We then recompute an approximate Schur Complement $\widetilde{H}_x$ of P_x.
  - If y is an interior vertex of some region, then we handle it in the same way as we did for the interior vertex x.
  - We treat the edge (x, y) as a new region containing only this edge.

Observe that for each insertion, the number of vertices in any region is always at most r, and we perform only a constant number of calls to APPROXSCHUR, Lemma 15 implies that the time to handle an edge insertion is $\tilde{O}(r\varepsilon^{-2})$. Furthermore, since each edge insertion may increase by a constant the number of boundary nodes and the total number of regions.

We now describe the DELETE operation. If we delete some edge $e = (x, y)$, let P_i be the region such that both $x, y \in P_i$. We remove the edge from P_i, and then recompute an approximate Schur Complement $\widetilde{H}_i$ of P_i with respect to its boundary. By Lemma 15, the cost of this resparsification step is bounded by $\tilde{O}(r\varepsilon^{-2})$.

Since we recompute the data structure every $\Theta(n/r)$ operations, the amortized update time is $\tilde{O}\left(\frac{n\varepsilon^{-2}}{n/r} + r\varepsilon^{-2}\right) = \tilde{O}(r\varepsilon^{-2})$.

Handing Queries. In order to return a $(1+\varepsilon)$-approximation of the energy of $s-t$ electrical flow for an ELECTRICALFLOW(s, t) query, it suffices to return a $(1 - \frac{\varepsilon}{2})$-approximation of the effective $s - t$ resistance, for which we first need to review the static algorithm for computing effective resistance. The following result is due to Durfee et al. [13] (which builds and/or improves upon [10, 28, 39]).

▶ **Theorem 16** ([13]). *Fix $\varepsilon \in (0, 1/2)$ and let $G = (V, E, \mathbf{w})$ be a weighted graph with n vertices and m edges. There is an algorithm* EFFECTIVERESISTANCE *that computes a value ψ such that $(1-\varepsilon)R_G(s,t) \le \psi \le (1+\varepsilon)R_G(s,t)$, in time $\tilde{O}(m + \frac{n}{\varepsilon^2})$ with high probability.*

To answer the query ELECTRICALFLOW(s, t), we will form a smaller auxiliary graph that is the union of the regions containing s, t and the approximate Schur Complements of the remaining regions with respect to their boundaries, and output the approximate effective $s - t$ resistance of the smaller graph. More precisely, let P_s and P_t be two regions that contain s and t, respectively. Let $\mathcal{J}$ denote the index set of all the remaining regions, i.e., $\mathcal{J} = \{i : P_i \in \mathcal{P} \setminus \{P_s, P_t\}\}$. For each region P_i such that $i \in \mathcal{J}$, as before, let $\widetilde{H}_i$ be the approximate Schur Complement of P_i that we have maintained. Now we form an auxiliary graph H by taking the union over the regions P_s and P_t and all the approximate Schur Complements of the remaining regions, i.e., $H = P_s \cup P_t \cup \bigcup_{i \in \mathcal{J}} \widetilde{H}_i$. We then run the algorithm EFFECTIVERESISTANCE on H with $\varepsilon = \frac{\varepsilon}{6}$ to obtain an estimator ψ and return $c_H(s, t) := (1 - \frac{\varepsilon}{6})\psi$. Next we show that the returned value is a good approximation to the actual effective resistance.

▶ **Lemma 17.** *Fix $\varepsilon \in (0,1)$. Let $G = (V, E, \mathbf{w})$ be some current graph and $s, t \in V$. Further, let $H = P_s \cup P_t \cup \bigcup_{i \in \mathcal{J}} \widetilde{H}_i$ be defined as above and let $c_H(s,t)$ be the value returned as above by invoking* EffectiveResistance *on H. Then, with high probability, we get*

$$(1 - \frac{\varepsilon}{2})R_G(s,t) \leq c_H(s,t) \leq (1 - \frac{\varepsilon}{2})R_G(s,t).$$

Proof. For the sake of analysis, we divide the sequence of updates into intervals each consisting of $T_{\text{div}} = \Theta(n/r)$ operations. Let I be the interval in which the query is made. Let $G^{(0)}$ denote the graph at the beginning of I. We compute the data structure $\mathcal{D}(G^{(0)})$ of $G^{(0)}$, which contains an r-division $\mathcal{P}^{(0)}$ and the corresponding approximate Schur Complements $\widetilde{H}_i^{(0)}$. As mentioned before, with probability at least $1 - \frac{1}{n^2}$, each of the graphs $\widetilde{H}_i^{(0)}$ will be a $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier of the exact Schur Complement $H_i^{(0)}$of the corresponding region with respect to its boundary vertex set.

Let G be the current graph when the query is made, which is formed from $G^{(0)}$ after some updates in I. Let $\mathcal{P} = \{P_i\}_i, \widetilde{H}_i, 1 \leq i \leq O(n/r)$ be the r-division and the approximate Schur Complements in the current data structure, respectively. Let H_i denote the exact Schur Complement of the region P_i with respect to its boundary vertex set. Since the total number of updates in I is $\Theta(n/r)$, and each update only involves a constant number of invocations of ApproxSchur with error probability $1/n^3$ that recomputes the approximate Schur Complements of some regions, we have that with probability at least $1 - O(n/r) \cdot \frac{1}{n^3} \geq 1 - \frac{1}{n^2}$, these recomputed approximate Schur Complements are $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifiers of the corresponding exact Schur Complements. Therefore, for the current graph G and its data structure, with probability $1 - 2 \cdot \frac{1}{n^2} = 1 - \frac{2}{n^2}$, for all i, the graph $\widetilde{H}_i$ is a $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier of H_i. In the following, we will condition on this event.

Recall that P_s and P_t are two regions that contain s and t, respectively. Consider the graph $G' = P_s \cup P_t \cup \bigcup_{i \in \mathcal{J}} H_i$. We have the following lemma whose proof is deferred to the full version of the paper.

▶ **Lemma 18.** *For any two vertices $u, v \in V(G')$, it holds that $R_G(u,v) = R_{G'}(u,v)$.*

It follows from the above lemma that $R_G(s,t) = R_{G'}(s,t)$. We next argue that H is a $(1 \pm \frac{\varepsilon}{6})$-resistance sparsifier to G' with high probability. First, note that each of the subgraphs P_s, P_t, H_i and $\widetilde{H}_i$, $i \in \mathcal{J}$ can be treated as graphs defined on the same vertex set $V(G')$ with appropriate isolated vertices. Second, since for each $i \in \mathcal{J}$, $\widetilde{H}_i$ is $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier of H_i, and P_s, P_t are sparsifiers of itself, we know that by Lemma 10 about the decomposability of sparsifiers, H is a $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier of G'. Since every $(1 \pm \frac{\varepsilon}{6})$-spectral sparsifier is a $(1 \pm \frac{\varepsilon}{6})$-resistance sparsifier, it holds that

$$(1 - \frac{\varepsilon}{6})R_H(s,t) \leq R_{G'}(s,t) \leq (1 + \frac{\varepsilon}{6})R_H(s,t). \tag{1}$$

Since by definition we have $c_H(s,t) := (1 - \frac{\varepsilon}{6})\psi$, Theorem 16 implies that

$$(1 - \frac{\varepsilon}{6})^2 R_H(s,t) \leq (1 - \frac{\varepsilon}{6})\psi \leq (1 - \frac{\varepsilon}{6})(1 + \frac{\varepsilon}{6})R_H(s,t), \tag{2}$$

with high probability. Combining (1) and (2) we get

$$\frac{(1 - \frac{\varepsilon}{6})^2}{(1 + \frac{\varepsilon}{6})} R_{G'}(s,t) \leq (1 - \frac{\varepsilon}{6})\psi \leq (1 + \frac{\varepsilon}{6})R_{G'}(s,t),$$

which in turn along with $R_G(s,t) = R_{G'}(s,t)$ imply that,

$$(1 - \frac{\varepsilon}{2})R_G(s,t) \leq (1 - \frac{\varepsilon}{6})\psi \leq (1 + \frac{\varepsilon}{2})R_G(s,t).$$

Therefore, with high probability, the algorithm outputs a $(1-\frac{\varepsilon}{2})$-approximation to the effective $s-t$ resistance. ◀

To bound the query time, we need to bound the size of the $H = P_s \cup P_t \cup \bigcup_{i\in\mathcal{J}} \widetilde{H}_i$. As in the proof of Lemma 17, we let $G^{(0)}$ denote the graph right after the last rebuilding of the data structure. Let $\mathcal{P}^{(0)}$ denote the corresponding r-division. By definition, for each $P \in \mathcal{P}^{(0)}$, $|P| \leq r$ and the size of all the boundary vertices is $c_2 n/\sqrt{r}$. By Lemma 6, we have that $\sum_{P\in\mathcal{P}^{(0)}} |\partial_{G^{(0)}}(P)| \leq O(n/\sqrt{r})$, i.e., the sum of the numbers of boundary vertices over all regions of $G^{(0)}$ is at most $O(n/\sqrt{r})$.

Note that there will be at most $T_{\text{div}} = \Theta(n/r)$ updates between $G^{(0)}$ and G, the graph to which the query is performed, and each update can only increase the number of boundary vertices and the total number of regions by a constant. These facts imply that the size of all boundary nodes is $O(n/\sqrt{r})$. Therefore, we have that $|V(H)| \leq O(r + n/\sqrt{r})$, and that the sum of the numbers of boundary vertices of the regions of G is at most $O(n/\sqrt{r})$, i.e., $\sum_i |V(\widetilde{H}_i)| \leq O(n/\sqrt{r})$.

On the other hand, by Lemma 15, for each i, $|E(\widetilde{H}_i)| = O(|V(\widetilde{H}_i)| \cdot \varepsilon^{-2}\log n)$. Thus,

$$\begin{aligned} |E(H)| &\leq |E(P_s)| + |E(P_t)| + \sum_i |E(\widetilde{H}_i)| \leq O(r) + \sum_i |V(\widetilde{H}_i)| \cdot O(\varepsilon^{-2}\log n) \\ &= O((r + n/\sqrt{r})\varepsilon^{-2}\log n). \end{aligned}$$

By Theorem 16, it follows that the worst-case query time is $\tilde{O}((r + n/\sqrt{r})\varepsilon^{-2})$.

To achieve asymptotically the same worst-case update time, we use a standard global rebuilding technique (see the full version for details), which then finishes the proof of Theorem 1 when the input graph is planar.

Extension to Minor-Free Graphs. In the following, we briefly discuss how one can adapt the previous dynamic algorithms for planar graphs to minor-free graphs.

The key observation is that since the approximate Schur Complement can be constructed in nearly-linear time for any graph, it suffices for us to efficiently maintain an r-division of any minor-free graph, i.e., we need fast algorithms for computing a *separator* of order $\sqrt{n}$ in such graphs. (A separator is a subset S of vertices whose deletion will partition the graph into connected components, each of size at most $\frac{2n}{3}$). Kawarabayashi and Reed [21] showed that for any K_t-minor-free graph G, one can construct in $O(n^{1+\xi})$ time a separator of size $O(\sqrt{n})$ for G, for any constant $\xi > 0$ and constant t. (The $O(\cdot)$ notation for the running time hides huge dependency on t.) Applying this separator construction recursively as in Frederickson's algorithm [15], we can maintain an r-division of any K_t-minor-free G in $\tilde{O}(n^{1+\xi})$ time. Furthermore, by analysis in [15], it holds that the total sum of the sizes of all boundary vertex sets is also bounded by $O(\frac{n}{\sqrt{r}})$ as guaranteed by Lemma 6 for planar graphs.

Now we can dynamically maintain the data structure for electrical flows in minor-free graphs almost the same as we did for planar graphs, except that we use the above $\tilde{O}(n^{1+\xi})$ time algorithm to compute the r-divisions. Thus, the time to compute the data structure for any minor-free graph is then $\tilde{O}(n^{1+\xi} + n\varepsilon^{-2})$, for arbitrarily small constant $\xi > 0$. Then from previous analysis, the worst-case update time is $\tilde{O}(n^{\xi} r\varepsilon^{-2})$ update time, and the worst-case query time is $\tilde{O}((r + n/\sqrt{r})\varepsilon^{-2})$. This completes the proof of Theorem 1.

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 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. of the 55th FOCS*, pages 434–443, 2014.

3 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.

4 Ittai Abraham, Shiri Chechik, and Kunal Talwar. Fully dynamic all-pairs shortest paths: Breaking the $o(n)$ barrier. In *Proc. of the 17th APPROX*, 2014.

5 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.

6 Noga Alon and Raphael Yuster. Solving linear systems through nested dissection. In *Proc. of the 51st FOCS*, pages 225–234, 2010.

7 Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. In *Proc. of the 25th SODA*, pages 279–293, 2014.

8 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

9 Aaron Bernstein. Fully dynamic $(2+\epsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time. In *Proc. of the 50th FOCS*, pages 693–702, 2009.

10 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.

11 Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *Proc. of the 44th STOC*, pages 855–874. ACM, 2012.

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

13 David Durfee, Rasmus Kyng, John Peebles, Anup B. Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proc. of the 49th STOC*, pages 730–742, 2017.

14 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.

15 Greg N Federickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.

16 Zvi Galil, Giuseppe F. Italiano, and Neil Sarnak. Fully dynamic planarity testing with applications. *J. ACM*, 46(1):28–91, 1999.

17 MT Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and System Sciences*, 3(51):374–389, 1995.

18 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.

19 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.

20 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Proc. of the 55th FOCS*, pages 561–570, 2014.

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

22 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proc. of the 25th SODA*, pages 217–226, 2014.

23 Jonathan A. Kelner and Alex Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013.

24 Richard Kenyon. The laplacian on planar graphs and graphs on surfaces. *Current Developments in Mathematics*, 2011.

25 Philip N Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proc. of the 45th STOC*, pages 505–514, 2013.

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

27 Ioannis Koutis and Gary L. Miller. A linear work, $o(n^{1/6})$ time, parallel algorithm for solving planar laplacians. In *Proc. of the 18th SODA*, pages 1002–1011, 2007.

28 Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. *SIAM J. Comput.*, 43(1):337–354, 2014.

29 Robert Krauthgamer, Huy L Nguyen, and Tamar Zondiner. Preserving terminal distances using minors. *SIAM Journal on Discrete Mathematics*, 28(1):127–141, 2014.

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

31 Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proc. of the 45th STOC*, pages 755–764, 2013.

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

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

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

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

36 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *Proc. of the 50th FOCS*, pages 3–12, 2009.

37 Richard Peng. Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time. In *Proc. of the 27th SODA*, pages 1862–1867, 2016.

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

39 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.

40 Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Analysis Applications*, 35(3):835–885, 2014.

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

42 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proc. of the 44th STOC*, pages 887–898, 2012.