Average Sensitivity of Graph Algorithms*

Nithin Varma† Yuichi Yoshida‡§

Abstract

In modern applications of graph algorithms, where the graphs of interest are large and dynamic, it is unrealistic to assume that an input representation contains the full information of a graph being studied. Hence, it is desirable to use algorithms that, even when provided with only a (large) subgraph, output solutions that are close to the solutions output when the whole graph is available. We formalize this feature by introducing the notion of average sensitivity of graph algorithms, which is the average earth mover's distance between the output distributions of an algorithm on a graph and its subgraph obtained by removing an edge, where the average is over the edges removed and the distance between two outputs is the Hamming distance.

In this work, we initiate a systematic study of average sensitivity. After deriving basic properties of average sensitivity such as composition, we provide efficient approximation algorithms with low average sensitivities for concrete graph problems, including the minimum spanning forest problem, the global minimum cut problem, the minimum s-t cut problem, and the maximum matching problem. In addition, we prove that the average sensitivity of our global minimum cut algorithm is almost optimal, by showing a nearly matching lower bound. We also show that every algorithm for the 2-coloring problem has average sensitivity linear in the number of vertices. One of the main ideas involved in designing our algorithms with low average sensitivity is the following fact; if the presence of a vertex or an edge in the solution output by an algorithm can be decided locally, then the algorithm has a low average sensitivity, allowing us to reuse the analyses of known sublinear-time algorithms and local computation algorithms. Using this fact in conjunction with our average sensitivity lower bound for 2-coloring, we show that every local computation algorithm for 2-coloring has query complexity linear in the number of vertices, thereby answering an open question.

*A full version of this paper is available at `https://arxiv.org/abs/1904.03248`. Much of this research was done while the first author was visiting the second author.

†University of Haifa, Israel.

‡National Institute of Informatics, Japan.

§JST PRESTO, Japan.

1 Introduction

In modern applications of graph algorithms, where the graphs of interest are large and dynamic, it is unrealistic to assume that an input representation contains the full information of a graph being studied. For example, consider a social network, where a vertex corresponds to a user of the social network service and an edge corresponds to a friendship relation. It is reasonable to assume that users do not always update new friendship relations on the social network service, and that sometimes they do not fully disclose their friendship relations because of security or privacy reasons. Hence, we can only obtain an approximation G' to the true social network G. This brings out the need for algorithms that can extract information on G by solving a problem on G'. Moreover, as the solutions output by a graph algorithm are often used in applications such as detecting communities [37, 38], ranking nodes [42], and spreading influence [22], the solutions output by an algorithm on G' should be close to those output on G.

We adopt a popular model that the n-vertex input graph G' at hand is a randomly chosen (large) subgraph of an unknown true graph G [1, 25, 39]. Intuitively, a deterministic algorithm $\mathcal{A}$ is said to be *stable-on-average* if the Hamming distance $d_{\mathrm{Ham}}\big(\mathcal{A}(G), \mathcal{A}(G')\big)$ is small, where $\mathcal{A}(G)$ and $\mathcal{A}(G')$ are outputs of $\mathcal{A}$ on G and G', respectively. Here, outputs are typically vertex sets or edges sets and we assume that they are represented appropriately using binary strings. More specifically, for an integer $k \geq 1$, we say that the *k-average sensitivity* of a deterministic algorithm $\mathcal{A}$ is

$$\mathop{\mathbb{E}}_{\{e_1,\dots,e_k\}\sim\binom{E}{k}}\Big[d_{\mathrm{Ham}}\big(\mathcal{A}(G), \mathcal{A}(G-\{e_1,\dots,e_k\})\big)\Big] \tag{1.1}$$

for every graph $G = (V, E)$, where $\{e_1, \dots, e_k\}$ is sampled uniformly at random from $\binom{E}{k}$, the set of all subsets of E of cardinality k, and where $G - F$ for a set of edges $F \subseteq E$ denotes the subgraph obtained from G by removing F. When $k = 1$, we call the k-average sensitivity simply *average sensitivity*. We say that algorithms with low average sensitivity are *stable-on-average*. Although we focus on graphs here, we note that our definition can also be extended to the study of combinatorial objects other than graphs such as strings and constraint satisfaction problems.

An algorithm that outputs the same solution regardless of the input has the least possible average sensitivity, even though it is definitely useless. Hence, the key question in a study of average sensitivity is to reveal trade-offs between solution quality and average sensitivity for various problems.

EXAMPLE 1. *Consider the algorithm that, given a graph $G = (V, E)$ on n vertices, outputs the set of vertices of degree at least $n/2$. As removing an edge changes the degree of exactly two vertices, the sensitivity of this algorithm is at most 2.*

EXAMPLE 2. *Consider the s-t shortest path problem, where given a graph $G = (V, E)$ and two vertices $s, t \in V$, we are to output the set of edges in a shortest path from s to t. Since the length of a shortest path is always bounded by n, where n is the number of vertices, every deterministic algorithm has average sensitivity $O(n)$. Indeed, there exists a graph for which this trivial upper bound is tight. Think of a cycle of even length n and two vertices s, t in diametrically opposite positions. Consider an arbitrary deterministic algorithm $\mathcal{A}$, and assume that it outputs a path P (of length $n/2$) among the two shortest paths from s to t. With probability half, an edge in P is removed, and $\mathcal{A}$ must output the other path Q (of length $n/2$) from s to t. Hence, the average sensitivity must be $1/2 \cdot (n/2) = \Omega(n)$. In this sense, there is no deterministic algorithm with nontrivial average sensitivity for the s-t shortest path problem.*

We also generalize our definition of average sensitivity to apply to randomized algorithms. Let $\mathcal{A}(G)$ denote the output distribution of $\mathcal{A}$ on G. Let $d_{\mathrm{EM}}(\mathcal{A}(G), \mathcal{A}(G'))$ denote the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}(G')$, where the distance between two outputs is measured by the Hamming distance. Specifically, $d_{\mathrm{EM}}(\mathcal{A}(G), \mathcal{A}(G'))$ is equal to $\min_{\mathcal{D}} \left[\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[d_{\mathrm{Ham}}(x, y)\right]\right]$, where $\mathcal{D}$ denotes a distribution over pairs (x, y) of outputs of $\mathcal{A}$ such that the left and right marginals of $\mathcal{D}$ are equal to $\mathcal{A}(G)$ and $\mathcal{A}(G')$, respectively. Then, for an integer $k \geq 1$, the *k-average sensitivity* of a randomized algorithm $\mathcal{A}$ is

$$\mathbb{E}_{\{e_1,\ldots,e_k\}\sim\binom{E}{k}} \left[d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}(G - \{e_1, \ldots, e_k\})\big)\right] \tag{1.2}$$

where $\{e_1, \ldots, e_k\}$ is sampled uniformly at random from $\binom{E}{k}$. Note that when the algorithm $\mathcal{A}$ is deterministic, (1.2) matches the definition of the average sensitivity for deterministic algorithms.

REMARK 1. *The k-average sensitivity of an algorithm $\mathcal{A}$ with respect to the total variation distance can be defined as $\mathbb{E}_{\{e_1,\ldots,e_k\}\sim\binom{E}{k}} \left[d_{\mathrm{TV}}\big(\mathcal{A}(G), \mathcal{A}(G - \{e_1, \ldots, e_k\})\big)\right]$, where $d_{\mathrm{TV}}(\cdot, \cdot)$ denotes the total variation distance between two distributions. It is easy to observe that, if the k-average sensitivity of an algorithm with respect to the total variation distance is at most $\gamma(G)$, then its k-average sensitivity is bounded by $\mathsf{H} \cdot \gamma(G)$, where the H is the maximum over Hamming weights of all solutions output (with nonzero probability) by running $\mathcal{A}$ on $G = (V, E)$ and on all the graphs in $\{G - \{e_1, \ldots, e_k\} : \{e_1, \ldots, e_k\} \in \binom{E}{k}\}$.*

EXAMPLE 3. *Randomness does not help improve the average sensitivity of algorithms for the s-t shortest path problem. Think of the cycle graph given in Example 2, and suppose that a randomized algorithm $\mathcal{A}$ outputs the s-t paths P and Q with probability p and $q = 1 - p$, respectively. Then, the average sensitivity is $p \cdot 1/2 \cdot (n/2) + q \cdot 1/2 \cdot (n/2) = \Omega(n)$.*

1.1 Basic properties of average sensitivity

Our definition of average sensitivity has many nice properties. In this section, we discuss some such properties of average sensitivity that are useful in the design of our stable-on-average algorithms. We denote by $\mathcal{G}$ the (infinite) set consisting of all graphs. Given a graph $G = (V, E)$ and $e \in E$, we use $G - e$ as a shorthand for $G - \{e\}$. We use n and m to denote the number of vertices and edges in the input graph, respectively.

Bounds on k-average sensitivity from bounds on average sensitivity. This is one of the most important properties of our definition of average sensitivity. It essentially says that a bound on the (1-)average sensitivity of an algorithm can be used to obtain a bound on the k-average sensitivity of that algorithm for $k \geq 1$. In other words, it is enough to analyze the average sensitivity of an algorithm with respect to the removal of a single edge.

THEOREM 1.1. *Let $\mathcal{A}$ be an algorithm for a graph problem with average sensitivity at most $f(n, m)$. Then, for any integer $k \geq 1$, the algorithm $\mathcal{A}$ has k-average sensitivity at most $\sum_{i=1}^{k} f(n, m - i + 1)$.*

In particular, if the average sensitivity of an algorithm is bounded from above by a nondecreasing function of the number of edges, then its k-average sensitivity is at most k times the upper bound on its average sensitivity.

Sequential composition. Another useful feature of our definition of average sensitivity is that one can obtain a stable-on-average algorithm by sequentially applying several stable-on-average subroutines. The following two *sequential composition theorems* formalize this feature.

THEOREM 1.2 (SEQUENTIAL COMPOSITION) *Consider two randomized algorithms $\mathcal{A}_1 : \mathcal{G} \to \mathcal{S}_1, \mathcal{A}_2 : \mathcal{G} \times \mathcal{S}_1 \to \mathcal{S}_2$. Suppose that the average sensitivity of $\mathcal{A}_1$ with respect to the total variation distance is $\gamma_1(G)$ and the average sensitivity of $\mathcal{A}_2(\cdot, S_1)$ is $\beta_2^{(S_1)}(G)$ for any $G \in \mathcal{G}, S_1 \in \mathcal{S}_1$. Let $\mathcal{A} : \mathcal{G} \to \mathcal{S}_2$ be a randomized algorithm obtained by composing $\mathcal{A}_1$ and $\mathcal{A}_2$, that is, $\mathcal{A}(G) = \mathcal{A}_2(G, \mathcal{A}_1(G))$ for $G \in \mathcal{G}$. Then, the average sensitivity of $\mathcal{A}$ on a graph $G \in \mathcal{G}$ is $\mathsf{H} \cdot \gamma_1(G) + \mathbb{E}_{S_1 \sim \mathcal{A}_1(G)}\left[\beta_2^{(S_1)}(G)\right]$, where H denotes the maximum over Hamming weights of all the solutions output (with nonzero probability) by running $\mathcal{A}$ on G and all of the graphs in $\{G - e : e \in E\}$.*

Our second composition theorem is for the average sensitivity with respect to the total variation distance. This is also useful for analyzing the average sensitivity with respect to the earth mover's distance, as it can be bounded by the average sensitivity with respect to the total variation distance times the maximum over Hamming weights of solutions output, as in Remark 1.

THEOREM 1.3 (SEQUENTIAL COMPOSITION W.R.T. THE TV DISTANCE) *Consider ℓ randomized algorithms $\mathcal{A}_i : \mathcal{G} \times \prod_{j=1}^{i-1} \mathcal{S}_j \to \mathcal{S}_i$ for $i \in \{1, \ldots, \ell\}$. Suppose that, for each $i \in \{1, \ldots, \ell\}$, the average sensitivity of $\mathcal{A}_i(\cdot, S_1, \ldots, S_{i-1})$ is $\gamma_i(G)$ with respect to the total variation distance for every $G \in \mathcal{G}$ and $S_1 \in \mathcal{S}_1, \ldots, S_{i-1} \in \mathcal{S}_{i-1}$. Consider a sequence of computations $S_1 = \mathcal{A}_1(G), S_2 = \mathcal{A}_2(G, S_1), \ldots, S_\ell = \mathcal{A}_\ell(G, S_1, \ldots, S_{\ell-1})$ for $G \in \mathcal{G}$. Let $\mathcal{A} : \mathcal{G} \to \mathcal{S}_\ell$ be a randomized algorithm that performs this sequence of computations on input G and outputs S_ℓ. Then, the average sensitivity of $\mathcal{A}$ on a graph $G \in \mathcal{G}$ with respect to the total variation distance is at most $\sum_{i=1}^{\ell} \gamma_i(G)$.*

Parallel composition. It is often the case that there are multiple algorithms that solve the same problem albeit with different average sensitivity guarantees. Such stable-on-average algorithms can be combined via *parallel composition*, where we run these algorithms according to a distribution determined by the input graph. The advantage of parallel composition is that the average sensitivity of the resulting algorithm might be better than that of the component algorithms.

THEOREM 1.4 (PARALLEL COMPOSITION) *Let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ be algorithms for a graph problem with average sensitivities $\beta_1(G), \ldots, \beta_\ell(G)$, respectively. Let $\mathcal{A}$ be an algorithm that, given a graph G, runs $\mathcal{A}_i$ with probability $\rho_i(G)$ for $i \in \{1, \ldots, \ell\}$, where $\sum_{i \in \{1,\ldots,\ell\}} \rho_i(G) = 1$. Let H denote the maximum over Hamming weights of all solutions output (with nonzero probability) by running $\mathcal{A}$ on G and on all the graphs in $\{G - e : e \in E\}$. Then the average sensitivity of $\mathcal{A}$ is at most $\sum_{i \in \{1,\ldots,\ell\}} \rho_i(G) \cdot \beta_i(G) + \mathsf{H} \cdot \mathbb{E}_{e \sim E}\left[\sum_{i \in \{1,\ldots,\ell\}} |\rho_i(G) - \rho_i(G - e)|\right]$.*

In this paper, we use the above theorem extensively to combine algorithms with different average sensitivities.

1.2 Connection to sublinear-time algorithms

We show a relationship between the average sensitivity of an algorithm and the query complexity of a sublinear-time algorithm [40, 15, 51] that simulates oracle access to the solution output by the former algorithm. Roughly speaking, we show, in Theorem 1.5, that the average sensitivity of an algorithm $\mathcal{A}$ is bounded by the query complexity of another algorithm $\mathcal{O}$, which we call a *solution oracle*, where $\mathcal{O}$ *queries* the edges of the graph G and simulates oracle access to the solution produced by $\mathcal{A}$ on input G. We first formalize the notion of a solution oracle.

DEFINITION 1 (SOLUTION ORACLE) *Consider a deterministic algorithm $\mathcal{A} : \mathcal{G} \to \mathcal{S}$ for a graph problem, where each solution output by $\mathcal{A}$ is a subset of the set of edges of the input graph. An algorithm $\mathcal{O}$ is a* solution oracle *for $\mathcal{A}$ if $\mathcal{O}$ satisfies:*

- *$\mathcal{O}$ has access to a graph $G = (V, E)$, which is represented as adjacency lists, via* neighbor queries*, where each query is of the form (v, i) for $v \in V$ and $i \in [|V|]$ and the answer is the i-th neighbor of vertex v in its adjacency list (and a special symbol if i is larger than the degree of v),*
- *given an edge $e \in E$ as input, $\mathcal{O}$ queries G and outputs whether e is contained in the solution obtained by running $\mathcal{A}$ on G.*

Given a randomized algorithm $\mathcal{A}$ and an associated function $r : \mathbb{N} \to \mathbb{N}$, the solution oracle $\mathcal{O}$ of $\mathcal{A}$ first generates a random string $\pi \in \{0,1\}^{r(|V|)}$ and then runs the solution oracle $\mathcal{O}_\pi$ of the deterministic algorithm $\mathcal{A}_\pi$ obtained by fixing the randomness of $\mathcal{A}$ to π.

Note that an analogous definition can be made for algorithms that output a subset of vertices.

THEOREM 1.5 (SUBLINEARITY IMPLIES LOW AVERAGE SENSITIVITY) *Consider a randomized algorithm $\mathcal{A} : \mathcal{G} \to \mathcal{S}$ for a graph problem, where each solution output by $\mathcal{A}$ is a subset of the set of edges in the input graph. Assume that there exists a solution oracle $\mathcal{O}$ for $\mathcal{A}$ such that $\mathcal{O}$ makes at most $q(G)$ queries to G in expectation, where this expectation is taken over the random coins of $\mathcal{O}$ and over input edges $e \in E$. Then, $\mathcal{A}$ has average*

sensitivity at most $q(G)$. Moreover, given the promise that the input graphs satisfy $|E| \geq |V|$, the statement applies also to algorithms for which each solution is a subset of the set of vertices in the input graph.

We use Theorem 1.5 to design a stable-on-average matching algorithm (Theorem 5.4) based on a sublinear-time matching algorithm due to Yoshida et al. [51].

Closely related to Theorem 1.5 is Corollary 1.1, which says that if a graph problem has a *local computation algorithm* (LCA), then one can design a stable-on-average algorithm for that problem. LCAs, whose definition we give below, were introduced by Rubinfeld et al. [47] and has been widely studied ever since [2, 11, 16, 26, 28, 29, 30, 31, 32, 43, 46]. For more information on LCAs, we refer the interested reader to an excellent survey on the topic by Levi and Medina [27].

Definition 2 (Local Computation Algorithm (LCA)) *Consider a graph problem $\mathcal{P} : \mathcal{G} \to \mathcal{S}$, where the output to the problem is a subset of edges of the input graph. Let $\delta : \mathbb{N} \to [0,1]$ and $q, r : \mathbb{N} \to \mathbb{N}$. A (q, r, δ)-LCA for $\mathcal{P}$ is an algorithm $\mathcal{L}$ that, given query access to a graph $G = (V, E)$ (as in Definition 1), first generates a random string $\pi \in \{0,1\}^{r(|V|)}$, and satisfies:*

- *given an input $e \in E$, the algorithm $\mathcal{L}$ makes at most $q(|V|)$ queries to G and answers whether e is part of a solution to the problem $\mathcal{P}$ on graph G, and*
- *the answers of $\mathcal{L}$ to all possible input edges are consistent with a single feasible solution to $\mathcal{P}$ on G.*

For every graph G, the probability (over the choice of random string) that there exists an input edge for which $\mathcal{L}$ makes more than $q(|V|)$ queries is at most $\delta(|V|)$.

We mention that $\mathcal{L}$ is not allowed to perform any preprocessing on the graph. Additionally, the same set of edges is queried by $\mathcal{L}$ when the same edge is given as input multiple times.

Note that one can have an analogous definition of LCAs for graph problems where each solution is a subset of vertices. We have the following result which is a direct corollary of Theorem 1.5.

Corollary 1.1 (LCAs imply stable-on-average algorithms) *Consider a graph problem $\mathcal{P} : \mathcal{G} \to \mathcal{S}$. Let $\delta : \mathbb{N} \to [0,1]$ and $q, r : \mathbb{N} \to \mathbb{N}$. If $\mathcal{P}$ has a (q, r, δ)-LCA $\mathcal{L}$, then, there exists an algorithm $\mathcal{A}$ for $\mathcal{P}$, that on input $G = (V, E)$, has an average sensitivity of at most $q(|V|) + |E| \cdot \delta(|V|)$.*

Theorem 1.5 and Corollary 1.1 cement the intuition that strong locality guarantees for solutions output by an algorithm imply that the removal of edges from a graph affects only the presence of a few elements (edges or vertices) in the solution, which in turn implies low average sensitivity. On the contrapositive side, Corollary 1.1 implies that a lower bound on the average sensitivity of algorithms for a problem implies a lower bound on the query complexity of an LCA (with failure probability $o(1/n^2)$) for the same problem, where n denotes the number of vertices. Exploiting this result, we show that every LCA for 2-coloring has query complexity $\Omega(n)$, thereby answering an open question raised by Czumaj et al. [7]. We believe that this connection has the potential to shed more light on fundamental limits of LCAs and is of independent interest.

1.3 Stable-on-average algorithms for concrete problems We summarize, in Table 1, the average sensitivity bounds that we obtain for various concrete problems. We use n, m, OPT to denote the number of vertices, the number of edges, and the optimal value.

All of our algorithms run in polynomial time, and for $k \geq 1$, upper bounds on k-average sensitivity of these algorithms can be easily obtained using Theorem 1.1. Except in the case of our algorithm for the minimum spanning forest problem, our stable-on-average algorithms are all randomized. Our lower bounds hold for any (randomized) algorithm to solve the respective problems with the specified approximation guarantee.

For the minimum spanning forest problem, we show that the classical Kruskal's algorithm [23] has average sensitivity $O(n/m)$, which is at most 1, and is quite small considering that Kruskal's algorithm is deterministic and that the spanning forest can have $\Omega(m)$ edges. We also show a matching lower bound of $\Omega(n/m)$ for the minimum spanning forest problem, implying that the average sensitivity of Kruskal's algorithm is optimal. In contrast, we show that Prim's algorithm can have average sensitivity $\Omega(m)$ for a natural (and deterministic) rule of breaking ties among edges.

For the global minimum cut problem, we show that every algorithm that outputs the exact mincut has to have average sensitivity $\Omega(n)$. However, by allowing for a multiplicative approximation guarantee of $2 + \epsilon$ for $\epsilon > 0$, we design a global minimum cut algorithm with average sensitivity $n^{O(\frac{1}{\epsilon \mathsf{OPT}})}$. If $\mathsf{OPT} = \Omega(\log n)$, the average sensitivity of our algorithm is $O(1)$, which is quite small. We also prove a nearly tight lower bound on the average sensitivity of any algorithm that guarantees a purely multiplicative approximation to the minimum cut size. In particular, when OPT is $o(\log n)$, our lower bound matches, up to a polylogarithmic factor, our upper bound for 3-approximating the minimum cut size.

Table 1: Our results. Here n, m, OPT denote the number of vertices, the number of edges, and the optimal value, respectively, and $\epsilon \in (0,1)$ is an arbitrary constant. The notation $\widetilde{O}(\cdot)$ hides polylogarithmic factors in n. Approximation guarantees are multiplicative unless specified otherwise.

Problem	Output	Approximation Guarantee	Average Sensitivity	Reference
Minimum Spanning Forest	Edge set	1	$O(\frac{n}{m})$	Sec. 3
		$< \infty$	$\Omega(\frac{n}{m})$	Sec. 3
Global Minimum Cut	Vertex set	$2+\epsilon$	$n^{O(\frac{1}{\epsilon \mathsf{OPT}})}$	Sec. 4.1
		1	$\Omega(n)$	Full Version [50]
		$< \infty$	$\Omega\left(\frac{n\frac{1}{\mathsf{OPT}}}{\mathsf{OPT}^2}\right)$	Full Version [50]
Minimum s-t Cut	Vertex set	additive $O(n^{2/3})$	$O\left(n^{2/3}\right)$	Full Version [50]
Maximum Matching	Edge set	1/2	1	Sec. 5.2
		$1-\epsilon$	$\widetilde{O}\left(\left(\frac{\mathsf{OPT}}{\epsilon^3}\right)^{\frac{1}{1+\Omega(\epsilon^2)}}\right)$	Sec. 5.3
		1	$\Omega(n)$	Full Version [50]
Minimum Vertex Cover	Vertex set	2	2	Sec. 5.2
2-Coloring	Vertex set	—	$\Omega(n)$	Sec. 6

Our lower bound of $\Omega(n)$ on the average sensitivity of algorithms that output the exact global minimum cut also applies to the minimum s-t cut problem. In contrast, we show that it is possible to achieve average sensitivity of $O(n^{2/3})$ for the minimum s-t cut problem by allowing for an additive $O(n^{2/3})$ approximation.

We show that the average sensitivity of every algorithm that outputs the exact maximum matching is $\Omega(n)$, implying that some approximation is essential to obtain nontrivial average sensitivity. We also propose two stable-on-average approximation algorithms for maximum matching. Our first algorithm has approximation ratio 1/2 and average sensitivity at most 1. This result immediately implies a 2-approximation algorithm for the minimum vertex cover problem with average sensitivity at most 2. Our second algorithm for maximum matching has approximation ratio $1-\varepsilon$ and average sensitivity $\widetilde{O}\left(\left(\mathsf{OPT}/\varepsilon^3\right)^{1/(1+\Omega(\varepsilon^2))}\right)$ for every constant $\varepsilon \in (0,1)$[1].

In the 2-coloring problem, given a bipartite graph, we are to output one part in the bipartition. For this problem, we show a lower bound of $\Omega(n)$ for the average sensitivity, that is, there is no algorithm with nontrivial average sensitivity.

Implications on the query complexity of LCAs. Recall that, by Corollary 1.1, the average sensitivity lower bound for a problem implies an identical lower bound on the query complexity of an LCA (with failure probability $o(1/n^2)$) for the same problem. This implies that every LCA for exact maximum matching, global minimum cut, and minimum s-t cut has query compleity $\Omega(n)$. Additionally, every LCA giving a purely multiplicative approximation guarantee for the global minimum cut has query complexity $\Omega\left(\frac{n\frac{1}{\mathsf{OPT}}}{\mathsf{OPT}^2}\right)$. Additionally, as mentioned earlier, every LCA for 2-coloring has query complexity $\Omega(n)$, which answers an open question raised by Czumaj et al. [7].

1.4 Discussions on average sensitivity

In this section, we discuss the reasoning behind the various choices we made in coming up with our definition of average sensitivity.

Output representation. Average sensitivity is dependent on the output representation. For example, we can double the average sensitivity by duplicating the output. A natural idea for alleviating this issue is to normalize the average sensitivity by the maximum Hamming weight H of a solution. However, for minimization problems where the optimal value OPT could be much smaller than H, such a normalization can diminish subtle differences in average sensitivity, e.g., $O(\mathsf{OPT}^{1/2})$ vs $O(\mathsf{OPT})$. It is an interesting open question whether there is a canonical way to normalize average sensitivity so that the resulting quantity is independent of the output representation.

[1] Recently, Yoshida and Zhou [52] showed that there is an $(1-\varepsilon)$-approximation algorithm with the *worst-case* sensitivity $O_\epsilon(1)$, where the dependency on $1/\epsilon$ is triply exponential.

Sensitivity against adversarial edge removals. It is also natural to take the maximum, instead of the average, over edges in definitions (1.1) and (1.2), which can be seen as sensitivity against adversarial edge removals. Indeed a similar notion has been proposed to study algorithms for geometric problems [35]. However, in the case of graph algorithms, it is hard to guarantee that the output of an algorithm does not change much after removing an arbitrary edge. Moreover, by a standard averaging argument, one can say that for 99% of arbitrary edge removals, the sensitivity of an algorithm is asymptotically equal to the average sensitivity, which is sufficient in most cases.

Average sensitivity w.r.t. edge additions. As another variant of average sensitivity, it is natural to consider incorporating edge additions in definitions (1.1) and (1.2). If an algorithm is stable-on-average against edge additions, then in addition to the case of not knowing the true graph as we have discussed earlier, it will be useful for the case that the graph dynamically changes but we want to prevent the output of the algorithm from fluctuating too much. However, in contrast to removing edges, it is not always clear how we should add edges to the graph in definitions (1.1) and (1.2). A naive idea is sampling k pairs of vertices uniformly at random and adding edges between them. This procedure makes the graph close to a graph sampled from the Erdős-Rényi model [10], which does not represent real networks such as social networks and road networks well. To avoid this subtle issue, in this work, we focus on removing edges.

Alternative notion of average sensitivity for randomized algorithms. Consider a randomized algorithm $\mathcal{A}$ that, given a graph G on n vertices, generates a random string $\pi \in \{0,1\}^{r(n)}$ for some function $r : \mathbb{N} \to \mathbb{N}$, and then runs a deterministic algorithm $\mathcal{A}_\pi$ on G, where the algorithm $\mathcal{A}_\pi$ has π hardwired into it. Assume that $\mathcal{A}_\pi$ can be applied to any graph. It is also natural to define the average sensitivity of $\mathcal{A}$ as

$$\mathop{\mathbb{E}}_{e\sim E}\left[\mathop{\mathbb{E}}_{\pi}\left[d_{\mathrm{Ham}}\big(\mathcal{A}_\pi(G),\mathcal{A}_\pi(G-e)\big)\right]\right]. \tag{1.3}$$

In other words, we measure the expected distance between the outputs of $\mathcal{A}$ on G and $G-e$ when we feed the same string π to $\mathcal{A}$, over the choice of π and edge e. Note that (1.3) upper bounds (1.2) because, in the definition of the earth mover's distance, we optimally transport probability mass from $\mathcal{A}(G)$ to $\mathcal{A}(G-e)$ whereas, in (1.3), how the probability mass is transported is not necessarily optimal.

We can actually bound (1.3) for some of our algorithms. In this work, however, we focus on the definition (1.2) because the assumption that A_π can be applied to any graph does not hold in general, since the length of π might depend on parameters of the input graph other than the number of vertices. Moreover, bounding (1.3) is unnecessarily tedious and is not very enlightening.

1.5 Overview of our techniques In this section, we give a high level outline of the technical ideas used in the design and analysis of our stable-on-average algorithms and also in constructing our lower bounds.

Global minimum cut. For the global minimum cut problem, our algorithm is inspired by a differentially private algorithm[2] for the same problem by Gupta et al. [14]. Our algorithm, given a parameter $\varepsilon > 0$ and a graph G as input, first enumerates a list of cuts whose sizes are at most $(2+\varepsilon)\cdot\mathsf{OPT}$; this enumeration can be done in polynomial time as shown by Karger's theorem [18]. The algorithm then outputs a cut from the list with probability exponentially small in the product of the size of the cut and $O(1/\varepsilon\cdot\mathsf{OPT})$. The main argument in analyzing the average sensitivity of the algorithm is that the aforementioned distribution is very close (in earth mover's distance) to a related Gibbs distribution on the set of all cuts in the graph. Therefore the average sensitivity of the algorithm is of the same order as that of the average sensitivity of sampling a cut from such a Gibbs distribution, where the latter sampling task requires exponential time. We finally show that the average sensitivity of sampling a cut from this Gibbs distribution is at most $n^{O(1/\varepsilon\mathsf{OPT})}$.

Minimum s-t cut. The first stage of our algorithm consists of solving an LP relaxation for the minimum s-t cut problem in a stable way, for an appropriately defined notion of sensitivity. Given a graph $G=(V,E)$ and vertices $s,t\in V$, the relaxation contains variables $\boldsymbol{d}(\{u,v\})\in[0,1]$ for each $\{u,v\}\in\binom{V}{2}$, where these variables can be thought of as representing a *pseudometric* over the vertices. The constraints include triangle inequalities, and also a special constraint $\boldsymbol{d}(\{s,t\})=1$. The objective is to minimize the sum of the variables associated with the edges in G. Intuitively, if $\boldsymbol{d}(\{s,v\})$ is *large* in a solution to the linear program, then the vertex v falls on the *t-side* of the s-t cut represented by the solution.

Our stable LP solving strategy works by solving a related LP that is identical to the original LP, except for a regularization term added to the objective function. This regularization term is $n^{-1/3}$ times the ℓ_2 norm of the vector of variables $(\boldsymbol{d}(\{s,v\}) : v\in V)$, where we use $\|\boldsymbol{d}\|_s$ to denote this norm. It is easy to show that the

[2] We compare and contrast the definitions of average sensitivity and differential privacy in Section 1.6.

value of the optimal solution to the regularized LP is within an additive $O(n^{2/3})$ of the value of the optimal solution to the original LP. We also show that for all $e \in E$, the solutions output by our LP solver graphs G and $G - e$ are close to each other with respect to $\|\cdot\|_s$. Here, we use the fact that the regularized objective function is strongly convex with respect to $\|\cdot\|_s$.

Given a solution to the (regularized) LP relaxation, our rounding procedure samples a threshold $\tau \in [0,1]$ uniformly at random and outputs the set S consisting of all vertices $u \in V$ such that $d(\{s,u\}) \leq \tau$. The approximation guarantee of this algorithm follows from the fact that we are rounding based on a near optimal solution to the linear programming relaxation. To analyze the average sensitivity of the algorithm, we first show that the earth mover's distance (with respect to Hamming distance) between the outputs of the rounding procedure for inputs $\boldsymbol{d}, \boldsymbol{d}' \in [0,1]^{\binom{V}{2}}$ is bounded by the ℓ_1 distance between the vectors $(\boldsymbol{d}(\{s,v\}) : v \in V)$ and $(\boldsymbol{d}'(\{s,v\}) : v \in V)$. Combining this with the bound on the average sensitivity of our LP solving strategy, we obtain our final bound on the average sensitivity for our algorithm to approximate the minimum s-t cut.

Maximum matching. Our stable-on-average $\frac{1}{2}$-approximation algorithm for the maximum matching problem first considers a uniformly random ordering of the edges in the input graph, and then greedily adds edges to the matching according to that ordering. In the context of dynamic distributed algorithms, Censor-Hillel et al. [6] showed that at most 1 edge changes in the matching (in expectation) due to the removal of a uniformly random edge, where the expectation is taken over the edge removed and the ordering of edges. This result immediately implies that the average sensitivity of this randomized greedy matching algorithm is at most 1. In addition, it implies a 2-approximation algorithm for minimum vertex cover with average sensitivity at most 2.

There are several components to the design and analysis of our stable-on-average $(1-\varepsilon)$-approximation algorithm. Our starting point is the observation (Theorem 1.5) that the existence of a sublinear-time solution oracle $\mathcal{O}$ (see Definition 1) for an algorithm $\mathcal{A}$ implies that $\mathcal{A}$ is stable-on-average. We use Theorem 1.5 to bound the average sensitivity of a $(1-\varepsilon')$-approximation algorithm $\mathcal{A}$ for the maximum matching problem, where $\varepsilon' = \Omega(\varepsilon)$. Specifically, $\mathcal{A}$ constructs a matching by considering augmenting paths of increasing length, and augmenting the (initially empty) matching iteratively, where the paths of each length are considered in a uniformly random order. Yoshida et al. [51] constructed a sublinear-time solution oracle that, given a uniformly random edge $e \in E$ as input, makes $O\left(\Delta^{O(1/(\varepsilon')^2)}\right)$ queries to G in expectation and answers whether e is in the matching output by $\mathcal{A}$ on G, where the expectation is over the choice of input e and the randomness in $\mathcal{A}$, and Δ is the maximum degree of G. Combined with Theorem 1.5, this implies that the average sensitivity of $\mathcal{A}$ is $O\left(\Delta^{O(1/(\varepsilon')^2)}\right)$.

Next, we transform $\mathcal{A}$ to also work for graphs of unbounded degree as follows. The idea is to remove vertices of degree at least $\frac{m}{\varepsilon' \mathsf{OPT}}$ from the graph and run $\mathcal{A}$ on the resulting graph. This transformation affects the approximation guarantee only by an additive $\varepsilon'\mathsf{OPT}$ term, since the number of such *high degree* vertices is small. However, this thresholding procedure might itself have high average sensitivity, since the thresholds for G and $G - e$ can be very different for all $e \in E$.

We circumvent this issue by using a Laplace random variable L as the threshold, where the distribution of L is tightly concentrated around $\frac{m}{\varepsilon' \mathsf{OPT}}$. We use our sequential composition theorem (Theorem 1.2) in order to analyze the average sensitivity of the resulting procedure, where we consider the instantiation of the Laplace random threshold as the first algorithm, and the remaining steps in the procedure as the second algorithm. The first term in the expression given by Theorem 1.2 turns out to be a negligible quantity and is easy to bound. The main task in bounding the second term is to bound, for all $x \in \mathbb{R}$, the average sensitivity of a procedure $\mathcal{A}_x$ that, on the input graph G, removes all vertices of degree at least x from G and runs the augmenting paths-based matching algorithm. The heart of the argument in bounding this average sensitivity is that given a solution oracle $\mathcal{O}$ with query complexity $q(\Delta)$ for an algorithm $\mathcal{A}$, we can, for all $x \in \mathbb{R}$, construct a solution oracle $\mathcal{O}_x$ for the algorithm $\mathcal{A}_x$. Moreover, the query complexity of $\mathcal{O}_x$ is at most $O(x^2 q(x))$. By Theorem 1.5, this is also a bound on the average sensitivity of $\mathcal{A}_x$. Using this, we bound the second term in the expression given by Theorem 1.2 as $\mathbb{E}_L\left[O(L^2 q(L))\right] = O\left(\left(\frac{m}{\varepsilon' \mathsf{OPT}}\right)^{O(1/(\varepsilon')^2)}\right)$.

An issue with the aforementioned matching algorithm is that its average sensitivity is poor for graphs with small values of OPT. We observe that, in contrast to this, the algorithm that simply outputs the lexicographically smallest maximum matching has average sensitivity $O(\mathsf{OPT}^2/m)$, since the output matching stays the same unless an edge in the matching is removed. We obtain our final stable-on-average $(1-\varepsilon)$-approximation algorithm for the maximum matching problem by running these two algorithms according to a probability distribution determined by the input

graph. Using our parallel composition theorem, we bound the average sensitivity of the resultant algorithm as $\widetilde{O}\left(\left(\mathsf{OPT}/\varepsilon^3\right)^{1/(1+\Omega(\varepsilon^2))}\right)$.

2-coloring. To show our $\Omega(n)$ lower bound on the average sensitivity for 2-coloring, consider the set of all paths on n vertices and the set of all graphs obtained by removing exactly one edge from these paths (called 2-part-paths). A path has exactly two ways of being 2-colored and a 2-path has four ways of being 2-colored. A path and 2-part-path are neighbors if the latter is obtained from the former by removing an edge. A 2-part-path has at most four neighbors. The output distribution of any 2-coloring algorithm $\mathcal{A}$ on a 2-part-path can be close (in earth mover's distance) only to those of at most 2 of its neighboring paths. If $\mathcal{A}$, however, has low average sensitivity, the output distributions of $\mathcal{A}$ have to be close on a large fraction of pairs of neighboring graphs, which gives a contradiction.

1.6 Related work

In this section, we discuss some of the related sensitivity definitions from earlier work.

Average sensitivity of network centralities. *(Network) centrality* is a collective name for indicators that measure importance of vertices or edges in a network. Notable examples are closeness centrality [3, 4, 48], harmonic centrality [33], betweenness centrality [12], and PageRank [42]. To compare these centralities qualitatively, Murai and Yoshida [36] recently introduced the notion of average-case sensitivity for centralities. Fix a vertex centrality measure c; let $c_G(v)$ denote the centrality of a vertex $v \in V$ in a graph $G = (V, E)$. Then, the *average-case sensitivity* of c on G is defined as

$$S_c(G) = \mathop{\mathbb{E}}_{e\sim E} \mathop{\mathbb{E}}_{v\sim V} \frac{|c_{G-e}(v) - c_G(v)|}{c_G(v)},$$

where e and v are sampled uniformly at random. They showed various upper and lower bounds for centralities. See [36] for details.

Since a centrality measure assigns real values to vertices, they studied the relative change of the centrality values upon removal of random edges. As our focus in this work is on graph algorithms, our notion (1.2) measures the Hamming distance between solutions when one removes random edges.

Average sensitivity of spectral clustering. In a subsequent work to this study, Pan and Yoshida [44] analyzed the average sensitivity of k-way spectral clustering to assess its reliability and efficiency. They showed that the average sensitivity is proportional to $\lambda_k/\lambda_{k+1}^2$, where λ_i is the i-th smallest eigenvalue of the normalized Laplacian. This result and higher-order Cheeger's inequality [24] suggest that, if the input graph can be well partitioned into k components but not more, k-way spectral clustering is stable-on-average and hence can be safely used. If not, k-way spectral clustering may not be stable-on-average, but then we should not try to find k-way clustering in the first place.

Differential privacy. *Differential privacy* [8] is a notion closely related to average sensitivity. Assuming the existence of a neighbor relation over inputs, the definition of differential privacy requires that the distributions of outputs on neighboring inputs are similar. The variant of differential privacy closest to our definition of average sensitivity is edge differential privacy introduced by Nissim et al. [41] and further studied by [17, 14, 20, 21, 19, 45]. Here, the neighbors of a graph $G = (V, E)$ are defined to be $\{G - e\}_{e\in E}$. For $\varepsilon > 0$, we say that an algorithm is *ε-differentially private* if for all $e \in E$,

$$\begin{aligned}(1.4)\qquad &\exp(-\varepsilon) \cdot \Pr[\mathcal{A}(G - e) \in \mathcal{S}] \le \Pr[\mathcal{A}(G) \in \mathcal{S}] \\ &\le \exp(\varepsilon) \cdot \Pr[\mathcal{A}(G - e) \in \mathcal{S}]\end{aligned}$$

for any set of solutions $\mathcal{S}$.

Differential privacy has stricter requirements than average sensitivity. Firstly, differential privacy is a *worst-case* sensitivity notion. Moreover, since differential privacy guarantees that the probabilities of outputting a specific solution on G and $G - e$ are close to each other, the total variation distance between the two distributions $\mathcal{A}(G)$ and $\mathcal{A}(G - e)$ must be small. The earth mover's distance between two output distributions can be small even if the total variation distance between them is large, and therefore, even if an algorithm is not differentially private, it could still be stable-on-average. Despite these differences, our stable-on-average algorithm for the global minimum cut problem is inspired by a differentially private algorithm for the same problem [14].

Generalization and stability of learning algorithms. Generalization [49] is a fundamental concept in statistical learning theory. Given samples $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ from an unknown true distribution $\mathcal{D}$ over a dataset, the goal of a learning algorithm $\mathcal{L}$ is to output a parameter θ that minimizes *expected loss* $\mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}}[\ell(\boldsymbol{z};\theta)]$, where $\ell(\boldsymbol{z};\theta)$ is the loss incurred by a sample $\boldsymbol{z}$ with respect to a parameter θ. As the true distribution $\mathcal{D}$ is unknown, a frequently used approach in learning is to compute a parameter θ that minimizes the *empirical loss* $\frac{1}{n} \cdot \sum_{i=1}^{n} \ell(\boldsymbol{z}_i;\theta)$, which is an unbiased estimator of the expected loss and is purely a function of the available samples. The *generalization error* of a learner $\mathcal{L}$ is a measure of how close the empirical loss is to the expected loss as a function of the sample size n.

One technique to reduce the generalization error is to add a *regularization* term to the loss function being

minimized [5]. This also ensures that the learned parameter θ does not change much with respect to minor changes in the samples being used for learning. Therefore, in a sense, learning algorithms that use regularization can be considered as being stable according to our definition of sensitivity.

Bousquet and Elisseeff [5] defined a notion of stability for learning algorithms in relation to reducing the generalization error. Their stability notion requires that the empirical loss of the learning algorithm does not change much by removing or replacing *any* sample in the input data. In contrast, in our definition of average sensitivity, we consider removing *random* edges from a graph and measure the change in the output solution rather than that in the objective value.

1.7 Organization We show our stable-on-average algorithms for the minimum spanning forest problem, the global minimum cut problem, and the maximum matching problem problems in Sections 3, 4, and 5, respectively. We show a linear lower bound for the 2-coloring problem in Section 6. We discuss the connection of average sensitivity to sublinear-time algorithms in Section 7.

We refer the reader to the full version of this paper [50] for our stable-on-average algorithm for the minimum s-t cut problem, our lower bounds on the average sensitivity of algorithms for the global minimum cut problem and the maximum matching problem, and also the proofs of basic properties of average sensitivity. Additionally, the full version also contains the proofs omitted from the technical sections presented here.

2 Preliminaries

For a positive integer n, let $[n] = \{1, 2, \ldots, n\}$. Let $G = (V, E)$ be a graph whose vertex set is V and edge set is E. We denote by $\mathcal{G}$ the (infinite) set consisting of all graphs. We often use the symbols n, m, Δ to denote the number of vertices, the number of edges, and the maximum degree of a vertex, respectively, in the input graph. We use $\mathsf{OPT}(G)$ to denote the optimal value of a graph G in the graph problem we are concerned with. We simply write OPT when G is clear from the context. For an edge $e \in E$, we denote by $G - e$ the graph obtained by removing e from G. Similarly, for a subset of edges $F \subseteq E$, we denote by $G - F$ the graph obtained by removing every edge in F from G. For a subset of edges $F \subseteq E$, let $V(F)$ denote the set of vertices incident to an edge in F. For a positive integer $k \leq |E|$, we use the notation $\binom{E}{k}$ to denote the set of all subsets of E of cardinality k. For a subset of vertices S, let $G[S]$ be the subgraph of G induced by S. We denote by $\mathbb{R}_+$ the set of non-negative real numbers. For vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we use $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ to denote the inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$.

2.1 Exponential Mechanism The *exponential mechanism* [34] is an algorithm that, given a vector $\boldsymbol{x} \in \mathbb{R}^n$ and a real number $\eta > 0$, returns an index $i \in [n]$ with probability proportional to $e^{-\eta \boldsymbol{x}(i)}$. Just as the exponential mechanism is useful to design differentially private algorithms, it is also useful to design stable-on-average algorithms. Lemma 2.1 formalizes this statement.

LEMMA 2.1. *Let $\eta > 0$ and let $\mathcal{A}$ be the algorithm that, given a vector $\boldsymbol{x} \in \mathbb{R}^n$, applies the exponential mechanism to $\boldsymbol{x}$ and η. Then for any $t > 0$, we have*

$$\Pr_{i \sim \mathcal{A}(\boldsymbol{x})} \left[\boldsymbol{x}(i) \geq \mathsf{OPT} + \frac{\log n}{\eta} + \frac{t}{\eta} \right] \leq e^{-t},$$

where $\mathsf{OPT} = \min_{i \in [n]} \boldsymbol{x}(i)$. Moreover, for all $\boldsymbol{x}' \in \mathbb{R}^n$, we have

$$d_{\mathrm{TV}}(\mathcal{A}(\boldsymbol{x}), \mathcal{A}(\boldsymbol{x}')) = O\left(\eta \cdot \|\boldsymbol{x} - \boldsymbol{x}'\|_1\right).$$

The proof of Lemma 2.1 can be found in the full version. By setting $\eta = \log n / \epsilon$ and replacing t with $t \log n$, we get the following:

LEMMA 2.2. *Let $\epsilon > 0$. There exists an algorithm $\mathcal{A}_\epsilon$ such that, given a vector $\boldsymbol{x} \in \mathbb{R}^n$ outputs $i \in [n]$ such that*

$$\Pr_{i \sim \mathcal{A}_\epsilon(\boldsymbol{x})} \left[\boldsymbol{x}(i) \geq \mathsf{OPT} + \epsilon(1 + t) \right] \leq n^{-t},$$

for any $t > 0$, where $\mathsf{OPT} = \min_{i \in [n]} \boldsymbol{x}(i)$. Moreover, for all $\boldsymbol{x}' \in \mathbb{R}^n$, we have

$$d_{\mathrm{TV}}(\mathcal{A}_\epsilon(\boldsymbol{x}), \mathcal{A}_\epsilon(\boldsymbol{x}')) = O\left(\|\boldsymbol{x} - \boldsymbol{x}'\|_1 \cdot \frac{\log n}{\epsilon} \right).$$

3 Warm Up: Minimum Spanning Forest

To get intution about average sensitivity of algorithms, we start with the *minimum spanning forest problem*. In this problem, we are given a weighted graph $G = (V, E, w)$, where $w : E \to \mathbb{R}$ is a weight function on edges, and we want to find a forest of the minimum total weight including all the vertices.

Recall that Kruskal's algorithm [23] works as follows: Iterate over edges in the order of increasing weights, where we break ties arbitrarily. At each iteration, add the current edge to the solution if it does not form a cycle with the edges already added. The following theorem states that this simple and deterministic algorithm is stable-on-average.

THEOREM 3.1. *The average sensitivity of Kruskal's algorithm is $O(n/m)$.*

Proof. Let $G = (V, E)$ be the input graph and T be the spanning forest obtained by running Kruskal's algorithm on G, where T is represented by its set of edges. We consider how the output changes when we remove an edge $e \in E$ from G.

If the edge e does not belong to T, clearly the output of Kruskal's algorithm on $G - e$ is also T.

Suppose that the edge e belongs to T. Let T_1 and T_2 be the two trees rooted at the endpoints of e obtained by removing e from T. If $G - e$ is not connected, that is, e is a bridge in G, then Kruskal's algorithm outputs $T_1 \cup T_2$ on $G - e$. If $G - e$ is connected, then let e' be the first edge considered by Kruskal's algorithm among all the edges connecting $G[V(T_1)]$ and $G[V(T_2)]$, where $V(T_i)$ is the vertex set of T_i for $i \in [2]$. Then, Kruskal's algorithm outputs $T_1 \cup T_2 \cup \{e'\}$ on $G - e$. It follows that the Hamming distance between T and the output of the algorithm on $G - e$ is at most 2.

Therefore, the average sensitivity of Kruskal's algorithm is at most

$$\frac{m - |T|}{m} \cdot 0 + \frac{|T|}{m} \cdot 2 = O\left(\frac{n}{m}\right).$$

□

Indeed, it is not hard to show a matching lower bound.

THEOREM 3.2. *The average sensitivity of a (possibly randomized) algorithm for the minimum spanning forest problem is* $\Omega(n/m)$.

Proof. Let $\mathcal{A}$ be an algorithm for the minimum spanning forest problem, and let $G = (V, E)$ be a connected graph with n vertices and m edges.

For each $e \in E$, let μ_e be the probability distribution over $\mathcal{F}(G) \times \mathcal{F}(G - e)$ such that $d_{\mathrm{EM}}(\mathcal{A}(G), \mathcal{A}(G - e)) = \mathbb{E}_{(F,F_e)\sim\mu_e} |F \triangle F_e|$, where $\mathcal{F}(G)$ is the set of spanning forests of G. Note that the marginal distribution of μ_e on the first coordinate is identical for all $e \in E$. Let $\mu_{e,F}$ be the distribution over $\mathcal{F}(G - e)$ defined as (the second coordinate of) the distribution μ_e conditioned on the first coordinate being F. Then, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{e\sim E}\left[d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}(G-e)\big)\right] = \mathop{\mathbb{E}}_{e\sim E}\mathop{\mathbb{E}}_{(F,F_e)\sim\mu_e} |F \triangle F_e| \\
&= \mathop{\mathbb{E}}_{F\sim\mathcal{F}(G)}\mathop{\mathbb{E}}_{e\sim E}\mathop{\mathbb{E}}_{F_e\sim\mu_{e,F}} |F \triangle F_e| \\
&= \mathop{\mathbb{E}}_{F\sim\mathcal{F}(G)}\left[\frac{1}{m}\sum_{e\in E}\mathop{\mathbb{E}}_{F_e\sim\mu_{e,F}} |F \triangle F_e|\right] \\
&\geq \mathop{\mathbb{E}}_{F\sim\mathcal{F}(G)}\left[\frac{1}{m}\sum_{e\in F}\mathop{\mathbb{E}}_{F_e\sim\mu_{e,F}} |F \triangle F_e|\right] \\
&\geq \mathop{\mathbb{E}}_{F\sim\mathcal{F}(G)}\left[\frac{1}{m}\sum_{e\in F}\mathop{\mathbb{E}}_{F_e\sim\mu_{e,F}} 1\right] \\
&= \frac{n-1}{m} = \Omega\left(\frac{n}{m}\right),
\end{aligned}$$

where in the second inequality we used the fact that $e \in F$ and $e \notin F_e$. □

In the full version, we show that Prim's algorithm, another classical algorithm for the minimum spanning forest problem, has average sensitivity $\Omega(m)$ for a certain natural tie breaking rule. We mention that this lower bound holds even for unweighted graphs.

4 Global Minimum Cut

For a graph $G = (V, E)$ and a vertex set $S \subseteq V$, we define $\mathsf{cost}(G, S)$ to be the number of edges in E that cross the cut $(S, V \setminus S)$. Then in the *global minimum cut problem*, given a graph $G = (V, E)$, we want to compute a vertex set $\emptyset \subsetneq S \subsetneq V$ that minimizes $\mathsf{cost}(G, S)$. In this section, we discuss upper and lower bounds on the average sensitivity for the global minimum cut problem.

4.1 Upper bound In this section, we show the following.

THEOREM 4.1. *For $\varepsilon > 0$, there exists a polynomial time algorithm for the global minimum cut problem with approximation ratio $2 + \varepsilon$ and average sensitivity $n^{O(1/\varepsilon\mathsf{OPT})}$.*

Let OPT be the minimum size of a cut in G. Our algorithm enumerates cuts of small size and then output a vertex set S with probability $\exp(-\alpha \cdot \mathsf{cost}(G, S))$ for a suitable α. See Algorithm 1 for details.

Algorithm 1: STABLE ALGORITHM FOR GLOBAL MINIMUM CUT

Input: undirected graph $G = (V, E)$, $\varepsilon > 0$

1 Compute the value OPT;
2 Let $\alpha \leftarrow \frac{(2+1/\varepsilon)\log n}{\mathsf{OPT}}$ denote a parameter;
3 Enumerate all cuts of size at most $(2 + 7\varepsilon)\mathsf{OPT} + 2\varepsilon$;
4 Sample a vertex set S (from among the cuts enumerated) with probability proportional to $\exp(-\alpha \cdot \mathsf{cost}(G, S))$;
5 **return** S.

The approximation ratio of the Algorithm 1 is $2+9\varepsilon$: It clearly holds when $\mathsf{OPT} \geq 1$, and it also holds when $\mathsf{OPT} = 0$ because we only output a cut of size zero (for $\varepsilon < 1/2$). The following theorem due to Karger [18] directly implies that it runs in time polynomial in the input size for any constant $\varepsilon > 0$.

THEOREM 4.2 ([18]) *Given a graph G on n vertices with the minimum cut size c and a parameter $\alpha \geq 1$,*

the number of cuts of size at most $\alpha \cdot c$ is at most $n^{2\alpha}$ and can be enumerated in time polynomial (in n) per cut.

We now show that Algorithm 1 is stable-on-average.

LEMMA 4.1. *The average sensitivity of Algorithm 1 is at most*

$$\beta(G) = \frac{n}{m} \cdot n^{(2+1/\varepsilon)/\mathsf{OPT}} \cdot ((2+7\varepsilon)\,\mathsf{OPT} + 2\varepsilon) + o(1).$$

As we have $\mathsf{OPT} \leq 2m/n$, the average sensitivity can be bounded by $n^{O(1/\varepsilon\mathsf{OPT})}$, and Theorem 4.1 follows by replacing ε with $\varepsilon/9$.

Proof. If $\mathsf{OPT} = 0$, then the claim trivially holds because the right hand size is infinity. Hence in what follows, we assume $\mathsf{OPT} \geq 1$.

Let $\mathcal{A}$ denote Algorithm 1. Consider an (inefficient) algorithm $\mathcal{A}'$ that on input G, outputs a cut $S \subseteq V$ (from among all the cuts in G) with probability proportional to $\exp(-\alpha \cdot \mathsf{cost}(G,S))$. For a graph $G = (V,E)$, let $\mathcal{A}(G)$ and $\mathcal{A}'(G)$ denote the output distribution of algorithms $\mathcal{A}$ and $\mathcal{A}'$ on input G, respectively. For $G = (V,E)$ and $S \subseteq V$, let $p_G(S)$ and $p'_G(S)$ be shorthands for the probabilities that S is output on input G by algorithms $\mathcal{A}$ and $\mathcal{A}'$, respectively.

We first bound the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}'(G)$ for a graph $G = (V,E)$. To this end, we define $Z = \sum_{S\subseteq V:\mathsf{cost}(G,S)\leq \mathsf{OPT}+b} \exp(-\alpha \cdot \mathsf{cost}(G,S))$, and $Z' = \sum_{S\subseteq V} \exp(-\alpha \cdot \mathsf{cost}(G,S))$ where $b = (1+7\varepsilon)\mathsf{OPT} + 2\varepsilon$. Note that $Z \leq Z'$ and the quantity $\frac{Z'-Z}{Z'}$ is the total probability mass assigned by algorithm $\mathcal{A}'$ to cuts $S \subseteq V$ such that $\mathsf{cost}(G,S) > \mathsf{OPT} + b$.

Now, we start with $\mathcal{A}'(G)$. For each $S \subseteq V$ such that $\mathsf{cost}(G,S) \leq \mathsf{OPT} + b$, keep at least $\frac{Z}{Z'} \cdot p'_G(S)$ mass with a cost of 0 and move a mass of at most $p'_G(S) - \frac{Z}{Z'} \cdot p'_G(S)$ at a cost of $n \cdot (p'_G(S) - \frac{Z}{Z'} \cdot p'_G(S))$. For each $S \subseteq V$ such that $\mathsf{cost}(G,S) > \mathsf{OPT} + b$, we move a mass of $p'_G(S)$ at a cost of $n \cdot p'_G(S)$. The total cost of moving masses is then equal to:

$$\begin{aligned}
&d_{\mathrm{EM}}\left(\mathcal{A}(G), \mathcal{A}'(G)\right) \\
&\leq n \cdot \sum_{S\subseteq V:\mathsf{cost}(G,S)\leq \mathsf{OPT}+b} p'_G(S)\left(1 - \frac{Z}{Z'}\right) \\
&+ n \cdot \sum_{S\subseteq V:\mathsf{cost}(G,S)> \mathsf{OPT}+b} p'_G(S) \\
&= \frac{n(Z'-Z)}{Z'}\left(\sum_{S\subseteq V:\mathsf{cost}(G,S)\leq \mathsf{OPT}+b} p'_G(S) + 1\right) \\
&\leq \frac{2n(Z'-Z)}{Z'}.
\end{aligned}$$

Let n_t stand for the number of cuts of cost at most $\mathsf{OPT} + t$ in G. By Karger's theorem (Theorem 4.2), we have that $n_t \leq n^{2+2t/\mathsf{OPT}}$. Then, we have

$$\begin{aligned}
\frac{Z'-Z}{Z'} &\leq \sum_{t>b} \exp(-\alpha t) \cdot (n_t - n_{t-1}) \\
&\leq (\exp(\alpha)-1) \cdot \sum_{t>b} \exp(-\alpha t) n_t \\
&\leq (\exp(\alpha)-1)n^2 \cdot \sum_{t>b} n^{2t/\mathsf{OPT}} \cdot \exp(-\alpha t) \\
&\leq (\exp(\alpha)-1)n^2 \cdot \sum_{t>b} n^{-t/\varepsilon\mathsf{OPT}} \\
&\leq (\exp(\alpha)-1)n^2 \cdot \frac{n^{-(b+1)/\varepsilon\mathsf{OPT}}}{1 - n^{-1/\varepsilon\mathsf{OPT}}} \\
&= \left(n^{(2+1/\varepsilon)/\mathsf{OPT}} - 1\right) \cdot \left(1 + \frac{1}{n^{1/\varepsilon\mathsf{OPT}} - 1}\right) \cdot \frac{n^2}{n^{(b+1)/\varepsilon\mathsf{OPT}}} \\
&\leq n^{(2+1/\varepsilon)/\mathsf{OPT}} \cdot \left(1 + \frac{\varepsilon n}{\log n}\right) \cdot \frac{n^2}{n^{(b+1)/\varepsilon\mathsf{OPT}}} \\
&= O\left(\frac{\varepsilon n^{3+(2+1/\varepsilon)/\mathsf{OPT}}}{n^{(b+1)/\varepsilon\mathsf{OPT}}}\right) = O\left(\frac{\varepsilon}{n^{4+1/\varepsilon}}\right).
\end{aligned}$$

The last inequality above follows from our choice of b. Therefore, the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}'(G)$ is $d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}'(G)\big) \leq O(\frac{\varepsilon}{n^{3+1/\varepsilon}})$.

In addition, we can bound the expected size of the cut output by $\mathcal{A}'$ on G as follows. The total probability mass assigned by algorithm $\mathcal{A}'$ to cuts of size larger than $\mathsf{OPT} + b$ is equal to $\frac{Z'-Z}{Z'} = O\left(\frac{\varepsilon}{n^{4+1/\varepsilon}}\right)$. Hence, the expected size of the cut output by $\mathcal{A}'$ on G is at most $\mathsf{OPT} + b + m \cdot O(\frac{\varepsilon}{n^{4+1/\varepsilon}}) = (2+7\varepsilon)\mathsf{OPT} + 2\varepsilon + O(\frac{\varepsilon m}{n^{4+1/\varepsilon}})$.

We now bound the earth mover's distance between $\mathcal{A}'(G)$ and $\mathcal{A}'(G-e)$ for an arbitrary edge $e \in E$. Let Z'_e denote the quantity $\sum_{S\subseteq V} \exp(-\alpha \cdot \mathsf{cost}(G-e,S))$. Since the cost of every cut in $G-e$ is at most the cost of the same cut in G, we have that $Z' \leq Z'_e$ and therefore, $p'_G(S) = \frac{\exp(-\alpha\cdot\mathsf{cost}(G,S))}{Z'} \leq \frac{\exp(\alpha\cdot\mathsf{cost}(G-e,S))}{Z'_e} \cdot \frac{Z'_e}{Z'} = p'_{G-e}(S) \cdot \frac{Z'_e}{Z'}$.

We transform $\mathcal{A}'(G)$ into $\mathcal{A}'(G-e)$ as follows. For each $S \subseteq V$, we leave a probability mass of at most $p'_{G-e}(S)$ at S with zero cost and move a mass of $\max\{0, p'_G(S) - p'_{G-e}(S)\}$ to any other point at a cost of at most $n \cdot \max\{0, p'_G(S) - p'_{G-e}(S)\} \leq n \cdot \left(\frac{Z'_e}{Z'} - 1\right) \cdot p'_G(S)$. Hence, $d_{\mathrm{EM}}\left(\mathcal{A}'(G), \mathcal{A}'(G-e)\right) \leq n \cdot \left(\frac{Z'_e}{Z'} - 1\right) \cdot \sum_{S\subseteq V} p'_G(S) = n \cdot \left(\frac{Z'_e}{Z'} - 1\right)$.

By the triangle inequality, the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}(G-e)$ can be bounded as

$$\begin{aligned}
&d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}(G-e)\big) \\
&\leq d_{\mathrm{EM}}\left(\mathcal{A}(G), \mathcal{A}'(G)\right) + d_{\mathrm{EM}}\left(\mathcal{A}'(G), \mathcal{A}'(G-e)\right)
\end{aligned}$$

$$+ d_{\mathrm{EM}}\left(\mathcal{A}'(G-e), \mathcal{A}(G-e)\right)$$
$$\leq n \cdot \left(\frac{Z'_e}{Z'} - 1\right) + O\left(\frac{2\varepsilon}{n^{2+1/\varepsilon}}\right).$$

Hence, the average sensitivity of $\mathcal{A}$ is bounded as:

$$\begin{aligned}\beta(G) &= \mathop{\mathbb{E}}_{e\sim E} d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}(G-e)\big) \\ &\leq O\left(\frac{2\varepsilon}{n^{3+1/\varepsilon}}\right) + n \cdot \mathop{\mathbb{E}}_{e\sim E}\left(\frac{Z'_e}{Z'} - 1\right) \\ &= O\left(\frac{2\varepsilon}{n^{3+1/\varepsilon}}\right) + \frac{n}{mZ'}\sum_{e\in E}(Z'_e - Z').\end{aligned}$$

Now,

$$\begin{aligned}&\sum_{e\in E}(Z'_e - Z') \\ &= \sum_{e\in E}\sum_{S\subseteq V : e \text{ crosses } S}\Big(\exp(-\alpha\cdot\mathsf{cost}(G-e,S)) \\ &\qquad\qquad - \exp(-\alpha\cdot\mathsf{cost}(G,S))\Big) \\ &= (\exp(\alpha)-1)\cdot\sum_{e\in E}\sum_{S\subseteq V : e \text{ crosses } S}\exp(-\alpha\cdot\mathsf{cost}(G,S)) \\ &= (\exp(\alpha)-1)\cdot\sum_{S\subseteq V}\mathsf{cost}(G,S)\cdot\frac{\exp(-\alpha\cdot\mathsf{cost}(G,S))}{Z'}.\end{aligned}$$

The summation above is equal to the expected size of the cut output by algorithm $\mathcal{A}'$ on input G. We argued that it is at most $(2+7\varepsilon)\mathsf{OPT} + 2\varepsilon + O(\frac{\varepsilon m}{n^{4+1/\varepsilon}})$. Hence, the average sensitivity of $\mathcal{A}$ is at most $\frac{n}{m}\cdot n^{(2+1/\varepsilon)/\mathsf{OPT}}\cdot((2+7\varepsilon)\,\mathsf{OPT} + 2\varepsilon) + O\left(\frac{\varepsilon n^{(2+1/\varepsilon)/\mathsf{OPT}+2}}{n^{3+1/\varepsilon}}\right) = \frac{n}{m}\cdot n^{(2+1/\varepsilon)/\mathsf{OPT}}\cdot((2+7\varepsilon)\,\mathsf{OPT} + 2\varepsilon) + o(1)$, as $\mathsf{OPT} \geq 1$. ∎

5 Maximum Matching

A vertex-disjoint set of edges is called a *matching*. In the maximum matching problem, given a graph, we want to find a matching of the maximum size. In this section, we describe several algorithms with low average sensitivity that approximate the maximum matching in a graph.

5.1 Lexicographically smallest matching In this section, we describe an algorithm that computes a maximum matching in a graph with average sensitivity at most OPT^2/m and prove Theorem 5.1, where OPT is the maximum size of a matching.

First, we define some ordering among vertex pairs. Then, we can naturally define the lexicographical order among matchings by considering a matching as a sorted sequence of vertex pairs. Then, our algorithm simply outputs the lexicographically smallest matching. Note that this can be done in polynomial time using Edmonds' algorithm [9].

Algorithm 2: Randomized Greedy Algorithm

Input: undirected unweighted graph $G = (V, E)$

1 Sample a uniformly random ordering π of edges in E;
2 Set $M \leftarrow \emptyset$;
3 Consider edges one by one according to π and add an edge (u, v) to M only if both u and v are unmatched in M;
4 **return** M.

Theorem 5.1. *Let $\mathcal{A}$ be the algorithm that outputs the lexicographically smallest maximum matching. Then, the average sensitivity of $\mathcal{A}$ is at most OPT^2/m, where OPT is the maximum size of a matching.*

Proof. For a graph $G = (V, E)$, let $M(G) \subseteq E$ be its lexicographically smallest maximum matching. As long as $e \notin M$, we have $M(G) = M(G-e)$. Hence, the average sensitivity of the algorithm is at most $\frac{\mathsf{OPT}}{m}\cdot\mathsf{OPT} + \left(1 - \frac{\mathsf{OPT}}{m}\right)\cdot 0 = \frac{\mathsf{OPT}^2}{m}$. ∎

Remark 2. *Consider the path graph $P_n = (\{1,\dots,n\}, E)$, where $E = \{(i, i+1) : i \in [n-1]\}$. The average sensitivity of the above algorithm on P_n is $\Omega(\frac{\mathsf{OPT}^2}{m})$. Hence the above analysis of the average sensitivity is tight.*

5.2 Greedy matching algorithm In this section, we analyze the average sensitivity of the randomized greedy algorithm (Algorithm 2) that outputs a maximal matching. It is evident that Algorithm 2 runs in polynomial time and that the matching it outputs has size at least $\frac{1}{2}$ the size of a maximum matching in the input graph.

Theorem 5.2. *Algorithm 2 is a $\frac{1}{2}$-approximation algorithm for the maximum matching problem and has average sensitivity at most 1.*

Proof. For a permutation π of edges in E, let $M_\pi(G)$ denote the matching obtained by running Algorithm 2 on a graph G. Using [6, Theorem 1], we get that for every $e \in E(G)$, it holds that $\mathbb{E}_\pi\left[\mathsf{Ham}(M_\pi(G), M_\pi(G-e))\right] \leq 1$. This implies that the average sensitivity of Algorithm 2 is at most 1. ∎

A vertex set $S \subseteq V$ in a graph $G = (V, E)$ is called a *vertex cover* if every edge in E is incident to a vertex in S. In the *minimum vertex cover problem*, given a graph G, we want to compute a vertex cover of the minimum size. It is well known that, for any maximal matching M, the vertex set consisting of all endpoints of edges

in M is a 2-approximate vertex cover. The following theorem is immediate from Theorem 5.2.

THEOREM 5.3. *There exists a 2-approximation algorithm for the minimum vertex cover problem with average sensitivity at most 2.*

5.3 Matching algorithm based on augmenting paths In this section, we describe a $(1-\varepsilon)$-approximation algorithm for the maximum matching problem with average sensitivity $\tilde{O}\left(\mathsf{OPT}^{\frac{c}{c+1}}/\varepsilon^{\frac{3c}{c+1}}\right)$ for $c = O(1/\varepsilon^2)$ in Theorem 5.7. The basic building block is a $(1-\varepsilon)$-approximation algorithm (Algorithm 3) for maximum matching that is based on iteratively augmenting a matching with greedily chosen augmenting paths of increasing lengths. In Theorem 5.4, we show that the average sensitivity of this algorithm is $\Delta^{O(1/\varepsilon^2)}$, where Δ is the maximum degree of the input graph. We obtain Theorem 5.4 by applying Theorem 1.5 to a result by Yoshida et al. [51].

We then apply Theorem 5.5 to Theorem 5.4 in order to get rid of the dependence of the average sensitivity on the maximum degree and obtain Theorem 5.6. We then combine (using Theorem 1.4, the parallel composition theorem) the algorithm guaranteed by Theorem 5.6 with the algorithm guaranteed by Theorem 5.1 to obtain Theorem 5.7.

5.3.1 Greedy matching algorithm based on augmenting paths In this section, we present an approximation algorithm that starts with an empty matching and then iteratively improves its size with augmenting paths of increasing lengths. We show that the average sensitivity of this algorithm can be bounded using Theorem 1.5.

Algorithm 3: GREEDY AUGMENTING PATHS ALGORITHM

Input: undirected unweighted graph $G = (V, E)$, parameter $\varepsilon \in (0, 1)$

$M_0 \leftarrow \emptyset$;
for $i \in \{1, 2, \ldots \lceil \frac{1}{\varepsilon} - 1 \rceil\}$ **do**
  Let A_i denote the set of augmenting paths of length $2i-1$ for the matching M_{i-1};
  Let A'_i denote a maximal set of disjoint paths from A_i, where A'_i is made from a random ordering of A_i;
  $M_i \leftarrow M_{i-1} \triangle A'_i$.
return $M_{\lceil \frac{1}{\varepsilon} - 1 \rceil}$.

THEOREM 5.4. *Algorithm 3 with parameter $\varepsilon > 0$ has approximation ratio $1-\varepsilon$ and average sensitivity $\Delta^{O(1/\varepsilon^2)}$, where Δ is the maximum degree of the input graph.*

Proof. For all $k \geq 0$, it is known that $|M_k| \geq \frac{k}{k+1} \cdot |M^*|$ [13], where M^* denotes a maximum matching in G. Hence, the matching $M_{\lceil \frac{1}{\varepsilon} - 1 \rceil}$ is a $(1-\varepsilon)$-approximation to M^*.

Yoshida et al. [51, Theorem 3.7] show that for all $k \geq 0$, determining whether a uniformly random edge $e \sim E$ belongs to M_k can be done by querying at most $\Delta^{O(k^2)}$ edges in expectation, where Δ is the maximum degree of G. Applying Theorem 1.5 to this result, we can see that the average sensitivity of Algorithm 3 with parameter $\varepsilon > 0$ and input G is $\Delta^{O(1/\varepsilon^2)}$, where Δ is the maximum degree of G. ∎

5.3.2 Stable-on-average thresholding transformation In this section, we show a transformation from matching algorithms whose average sensitivity is a function of the maximum degree to matching algorithms whose average sensitivity does not depend on the maximum degree. This is done by adding to the algorithm, a preprocessing step that removes vertices from the input graph, where the removed vertices have degree at least an appropriate random threshold. Such a transformation helps us to design stable-on-average algorithms for graphs with unbounded degree. Let $\mathsf{Lap}(\mu, \phi)$ denote the Laplace distribution with a location parameter μ and a scale parameter ϕ.

THEOREM 5.5. *Let $\mathcal{A}'$ be a randomized algorithm for the maximum matching problem such that the size of the matching output by $\mathcal{A}'$ on a graph G is always at least $a \cdot \mathsf{OPT}$ for some $a \geq 0$. In addition, assume that there exists a solution oracle $\mathcal{O}$ (see Definition 1) for $\mathcal{A}'$ that makes at most $q(\Delta)$ queries to G in expectation, where Δ is the maximum degree of G, and the expectation is taken over the random coins of $\mathcal{A}'$ and edges $e \in E$. Let $\delta > 0$ and τ be a non-negative function on graphs. Then, there exists an algorithm $\mathcal{A}$ for the maximum matching problem with average sensitivity $\beta(G) \leq O\left(\frac{K_G}{\delta(\tau(G) - K_G)} + \exp\left(-\frac{1}{\delta}\right)\right) \cdot \mathsf{OPT} + \mathbb{E}_L\left[(2L-2)^2 q(L)\right]$, where L is a random variable distributed as $\mathsf{Lap}(\tau(G), \delta\tau(G))$ and $K_G = \max_{e \in E(G)} |\tau(G) - \tau(G - e)|$. Moreover, the expected size of the matching output by $\mathcal{A}$ is at least*

$$a \cdot \mathsf{OPT} - \frac{am}{(1 - \delta \ln(\mathsf{OPT}/2)) \cdot \tau(G)} - a.$$

The following fact will be useful in the proof of Theorem 5.5.

PROPOSITION 5.1. *Let L be a random variable distributed as $\mathsf{Lap}(\mu, \phi)$. Then, $\Pr[L < (1-\varepsilon)\mu] \leq$*

$\exp(-\varepsilon\mu/\phi)/2$. *Similarly,* $\Pr[L > (1+\varepsilon)\mu] \leq \exp(-\varepsilon\mu/\phi)/2$.

Proof. [Proof of Theorem 5.5] The algorithm $\mathcal{A}$ is given below.

Algorithm $\mathcal{A}$: On input $G = (V, E)$,

1. Sample a random variable L according to the distribution $\mathsf{Lap}(\tau(G), \delta\tau(G))$.
2. Let $[G]_L$ be the graph obtained after removing from G all vertices of degree at least L.
3. Run $\mathcal{A}'$ on $[G]_L$.

We first bound the average sensitivity of $\mathcal{A}$. We can think of $\mathcal{A}$ as being sequentially composed of two algorithms, where the first algorithm takes in a graph $G = (V, E)$ and outputs a number $L \sim \mathsf{Lap}(\tau(G), \delta\tau(G))$. The second algorithm takes both L and G and runs $\mathcal{A}'$ on $[G]_L$.
Let L_e for $e \in E$ denote a Laplace random variable distributed as $\mathsf{Lap}(\tau(G-e), \delta\tau(G-e))$. Using Theorem 1.2, we get that the average sensitivity of $\mathcal{A}$ is bounded by $\mathsf{OPT} \cdot \mathbb{E}_{e\sim E}[d_{\mathrm{TV}}(L, L_e)] + \mathbb{E}_L\left[\mathbb{E}_{e\sim E}\left[d_{\mathrm{EM}}(\mathcal{A}'([G]_L), \mathcal{A}'([G-e]_L))\right]\right]$.

Claim 1. *For* $x \in \mathbb{R}$, $\mathbb{E}_{e\sim E}\left[d_{\mathrm{EM}}\big(\mathcal{A}'([G]_x), \mathcal{A}'([G-e]_x)\big)\right] \leq (2x-2)^2 q(x)$.

Proof. Fix $x \in \mathbb{R}$. In order to bound the term $\mathbb{E}_{e\sim E}\left[d_{\mathrm{EM}}\big(\mathcal{A}'([G]_x), \mathcal{A}'([G-e]_x)\big)\right]$, consider the following algorithm $\mathcal{A}'_x$. On input $G = (V, E)$, the algorithm $\mathcal{A}'_x$ first removes every vertex of degree at least x from G and then runs $\mathcal{A}'$ on the resulting graph. Hence, the quantity $\mathbb{E}_{e\sim E}\left[d_{\mathrm{EM}}\big(\mathcal{A}'([G]_x), \mathcal{A}'([G-e]_x)\big)\right]$ denotes the average sensitivity of $\mathcal{A}'_x$.

In order to bound the average sensitivity of $\mathcal{A}'_x$, construct a solution oracle $\mathcal{O}_x$ for $\mathcal{A}'_x$ as follows. The oracle $\mathcal{O}_x$, when given access to a graph $G = (V, E)$ and input e sampled uniformly at random from E, does the following. It first checks whether at least one of the endpoints of e has degree at least x. If so, it returns that e does not belong to the solution obtained by running $\mathcal{A}'_x$ on G. Otherwise, it runs $\mathcal{O}$ with access to $[G]_x$ and e as input and outputs the answer of $\mathcal{O}$.

We can analyze the query complexity of $\mathcal{O}_x$ as follows. Call an edge $e \in E$ *alive* if both the endpoints of e have degree less than x. Otherwise, e is *dead.*

The oracle $\mathcal{O}_x$ can check whether an edge $e = (u, v)$ is alive or not by querying at most $2x-2$ edges incident to e. In particular $\mathcal{O}_x$ examines the neighbors of u and v one by one, and, as soon $\mathcal{O}_x$ encounters $x-1$ distinct neighbors (excluding u or v themselves) for either u or v, $\mathcal{O}_x$ can declare e to be a dead edge.

If the edge $e \in E$ input to $\mathcal{O}_x$ is a dead edge, therefore, $\mathcal{O}_x$ queries at most $2x-2$ edges and returns that e cannot be part of a solution to running $\mathcal{A}'_x$ on G.

If the input edge $e \in E$ is alive, then we know that it is a uniformly random alive edge. By the guarantee on $\mathcal{O}$, we then know that $\mathcal{O}$ makes at most $q(x)$ queries to the alive edges in expectation over the randomness of $\mathcal{A}'$ and the choice of the input alive edge, since the maximum degree of $[G]_x$ is at most x. In order for the oracle $\mathcal{O}_x$ to simulate oracle access to $[G]_x$ for the purpose of answering queries made by oracle $\mathcal{O}$, for each alive edge e queried by $\mathcal{O}$, the oracle $\mathcal{O}_x$ has to query each edge incident to e in G and determine which among these are alive. Since e is alive, both endpoints of e have degrees less than x. Hence, $\mathcal{O}_x$ need only check whether at most $2x-2$ edges incident to e are alive or not. This can be done by querying $(2x-2)^2$ edges in E in total.

Combining all of the above, the expected query complexity of $\mathcal{O}_x$ is at most $(2x-2)^2 q(x)$, where the expectation is taken over the edges of $e \in E$ and the randomness in $\mathcal{A}_x$.

Therefore, by Theorem 1.5, we get that the average sensitivity of algorithm $\mathcal{A}_x$ is bounded by $(2x-2)^2 q(x)$. ∎

We now bound the quantity $\mathbb{E}_{e\sim E}[d_{\mathrm{TV}}(L, L_e)]$.

Claim 2. *For any* $e \in E$, *we have*

$$d_{\mathrm{TV}}(L, L_e) \leq O\left(\frac{K}{\delta(\tau - K)} + \exp\left(-\frac{1}{\delta}\right)\right).$$

Proof. Let $f_L, f_{L_e} \colon \mathbb{R} \to \mathbb{R}$ be the probability density functions of the Laplace random variables L and L_e, respectively. Let $\tau = \tau(G)$, $\tau_e = \tau(G_e)$, and $K = K_G$. Then

$$\begin{aligned}\frac{f_L(x)}{f_{L_e}(x)} &= \frac{\frac{1}{2\delta\tau}\exp\left(-\frac{|x-\tau|}{\delta\tau}\right)}{\frac{1}{2\delta\tau_e}\exp\left(-\frac{|x-\tau_e|}{\delta\tau_e}\right)} \\ &= \frac{\tau_e}{\tau}\exp\left(\frac{|x-\tau_e|}{\delta\tau_e} - \frac{|x-\tau|}{\delta\tau}\right) \\ &= \left(1 - \frac{\tau-\tau_e}{\tau}\right)\exp\left(\frac{\tau|x-\tau_e| - \tau_e|x-\tau|}{\delta\tau\tau_e}\right).\end{aligned}$$

A direct calculation shows that for $0 \leq x \leq 2\max\{\tau, \tau_e\}$, we have

$$\begin{aligned}&\left(1 - \frac{K}{\tau}\right)\exp\left(\frac{-2K}{\delta(\tau-K)}\right) \\ &\leq \frac{f_L(x)}{f_{L_e}(x)} \leq \left(1 + \frac{K}{\tau}\right)\exp\left(\frac{2K}{\delta(\tau-K)}\right).\end{aligned}$$

This implies that for all $S \subseteq [0, 2\max\{\tau, \tau_e\}]$,

$$\left(1 - \frac{K}{\tau}\right)\exp\left(\frac{-2K}{\delta(\tau-K)}\right) - 1$$

$$\leq \Pr[L \in S] - \Pr[L_e \in S] \leq \left(1 + \frac{K}{\tau}\right) \exp\left(\frac{2K}{\delta(\tau - K)}\right) - 1.$$

By Proposition 5.1, the probability that L (and L_e as well) falls in the range $[-\infty, 0] \cup [2\max\{\tau, \tau_e\}, \infty]$ is bounded by $\exp(-1/\delta)$. Hence, total variation distance between L and L_e is $O\left(\frac{K}{\delta(\tau-K)} + \exp\left(-\frac{1}{\delta}\right)\right)$. ∎

Therefore, the average sensitivity of $\mathcal{A}$ is bounded as
$\mathbb{E}_{e\sim E}\, d_{\mathrm{EM}}\big(\mathcal{A}(G), \mathcal{A}(G-e)\big)$
$\leq O\left(\frac{K}{\delta(\tau-K)} + \exp\left(-\frac{1}{\delta}\right)\right) \cdot \mathsf{OPT} + \mathbb{E}_L\left[(2x-2)^2 q(x)\right]$.

We now bound the approximation guarantee of $\mathcal{A}$. By Proposition 5.1,

$$\Pr\left[L < \left(1 - \delta \ln\left(\frac{\mathsf{OPT}}{2}\right)\right) \cdot \tau(G)\right] \leq \frac{1}{\mathsf{OPT}}.$$

Therefore, with probability at least $1 - 1/\mathsf{OPT}$, only those vertices with degree at least $(1 - \delta\ln(\mathsf{OPT}/2)) \cdot \tau(G)$ are removed from G. The number of such vertices is at most $\frac{m}{(1-\delta\ln(\mathsf{OPT}/2))\cdot\tau(G)}$. Therefore, with probability at least $1 - 1/\mathsf{OPT}$, the size of a maximum matching in the resulting graph is at most $\frac{m}{(1-\delta\ln(\mathsf{OPT}/2))\cdot\tau(G)}$ smaller than that of G. With probability at most $1/\mathsf{OPT}$, the size of a maximum matching in the resulting instance could be smaller by an additive term of at most OPT. Hence, the expected size of a maximum matching in the new instance is at least

$$\mathsf{OPT} - \frac{m}{(1 - \delta\ln(\mathsf{OPT}(G)/2)) \cdot \tau(G)} - 1.$$

The statement on approximation guarantee follows. ∎

5.3.3 Average sensitivity of the greedy augmenting paths algorithm with thresholding

THEOREM 5.6. *Let $\varepsilon \in (0,1)$ be a parameter. There exists an algorithm with approximation ratio $1-\varepsilon$ and average sensitivity $O\left(\frac{\varepsilon}{1-\varepsilon}\log n\right) + \left(\frac{m}{\varepsilon^3\mathsf{OPT}}\right)^{O(1/\varepsilon^2)}$.*

Proof. The algorithm guaranteed by the theorem statement is as follows.

Algorithm $\mathcal{A}_\varepsilon$: On input $G = (V, E)$,

1. Compute OPT.
2. If $\mathsf{OPT} \leq \frac{2}{\varepsilon} + 1$ or $m \leq \frac{1}{3\varepsilon}$, then output an arbitrary maximum matching.
3. Otherwise, run the algorithm obtained by applying Theorem 5.5 with the setting $\tau := \tau(G) = \frac{m}{\varepsilon'\mathsf{OPT}}$ and $\delta := \frac{1}{2\ln n}$ to Algorithm 3 run with parameter ε', where $\varepsilon' = \frac{\varepsilon}{3} - \frac{1}{3\mathsf{OPT}}$.

Approximation guarantee: If $\mathsf{OPT} \leq \frac{1}{\varepsilon} + 1$ or $m \leq \frac{1}{2\varepsilon}$, the approximation guarantee is clear. Otherwise, since Algorithm 3 outputs a maximal matching whose size is always at least $(1-\varepsilon') \cdot \mathsf{OPT}$, the size of the matching output by $\mathcal{A}_\varepsilon$ is at least $(1-\varepsilon') \cdot \mathsf{OPT} - \frac{\varepsilon'\cdot(1-\varepsilon')\cdot\mathsf{OPT}}{1 - \frac{\ln(\mathsf{OPT}/2)}{2\ln n}} - (1-\varepsilon')$, which is at least $(1-\varepsilon)\cdot\mathsf{OPT}$ by the setting of ε' and the fact that $\frac{\ln(\mathsf{OPT}/2)}{2\ln n} \leq \frac{1}{2}$.

Average sensitivity: If $\mathsf{OPT} \leq \frac{2}{\varepsilon} + 1$ or $m \leq \frac{1}{3\varepsilon}$, the average sensitivity of $\mathcal{A}_\varepsilon$ is bounded by $O(\frac{1}{\varepsilon})$, since the size of maximum matching in G is small and it can decrease only by at most 1 by the removal of an edge.

We now analyze the average sensitivity of $\mathcal{A}_\varepsilon$ for the case that $\mathsf{OPT} > \frac{2}{\varepsilon} + 1$ and $m > \frac{1}{3\varepsilon}$. Let $c = O(1/\varepsilon^2)$. The average sensitivity of the algorithm resulting from applying Theorem 5.5 to Algorithm 3 is bounded as:

$$O\left(\frac{K_G}{\delta(\tau - K_G)} + \exp\left(-\frac{1}{\delta}\right)\right) \cdot \mathsf{OPT} + \int_0^\infty (2x-2)^2 \cdot x^c \cdot \frac{1}{2\delta\tau} \cdot \exp\left(-\frac{|x-\tau|}{\delta\tau}\right) \mathrm{d}x. \tag{5.5}$$

To obtain the above expression, we used the fact (from [51, Theorem 3.7]) that $q(x) \leq x^c$ when $x > 0$ and $q(x) = 0$ otherwise.

The second term of (5.5) can be bounded as $\left(\frac{m}{\varepsilon^3\mathsf{OPT}}\right)^{O(1/\varepsilon^2)}$.

In order to bound the first term of (5.5), note that

$$\begin{aligned} K_G &= \max_{e\in E} |\tau(G) - \tau(G-e)| \\ &= 3\max_{e\in E}\left|\frac{m}{\varepsilon\mathsf{OPT}(G) - 1} - \frac{m-1}{\varepsilon\mathsf{OPT}(G-e) - 1}\right| \\ &\leq \frac{3}{\varepsilon\mathsf{OPT}(G) - 1}\max\left\{1, \frac{\varepsilon(m - \mathsf{OPT}(G)) + 1}{\varepsilon(\mathsf{OPT}(G) - 1) - 1}\right\}. \end{aligned}$$

The proof of inequality above (that we omit) uses the fact that for numbers $u, v, w \geq 0$ such that $u \leq v$ and $w(v-1) - 1 \geq 0$, we have that $\frac{u}{wv-1} \leq \frac{u-1}{w(v-1)-1}$.

Since $\frac{\varepsilon(m-\mathsf{OPT})+1}{\varepsilon(\mathsf{OPT}-1)-1}$ is a nonincreasing function of OPT and $\mathsf{OPT} > \frac{2}{\varepsilon} + 1$, we have that $\frac{\varepsilon(m-\mathsf{OPT})+1}{\varepsilon(\mathsf{OPT}-1)-1} < \varepsilon m$. Hence, $K_G < \frac{3}{\varepsilon\mathsf{OPT}-1}\max\{1, \varepsilon m\} = \frac{9\varepsilon m}{\varepsilon\mathsf{OPT}-1}$, since $m > \frac{1}{3\varepsilon}$ and therefore, we have that $\tau - K_G \geq \tau(1 - 9\varepsilon)$. Hence, the first term of (5.5) can be upper bounded by

$$\begin{aligned} &O\left(\frac{K_G}{\delta\tau(1-9\varepsilon)} + \exp\left(-\frac{1}{\delta}\right)\right) \cdot \mathsf{OPT} \\ &= O\left(\frac{9\varepsilon m}{\varepsilon\mathsf{OPT}-1} \cdot \frac{1}{1-9\varepsilon} \cdot \frac{2\ln n}{\frac{m}{\varepsilon\mathsf{OPT}-1}} + \frac{\mathsf{OPT}}{n^2}\right) \\ &= O\left(\frac{\varepsilon}{1-\varepsilon} \cdot \log n\right). \end{aligned}$$

Hence, the average sensitivity of the algorithm obtained can be bounded by $\beta(G) = \max\left\{O\left(\frac{1}{\varepsilon}\right), O\left(\frac{\varepsilon}{1-\varepsilon}\cdot\log n\right) + \left(\frac{m}{\varepsilon^3\mathsf{OPT}}\right)^{O(1/\varepsilon^2)}\right\}$
$= O\left(\frac{\varepsilon}{1-\varepsilon}\log n\right) + \left(\frac{m}{\varepsilon^3\mathsf{OPT}}\right)^{O(1/\varepsilon^2)}$. ∎

5.3.4 Average sensitivity of a combined matching algorithm In this section, we combine the algorithms guaranteed by Theorems 5.1 and 5.6 in order to get a matching algorithm with improved sensitivity.

THEOREM 5.7. *Let $\varepsilon \in (0,1)$ be a parameter. There exists an algorithm with approximation ratio $1-\varepsilon$ and average sensitivity*

$$\mathrm{OPT}(G)^{\frac{c}{c+1}} \cdot O\left(\left(\frac{\varepsilon}{1-\varepsilon}\cdot \log n\right)^{\frac{1}{c+1}} + \frac{1}{\varepsilon^{\frac{3c}{c+1}}}\right)$$

for $c = O(1/\varepsilon^2)$.

Proof. Let $c = O(1/\varepsilon^2)$. The algorithm guaranteed by the theorem is given as Algorithm 4. The bounds on approximation guarantee and average sensitivity are both straightforward when $\mathsf{OPT} < 2c$ or $m < 2c$.

Algorithm 4: COMBINED ALGORITHM TO $(1-\varepsilon)$-APPROXIMATE MAXIMUM MATCHING

Input: undirected unweighted graph $G = (V, E)$

Compute OPT.;
if $\mathsf{OPT} < 2c$ *or* $m < 2c$ **then**
  return an arbitrary maximum matching in G.
else
  Let $f(G) \leftarrow \frac{\mathsf{OPT}^2}{m}$ and $g(G) \leftarrow \frac{\varepsilon}{(1-\varepsilon)} \cdot \log n + \left(\frac{m}{\varepsilon^3 \mathsf{OPT}}\right)^c$;
  Run the algorithm given by Theorem 5.1 with probability $\frac{g(G)}{f(G)+g(G)}$ and run the algorithm given by Theorem 5.6 with the remaining probability.

The approximation guarantee in the case when $\mathsf{OPT} \geq 2c$ and $m \geq 2c$ is also straightforward since Algorithm 4 is simply a distribution over algorithms guaranteed by Theorem 5.1 and Theorem 5.6.

We now bound the average sensitivity of Algorithm 4 when $\mathsf{OPT} \geq 2c$ and $m \geq 2c$. Let $\rho(G)$ denote the probability $\frac{g(G)}{f(G)+g(G)}$. By Theorem 1.4, the average sensitivity is at most

$$\frac{O(f(G))\cdot g(G) + O(g(G))\cdot f(G)}{f(G)+g(G)} + 2\mathsf{OPT} \cdot \mathop{\mathbb{E}}_{e\sim E}\left[|\rho(G) - \rho(G-e)|\right]. \tag{5.6}$$

We first bound the quantity $\mathbb{E}_{e\sim E}\left[|\rho(G) - \rho(G-e)|\right]$.

CLAIM 3. *For every graph $G = (V, E)$ such that $\mathsf{OPT} \geq c+1$, and for every $e \in E$,*

$$\left(1 - \frac{c}{m}\right)\cdot g(G) \leq g(G-e) \leq \left(1 + \frac{c}{\mathsf{OPT}-c}\right)\cdot g(G).$$

CLAIM 4. *For every graph $G = (V, E)$ and every $e \in E$,*

$$f(G)\cdot\left(1 - \frac{2}{\mathsf{OPT}}\right) \leq f(G-e) \leq f(G)\cdot\left(1 + \frac{1}{m-1}\right).$$

CLAIM 5. *For every graph $G = (V, E)$ such that $\mathsf{OPT} \geq 2c$ and $m \geq 2c$, and for every $e \in E$,*

$$\rho(G)\cdot\left(1 - \frac{2c}{\mathsf{OPT}-c}\right) \leq \rho(G-e) \leq \rho(G)\cdot\left(1 + \frac{5c}{\mathsf{OPT}-c}\right).$$

Thus, for all $e \in E$, we have that $|\rho(G) - \rho(G-e)| \leq \max\left\{\frac{2c}{\mathsf{OPT}-c}, \frac{5c}{\mathsf{OPT}-c}\right\}\cdot\rho(G) = \frac{5c\rho(G)}{\mathsf{OPT}-c}$. Hence, $\mathbb{E}_{e\sim E}[|\rho(G) - \rho(G-e)|] \leq \frac{5c\rho(G)}{\mathsf{OPT}-c}$.

Therefore, the average sensitivity of Algorithm 4 is at most

$$\begin{aligned}
&\frac{O(f(G))\cdot g(G) + O(g(G))\cdot f(G)}{f(G)+g(G)} \\
&+ 2\mathsf{OPT}\cdot\mathop{\mathbb{E}}_{e\sim E}[|\rho(G) - \rho(G-e)|] \\
&= O\left(\frac{f(G)^{c/(c+1)}g(G)^{1/(c+1)}}{\frac{g(G)^{1/(c+1)}}{f(G)^{1/(c+1)}} + \frac{f(G)^{c/(c+1)}}{g(G)^{c/(c+1)}}}\right) + O\left(\frac{\mathsf{OPT}\, c\rho(G)}{\mathsf{OPT}}\right) \\
&= O\left(f(G)^{c/(c+1)}g(G)^{1/(c+1)}\right) + O(1/\varepsilon^2).
\end{aligned}$$

To obtain the first term of the expression resulting from the first equality, we divide both the numerator and denominator by $f(G)^{\frac{1}{c+1}}\cdot g(G)^{\frac{c}{c+1}}$. The second term of the second equality above follows since $\frac{\mathsf{OPT}}{\mathsf{OPT}-c} \leq 2$ as $\mathsf{OPT} \geq 2c$.

Using further calculations on the resulting expression, we can bound the average sensitivity as $O\left(\mathsf{OPT}^{c/(c+1)}\left(\left(\frac{\varepsilon}{1-\varepsilon}\right)^{1/(c+1)}\log^{1/(c+1)} n + \frac{1}{\varepsilon^{3c/(c+1)}}\right)\right)$. ∎

6 2-Coloring

In the 2-*coloring problem*, given a bipartite graph $G = (V, E)$, we are to output a (proper) 2-coloring on G, that is, an assignment $f : V \to \{0, 1\}$ such that $f(u) \neq f(v)$ for every edge $(u, v) \in E$. Clearly this problem can be solved in linear time. In this section, however, we show that there is no stable-on-average algorithm for the 2-coloring problem.

THEOREM 6.1. *Any (randomized) algorithm for the 2-coloring problem has average sensitivity $\Omega(n)$.*

Proof. Suppose that there is a (randomized) algorithm $\mathcal{A}$ whose average sensitivity is at most βn for $\beta < 1/256$. In what follows, we assume that n, that is, the number of vertices in the input graph, is a multiple of 16.

Let $\mathcal{P}_n$ be the family of all possible paths on n vertices, and let $\mathcal{Q}_n$ be the family of all possible graphs on n vertices consisting of two paths. Note that $|\mathcal{P}_n| = n!/2$ and $|\mathcal{Q}_n| = (n-1)n!/4$. Consider a bipartite graph $H = (\mathcal{P}_n, \mathcal{Q}_n; E)$, where a pair (P, Q) is in E if and only if Q can be obtained by removing an edge in P. Note that each $P \in \mathcal{P}_n$ has $n-1$ neighbors in H and each $Q \in \mathcal{Q}_n$ has four neighbors in H.

We say that an edge $(P, Q) \in E$ is *intimate* if $d_{\mathrm{EM}}\big(\mathcal{A}(P), \mathcal{A}(Q)\big) \leq 8\beta n$. We observe that for every $P \in \mathcal{P}_n$, at least a 7/8-fraction of the edges incident to P are intimate; otherwise

$$\mathop{\mathbb{E}}_{e \sim E(P)} \left[d_{\mathrm{EM}}\big(\mathcal{A}(P), \mathcal{A}(P-e)\big) \right] > \frac{1}{8} \cdot 8\beta n = \beta n,$$

which is a contradiction, where $E(P)$ denotes the set of edges in P.

We say that a graph $Q \in \mathcal{Q}_n$ is *heavy* if both components of Q have at least $n/16$ vertices, and say that an edge $(P, Q) \in E$ is *heavy* if Q is heavy. We observe that for every $P \in \mathcal{P}_n$, at least a 7/8-fraction of the edges incident to P are heavy.

We say that an edge $(P, Q) \in E$ is *good* if it is intimate and heavy. Observe that for every $P \in \mathcal{P}_n$, by the union bound, at least a 3/4-fraction of the edges incident to P are good. In particular, this means that the fraction of good edges in H is at least 3/4. Hence, there exists $Q^* \in \mathcal{Q}_n$ that has at least three good incident edges; otherwise the fraction of good edges in H is at most $2/4 = 1/2$, which is a contradiction.

Let $f_1, \ldots, f_4$ be the four 2-colorings of Q^*. As Q^* has three good incident edges, without loss of generality, there are adjacent paths $P_1, P_2 \in \mathcal{P}_n$ such that both (P_1, Q^*) and (P_2, Q^*) are good, and there is no assignment that is a 2-coloring for both P_1 and P_2. Without loss of generality, we assume that f_1, f_2 are 2-colorings of P_1, and f_3, f_4 are 2-colorings of P_2. Note that $d_{\mathrm{Ham}}(f_i, f_j) \geq n/16$ for $i \neq j$ because Q is heavy. Let $q_i = \Pr[\mathcal{A}(Q^*) = f_i]$ for $i \in [4]$. As the edge (P_1, Q^*) is intimate, we have

$$\begin{aligned}
&8\beta n \geq d_{\mathrm{EM}}\big(\mathcal{A}(P_1), \mathcal{A}(Q^*)\big) \\
&\geq \frac{n}{16}\big(\big|\Pr[\mathcal{A}(P_1) = f_1] - q_1\big| \\
&+ \big|\Pr[\mathcal{A}(P_1) = f_2] - q_2\big| + q_3 + q_4\big) \\
&= \frac{n}{16}\big(\big|\Pr[\mathcal{A}(P_1) = f_1] - q_1\big| \\
&+ \big|\Pr[\mathcal{A}(P_1) = f_2] - q_2\big| + 1 - q_1 - q_2\big)
\end{aligned}$$

and hence we must have $q_1 + q_2 \geq 1 - 128\beta$. Considering $d_{\mathrm{EM}}\big(\mathcal{A}(P_2), \mathcal{A}(Q^*)\big)$, we also have $q_3 + q_4 \geq 1 - 128\beta$. However, $1 = q_1 + q_2 + q_3 + q_4 \geq (1 - 128\beta) + (1 - 128\beta) = 2 - 256\beta > 1$ as $\beta < 1/256$, which is a contradiction. ∎

7 Sublinearity implies low average sensitivity

In this section, we prove Theorem 1.5, which show that the existence of a sublinear-time solution oracle (Definition 1) for an algorithm $\mathcal{A}$ implies that the average sensitivity of $\mathcal{A}$ is bounded by the query complexity of that oracle. The proofs of other general results on average sensitivity can be found in the full version [50].

THEOREM 7.1 (SUBLINEARITY IMPLIES LOW AVERAGE SENSITIVITY) *Consider a randomized algorithm $\mathcal{A} : \mathcal{G} \to \mathcal{S}$ for a graph problem, where each solution output by $\mathcal{A}$ is a subset of the set of edges in the input graph. Assume that there exists a solution oracle $\mathcal{O}$ for $\mathcal{A}$ such that $\mathcal{O}$ makes at most $q(G)$ queries to G in expectation, where this expectation is taken over the random coins of $\mathcal{O}$ and over input edges $e \in E$. Then, $\mathcal{A}$ has average sensitivity at most $q(G)$. Moreover, given the promise that the input graphs satisfy $|E| \geq |V|$, the statement applies also to algorithms for which each solution is a subset of the set of vertices in the input graph.*

Proof. We prove the theorem for the case that solutions output by $\mathcal{A}$ are subsets of edges of the input graph. It can be easily modified to work for the case that the solutions output by $\mathcal{A}$ are subsets of vertices of the input graph in which case, we will use the technical condition that $n \leq m$.

Without loss of generality, assume that $\mathcal{A}$ uses $r(n)$ random bits when run on graphs of n vertices[3]. Consider a graph $G = (V, E)$ that $\mathcal{O}$ gets access to. For $e \in E$ and a string $\pi \in \{0,1\}^{r(n)}$, let $Q_{e,\pi}$ denote the set of edges in E queried by $\mathcal{O}$ on input e, while simulating the run of $\mathcal{A}$ with π as the random string. The set $Q_{e,\pi}$ denotes the set of edges e' such that the status of e in the solutions output by $\mathcal{A}$ with randomness π on inputs G and $G - e'$ could be different. For each edge $e' \in E$ and string $\pi \in \{0,1\}^{r(n)}$, define $R_{e',\pi}$ as the set of edges $e \in E$ such that $e' \in Q_{e,\pi}$.

By definition, for each $\pi \in \{0,1\}^{r(n)}$, we have $\sum_{e \in E} |R_{e,\pi}| = \sum_{e \in E} |Q_{e,\pi}|$. Hence we have:

$$\sum_{\pi \in \{0,1\}^{r(n)}} \sum_{e \in E} |R_{e,\pi}| = \sum_{\pi \in \{0,1\}^{r(n)}} \sum_{e \in E} |Q_{e,\pi}|,$$

and

$$\mathop{\mathbb{E}}_{\pi \sim \{0,1\}^{r(n)}} \mathop{\mathbb{E}}_{e \sim E} |R_{e,\pi}| = \mathop{\mathbb{E}}_{\pi \sim \{0,1\}^{r(n)}} \mathop{\mathbb{E}}_{e \sim E} |Q_{e,\pi}| \leq q(G),$$

where the last inequality follows from our assumption on $\mathcal{O}$.

[3] If $r(G)$ is the length of the random string used for G, we can simply set $r(n) = \max\{r(G) : G = (V, E), |V| = n\}$. If we do not need $r(n)$ bits for some particular graph G on n vertices, we can just throw away the unused bits.

For $\pi \in \{0,1\}^{r(n)}$ and $e \in E$, the set $R_{e,\pi}$ contains the set of edges whose presence in the solution could be affected by the removal of e from G. Therefore, it is a superset of the set of edges contained in the symmetric difference between the outputs of $\mathcal{A}$ on inputs G and $G-e$ when run with π as the random string.

Let $\mathcal{H}_{\mathcal{A},\pi}(G,G')$ denote the Hamming distance between the outputs of the algorithm $\mathcal{A}$ on inputs G and G' when run with π as the random string. As per this notation, for each $e \in E$,

$$\mathop{\mathbb{E}}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e) \leq \mathop{\mathbb{E}}_{\pi\in\{0,1\}^{r(n)}} |R_{e,\pi}|.$$

The following claim relates the quantity on the left hand side of the above inequality with the average sensitivity of $\mathcal{A}$.

Claim 6. *The average sensitivity of $\mathcal{A}$ is bounded as*

$$\beta(G) \leq \mathop{\mathbb{E}}_{e\in E(G)} \mathop{\mathbb{E}}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e).$$

Proof. Fix $G \in \mathcal{G}$ and $e \in E(G)$. We first bound the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}(G-e)$, where $\mathcal{A}(G)$ and $\mathcal{A}(G-e)$ are the output distributions of $\mathcal{A}$ on inputs G and $G-e$, respectively. For $S \in \mathcal{S}$, let $p_G(S)$ and $p_{G-e}(S)$ denote the probabilities that $\mathcal{A}$ outputs S on G and $G-e$, respectively. We start with $\mathcal{A}(G)$. Consider a string $\pi \in \{0,1\}^{r(n)}$. Let $S \in \mathcal{S}$ denote the output of $\mathcal{A}$ on input G when using the string π as its random string. Let S' denote the output that is generated when running $\mathcal{A}$ on input $G-e$ with π as the random string. We move a mass of $\frac{1}{2^{r(n)}}$ (corresponding to the string π) from $p_G(S)$ to $p_G(S')$ at a cost of $\frac{d_{\mathrm{Ham}}(S,S')}{2^{r(n)}}$. Moving masses corresponding to every string $\pi \in \{0,1\}^{r(n)}$ this way, we can transform $\mathcal{A}(G)$ to $\mathcal{A}(G-e)$. The total cost incurred during this transformation is $\mathbb{E}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e)$. Therefore the earth mover's distance between $\mathcal{A}(G)$ and $\mathcal{A}(G-e)$ is at most $\mathbb{E}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e)$. Therefore the average sensitivity of $\mathcal{A}$ is $\beta(G) \leq \mathbb{E}_{e\in E(G)} \mathbb{E}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e)$. ∎

Therefore, the average sensitivity of $\mathcal{A}$ is $\beta(G) \leq \mathbb{E}_{e\sim E} \mathbb{E}_{\pi\in\{0,1\}^{r(n)}} \mathcal{H}_{\mathcal{A},\pi}(G,G-e) \leq \mathbb{E}_{e\sim E} \mathbb{E}_{\pi\in\{0,1\}^{r(n)}} |R_{e,\pi}| \leq q(G)$. ∎

We now prove Corollary 1.1 which says that the existence of an LCA (Definition 2) for a graph problem implies the existence of a stable-on-average algorithm for the same problem.

Corollary 1.1 (LCAs imply stable-on-average algorithms) *Consider a graph problem $\mathcal{P} : \mathcal{G} \to \mathcal{S}$. Let $\delta : \mathbb{N} \to [0,1]$ and $q, r : \mathbb{N} \to \mathbb{N}$. If $\mathcal{P}$ has a (q,r,δ)-LCA $\mathcal{L}$, then, there exists an algorithm $\mathcal{A}$ for $\mathcal{P}$, that on input $G=(V,E)$, has an average sensitivity of at most $q(|V|) + |E| \cdot \delta(|V|)$.*

Proof. Assume without loss of generality that each solution in $\mathcal{S}$ is a subset of edges of its preimage with respect to $\mathcal{P}$. Consider the algorithm $\mathcal{A}$ that, on input $G=(V,E)$, constructs a solution to $\mathcal{P}$ by running $\mathcal{L}$ on each edge $e \in E$ and combining the outputs of $\mathcal{L}$. It is clear that $\mathcal{L}$ is a solution oracle (Definition 1) for the algorithm $\mathcal{A}$. Hence, the average sensitivity of $\mathcal{A}$ is upper bounded by the expected number of queries made by $\mathcal{L}$, which is at most $q(|V|) + |E| \cdot \delta(|V|)$. ∎

Acknowledgments. We are grateful to anonymous reviewers for suggesting a major improvement to the average sensitivity analysis of Algorithm 2. We thank Tasuku Soma and Samson Zhou for several helpful discussions. We extend our gratitude to Sofya Raskhodnikova for helpful comments that improved the presentation of this article. Y.Y. is supported by JST, PRESTO Grant Number JPMJPR192B, Japan. N.V. thanks the NII International Internship Program for supporting his visit to NII, Tokyo where most of this work was done.

References

[1] N. K. Ahmed, J. Neville, and R. Kompella. Network sampling. *ACM Transactions on Knowledge Discovery from Data*, 8(2):1–56, 2014.

[2] N. Alon, R. Rubinfeld, S. Vardi, and N. Xie. Space-efficient local computation algorithms. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1132–1139, 2012.

[3] A. Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.

[4] M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, pages 499–526, 2002.

[6] K. Censor-Hillel, E. Haramaty, and Z. S. Karnin. Optimal dynamic distributed MIS. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 217–226, 2016.

[7] A. Czumaj, Y. Mansour, and S. Vardi. Sublinear graph augmentation for fast query implementation. In *Proceedings of the 16th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 181–203, 2018.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference (TCC)*, pages 265–284, 2006.

[9] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, pages 449–467, 1965.

[10] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[11] G. Even, M. Medina, and D. Ron. Best of two local models: Centralized local and distributed local algorithms. *Inf. Comput.*, 262(Part):69–89, 2018.

[12] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[14] A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar. Differentially private combinatorial optimization. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1106–1125, 2010.

[15] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009.

[16] A. Hassidim, Y. Mansour, and S. Vardi. Local computation mechanism design. *ACM Trans. Economics and Comput.*, 4(4):21:1–21:24, 2016.

[17] M. Hay, C. Li, G. Miklau, and D. D. Jensen. Accurate estimation of the degree distribution of private networks. In *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM)*, pages 169–178, 2009.

[18] D. R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 21–30, 1993.

[19] V. Karwa, S. Raskhodnikova, A. D. Smith, and G. Yaroslavtsev. Private analysis of graph structure. *ACM Transactions on Database Systems*, 39(3):22:1–22:33, 2014.

[20] V. Karwa and A. B. Slavkovic. Differentially private graphical degree sequences and synthetic graphs. In *Proceedings of the International Conference on Privacy in Statistical Databases (PSD)*, pages 273–285, 2012.

[21] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. D. Smith. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography (TCC)*, pages 457–476, 2013.

[22] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

[23] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[24] J. R. Lee, S. Oveis Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM*, 61(6):37, 2014.

[25] S. H. Lee, P.-J. Kim, and H. Jeong. Statistical properties of sampled networks. *Physical Review E*, 73(1), 2006.

[26] C. Lenzen and R. Levi. A centralized local algorithm for the sparse spanning graph problem. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 87:1–87:14, 2018.

[27] R. Levi and M. Medina. A (centralized) local guide. *Bulletin of the EATCS*, 122, 2017.

[28] R. Levi, D. Ron, and R. Rubinfeld. Local algorithms for sparse spanning graphs. *Algorithmica*, 82(4):747–786, 2020.

[29] R. Levi, R. Rubinfeld, and A. Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. *Algorithmica*, 77(4):971–994, 2017.

[30] Y. Mansour, B. Patt-Shamir, and S. Vardi. Constant-time local computation algorithms. *Theory Comput. Syst.*, 62(2):249–267, 2018.

[31] Y. Mansour, A. Rubinstein, S. Vardi, and N. Xie. Converting online algorithms to local computation algorithms. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 653–664, 2012.

[32] Y. Mansour and S. Vardi. A local computation approximation scheme to maximum matching. In *Proceedings of 16th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 260–273, 2013.

[33] M. Marchiori and V. Latora. Harmony in the small-world. *Physica A: Statistical Mechanics and its Applications*, 285(3-4):539–546, 2000.

[34] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[35] W. Meulemans, B. Speckmann, K. Verbeek, and J. Wulms. A framework for algorithm stability and its application to kinetic euclidean MSTs. In *Proceedings of the 13th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 805–819, 2018.

[36] S. Murai and Y. Yoshida. Sensitivity analysis of centralities on unweighted networks. In *Proceedings of the 2019 World Wide Web Conference (WWW)*, pages 1332–1342, 2019.

[37] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[38] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[39] M. E. J. Newman. Network structure from rich but noisy data. *Nature Physics*, 14(6):542–545, 2018.

[40] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–336, 2008.

[41] K. Nissim, S. Raskhodnikova, and A. D. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 75–84, 2007.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[43] M. Parter, R. Rubinfeld, A. Vakilian, and A. Yodpinyanee. Local computation algorithms for spanners. In *Proceedings of 10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 58:1–58:21, 2019.

[44] P. Peng and Y. Yoshida. Average sensitivity of spectral clustering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020. to appear.

[45] S. Raskhodnikova and A. D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In *Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 495–504, 2016.

[46] O. Reingold and S. Vardi. New techniques and tighter bounds for local computation algorithms. *J. Comput. Syst. Sci.*, 82(7):1180–1200, 2016.

[47] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Proceedings of the 1st Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011.

[48] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[49] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning*. From Theory to Algorithms. Cambridge University Press, Cambridge, 2009.

[50] N. Varma and Y. Yoshida. Average Sensitivity of Graph Algorithms. *arXiv e-prints*, page arXiv:1904.03248, Apr 2019.

[51] Y. Yoshida, M. Yamamoto, and H. Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM Journal on Computing*, 41(4):1074–1093, 2012.

[52] Y. Yoshida and S. Zhou. Sensitivity analysis of the maximum matching problem. In *Proceedings of the 12th Innovations in Theoretical Computer Science (ITCS)*, 2021.