Dynamic Sketching for Graph Optimization Problems with Applications to Cut-Preserving Sketches[*]

Sepehr Assadi[†], Sanjeev Khanna[†], Yang Li[†], and Val Tannen[‡]

University of Pennsylvania, Philadelphia, US
{sassadi,sanjeev,yangli2,val}@cis.upenn.edu

------ **Abstract** --

In this paper, we introduce a new model for sublinear algorithms called *dynamic sketching*. In this model, the underlying data is partitioned into a large *static* part and a small *dynamic* part and the goal is to compute a summary of the static part (i.e, a *sketch*) such that given any *update* for the dynamic part, one can combine it with the sketch to compute a given function. We say that a sketch is *compact* if its size is bounded by a polynomial function of the length of the dynamic data, (essentially) independent of the size of the static part.

A graph optimization problem P in this model is defined as follows. The input is a graph $G(V, E)$ and a set $T \subseteq V$ of k terminals; the edges between the terminals are the dynamic part and the other edges in G are the static part. The goal is to summarize the graph G into a compact sketch (of size $\text{poly}(k)$) such that given any set Q of edges between the terminals, one can answer the problem P for the graph obtained by inserting all edges in Q to G, using only the sketch.

We study the fundamental problem of computing a maximum matching and prove tight bounds on the sketch size. In particular, we show that there exists a (compact) dynamic sketch of size $O(k^2)$ for the matching problem and any such sketch has to be of size $\Omega(k^2)$. Our sketch for matchings can be further used to derive compact dynamic sketches for other fundamental graph problems involving cuts and connectivities. Interestingly, our sketch for matchings can also be used to give an elementary construction of a *cut-preserving vertex sparsifier* with space $O(kC^2)$ for k-terminal graphs, which matches the best known upper bound; here C is the total capacity of the edges incident on the terminals. Additionally, we give an improved lower bound (in terms of C) of $\Omega(C/\log C)$ on size of cut-preserving vertex sparsifiers, and establish that progress on dynamic sketching of the *s-t* max-flow problem (either upper bound or lower bound) immediately leads to better bounds for size of cut-preserving vertex sparsifiers.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases Small-space Algorithms, Maximum Matchings, Vertex Sparsifiers

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2015.52

1 Introduction

Massive data sets are arising more and more frequently in many application domains. Traditional gold standards of computational efficiency, namely, linear-time and linear-space, no longer seem sufficient for managing and analyzing such massive data sets. As a result, a

35th IARCS Annual Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015).
Editors: Prahladh Harsha and G. Ramalingam; pp. 52–68
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

beautiful new area of sublinear algorithms has developed over the past two decades – these are algorithms whose resource requirements are substantially smaller than the size of the input on which they operate. A rich theory of sublinear algorithms has emerged, and has brought remarkable new insights into combinatorial structure of well-studied optimization problems (see, for instance, the surveys [23, 25, 27], and references therein).

In recent years, graph optimization problems have received a lot of attention in the study of sublinear algorithms in various models, and the streaming model of computation is one of the most popular examples. In the streaming model, an algorithm is presented with a stream of edge insertions and deletions and is required to give an answer to a pre-specified graph problem at the end of the stream. Unfortunately, for many fundamental graph problems, no small space streaming algorithm is possible. For instance, [10] showed that determining whether or not there is a path from a specific vertex s to a specified vertex t in a directed graph requires $\Omega(n^2)$ space even for streams with only edge insertions; here n denotes the number of vertices in the input graph. This immediately implies that computing the length of the s-t shortest path, the value of the minimum cut between s and t, or the edge/vertex connectivity between s and t also requires $\Omega(n^2)$ space since the output of these problems is non-zero only when there is a path from s to t. The same lower bound is also obtained for computing the size of the maximum matching [10]. In fact, most of recent works for graph problem focus on *approximation algorithms* developed under the *semi-streaming* model introduced in [10], where an algorithm is allowed to output an approximate answer while using space linear in n. But is there hope left for *exact* sublinear algorithms? More specifically, is there a non-trivial model where sublinear algorithms are achievable for outputting exact answers for fundamental graph problems like matchings, connectivities, cuts, etc.?

In this paper, we explore this direction by considering the case where the input graph only undergoes *local changes*, and study how local changes influence the solutions of several fundamental graph problems. The goal is to exploit the locality of these updates and compress the rest of the graph into a small-size *sketch* that is able to answer *queries* regarding a specific problem (e.g. the s-t edge connectivity problem) for every possible local changes made to the graph. We introduce a model in this spirit and in the rest of this section, we formally define the model, discuss the connection to existing models, and summarize our results.

1.1 The Dynamic Sketching Model

We define the *dynamic sketching model*, where algorithms are required to construct data structures (called sketches) that are composable with *local updates* to the underlying data. Specifically, for graph problems in the dynamic sketching model, we consider the following setup (see Section 1.1 in the full version [5] for a more general definition which is not restricted to graph problems). Given a graph optimization problem P, an input graph $G(V, E)$ on n vertices with k vertices identified as *terminals* $T = \{q_1, \ldots, q_k\}$, the goal of k-dynamic sketching for P is to construct a sketch Γ such that given any possible subset of the edges between the terminals (a *query*), we can solve the problem P using only the information contained in the sketch Γ. Formally,

▶ **Definition 1.** Given a graph-theoretic problem P, a k-dynamic sketching scheme for P is a pair of algorithms with the following properties.
 (i) A **compression algorithm** that given any input graph $G(V, E)$ with a set T of k terminals, outputs a data structure Γ (i.e, a dynamic sketch).
 (ii) An **extraction algorithm** that given any subset of the edges between the terminals, i.e, a **query** Q, and the sketch Γ, outputs the answer to the problem P for the graph, denoted by G^Q, obtained by inserting all edges in Q to G (without further access to G).

We allow both compression and extraction algorithms to be randomized and err with some small probability. Furthermore, we say a sketching scheme is *compact* if it constructs dynamic sketches of size poly(k), where the size of a sketch is measured by the number of machine words of length $O(\log n)$.

We should note right away that of course *not* every graph problem admits a compact dynamic sketch. For example, one can show that any dynamic sketch for the *maximum clique* problem or the *minimum vertex cover* problem requires $\min\{\Omega(n), 2^{\Omega(k)}\}$ space (see the full version of the paper [5], Section 6).

1.2   Connection to Existing Models

Streaming.   Any single-pass streaming algorithm with space requirement s can be used as a dynamic sketching scheme with a sketch of size s: run the streaming algorithm on graph $G(V, E)$ for the static data and store the state of the algorithm as the sketch; continue running the algorithm using the stored state when the dynamic data is presented. However, note that a streaming algorithm directly gives a *compact* scheme only when the space requirement is logarithmic in n, which, as we just discussed, is not the case for nearly all fundamental graph optimization problems. In the following, we use the *s-t* shortest path problem as an example to elaborate the distinction between the two models. Our results in Section 2, illustrates a similar distinction for the case of the maximum matching problem.

As we already mentioned, outputting the length of the *s-t* shortest path requires $\Omega(n^2)$ space in the streaming model. We now give a simple dynamic sketching scheme for the *s-t* shortest problem with a sketch of size $O(k^2)$. The input to the *s-t* shortest path problem in the k-dynamic sketching model is a weighted graph G, a set T of terminals, and two designated vertices s and t. Without loss of generality, we can assume s and t are terminals; otherwise we can add them to the set of terminals and record their edges to the other terminals in $O(k)$ space. The compression algorithm creates a graph H with $V(H) = T$, where for any pair of terminals q_i and q_j, a directed edge from q_i to q_j is added to H with weight equal to the weight of a shortest path from q_i to q_j in G. The size of H is $O(k^2)$. To obtain the answer for each query Q, the extraction algorithm adds the edges in Q to H, building a small graph H^Q, and compute the shortest path from s to t in H^Q. It is easy to see that the weights of the shortest paths between s and t in H^Q and G^Q are equal, thus H can be used as a dynamic sketch for the *s-t* shortest path problem.

Linear sketches.   A very strong notion of sketching for handling arbitrary changes to the original data is linear sketching, which corresponds to applying a randomized low-dimensional linear transformation to the input data. This allows for compressing the data into a smaller space while (approximately) preserving some desired property of the input. Moreover, composability of these sketches (as they are linear transformations) allow them to handle arbitrary changes to the input data. Linear sketching technique has been successfully applied to various graph problems, mainly involving cuts and connectivity [2, 3, 14] (see also [23] for a survey of such results in dynamic graph streams). However, these results use space that is prohibitively large for dynamic sketching (a linear dependence on n), and typically only yield approximation answers.

Kernelization.   Dynamic sketching shares some similarity to *Kernelization* developed in parametrized complexity [17, 16, 11] in the following two aspects. Firstly, the number of terminals k in dynamic sketching may be viewed as a parameter. However, the main difference here is that for dynamic sketching, k is a parameter of the *model*, while for kernelization, the

parameter is usually the size of the solution, which is the property of the input rather than the model. Secondly, since a kernel for an instance of a problem is defined to be an equivalent instance of the same problem with size bounded by a function of a fixed parameter of the problem, both dynamic sketching and kernelization are in the spirit of compression. However, the techniques developed in kernelization do not directly carry over to dynamic sketching for the following two reasons. Firstly, kernelization typically focuses only on static data and secondly, the space target in kernelization (which is different compare to dynamic sketching) is normally polynomial in the parameter (usually the size of the solution to the problem) which could be $\Omega(n)$ in the dynamic sketching model. Finally, it is worth mentioning that there are problems (e.g. *minimum vertex cover*) that admit polynomial size kernels, while it can be shown that the dynamic sketching for these problem require sketches of size $2^{\Omega(k)}$ (see the full version of the paper [5], Section 6).

Provisioning.   We should note that dynamic sketching shares some ancestry with *provisioning*, a technique developed by [8] for avoiding repeated expensive computations in what-if analysis, where the input data is formed by k known overlapping subsets of some universe, and the goal is to compress these subsets so as to answer a specific database query when only some of those subsets are presented at run-time. Note that a main distinction between the two model is that in provisioning the dynamic input is neither small nor local.

1.3   Our Results

Maximum matching.   The main focus of this paper is on the maximum matching problem [20] in the dynamic sketching model, and its applications to various others problems. We give a dynamic sketching scheme with a sketch of size $O(k^2)$, using a technique based on an algebraic formulation of the matchings introduced by Tutte [29]. At a high level, we store a sketch that computes the rank of the *Tutte matrix* (see Definition 4) of the underlying graph. Since the queries only affect $O(k^2)$ entries of the Tutte matrix, we can compress this matrix using algebraic operations into a few small matrices of dimensions $k \times k$. Storing the small matrices as the sketch and modifying the related entries when a query is presented allows us to compute the rank of the original Tutte matrix, and hence the maximum matching size. Furthermore, we prove that our sketching scheme is optimal in terms of its space requirement (up to a logarithmic factor). In particular, we show that any dynamic sketching scheme for the matching problem has to store a sketch of size $\Omega(k^2)$ bits. We emphasize that the lower bound is information-theoretic; it holds even if the compression and extraction algorithms are computationally unbounded.

Cut-preserving sketches.   Interestingly, we discovered that our scheme for matchings can be used to design a *cut-preserving sketch*, which is the information-theoretic version of a *cut-preserving vertex sparsifier* [12, 24, 19]. Given a capacitated graph G (assume all capacities are integers) with a set T of k terminals, a cut-preserving vertex sparsifier (or a sparsifier for short) of G is a graph H with $T \subseteq V(H)$ ($V(H)$ denotes the set of vertices of H) such that for any bipartition S and $T \setminus S$ of terminals, the value of the minimum cut between S and $T \setminus S$ in G is preserved in H. A vertex sparsifier where the stored data is not restricted to be a graph is called a cut-preserving sketch.

  In recent years, cut-preserving vertex sparsifiers have been extensively studied (see, for example, [6, 9, 7, 4]). For instance, exact sparsifiers with 2^{2^k} vertices are shown by [12, 15], and sparsifiers with $O(C^3)$ vertices are shown by [17], where C is the total capacity of the edges incident on the terminals. Additionally, the size of any exact sparsifier is shown to be

$2^{\Omega(k)}$ [18, 15]. Cut-preserving sketches are also studied in the literature [4, 18, 16], where the best construction is known to be of size $O(kC^2)$ by [16]. Moreover, the $2^{\Omega(k)}$ lower bound of [18] is also shown to hold for the cut-preserving sketches.

We show that our dynamic sketching scheme for matchings can be used to obtain an elementary construction of a cut-preserving sketch of size $O(kC^2)$ that matches the best known upper bound of [16]. [16] showed that given a graph G and a set of k terminals T, a single *gammoid* can be used to produce a matroid that encodes all *terminal vertex cuts*. The authors then use the result of [22] to show how to obtain a matrix representation of this gammoid with $O(k^2)$ entries of $O(k)$ bits each (see Corollary 3.2 of [16]). Using standard techniques, one can use this sketch for vertex cuts to obtain an sketch for edge cuts (i.e, a cut-preserving sketch) that requires $O(kC^2)$ space. Our construction, on the other hand, uses the connection between matchings and the Tutte matrix followed by a simple reduction from cut-preserving sketches to the maximum matching problem. We believe that the simplicity of this construction and its connection to dynamic sketches for the matching problem is of independent interest and gives further insights into the structure of cut-preserving sketches. Moreover, we prove an improved lower bound (in terms of C) of $\Omega(C/\log C)$ bits on the size of any cut-preserving sketch; prior to our work, the best lower bound in terms of C is $\Omega(C^\varepsilon)$ for some small constant $\varepsilon > 0$ obtained by [18].

s-t edge-connectivity and s-t maximum flow.   As it turns out, any cut preserving sketch can be (almost directly) used to obtain a dynamic sketching scheme for the s-t edge-connectivity problem. However, using our lower bound for cut-preserving sketches, the resulting sketch size for edge-connectivity would be $\Omega(C/\log C)$, where C could be as large as n (hence the sketch is not compact). To obtain a compact sketch for edge-connectivity, we further design a dynamic sketching scheme which directly uses our dynamic sketching scheme for matchings, and obtain compact sketches of size $O(k^4)$. We further establish that cut-preserving sketches are, in fact, more related to the s-t maximum flow problem, in the sense that progress on either upper bound or lower bound on size of dynamic sketches for the s-t maximum flow problem immediately leads to better bounds for size of cut-preserving sketches.

Minimum spanning tree.   Finally, we present an $O(k)$-size dynamic sketch for the minimum spanning tree (MST) problem. Our idea for creating a compact dynamic sketch for MST is as follows. First of all, it is easy to see that if we add an edge to a graph, an MST of the resulting graph can be created by adding the edge to an MST of the original graph. Hence, it is sufficient to store an MST H of the original graph as a sketch. But this sketch is of size $\Omega(n)$. We show that H can be compressed into a tree H' such that all leaf nodes are terminals and there are at most $O(k)$ internal nodes in this tree; moreover, for any query Q, the weights of the MSTs in G^Q and H'^Q are equal. Hence, H' can be stored as a dynamic sketch. Due to the space constraints, the proof of this result is deferred entirely to the full version of the paper [5] (see Section 5).

Organization.   The rest of the paper is organized as follows. We first introduce our dynamic sketching scheme for the maximum matching problem in Section 2 and prove its optimality in terms of the sketch size. Then, in Section 3.1, we show how to use our sketching scheme for the matching problem to construct a cut-preserving sketch. Next, we provide our improved lower bound on the size of cut-preserving sketches in Section 3.2. We further establish the connection between cut-preserving sketches and s-t edge-connectivity (and s-t maximum

flow) and introduce a compact dynamic sketch for edge connectivity in Section 4. Finally, we conclude the paper with some future directions in Section 5.

Notation. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$. The bold-face upper-case letters represent matrices. A matrix with a 'tilde' on top (e.g. $\widetilde{\mathbf{M}}$) denotes a symbolic matrix, i.e, a matrix containing formal variables. For any prime p, $\mathbb{Z}_p$ denotes the field of integers modulo p.

For any undirected graph G, we use $\nu(G)$ to denote the size of a maximum matching in G. For any directed graph $G(V, E)$, an edge $e = (u, v)$ is directed from u to v, where we say u is the *tail* and v is the *head* of e. For any vertex $v \in V$, $d^+(v)$ (resp. $d^-(v)$) denotes the number of outgoing (resp. incoming) edges of v. For a capacitated graph, $c^+(v)$ (resp. $c^-(v)$) denotes the total capacity of the outgoing (resp. incoming) edges of v. We assume all the capacities are integers and can be stored in a single machine word of size $O(\log n)$.

2 The Maximum Matching Problem

In this section, we provide our results for the maximum matching problem. In particular, we show that,

▶ **Theorem 2.** *For any $0 < \delta < 1$, there exists a randomized k-dynamic sketching scheme for the maximum matching problem with a sketch of size $O(k^2 \log{(1/\delta)})$, which answers any query correctly with probability at least $1 - \delta$.*

Furthermore, we prove that the sketch size obtained in Theorem 2 is tight (up to an $O(\log n)$ factor). Formally,

▶ **Theorem 3.** *For any $k \geq 2$, any k-dynamic sketching scheme for the maximum matching problem that answers any query correctly with probability at least $2/3$, requires a dynamic sketch of size $\Omega(k^2)$ bits.*

Our sketching scheme for the proof of Theorem 2 relies on an algebraic formulation for the matching problem due to Tutte [29]. In the remainder of this section, we present this algebraic formulation, state our sketching scheme for matchings and proves its correctness and then present our lower bound result.

Algebraic formulation for the matching problem. The following matrix was first introduced by Tutte [29].

▶ **Definition 4** (Tutte matrix [29]). Suppose $G(V, E)$ is an undirected graph. The *Tutte matrix* of G is the following *symbolic* matrix $\widetilde{\mathbf{M}}$ of dimension $n \times n$.

$$\widetilde{\mathbf{M}}_{i,j} = \begin{cases} x_{i,j} & \text{if } (i, j) \in E \text{ and } i < j \\ -x_{j,i} & \text{if } (i, j) \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

where the $x_{i,j}$ are distinct formal variables.

Lovász [21] established the following result for computing the size of a maximum matching using Tutte matrix (see also [26] for more details on performing the computations over a finite field).

▶ **Lemma 5** ([21, 26]). *Let G be an undirected graph with n vertices and the maximum matching size of $\nu(G)$. For any prime $p > n$, let $\mathbb{Z}_p$ be the field of integers modulo p. Suppose $\widetilde{\mathbf{M}}$ is the Tutte matrix of G and $\mathbf{M}$ is the matrix obtained by evaluating each variable in $\widetilde{\mathbf{M}}$ by a number chosen independently and uniformly at random from $\mathbb{Z}_p$; then:*

$$\Pr\left(\mathrm{rank}(\mathbf{M}) = 2\nu(G)\right) \geq 1 - \frac{n}{p}$$

Note that the computation of $\mathrm{rank}(\mathbf{M})$ is also done over the field $\mathbb{Z}_p$.

2.1 An $O(k^2)$ size upper bound

In this section, we provide our k-dynamic sketching scheme for the maximum matching problem and prove Theorem 2.

Notation. Suppose the input is an undirected graph $G(V, E)$ with a set $T = \{q_1, \ldots, q_k\}$ of k terminals. Let p be any prime of magnitude $\Theta(n/\delta)$; we perform the algebraic computations in the field $\mathbb{Z}_p$. Let $\widetilde{\mathbf{M}}$ be the Tutte matrix of the graph obtained by adding all edges between the terminals to G, where the first k rows and k columns correspond to the vertices in T. We decompose $\widetilde{\mathbf{M}}$ into four sub matrices $\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}$, and $\widetilde{\mathbf{D}}$ as follows:

$$\widetilde{\mathbf{M}} = \begin{bmatrix} \widetilde{\mathbf{A}}_{k \times k} & \widetilde{\mathbf{B}}_{k \times (n-k)} \\ \widetilde{\mathbf{C}}_{(n-k) \times k} & \widetilde{\mathbf{D}}_{(n-k) \times (n-k)} \end{bmatrix}$$

Compression algorithm: The compression algorithm consists of 4 steps. Each of them performs a simple algebraic manipulation on the Tutte matrix $\widetilde{\mathbf{M}}$.

Step 1. For each non-zero entry of $\widetilde{\mathbf{M}}$ that corresponds to an edge in G (i.e., not between the terminals), assign an integer chosen uniformly at random from $\mathbb{Z}_p$. Denote the resulting matrix by,

$$\widetilde{\mathbf{M}}_1 = \begin{bmatrix} \widetilde{\mathbf{A}}_{k \times k} & \mathbf{B}_{k \times (n-k)} \\ \mathbf{C}_{(n-k) \times k} & \mathbf{D}_{(n-k) \times (n-k)} \end{bmatrix}$$

Note that except for $\widetilde{\mathbf{A}}$, all sub-matrices in $\widetilde{\mathbf{M}}_1$ are *no longer* symbolic.

Step 2. Let $r = \mathrm{rank}(\mathbf{D})$. Use *elementary row and column operations* to change $\mathbf{D}$ into a *diagonal* matrix $\mathrm{diag}(1, \ldots, 1, 0, \ldots, 0)$ with only r non-zero entries. Note that after this process, matrices $\mathbf{B}$ and $\mathbf{C}$ would also change, but the symbolic matrix $\widetilde{\mathbf{A}}$ remains unchanged. We denote the matrix $\widetilde{\mathbf{M}}_1$ after this process by,

$$\widetilde{\mathbf{M}}_2 = \begin{bmatrix} \widetilde{\mathbf{A}}_{k \times k} & \mathbf{X}_{k \times r} & \mathbf{B}'_{k \times (n-k-r)} \\ \mathbf{Y}_{r \times k} & \mathbf{I}_{r \times r} & \mathbf{0}_{r \times (n-k-r)} \\ \mathbf{C}'_{(n-k-r) \times k} & \mathbf{0}_{(n-k-r) \times r} & \mathbf{0}_{(n-k-r) \times (n-k-r)} \end{bmatrix}$$

Step 3. Use the sub-matrix $\mathbf{I}_{r \times r}$ in $\widetilde{\mathbf{M}}_2$ to zero out the matrix $\mathbf{X}$ by elementary *row* operations. Similarly, zero out $\mathbf{Y}$ by elementary *column* operations. Note that after this process, the matrix $\widetilde{\mathbf{A}}$ would be added by a linear combination of the rows in $\mathbf{Y}$, denoted by $\mathbf{A}'$. Denote the resulting matrix by,

$$\widetilde{\mathbf{M}}_3 = \begin{bmatrix} \widetilde{\mathbf{A}}_{k \times k} + \mathbf{A}'_{k \times k} & \mathbf{0}_{k \times r} & \mathbf{B}'_{k \times (n-k-r)} \\ \mathbf{0}_{r \times k} & \mathbf{I}_{r \times r} & \mathbf{0} \\ \mathbf{C}'_{(n-k-r) \times k} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Step 4. Consider the matrix $\mathbf{B}'$ in $\widetilde{\mathbf{M}}_3$; pick a maximal set of *linearly independent* columns from $\mathbf{B}'$ (if less than k columns are picked, arbitrarily pick from the remaining columns until having picked k columns), denoted by $\mathbf{B}''_{k \times k}$. Do the same for the matrix $\mathbf{C}'$ (but using linearly independent rows) and create $\mathbf{C}''_{k \times k}$. Finally, pick k^2 numbers from $\mathbb{Z}_p$, independently and uniformly at random and form a matrix of dimension $k \times k$, denoted by $\hat{\mathbf{A}}$. Store the value r (i.e., the rank of $\mathbf{D}$), the matrix $\hat{\mathbf{A}}$, and three $k \times k$ matrices $\mathbf{A}'$, $\mathbf{B}''$ and $\mathbf{C}''$ as the sketch.

Extraction algorithm: Given a query Q, create the matrix $\hat{\mathbf{A}}_Q$ from $\hat{\mathbf{A}}$ by zeroing out every entry that corresponds to an edge *not* in Q. Evaluate $\widetilde{\mathbf{A}}$ by $\hat{\mathbf{A}}_Q$ and obtain a (non-symbolic) matrix $\mathbf{A}$. Construct a matrix $\hat{\mathbf{M}}$ as follows,

$$\hat{\mathbf{M}} = \begin{bmatrix} \mathbf{A}_{k \times k} + \mathbf{A}'_{k \times k} & \mathbf{B}''_{k \times k} \\ \mathbf{C}''_{k \times k} & \mathbf{0}_{k \times k} \end{bmatrix}$$

Return $\left(\text{rank}(\hat{\mathbf{M}}) + r\right)/2$ as the maximum matching size.

We now prove the correctness of this scheme and show that it satisfies the bound given in Theorem 2 and hence prove this theorem.

Proof of Theorem 2. Since the prime p is of magnitude $\Theta(n/\delta)$, any number in $\mathbb{Z}_p$ requires $O(\log(n/\delta)) = O(\log n + \log(1/\delta))$ bits to store, which is at most $O(\log(1/\delta))$ machine words. The compression algorithm stores a number r, which needs $O(\log n)$ bits, four matrices of dimension $k \times k$, where each entry is a number in $\mathbb{Z}_p$ and requires $O(\log(1/\delta))$ space. Therefore, the total sketch size is $O(k^2 \log(1/\delta))$. We now prove the correctness.

We need to show that for each query Q, the extraction algorithm correctly outputs the matching size with probability at least $1 - \delta$. By Lemma 5,

$$\Pr\left(\text{rank}(\mathbf{M}) = 2\nu(G^Q)\right) \geq 1 - \frac{n}{p} \geq 1 - \delta$$

Here $\mathbf{M}$ is the (randomly evaluated) Tutte matrix of the graph obtained by applying the query Q to G, i.e., G^Q. Since the extraction algorithm outputs $\left(\text{rank}(\hat{\mathbf{M}}) + r\right)/2$ as the matching size, it suffices for us to show that $\text{rank}(\hat{\mathbf{M}}) + r = \text{rank}(\mathbf{M})$.

More specifically, the extraction algorithm evaluates $\widetilde{\mathbf{A}}$ by assigning a (pre-selected) random number to each entry that corresponds to an edge in Q, i.e, the matrix $\hat{\mathbf{A}}_Q$. For the sake of analysis, assume this is done before the compression algorithm is executed. Then, at the first step of the compression algorithm, all entries of the matrices $\mathbf{B}, \mathbf{C}, \mathbf{D}$ are randomly and independently evaluated. Combined with evaluating $\widetilde{\mathbf{A}}$ by $\hat{\mathbf{A}}_Q$, the resulting matrix (denoted by $\mathbf{M}_1$) is obtained from randomly and independently evaluating every non-zero entry of the Tutte matrix of the graph G^Q. In other words, it suffices to show that $\text{rank}(\mathbf{M}_1) = \text{rank}(\hat{\mathbf{M}}) + r$.

Since step 2 and step 3 only perform elementary row/column operations on the matrix, the rank does not change. For the matrix $\widetilde{\mathbf{M}}_3$ obtained after step 3, denote by $\mathbf{M}_3$ the matrix after evaluating the $\widetilde{\mathbf{A}}$ part in $\widetilde{\mathbf{M}}_3$. $\mathbf{M}_3$ is non-symbolic and it suffices to prove that $\text{rank}(\mathbf{M}_3) = \text{rank}(\hat{\mathbf{M}}) + r$. Note that after reordering rows and columns of $\mathbf{M}_3$, $\mathbf{M}_3$ can be rewritten as

$$\begin{bmatrix} \mathbf{A}_{k \times k} + \mathbf{A}'_{k \times k} & \mathbf{B}'_{k \times (n-k-r)} & \mathbf{0} \\ \mathbf{C}'_{(n-k-r) \times k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{r \times r} \end{bmatrix}$$

Therefore, the rank of $\mathbf{M}_3$ is equal to r plus the rank of the following sub-matrix of $\mathbf{M}_3$.

$$\mathbf{M}_4 = \begin{bmatrix} \mathbf{A}_{k \times k} + \mathbf{A}'_{k \times k} & \mathbf{B}'_{k \times (n-k-r)} \\ \mathbf{C}'_{(n-k-r) \times k} & \mathbf{0} \end{bmatrix}$$

We now show that $\mathbf{M}_4$ has the same rank as $\hat{\mathbf{M}}$. Since the matrix $\mathbf{C}''$ (in step 4 of the compression algorithm) contains a maximal set of linearly independent rows of $\mathbf{C}'$, each remaining row of $\mathbf{C}'$ is a linear combination of the rows in $\mathbf{C}''$. Therefore, all remaining rows in $\mathbf{C}'$ can be zero-out using elementary row operations. Hence, the rank of $\mathbf{M}_4$ is equal to the rank of the following matrix

$$\mathbf{M}_5 = \begin{bmatrix} \mathbf{A}_{k \times k} + \mathbf{A}'_{k \times k} & \mathbf{B}'_{k \times (n-k-r)} \\ \mathbf{C}''_{k \times k} & \mathbf{0} \\ \mathbf{0}_{(n-2k-r) \times k} & \mathbf{0} \end{bmatrix}$$

Similarly, using elementary column operations, the sub-matrix $\mathbf{B}'$ in $\mathbf{M}_5$ can be made into $[\mathbf{B}'' \ \mathbf{0}_{k \times (n-2k-r)}]$ without changing the rank, and the resulting matrix has the same rank as $\hat{\mathbf{M}}$. ◀

2.2 An $\Omega(k^2)$ size lower bound

In this section, we prove an $\Omega(k^2)$ bits lower bound on the sketch size of any k-dynamic sketching scheme for the matching problem, which implies that our space upper bound in Theorem 2 is tight (up to a logarithmic factor). We establish this lower bound by reducing from the MEMBERSHIP problem studied in communication complexity defined as follows.

The MEMBERSHIP **Problem**

Input: Alice is given a set $S \subseteq [N]$ and Bob is given an element $e^* \in [N]$.
Goal: Alice has to send a message to Bob such that Bob can determine whether $e^* \in S$ or not.

It is well-known that in order for Bob to succeed with probability at least 2/3, Alice has to send a message of size $\Omega(N)$ bits [1], where the probability is taken over the random coin tosses of Alice and Bob.

Reduction. For simplicity, assume N is a perfect square. Given any $S \subseteq [N]$, Alice constructs a graph $G(V, E)$ with a set T of k terminals as follows:

- The vertex set $V = \{u, w\} \cup V_1 \cup V_2 \cup V_3 \cup V_4$, where $|V_i| = \sqrt{N}$ for any $i \leq 4$ and $T = \{u, w\} \cup V_1 \cup V_4$. We will use $v_j^{(i)}$ to denote the j-th vertex in V_i.
- For any $i \in [\sqrt{N}]$, $v_i^{(1)}$ (resp. $v_i^{(3)}$) is connected to $v_i^{(2)}$ (resp. $v_i^{(4)}$); i.e, there is perfect matching between V_1 and V_2 (resp. V_3 and V_4).
- Fix a bijection $\sigma : [N] \mapsto [\sqrt{N}] \times [\sqrt{N}]$; for any element $e \in S$ with $\sigma(e) = (i, j)$, $v_i^{(2)}$ is connected to $v_j^{(3)}$.

Note that in this construction, $n = 4\sqrt{N} + 2$ and $k = 2\sqrt{N} + 2$, and initially there is no edge between the terminals.

Alice constructs this graph, run the compression algorithm of the dynamic sketching scheme on it and sends the sketch to Bob. Let Q be the query in which, for $\sigma(e^*) = (i, j)$, u is connected to $v_i^{(1)}$ and $v_j^{(4)}$ is connected to w. Bob queries the sketch with Q, finds

the maximum matching size in G^Q, and returns $e^* \in [S]$ iff the maximum matching size is $2\sqrt{N} + 1$ in G^Q.

The proof of the following lemma can be found in the full version [5] (Lemma 2.2).

▶ **Lemma 6.** $\nu(G^Q) = 2\sqrt{N} + 1$ *if and only if* $e^* \in S$.

Theorem 3 now follows from Lemma 6, along with the lower bound of $\Omega(N) = \Omega(k^2)$ on the communication complexity of the MEMBERSHIP problem.

3   Cut-Preserving Sketches

We establish a connection between k-dynamic sketching schemes for the maximum matching problem and cut-preserving sketches. In particular, we use our dynamic sketching scheme for the matching problem in Section 2 to design an exact cut-preserving sketch (i.e, an information-theoretic vertex sparsifier) with size $O(kC^2)$, where C is the total capacity of the edges incident on the terminals. This matches the best known upper bound on the space requirement of cut-preserving sketches. We further provide an improved lower bound of $\Omega(C/\log C)$ on the size of any cut-preserving sketch. Throughout this section, we will use the term *bipartition cut* to refer to a cut between a bipartition of the terminals and the term *terminal cut* to refer to a cut which separates two arbitrary disjoint subsets of terminals (not necessarily a bipartition). With a slight abuse of notation, we refer to the value of the minimum cut for a bipartition/terminal cut as the value of the bipartition/terminal cut directly.

Before we present our results, we make a general remark about the property of all cut-preserving sparsifiers (and sketches) that is also used crucially in our lower bound proof. In [18], a *generalized* sparsifier is defined as a sparsifier that preserves the minimum cut between all disjoint subsets of terminals, i.e, terminal cuts and not only bipartition cuts. The authors then point out that their upper bound results, as well as the previous constructions of cut sparsifiers in [12], also satisfy this general definition. The following simple claim gives an explanation why all known cut sparsifiers satisfy this general definition.

▶ **Claim 1.** *Suppose H is a cut sparsifier of the graph $G(V, E)$ with terminals T that preserves the value of all bipartition cuts. Then, H also preserves the value of all terminal cuts.*

Proof. For any two disjoint subsets of terminals $A, B \subseteq T$, any cut separating A and B in G must form a bipartition $(S, \overline{S})$ of the terminals, and since H preserves the value of all minimum cuts like $(S, \overline{S})$, the (A, B) minimum cut value in H is also equal to the minimum cut value in G. In the case that H is a cut-preserving sketch, the (A, B) minimum cut can be answered by querying H with all bipartition cuts that separate (A, B), and outputting the smallest value. ◀

3.1   An $O(kC^2)$ Size Cut-Preserving Sketch

In this section, we construct a cut-preserving sketch for any digraph G and a set of terminals T. We achieve this by constructing an instance G' of the maximum matching problem in the dynamic sketching model and show that the value of any terminal cut (A, B) in G can be computed using a carefully designed query for the maximum matching size in G'. Our reduction is based on a classical result relating edge connectivity and bipartite matching due to [13] (see also [28], Section 16.7).

▶ **Theorem 7.** *For any directed graph G with a set of k terminals, there is an exact cut-preserving sketch that uses space $O(kC^2)$, where C is the total capacity of the edges incident on the terminals.*

Without loss of generality, we will replace each edge in G with capacity c_e with c_e parallel edges and still denote the new graph with G. Consider the following cut-preserving sketch.

A cut-preserving sketch

■ **Input:** A graph G with m edges and a set T of terminals.
■ **Compression:** Construct a bipartite graph $G'(L, R, E')$ with terminals T' as follows and create a dynamic sketch for the maximum matching problem for G' and T'.
    a. For each edge e in G, create two vertices e^- (in L) and e^+ (in R).
    b. For any terminal q in T and any outgoing (resp. incoming) edge e of q, create a vertex $q^{\rightarrow e}$ in R (resp. $q^{\leftarrow e}$ in L). $q^{\rightarrow e}$ (resp. $q^{\leftarrow e}$), along with e^- (resp. e^+), belongs to the set T' of terminals in G'.
    c. For each edge e in G, there is an edge between vertices e^- and e^+ in G'.
    d. For any two edges e_1 and e_2 in G where the tail of e_1 is the head of e_2, there is an edge between the vertices e_1^+ and e_2^- in G'.
■ **Extraction:** Given any two disjoint subsets $A, B \subseteq T$, let $Q_{A,B}$ be the query where for any terminal q in A (resp. in B) and any outgoing (resp. incoming) edge e of q, an edge between the vertices $q^{\rightarrow e}$ and e^- (resp. $q^{\leftarrow e}$ and e^+) is inserted in G'.
Return $\nu(G'^{Q_{A,B}}) - m$, where $\nu(G'^{Q_{A,B}})$ is the maximum matching size in $G'^{Q_{A,B}}$.

The total number of terminals in G' is $2C$, and the total number of different A-B pairs (i.e., the total number of different possible queries) is at most 3^k. To ensure that every query is answered correctly, by Theorem 2, the sketch size is $O(kC^2)$. We now prove the correctness.

Proof. For any $A, B \subseteq T$, denote the value of the minimum A-B cut (which is equal to the edge-connectivity from A to B), by $c(A, B)$. Recall that for a graph G, $\nu(G)$ denotes the maximum matching size in G. We prove that $\nu(G'^{Q_{A,B}}) - m = c(A, B)$. Let M be the matching in $G^{Q_{A,B}}$ where for each edge e in G, e^- is matched with e^+; hence $|M| = m$.

We first show that if $c(A, B) = l$, then M can be augmented by l vertex disjoint paths and hence $\nu(G'^{Q_{A,B}}) \geq m + l$. There are l edge-disjoint path $P_1, P_2, \ldots, P_l$ from A to B in G. For each path $P_i = (e_1, e_2, \ldots, e_j)$, where e_1 starts with a terminal $q_a \in A$ and e_j ends with a terminal $q_b \in B$, create a path $P_i' = (q_a^{\rightarrow e_1}, e_1^-, e_1^+, e_2^-, \ldots, e_j^+, q_b^{\leftarrow e_j})$ in $G'^{Q_{A,B}}$. It is straightforward to verify that the P_i' paths are valid *vertex-disjoint* paths in $G'^{Q_{A,B}}$ and moreover, form disjoint augmenting paths of the matching M.

We now show that if the maximum matching M^* in G' is of size $m + l$, then $c(A, B) \geq l$. The symmetric difference between M and M^* forms a graph with l augmenting paths of the matching M. Each augmenting path must start and end with a vertex of the form $q^{\rightarrow e}$ or $q^{\leftarrow e}$ since they are the only vertices that are unmatched in M. Since every $q^{\rightarrow e}$ is in R and every $q^{\leftarrow e}$ is in L, each augmenting path must start with a $q^{\rightarrow e}$ vertex and ends with a $q^{\leftarrow e}$ vertex. Using the reversed transformation as in the previous case, the l augmenting paths can be converted into l edge disjoint paths from A to B in G. ◀

3.2   An $\tilde{\Omega}(C)$ Size Lower Bound

In this section, we provide a lower bound on the size of any cut-preserving sketch.

▶ **Theorem 8.** *For any integer $C > 0$, any cut-preserving sketch for k-terminal undirected graphs, where the total capacity of edges incident on the terminals is equal to C, requires $\Omega(C/\log C)$ bits.*

To prove this lower bound, we show how to encode a binary vector of length $N := \Omega(C/\log C)$ in an undirected graph G, so that given only a cut-preserving sketch for G, one can recover any entry of this vector. Standard information-theoretical arguments (similar to the lower bound for the MEMBERSHIP problem in Section 2) then imply that size of the cut-preserving sketch has to be of size $\Omega(N) = \Omega(C/\log C)$. We emphasize that while in the proof we assume the cut-preserving sketch has to return the value of minimum cuts between any subsets (A, B) of the terminals (even when they are not a bipartition), by Claim 1, this is without loss of generality; hence, the lower bound holds also for cut-preserving sketches that only guarantee to preserve minimum cuts for bipartitions.

Construction.   Let $k' = k - 2$. For simplicity, assume k' is even, and let $N = \binom{k'}{\frac{k'}{2}}$. For any N-dimensional binary vector $\mathbf{v} \in \{0, 1\}^N$, we define a graph $G_{\mathbf{v}}(V, E)$ as follows:

Vertices: The set of vertices of $G_{\mathbf{v}}$ is $V = \{s, t\} \cup \{q_1, \dots, q_{k'}\} \cup \{u_1, \dots, u_{k'}\} \cup \{v_1, \dots, v_N\}$ and the k terminals are $T = \{s, q_1, \dots, q_{k'}, t\}$.

Edges: Let $\mathcal{S} = \{S_1, \dots, S_N\}$ be a collection of all $(k'/2)$-size subsets of $\{q_1, \dots, q_{k'}\}$. The set of edges are defined as:
- For any $i \in [k']$, there is an edge (q_i, u_i) with capacity N.
- For any $i \in [k']$, there is an edge (s, u_i) with capacity N.
- For any $j \in [N]$, there is an edge (v_j, t) with capacity 1.
- A vertex u_j is connected to a vertex v_i with an edge of capacity 1 iff $\mathbf{v}_i = 1$ or $q_j \notin S_i$. Additionally, if u_j is connected to v_i, there are two more edges $f_1 = (s, v_i)$ and $f_2 = (u_j, t)$ each with capacity 1.
- There is an edge (s, t) with capacity $kN - m$, where m is the number of edges between $\{u_1, \dots, u_{k'}\}$ and $\{v_1, \dots, v_N\}$.

To recover the vector $\mathbf{v}$ from a cut-preserving sketch of $G_{\mathbf{v}}$, we will consider the terminal cuts (A, B) where $A = \{s\} \cup S_i$ for some $S_i \in \mathcal{S}$ and $B = \{t\}$. We further denote the terminal cut (A, B) corresponding to picking $S_i \in \mathcal{S}$ in the part A by $\mathrm{TC}(S_i)$. We define the *output profile* of a graph $G_{\mathbf{v}} \in \mathcal{G}$ to be an N-dimensional vector $\mathbf{op}(G_{\mathbf{v}})$ where the i-th entry of $\mathbf{op}(G_{\mathbf{v}})$ is equal to the value of the terminal cut $\mathrm{TC}(S_i)$. We show that there is a one-to-one correspondence between the vector $\mathbf{v}$ and $\mathbf{op}(G_{\mathbf{v}})$.

▶ **Lemma 9.** *Let $\mathbf{1}$ be the N-dimensional vector of all ones. There exists a value c independent of $\mathbf{v}$ such that $\mathbf{op}(G_{\mathbf{v}}) = \mathbf{v} + c \cdot \mathbf{1}$.*

Proof.   Fix an $i \in [N]$ and consider $\mathrm{TC}(S_i)$. We argue that the maximum flow value from $\{s\} \cup S_i$ to $\{t\}$ is $(k+1)N - 1 + \mathbf{v}_i$; the lemma then follows from the max-flow min-cut duality and the choice of $c = (k+1)N - 1$.

In $G_{\mathbf{v}}$, we can first send a flow of size kN from s to t by sending one unit of flow along every $s \to v_l \to u_j \to t$ path for any edge of the form (u_j, v_l) (m units of flow in total) and $kN - m$ units of flow over the (s, t) edge. After this process, the residual graph of $G_{\mathbf{v}}$ becomes

a directed graph where any edge of the form (u_j, v_l) is directed from u_j to v_l. Now consider any vertex v_p where $p \neq i$. There exists at least one terminal $q_j \in S_i$ (in fact, in $S_i \setminus S_p$), such that there is an edge between u_j and v_p in $G_{\mathbf{v}}$. Since in the residual graph of $G_{\mathbf{v}}$, this edge is directed from u_j to v_p, we can send one unit of flow over this edge also through the path $q_j \to u_j \to v_p \to t$. Hence, in $G_{\mathbf{v}}$, we can always send $kN + N - 1 = (k+1)N - 1$ units of flow from $\{s\} \cup S_i$ to $\{t\}$.

First suppose the i-th entry of $\mathbf{v}$ is equal to 1; then there is an edges from u_j to v_i for any $q_j \in S_i$. In particular, we can send one extra unit of flow over one of these edges to t, hence having a flow of size $(k+1)N$ entering t. Since the total capacity of the edges incident on t is $(k+1)N$, this ensures that the max-flow is also $(k+1)N$.

Now suppose the i-th entry of $\mathbf{v}$ is equal to 0. For the vertex v_i, by construction, there is no edge from any u_j to v_i, where $q_j \in S_i$. Hence in the residual graph of $G_{\mathbf{v}}$, there is no path from $\{s\} \cup S_i$ to $\{t\}$, meaning that the maximum flow in this case is $(k+1)N - 1$. This completes the proof.   ◀

Proof of Theorem 8. Lemma 9 ensures that for any graph $G_{\mathbf{v}}$, there is a one-to-one correspondence between the value of i-th entry in $\mathbf{v}$ and i-th entry in $\mathbf{op}(G_{\mathbf{v}})$. Assuming that the cut-preserving sketch is able to answer each terminal cut (deterministically or even with a sufficiently small constant probability of error), we can recover i-th bit of $\mathbf{v}_i$, from the i-th index in $\mathbf{op}(G_{\mathbf{v}})$ with a constant probability. Standard information-theoretical arguments imply that the size of the cut-preserving has to be $\Omega(N)$. Moreover, since $N = 2^{\Omega(k)} = \Omega(C/k) = \Omega(C/\log C)$ in this construction, we obtain the final bound of $\Omega(C/\log C)$ bits on the sketch size.   ◀

We should point out for the case of randomized cut-preserving sketches that are only guaranteed to have a constant probability of failure over bipartition cuts (and not necessarily terminal cuts), we first need to reduce the probability of error to 2^{-k} before performing the described construction (and applying Claim 1) which results in a lower bound of $\Omega(C/\log^2 C)$.

We further point out that, as a corollary of Theorem 8, we also obtain a simple proof for a lower bound of $2^{\Omega(k)}$ on size of cut-preserving sketches (see [18, 15]).

4   The s-t Edge-Connectivity Problem

In this section, we study dynamic sketching for the s-t edge-connectivity problem. As it turns out, any cut-preserving sketch can be directly adapted to a dynamic sketching scheme for the s-t edge-connectivity problem as follows. Given a graph G with a set T of k terminals and two designated vertices s and t, create a cut-preserving sketch for G with terminals $T \cup \{s, t\}$. Note that given a query Q (i.e., a set of edges among T), the s-t minimum cut (which is equal to the s-t edge-connectivity) will partition $T \cup \{s, t\}$ into two sets T_s and T_t, where T_s contains s and T_t contains t. Hence, the minimum cut from T_s to T_t is equal to the minimum cut from s to t. The cut-preserving sketch can answer the minimum cut from T_s to T_t in the original graph, and the additional cut value caused by the query is simply the total number of the edges from T_s to T_t. Therefore, if we enumerate all possible partitions of the terminals that separate s and t, and compute the minimum cut for each partition as above, the smallest minimum cut among those partitions is equal to the minimum cut from s to t.

Nevertheless, by our lower bound on the size of cut-preserving sketches (Theorem 8), a dynamic sketching scheme constructed as above, will have a linear dependency on the total degree of the vertices in $T \cup \{s, t\}$, which could be as large as the number of vertices in the graph. To resolve this issue, we propose a scheme which directly uses our dynamic

sketching scheme for the maximum matching problem and achieve a sketch of size $O(k^4)$. The reduction is in the same spirit as the one we used for cut-preserving sketches. But note that the main differences is that unlike the case for cut-preserving sketches, in the dynamic sketching problem the set of edges in the original graph changes which require additional care.

▶ **Theorem 10.** *For any $\delta > 0$, there exists a randomized k-dynamic sketching scheme for the s-t edge-connectivity problem with a sketch of size $O(k^4 \log(1/\delta))$, which answers any query correctly with probability at least $1 - \delta$.*

Given a digraph G with two designated vertices s and t, along with a set of T terminals, recall that, for the query $Q_\forall$ where an edge is inserted between each (ordered) pair of terminals, $G^{Q_\forall}$ denotes the graph after applying the query $Q_\forall$ to G. Assume s does not have any incoming edge and t does not have any outgoing edge, since removing them will not affect the s-t edge-connectivity. We further assume that s are t are not terminals. This is without loss of generality since we can create two vertices s' and t' that are not terminals, while adding $d^+(s)$ (resp. $d^-(t)$) new vertices and for each of these vertices v, adding an edge from s' to v and v to s (resp. t to v and v to t'). In this new graph, the s'-t' edge connectivity is equal to the s-t edge-connectivity in G and s', t' are not terminals. Consider the following dynamic sketching scheme.

A dynamic sketching scheme for the s-t edge-connectivity problem

Input: A graph G with m edges, two designated vertices s and t, and a set T of k terminals.

Compression: Construct a bipartite graph $G'(L, R, E')$ with a set T' of terminals as follows and create a dynamic sketch for the maximum matching problem for G' and T'.
   a. For each edge e in $G^{Q_\forall}$, if e starts with s, create a vertex e^+ (in R), if e ends with t create a vertex e^- (in L), otherwise, create two vertices e^+ (in R) and e^- (in L).
   b. For each edge e between two terminals in G, create two vertices $\hat{e}^+$ and $\hat{e}^-$; $\hat{e}^-$ and $\hat{e}^+$, along with e^- and e^+, are terminals of G'.
   c. For each edge e in G where both e^- and e^+ exist, there is an edge between e^- and e^+.
   d. For any two edges e_1 and e_2 in $G^{Q_\forall}$ where the tail of e_1 is the head of e_2, there is an edge between e_1^+ and e_2^-.

Extraction: Given any query Q of G, let Q' be the query of G' where
   a. For each edge e in Q, insert an edge between e^- and e^+.
   b. For each edge e in $Q_\forall \setminus Q$, insert an edge between e^- and $\hat{e}^+$, and between e^+ and $\hat{e}^-$.
   c. Let the maximum matching size of $G'^{Q'}$ be ν. Output $\nu - (m + 2k^2 - |Q|)$.

The total number of terminals in G' is $4k^2$. Hence by Theorem 2, the sketch size is $O(k^4 \log(1/\delta))$. The correctness of the reduction is similar in spirit to the proof of Theorem 7 and is deferred to the full version [5] (see Section 4.1).

We conclude this section by remarking that there exists an equivalence between dynamic sketching the capacitated version of the s-t edge connectivity problem (i.e., the s-t maximum flow problem) and cut-preserving sketches. In particular,

▶ **Theorem 11.** *Any cut-preserving sketch can be adapted to a dynamic sketching scheme for the s-t maximum flow problem while increasing the number of terminals by at most* 2, *and vice versa.*

The proof of this theorem together with a detailed discussion on the similarity of the s-t maximum flow problem and cut-preserving sketches is provided in Section 4.2 of the full version of the paper [5]. However, we point out here that Theorem 11 combined with Theorem 8, proves a similar $2^{\Omega(k)}$ lower bound on size of dynamic sketches for the s-t maximum flow problem. In other words, this problem does not admit a compact dynamic sketch.

5   Conclusions

In this paper we have introduced *dynamic sketching*, a new approach for compressing data sets separated into static and dynamic parts. We studied dynamic sketching for graph problems where the dynamic part consists of k vertices and the edges between them may get modified in an arbitrary manner (a query). We showed that the maximum matching problem admits a sketch of size $O(k^2)$ and the space bound is tight. Moreover, this sketch can be used to obtain cut-preserving sketches of size $O(kC^2)$, and dynamic sketches for s-t edge-connectivity of size $O(k^4)$.

There are problems (even in P) for which any dynamic sketch requires $2^{\Omega(k)}$ space. An interesting direction for future work is to identify broad classes of problems that admit compact dynamic sketches, i.e, sketches of size poly(k).

Some data compression schemes (most notably, cut sparsifiers and kernelization results) generate as compressed representation an instance of the original problem, while the sketches we introduced do not fall into this category. A natural question is to understand if there exist polynomial-size "sparsifier-like" compressed representations for matchings and s-t edge connectivity in the dynamic sketching model.

Finally, while our work narrows the gap between upper and lower bounds on the size of a cut-preserving sketches, it remains an intriguing open question to get an asymptotically tight bound on the size of cut-preserving sketches.

References

1   Farid M. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theor. Comput. Sci.*, 157(2):139–159, 1996.

2   Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 459–467, 2012.

3   Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 5–14, 2012.

4   Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 279–293, 2014.

5   Sepehr Assadi, Sanjeev Khanna, Yang Li, and Val Tannen. Dynamic sketching for graph optimization problems with applications to cut-preserving sketches. *CoRR*, abs/1510.03252, 2015.

6   Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 265–274, 2010.

7   Julia Chuzhoy. On vertex sparsifiers with steiner nodes. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 673–688, 2012.

8   Daniel Deutch, Zachary G. Ives, Tova Milo, and Val Tannen. Caravan: Provisioning for what-if analysis. In *Sixth Biennial Conference on Innovative Data Systems Research, CIDR*, 2013.

9   Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX, and 14th International Workshop, RANDOM*, pages 152–165, 2010.

10 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *Automata, Languages and Programming: 31st International Colloquium, ICALP*, pages 531–543, 2004.

11 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC*, pages 133–142, 2008.

12 Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57(3):366–375, 1998.

13 Alan J Hoffman. Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. In *Proc. Sympos. Appl. Math*, volume 10, pages 113–127. World Scientific, 1960.

14 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 561–570, 2014.

15 Arindam Khan and Prasad Raghavendra. On mimicking networks representing minimum terminal cuts. *Inf. Process. Lett.*, 114(7):365–371, 2014.

16 Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 94–103, 2012.

17 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 450–459, 2012.

18 Robert Krauthgamer and Inbal Rika. Mimicking networks and succinct representations of terminal cuts. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1789–1799, 2013.

19 Frank Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, pages 47–56, 2010.

20 L. Lovász and D. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. American Mathematical Soc., 2009.

21 László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

22 Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009.

23 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

24 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 3–12, 2009.

25 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

26 Michael O. Rabin and Vijay V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, 1989.

27 Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM J. Discrete Math.*, 25(4):1562–1588, 2011.

28 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

29 William T Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.