Near-Optimal Differentially Private Graph Algorithms via the Multidimensional AboveThreshold Mechanism

Laxman Dhulipala
University of Maryland, College Park, MD, USA

Monika Henzinger
Institute of Science and Technology Austria, Klosterneuburg, Austria

George Z. Li
Carnegie Mellon University, Pittsburgh, PA, USA

Quanquan C. Liu
Yale University, New Haven, CT, USA

A. R. Sricharan
Faculty of Computer Science, UniVie Doctoral School Computer Science DoCS, University of Vienna, Austria

Leqi Zhu
University of Manitoba, Canada

Abstract

Many differentially private and classical non-private graph algorithms rely crucially on determining whether some property of each vertex meets a threshold. For example, for the k-core decomposition problem, the classic peeling algorithm iteratively removes a vertex if its induced degree falls below a threshold. The sparse vector technique (SVT) is generally used to transform non-private threshold queries into private ones with only a small additive loss in accuracy. However, a naive application of SVT in the graph setting leads to an amplification of the error by a factor of n due to composition, as SVT is applied to *every vertex*. In this paper, we resolve this problem by formulating a novel *generalized* sparse vector technique which we call the *Multidimensional AboveThreshold (MAT) Mechanism* which generalizes SVT (applied to vectors with one dimension) to vectors with *multiple* dimensions. When applied to vectors with n dimensions, we solve a number of important graph problems with better bounds than previous work.

Specifically, we apply our MAT mechanism to obtain a set of improved bounds for a variety of problems including k-core decomposition, densest subgraph, low out-degree ordering, and vertex coloring. We give a *tight* local edge differentially private (LEDP) algorithm for k-core decomposition that results in an approximation with $O(\varepsilon^{-1}\log n)$ additive error and *no* multiplicative error in $O(n)$ rounds. We also give a new $(2+\eta)$-factor multiplicative, $O(\varepsilon^{-1}\log n)$ additive error algorithm in $O(\log^2 n)$ rounds for any constant $\eta > 0$. Both of these results are asymptotically tight against our new lower bound of $\Omega(\log n)$ for any constant-factor approximation algorithm for k-core decomposition. Our new algorithms for k-core decomposition also directly lead to new algorithms for the related problems of densest subgraph and low out-degree ordering. Finally, we give novel LEDP differentially private defective coloring algorithms that use number of colors given in terms of the *arboricity* of the graph.

2012 ACM Subject Classification Security and privacy; Theory of computation

Keywords and phrases differential privacy, abovethreshold, densest subgraph

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.91

Related Version *Full Version*: `https://arxiv.org/abs/2508.02182`

Funding *Monika Henzinger and A. R. Sricharan*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (MoDynStruct, No. 101019564) and the Austrian Science Fund (FWF) grant DOI

10.55776/Z422 and grant DOI 10.55776/I5982. *Laxman Dhulipala and George Z. Li*: supported by NSF award number CNS-2317194. *Quanquan C. Liu*: supported by a Google Academic Research Award and by an NSF award number CCF-2453323.

1 Introduction

Designing and analyzing algorithms that satisfy *differential privacy* [23], the de facto standard for protecting user data, is a tricky task, with few general techniques. Among them, the *sparse vector technique (SVT)* is a well-known approach to make certain *threshold-based* computations privacy-preserving. This typically involves showing that the computation can be reduced to a sequence of black-box invocations of private *AboveThreshold* mechanisms, followed by some post-processing. Each such mechanism allows one to iteratively (and adaptively) query whether some real-valued property of the data is above some threshold, until the first "yes" response, at which point no further queries are possible.

While SVT was originally presented as a technique for *sequential* computations, recent work on concurrent composition [54] show that it is possible to interleave queries to different (AboveThreshold) mechanisms with (worst-case) privacy loss proportional to the number of concurrently queried mechanisms. However, this can be prohibitively large for applications (e.g., in parallel and distributed settings) where there is a large amount of concurrency and, in this case, one must exploit structural properties of the instance to obtain better guarantees.

To analyze these situations, we introduce a new mechanism, called *Multidimensional AboveThreshold (MAT)*, which allows one to query, *in parallel*, whether a number of different real-valued properties of the data are above some specified thresholds. We identify a condition on the sensitivity of the submitted queries that allows one to use MAT with far less privacy loss versus concurrent composition. By analyzing various non-private algorithms through the lens of our mechanism, we obtain private algorithms with substantially improved accuracy guarantees for a variety of important, recently studied graph problems.

All of our algorithms apply in the stringent, *local* model of (edge) differential privacy *(LEDP)*, where each user (here, a vertex in the graph) interactively discloses information about its data (here, its incident edges) in multiple *rounds* of communication with an *untrusted*[1] curator (or server), which produces the output. The *transcript* of messages sent throughout the interaction must be differentially private.

We now describe our contributions and the main related work in more detail (see Table 1 for a summary of our results). In the following, all graphs are simple (no self-loops), undirected, and contain n vertices. Furthermore, all mentioned additive error upper bounds hold *with high probability*, i.e., $1 - 1/n^c$, for any constant $c > 0$.

k-Core Decomposition. At a high level, core numbers measure the relative importance of vertices in a graph. Formally, given a graph G, the *core number* (or, *coreness*) of a vertex v in G is the largest $k_G(v) \in \mathbb{N}$ (or $k(v)$ if G is clear from context) for which there exists a subgraph H of G *containing* v such that every vertex in H has induced degree $\geq k_G(v)$. In the *k-core decomposition problem*, the goal is to output core numbers of all vertices in G.[2]

[1] The curator may accidentally leak their messages, e.g., due to security breaches. However, it is not malicious. This is referred to as "honest but curious" in the literature.

[2] We note that the k-core decomposition problem in the literature is sometimes used to refer to the problem of computing a hierarchy of *k-cores*, which are subgraphs of G induced by vertices with core number at least k [49]; note that it is not possible to emit such a hierarchy privately, so we (and prior work) focus on privately emitting core numbers for every vertex.

This problem has recently received much attention from the database and systems communities due its applications in community detection [1, 28, 31], clustering and data mining [16, 51], and other machine learning and graph analytic applications [37, 18, 19]; for a more comprehensive survey, see [43]. Due to the rapid increase of applications using sensitive user data, designing private algorithms for this problem is becoming increasingly important.

An algorithm is a (ϕ, ζ)-*approximation* for the k-core decomposition problem if, for every vertex v in the given graph, the *estimate* of the core number of v, as returned by the algorithm, lies between $k(v) - \zeta$ and $\phi \cdot k(v) + \zeta$. Dhulipala et al. [20] gave the first differentially private $(2+\eta, O(\varepsilon^{-1}\log^3 n))$-approximation algorithm for k-core decomposition in the ε-LEDP model, which runs in $O(\log n)$ rounds. They left it as an open question whether their multiplicative and additive approximation factors can be improved.

Our contributions. We answer their question in the affirmative. In particular, we give an *exact* (i.e., 1-approximate) algorithm and a $(2+\eta)$-approximate algorithm, both with $O(\varepsilon^{-1}\log n)$ additive error, which run in $O(n)$ and $O(\log^2 n)$ rounds, respectively.

▶ **Theorem 1.** *For $\varepsilon > 0$, there is an exact, $O(n)$-round ε-LEDP algorithm for the k-core decomposition problem, which has $O(\varepsilon^{-1}\log n)$ additive error, with high probability.*

▶ **Theorem 2.** *For $\varepsilon > 0$ and* constant *$\eta > 0$, there is a $(2+\eta)$-approximate, $O(\log^2 n)$-round ε-LEDP algorithm for the k-core decomposition problem, which has $O(\varepsilon^{-1}\log n)$ additive error, with high probability.*

Furthermore, we show that the additive errors of our algorithms are tight, by proving a matching lower bound of $\Omega(\varepsilon^{-1}\log n)$ on the additive error of any constant-approximation algorithm in the *centralized* model (EDP), where the algorithm has full access to user data (here, the input graph). Since the centralized model is less restricted than the local model, the same bound holds in the LEDP model, regardless of round complexity. We also show that interactivity is needed if one wants an exact algorithm with polylogarithmic additive error, by proving a lower bound of $\Omega(\sqrt{n})$ on the additive error of any exact, 1-round algorithm, when ε is a constant.

▶ **Theorem 3.** *For $\varepsilon > 0$ and $\phi \geq 1$, any ϕ-approximate ε-EDP algorithm for the k-core decomposition problem in the centralized model has $\Omega((\varepsilon\phi)^{-1}\log n)$ additive error, with constant probability.*

▶ **Theorem 4.** *For* constant *$\varepsilon > 0$, any 1-round, exact ε-LEDP algorithm for the k-core decomposition problem has $\Omega(\sqrt{n})$ additive error, with constant probability.*

Densest Subgraph. Another measure of importance, defined on *subsets* of vertices in a graph, is density. Formally, the *density* of a set of vertices S in a graph G, denoted $\rho_G(S)$ (or simply $\rho(S)$), is the ratio of the number of edges in the subgraph induced by S and the number of vertices in S, i.e., $|E(S)|/|S|$. In the *densest subgraph* problem, the goal is to output a set of vertices with the maximum density in a given graph.

This problem has been studied in the (centralized) private [29, 48], LEDP [21, 20], dynamic [13, 50], streaming [3, 12, 27, 46], and distributed [4, 53] settings; for a recent survey on densest subgraph and its variants, see [39].

An algorithm is a (ϕ, ζ)-*approximation* to the densest subgraph problem if the set of vertices S returned by the algorithm has density $\rho(S) \geq \rho^*/\phi - \zeta$, where ρ^* is the maximum density. In the ε-LEDP setting, Dhulipala et al. [20] gave a $(4+\eta, O(\varepsilon^{-1}\log^3 n))$-approximation for the densest subgraph problem. More recently, and concurrently with this work, Dinitz et al. [21] gave a $(2+\eta, O((\varepsilon\eta)^{-1}\log^2 n))$-approximation algorithm, which runs in $O(\log n)$ rounds, in the ε-LEDP setting, and as well as an $(1, O(\varepsilon^{-1}\sqrt{\log\delta^{-1}}\log n))$-approximation, which runs in $O(n^2\log n)$ rounds in the (ε,δ)-LEDP setting. On the other hand, there is a lower bound of $\Omega(\phi^{-1}\sqrt{\varepsilon^{-1}\log n})$ on the additive error of any ϕ-approximate algorithm for densest subgraph that satisfies (ε,δ)-edge differential privacy in the centralized model [29, 48] (when $\delta \le n^{-O(1)}$).

Our contributions. We give a condition on when a (private) (ϕ,ζ)-approximation algorithm for k-core decomposition can be transformed into a (private) $(2\phi,\zeta)$-approximation algorithm for densest subgraph. We show that our algorithms for k-core decomposition satisfy the condition, obtaining the following improved bounds for densest subgraph.

▶ **Theorem 5.** *For $\varepsilon > 0$, there is a 2-approximate, $O(n)$-round ε-LEDP algorithm for the densest subgraph problem, which has $O(\varepsilon^{-1}\log n)$ additive error, with high probability.*

▶ **Theorem 6.** *For $\varepsilon > 0$ and* constant *$\eta > 0$, there is a $(4+\eta)$-approximate, $O(\log^2 n)$-round ε-LEDP algorithm for the densest subgraph problem, which has $O(\varepsilon^{-1}\log n)$ additive error, with high probability.*

We also give a simple 1-round algorithm using the randomized response technique together with brute-force, exponential time search. We believe this algorithm serves as an interesting example of what a lower bound for 1-round algorithms must be able to reason about.

▶ **Theorem 7.** *For $\varepsilon > 0$, there is an exact, 1-round ε-LEDP algorithm for the densest subgraph problem, which has $O((1+\varepsilon^{-1})\sqrt{n})$ additive error, with high probability.*

Low Out-Degree Ordering. A *low out-degree orientation* of a (undirected) graph is an orientation of the edges of the graph such that the out-degree of any vertex is minimized. A *low out-degree ordering* is a permutation of the vertices, which implicitly encodes a low out-degree orientation, i.e., each edge is oriented from its earlier endpoint to its later endpoint (with respect to the permutation). In the *low out-degree ordering problem*, the goal is to output a low out-degree ordering of a given graph.

Differentially private low out-degree ordering was introduced by [20] as a way to release an implicit solution to the low out-degree orientation problem. The latter has attracted a lot of attention in the non-private setting (see, e.g., [9, 13, 38, 53] and references therein) and has been used to improve the running time of algorithms for matching [47, 33], coloring [34], and subgraph counting [7, 8, 42].

The *degeneracy* of a graph is the maximum core number of any vertex in the graph. It is known that degeneracy is equal to the minimum out-degree that is achievable through any ordering. Hence, we state the out-degree obtained through our algorithms in terms of α, the degeneracy of the input graph. An algorithm is a (ϕ,ζ)*-approximation* for the low out-degree ordering problem if the algorithm's output encodes an orientation where each vertex has out-degree at most $\le \phi\cdot\alpha+\zeta$. In the ε-LEDP model, Dhulipala et al. [20] gave a $(2+\eta, O(\varepsilon^{-1}\log^3 n))$-approximation to the low out-degree ordering problem, which runs in $O(\log n)$ rounds.

Our contributions. We improve the multiplicative factor and the additive error by giving an exact algorithm and a $(2+\eta)$-approximate algorithm, both with additive error $O(\varepsilon^{-1}\log n)$, which run in $O(n)$ and $O(\log n)$ rounds, respectively.

▶ **Theorem 8.** *For $\varepsilon > 0$, there is an exact, $O(n)$-round ε-LEDP algorithm algorithm for the low out-degree ordering problem, which has $O(\varepsilon^{-1}\log n)$ additive error, with high probability.*

▶ **Theorem 9.** *For $\varepsilon > 0$ and* constant *$\eta > 0$, there is a $(2+\eta)$-approximate, $O(\log^2 n)$-round ε-LEDP algorithm for the low out-degree ordering problem, which has $O(\varepsilon^{-1} \log n)$ additive error, with high probability.*

Defective Coloring. A (vertex) *coloring* of a graph G is an assignment of labels (or, *colors*) from the set $\{1, \ldots, n\}$ to the vertices of G. A coloring is (c, d)*-defective* if it uses at most c different colors, and each vertex has at most d neighbors with the same color; d is the *defectiveness* of the coloring. In the *defective coloring* problem, the goal is to output a (c, d)-defective coloring of a given graph with small c and d.

This problem (especially the case $d = 0$) has been widely studied in a variety of settings including the (centralized) private [15], dynamic [11, 34, 35, 52], distributed [30, 32, 45], and streaming [2, 10, 6] settings. In the ε-edge differential privacy setting, [15] recently gave the first (c, d)-defective coloring algorithm with $c = O(\Delta/\log n + \varepsilon^{-1})$ and $d = O(\log n)$, where Δ is the maximum degree of the input graph. They also showed that $d = \Omega(\log n/(\log c + \log \Delta))$ defectiveness is necessary if $\leq c$ colors are used. Although not explicitly stated, we believe their coloring algorithm can be also adapted to the ε-LEDP setting with the same bounds.

Our contributions. Using the approximate low out-degree ordering results and MAT, we give new algorithms for defective coloring under LEDP. In particular, the number of colors used by our algorithms scales with the arboricity α instead of the maximum degree. In small-arboricity graphs, the number of colors used by our algorithm can be significantly smaller than that of [15].

▶ **Theorem 10.** *For $\varepsilon > 0$, there is an $O(n)$-round ε-LEDP algorithm for the defective coloring problem that uses $O(1 + \alpha\varepsilon/\log n)$ colors, where α is the arborocity of the input graph, and has $O(\varepsilon^{-1} \log n))$ defectiveness, with high probability.*

▶ **Theorem 11.** *For $\varepsilon > 0$, there is an $O(\log^2 n)$-round ε-LEDP algorithm for the defective coloring problem that uses $O(\alpha\varepsilon \log n + \log^2 n)$ colors, where α is the arborocity of the input graph, and has $O(\varepsilon^{-1} \log n))$ defectiveness, with high probability.*

Efficient centralized algorithms. All of the bounds in the above theorems are stated for ε-LEDP algorithms but also are algorithms in the centralized model with the same approximation and additive error guarantees. All of our algorithms based on MAT can be implemented in the centralized model in near-linear time if we lose an additional multiplicative factor of $1 + \eta$ for any given constant $\eta > 0$. This nearly matches the best running times for sequential non-private k-core decomposition algorithms, while maintaining a near-optimal utility for private algorithms.

Concurrent, Independent Work. The recent concurrent, independent work of Dinitz et al. [21] provides a large set of novel results on (ε, δ)-DP, ε-LEDP, and (ε, δ)-LEDP densest subgraph, improving on the approximation guarantees of previous results. Their work is based on a recent result of Chekuri, Quanrud, and Torres [14] and the parallel densest subgraph algorithm of Bahmani, Kumar, and Vassilvitskii [4]. They offer improvements in the bounds in the central DP setting via the private selection mechanism of Liu and Talwar [41]. Note that their techniques are different from those presented here; furthermore, any c-approximate algorithm for densest subgraph translates to a $2c$-approximate algorithm for finding the degeneracy of the input graph (which is the value of the maximum k-core). Thus, no densest subgraph algorithm can obtain a better than 2-approximation of the maximum k-core value.

Subsequent Work. Using the MAT framework of this work, Dinitz et al. [22] gave new algorithms for node-differentially private bipartite matching, improving upon the results of [36]. They also gave the first algorithms for maximum matchings in general graphs under local edge-differential privacy, also crucially using the MAT framework. Complementing their algorithms, they show a matching lower bound for edge-private maximum matchings. This follow up work highlights the wide applicability of the MAT framework, giving nearly tight algorithms for various combinatorial problems under differential privacy.

Table 1 Summary of error bounds in the local model. Each upper bound is for a mechanism satisfying (ε, δ)-edge differential privacy and holds with high probability. An asterisk in the citation indicates that an algorithm is (ε, δ)-LEDP instead of ε-LEDP. Lower bounds are for ε-differential privacy and hold with constant probability.

Problem	Apx. Factor	Additive Error	Rounds	Citation
Core Decomposition	1	$O(\varepsilon^{-1}\log n)$	$O(n)$	Theorem 1
	1	$\Omega(\sqrt{n})$ (constant ε)	1	Theorem 4
	$2+\eta$	$O(\varepsilon^{-1}\log n)$	$O(\log^2 n)$	Theorem 2
	$2+\eta$	$O(\varepsilon^{-1}\log^3 n)$	$O(\log n)$	[20]
	ϕ	$\Omega((\varepsilon\phi)^{-1}\log n)$	any	Theorem 3
Densest Subgraph	1	$O(\varepsilon^{-1}\sqrt{\log(\delta^{-1})}\log n)$	$O(n^2\log n)$	[21]*
	1	$O((1+\varepsilon^{-1})\sqrt{n})$	1	Theorem 7
	ϕ	$\Omega(\phi^{-1}\sqrt{\varepsilon^{-1}\log n})$	any	[29, 48]
	2	$O(\varepsilon^{-1}\log n)$	$O(n)$	Theorem 5
	$2+\eta$	$O(\varepsilon^{-1}\eta^{-1}\log^2 n)$	$O(\eta^{-1}\log n)$	[21]
	$4+\eta$	$O(\varepsilon^{-1}\log^3 n)$	$O(\log n)$	[20]
	$4+\eta$	$O(\varepsilon^{-1}\log n)$	$O(\log^2 n)$	Theorem 6
Low Out-degree Ordering	1	$O(\varepsilon^{-1}\log n)$	$O(n)$	Theorem 8
	$2+\eta$	$O(\varepsilon^{-1}\log n)$	$O(\log^2 n)$	Theorem 9
	$2+\eta$	$O(\varepsilon^{-1}\log^3 n)$	$O(\log n)$	[20]
	# Colors	**Defectiveness**		
Defective Coloring	$O(1+\varepsilon\alpha/\log n)$	$O(\varepsilon^{-1}\log n)$	$O(n)$	Theorem 10
	$O(\varepsilon\alpha\log n+\log^2 n)$	$O(\varepsilon^{-1}\log n)$	$O(\log^2 n)$	Theorem 11
	$O(\varepsilon^{-1}+\Delta/\log n)$	$O(\log n)$	2	[15]
	c	$\Omega(\frac{\log(n)}{\log c+\log\Delta})$	any	[15]

2 Technical Overview

MAT. Our new *Multidimensional AboveThreshold* (MAT) mechanism generalizes the AboveThreshold mechanism [24] to support a d-dimensional vector of queries. Conceptually, one may think of MAT as running d copies of AboveThreshold in parallel, one per dimension. More precisely, suppose that the input dataset for (single-dimensional) AboveThreshold was from some domain $\mathcal{X}$, then the input for our mechanism is $X = (x_1, \ldots, x_d) \in \mathcal{X}^d$, a d-tuple of elements from $\mathcal{X}$. Our mechanism takes as input an adaptively chosen sequence $\vec{f_1}, \vec{f_2}, \ldots$ of *query vectors* and a *threshold vector* $\vec{T} = (T_1, \ldots, T_d)$. Each query vector specifies d *queries*, one for each dimension of the dataset, i.e., $\vec{f_i} = (f_{i,1}, \ldots, f_{i,d})$ and query $f_{i,j} : \mathcal{X} \to \mathbb{R}$ is on element x_j. The goal is to (privately) return, after each query vector $\vec{f_i}$, a response vector $\vec{a}_i \in \{\bot, \top\}^d$ indicating whether each query $f_{i,j}$ in the vector has *crossed corresponding* *threshold* T_j, i.e., $\vec{a}_i = (a_{i,1}, \ldots, a_{i,d})$ and $a_{i,j}$ indicates whether $f_{i,j}(x_j) \geq T_j$ ($\perp$ for "no", $\top$ for "yes"). After some query $f_{i,j}$ has crossed its threshold, then all future queries $f_{i',j}$, for $i' > i$, on element x_j are ignored (e.g., the response is always $\perp$). It is assumed that new query vectors are submitted until each element has had a query that has crossed its corresponding threshold. In a nutshell, MAT allows one to infer, for each element x_j, an estimate $\tilde{r}(j)$ of the index $r(j)$ of the *first* query $f_{r(j),j}$ on x_j that crosses the threshold T_j.

Sensitivity condition. We define a fairly technical condition on the *sensitivity* of the queries, generalizing the ℓ_1 sensitivity in the 1-dimensional case, such that the error of MAT can be bounded in terms of the sensitivity. For intuition, we first describe a simple case of our sensitivity condition.

Suppose that two datasets $X = (x_1, \ldots, x_d)$ and $X' = (x'_1, \ldots, x'_d)$ are neighboring only if they differ on at most $k \ll d$ coordinates, i.e., there exists a set $S \subseteq [d]$ of at most k indices such that $x_i = x'_i$ for all $i \notin S$ and $x_i \sim x'_i$ for all $i \in S$. Furthermore, suppose that the absolute difference of each query (on the appropriate element in X and X') is bounded by 1, i.e., for any query vector $\vec{f} = (f_1, \ldots, f_d)$, each query f_j satisfies $|f_j(x_j) - f_j(x'_j)| \leq 1$. Since MAT runs d copies of AboveThreshold in parallel and the queries are adaptive, applying concurrent composition [54] in this case would predict that the privacy loss is proportional to d. However, by coupling the outputs of the queries on the indices *not* in S (where X and X' are identical), it seems fairly intuitive that the actual privacy loss *should* be proportional to k. Hence, as the error of AboveThreshold is inversely proportional to the privacy loss parameter, to obtain ε-differential privacy, we *should* have at most $k \ll d$ times the original error of AboveThreshold in this setting, i.e., $O(k\varepsilon^{-1} \log n)$ with high probability.

More generally, for an arbitrary neighboring relation and arbitrary queries, roughly speaking, we define the sensitivity D as the worst-case (over pairs of neighboring datasets X and X') sum of the maximum absolute difference in the outputs of the queries (on X and X'); see Section 4 for a formal definition. In particular, for the simple case described above, $D = k$. Using this definition, we formalize the above intuition and prove that MAT guarantees an error of $O(D\varepsilon^{-1} \log n)$ in our applications, with high probability.

Applications of MAT. We apply MAT to get new private algorithms for the k-core decomposition, densest subgraph, low out-degree ordering, and defective coloring problems. In each case, given a graph $G = (V, E)$ with n vertices, we show that some non-private algorithm can be simulated via MAT, for suitable definitions of "dataset" and "queries".

As a simple example, for the core decomposition problem, a dataset $X = (x_u : u \in V)$ is an n-tuple containing the set of neighbors x_u of each vertex u in G. (Hence, the domain $\mathcal{X}$ is the set of all subsets of V.) A query $f_S : \mathcal{X} \to \mathbb{R}$ counts the number of neighbors of the vertex that are *not* among some fixed set of vertices $S \subseteq V$, shifted by some fixed amount i, and "mirrored" by n, i.e., $f_S(x) = n - (|x \setminus S| - i)$. (Notice that $f_S(x_u) \geq n$ means that vertex u has at most i neighbors that are not among S.) In this case, the absolute difference of each query on two neighboring datasets (i.e., graphs that differ by at most an edge) is at most 1 and there are at most two elements that differ for two neighboring datasets (namely, at the endpoints of the differing edge). Hence, the simple case mentioned above applies and MAT guarantees an error of $O(\varepsilon^{-1} \log n)$, with high probability.

For core decomposition, we apply MAT to analyze a variant of the classic peeling algorithm of Matula and Beck [44], which iteratively peels (removes) the lowest degree vertex in the graph. In our variant of the classical algorithm, we view the algorithm as running n iterations of a *peeling process*. In the ith iteration, the algorithm repeatedly peels all vertices with (induced) degree less than i until none are left. It is known that, at the end of the ith iteration, a vertex is alive (has not been peeled) if and only if it is in the i-core. It can be shown that the peeling process can be simulated by MAT via the queries mentioned above. Hence, with some post-processing, we can obtain core numbers with no multiplicative factor and the optimal (up to a constant factor against our lower bound) additive error of $O(\varepsilon^{-1} \log n)$, with high probability. Similarly, we apply MAT to analyze the more round-efficient $(2+\eta)$-approximate core decomposition algorithm of [20], obtaining optimal error, with high probability.

Using our new core decomposition algorithms, we achieve improved additive error bounds for both the densest subgraph and low out-degree ordering problems. In particular, we show that the 2-approximate densest subgraph with additive error $O(\varepsilon^{-1} \log n)$ is contained in the subgraph consisting of all vertices with the maximum estimated core numbers (as we run the peeling algorithm, it can be shown that these vertices necessarily induce a subgraph with sufficiently high minimum degree). Furthermore, all of our core decomposition algorithms directly return a low out-degree ordering with the same multiplicative and additive errors as the original core decomposition algorithms by taking the order in which the nodes are peeled.

We also apply MAT to defective coloring by using MAT to greedily pick colors for each vertex from the available colors that have not yet been occupied by too many neighbors. In particular, each dimension represents a vertex and color pair consisting of all vertices in the graph and all available colors. For dimension j representing pair (v, c), the queries ask whether the number of neighbors of v colored with c exceeded our defectiveness threshold $O(\varepsilon^{-1} \log n)$. If it has, we will remove c from v's available set of colors. Our low round coloring algorithm follows a similar structure, and is inspired by the non-private algorithm of [34] and uses our low out-degree ordering algorithm to implement a separate set of available colors for each of $O(\log^2 n)$ groups of vertices of similar core numbers.

Local lower bound. Our lower bound for 1-round, exact core decomposition in the local model is obtained by reduction to a problem in the *centralized* model, for which there is a strong lower bound. Specifically, it is known that, to privately answer $\Theta(n)$ *random* inner product queries on a secret $\{0,1\}^n$ vector, most responses need to have $\Omega(\sqrt{n})$ additive error [17]. Recently, [26] gave an elegant lower bound on the additive error of 1-round triangle counting algorithms in the local model using similar techniques.

In our case, we define a class of "query graphs", one per inner product query, in which the core number of a *fixed* vertex x is (roughly) the answer to the query. In addition to x, there are some *secret* vertices (and other vertices). For each secret vertex v, the existence of the edge $\{v, x\}$ is private information (which depends on the secret vector). Crucially, all neighborhoods of x and the secret vertices in *any possible query graph* appear in two specific query graphs, namely, ones corresponding to the all-ones and all-zeros query vectors, respectively. The neighborhoods of the remaining vertices do not depend on the secret vector.

Our approach to solving the inner product problem is now as follows. The centralized algorithm first simulates the 1-round local core decomposition algorithm on the two fixed graphs to determine the messages that x and the secret vertices would send in *any* query graph and saves these messages. Subsequently, when answering a query, the centralized algorithm *reuses* the saved messages for x and the secret vertices (without further privacy loss) when it simulates the core decomposition algorithm on the corresponding query graph. This allows it to answer many queries correctly.

Originally, we had a more complex, direct argument. We briefly mention it here to give an idea of what is going on "under the hood" of the reduction. Specifically, we showed that, on a class of random graphs (similar to the query graphs instantiated in our reduction), most transcripts of a 1-round core decomposition algorithm have the property that, conditioned on seeing the transcript, the coreness of some (fixed) vertex still has high variance, $\Omega(n)$. This implies that the standard deviation of the additive error is $\Omega(\sqrt{n})$. We prefer the reduction argument, as it is simpler and gives a stronger result.

One-round algorithm for densest subgraph. In our single-round exact densest subgraph algorithm, the vertices send the server a noisy version of the incidence matrix of the graph by using randomized response. In particular, each vertex sends its row of the incidence matrix and, for each bit, it sends the actual bit with probability $p = e^{\varepsilon}/1 + e^{\varepsilon}$; the flipped bit is sent with probability $1 - p$. Using this, the server can create an unbiased estimator $\tilde{E}(S)$ for the number of edges $E(S)$ inside any *fixed* subset of vertices S. The key observation is that the standard deviation σ of the estimated density, i.e., $\tilde{E}(S)/|S|$, is $O(1 + \varepsilon^{-1})$ and the estimated density is distributed binomially. Hence, by a Chernoff bound, the estimated density is $\Omega(\sqrt{\log t})$ standard deviations away from the actual density with probability at most $1/t$. By taking a union bound over all S, we can upper bound the total error over *all* (exponentially many) subsets by $\alpha = O(\sigma\sqrt{n})$ with high probability. Hence, a *brute-force*, exponential-time search, which computes the maximum $\tilde{E}(S)/|S|$ over all subsets S, finds a subset whose density is at most α from the maximum density, with high probability. The union bound appears to be necessary, but it is conceivable that there are more clever search strategies that do not require looking at exponentially many subsets S.

3 Differential Privacy Background

We give some basic definitions and background on differential privacy in this section, see [25] for more details. Since we will only be working with graphs, we will state these results in terms of edge differential privacy for graph algorithms; we will also use the terms differential privacy and edge differential privacy interchangeably. Edge-neighboring graphs is a well-studied model for differential privacy (see e.g. [40] for a survey of such results) and protects the privacy of edges with highly sensitive information like disease transmission graphs.

▶ **Definition 12.** *Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are* edge-neighboring*, denoted by $G_1 \sim G_2$, if $V_1 = V_2$ and $|E_1 \oplus E_2| = 1$ (i.e., they have the same vertex set and they differ by exactly one edge).*

▶ **Definition 13.** *We use $\mathcal{G}$ to denote the set of undirected graphs. An algorithm $\mathcal{M} : \mathcal{G} \to \mathcal{Y}$ is* (ε, δ)-edge differentially private *((ε, δ)-edge dp) if for all edge-neighboring graphs $G \sim G'$ and every $S \subseteq \mathcal{Y}$, we have*

$$\Pr[\mathcal{M}(G) \in S] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{M}(G') \in S] + \delta.$$

If $\delta = 0$, we say $\mathcal{M}$ is ε-edge differentially private (ε-edge DP). When $\mathcal{M}$ is defined over $\mathbb{N}$ or $\mathbb{R}$ instead, it is called ε-differentially private (ε-DP).

Next, we define two important properties of DP: composition and post-processing.

▶ **Lemma 14.** *Let $\mathcal{M}_1 : \mathcal{G} \to \mathcal{Y}_1$ and $\mathcal{M}_2 : \mathcal{G} \to \mathcal{Y}_2$ be ε_1- and ε_2-edge differentially private mechanisms, respectively. Then $\mathcal{M} : \mathcal{G} \to \mathcal{Y}_1 \times \mathcal{Y}_2$ defined by $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$ is $(\varepsilon_1 + \varepsilon_2)$-edge differentially private.*

▶ **Lemma 15.** *Let $\mathcal{M} : \mathcal{G} \to \mathcal{Y}$ be an ε-edge differentially private mechanism. and $f : \mathcal{Y} \to \mathcal{Z}$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : \mathcal{G} \to \mathcal{Z}$ is ε-edge differentially private.*

The ℓ_1-*sensitivity* of a function $f : \mathcal{G} \to \mathbb{R}^d$, denoted $\Delta_1(f)$, is the supremum of the quantity $\|f(x) - f(x')\|_1 = \sum_{i=1}^{d} |f(x)_i - f(x')_i|$, over all neighboring $G, G' \in \mathcal{G}$. The *Laplace* distribution (centered at 0) with *scale* $b > 0$ has probability density function $f(x) = \frac{1}{2b}\exp(-|x|/b)$. We write $X \sim \mathsf{Lap}(b)$ or just $\mathsf{Lap}(b)$ to denote a random variable X that is distributed according to the Laplace distribution with scale b.

▶ **Fact 3.1.** *If random variable* $X \sim \mathsf{Lap}(b)$*, then, for any* $\beta > 0$*,* $\Pr[|X| > b\log(1/\beta)] \leq \beta$*.*

▶ **Fact 3.2** (Theorem 3.6 in [25]). *Let* $f : \mathcal{G} \to \mathbb{R}^d$ *be any function, let* $\epsilon > 0$*, and let, for each* $i \in [d]$*,* $X_i \sim \mathsf{Lap}(\Delta_1(f)/\epsilon)$*. Then the* Laplace mechanism $\mathcal{A}(x) = f(x) + (X_1, \ldots, X_d)$ *is* ϵ*-differentially private.*

3.1 Local Edge Differential Privacy (LEDP)

The local model of differential privacy studies the setting where all private information is kept private by the individual parties, i.e. there exists no trusted curator. For graphs, the local model is called local edge differential privacy [20]. We now define this formally.

▶ **Definition 16.** *Let* G *be a graph. An* ε-**local randomizer** $R : \mathbf{a} \to \mathcal{Y}$ *for vertex* v *is an* ε*-edge differentially private algorithm that takes as input the set of its neighbors* $N(v)$ *in* G*, represented by an adjacency list* $\mathbf{a} = (b_1, \ldots, b_{|N(v)|})$*.*

Note that the information released via local randomizers is public to all nodes and the curator. The curator performs some computation on the released information and makes the result public via sending a message to all vertices. The overall computation is formalized via the notion of the transcript.

▶ **Definition 17.** *Let* G *be a graph. A* transcript π *is a vector consisting of 4-tuples* $(S_U^t, S_R^t, S_\varepsilon^t, S_Y^t)$ *– encoding the set of vertices chosen, the set of local randomizers assigned, the set of privacy parameters of the randomizers, and the set of randomized outputs produced – for each round* t*. Let* S_π *be the collection of all transcripts and* S_R *be the collection of all randomizers. Let STOP denote a special character indicating that the local randomizer's computation halts. A* protocol *is an algorithm* $\mathcal{A} : S_\pi \to (2^{[n]} \times 2^{S_R} \times 2^{\mathbb{R}^{\geq 0}} \times 2^{\mathbb{R}^{\geq 0}}) \cup \{STOP\}$ *mapping transcripts to sets of vertices, randomizers, and randomizer privacy parameters. The length of the transcript, as indexed by* t*, is its round complexity. Given* $\varepsilon > 0$*, a randomized protocol* $\mathcal{A}$ *on graph* G *is* ε-local edge differentially private *if the algorithm that outputs the entire transcript generated by* $\mathcal{A}$ *is* ε*-edge differentially private on graph* G*.*

The definition is difficult to parse, but it naturally corresponds to the intuition of vertices with private information communicating with an untrusted curator (represented by the protocol) so that the curator can compute some output that depends on the private data of the vertices (see also the discussion in [21]). At the beginning, the curator only knows the vertex set V and each vertex knows its incident edges, i.e., its neighborhood. In each round, a chosen subset of the vertices performs some computation using local randomizers on (a) their local edge information, (b) the outputs from previous rounds, and (c) the public information, so basically the whole transcript so far. The vertices then output a message which can be seen by all other vertices and the curator. The released information of each vertex is ε-dp since it computed based on the output of its local randomizer. In each round the curator can choose whom to query, what local randomizer they use, and what privacy parameters the randomizer uses. Since the curator's choice only depends on the transcripts from the previous rounds, this is exactly the definition of a protocol in the above definition.

Algorithm 1 Multidimensional AboveThreshold (MAT).

```
1 Input: Private graph $G$, adaptive queries $\{\vec{f}_1, \dots, \vec{f}_n\}$, threshold vector $\vec{T}$, privacy $\varepsilon$,
    sensitivity $D$.
2 Output: A sequence of responses $\{\vec{a}_1, \dots, \vec{a}_n\}$ where $a_{i,j}$ indicates if $f_{i,j}(G) \geq \vec{T}_j$
   1: for $j = 1, \dots, d$ do
   2:   $\hat{T}_j \leftarrow \vec{T}_j + \mathrm{Lap}(2D/\varepsilon)$
   3: end for
   4:
   5: for each query $\vec{f}_i \in \{\vec{f}_1, \dots, \vec{f}_n\}$ do
   6:   for $j = 1, \dots, d$ do
   7:     Let $\nu_{i,j} \leftarrow \mathrm{Lap}(4D/\varepsilon)$
   8:     if $f_{i,j}(G) + \nu_{i,j} \geq \hat{T}_j$ then
   9:       Output $a_{i,j} = \top$
  10:       Stop answering queries for coordinate $j$
  11:     else
  12:       Output $a_{i,j} = \perp$
  13:     end if
  14:   end for
  15: end for
```

We end the discussion with a short remark on stateless/stateful vertices. As defined above, a local randomizer is only allowed to look at the past (public) transcript and its (private) neighborhood list to compute its output for the current round. This formal definition disallows the use of private states that persist across rounds in the local memory of a vertex. Private stateful behaviour is nonetheless crucial for implementing MAT locally, since we require each vertex to store its noisy threshold privately. This is not a problem, since one can implement a protocol as in [5, Lemma 8] to simulate private states using local randomizers.

4 Generalized Sparse Vector Technique

We introduce a simple, novel multidimensional generalization of the AboveThreshold mechanism, which is the basis of the algorithms in this paper. In the standard AboveThreshold mechanism from Section 3.6 in [25], we are given as input a threshold T and an adaptive stream of sensitivity 1 queries $f_1, f_2, \dots$, and the goal is to output the first query which exceeds the threshold. In the multidimensional version, we have a d-dimensional vector of thresholds $\vec{T} = (T_1, \dots, T_d)$ and an adaptive stream of ℓ_1-sensitivity D vector-valued queries $\vec{f}_1, \vec{f}_2, \dots$. The goal is to output, for each coordinate $j \in [d]$, the first query i_j for which the j^{th} coordinate exceeds the threshold T_j. The pseudo-code is given in Algorithm 1.

Here D is defined as the sensitivity for *all* dimensions, i.e., for edge-neighboring $G \sim G'$,

$$D := \max_{G \sim G'} \left(\sum_{j \in [d]} \left(\max_{i \in [I_j]} \left(|f_{i,j}(G) - f_{i,j}(G')| \right) \right) \right), \tag{1}$$

where I_j is the number of queries in dimension j.

▶ **Theorem 18.** *Algorithm 1 is ε-differentially private.*

Proof. Fix $G \sim G'$ to be arbitrary edge-neighboring graphs. Let $\mathcal{A}(G)$ and $\mathcal{A}(G')$ denote the random variable representing the output of Algorithm 1 when run on G and G', respectively. For each coordinate $j \in [d]$, there is some $r(j) \in \{1, \ldots, n+1\}$ such that the output satisfies $a_{\cdot j} \in \{\top, \bot\}^{r(j)}$ and has the form that for all $i < r(j)$, $a_{i,j} = \bot$. We can assume without loss of generality that $a_{r(j),j} = \top$; if $r(j) < n+1$, this must be the case and if $r(j) = n+1$, we can simply ask an additional query which evaluates to ∞, independent of the dataset. Our goal is to show that for any output a, we have

$$\Pr[\mathcal{A}(G) = a] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(G') = a].$$

Observe that there are two sources of randomness in the algorithm: the noisy thresholds $\hat{T}_j$ for each $j \in [d]$ and the perturbations of the queries $\nu_{i,j}$ for each $j \in [d]$, $i \in [r(j)]$. We fix the random variables $\nu_{i,j}$ for each $j \in [d]$ and $i < r(j)$ by simply conditioning on their randomness; for neighboring graphs, we have a coupling between corresponding variables. For simplicity, we omit this from notation. The remaining random variables are $\hat{T}_j$ and $\nu_{r(j),j}$ for each $j \in [d]$; we will be taking probabilities over their randomness. The main observation needed is that the output of $\mathcal{A}$ is uniquely determined by the values of $r(j)$ for each $j \in [d]$. Thus, we can define the (deterministic, due to conditioning on the probabilities) quantity

$$g_j(G) \coloneqq \max\{f_{i,j}(G) + \nu_{i,j} : i < r(j)\}$$

for each $j \in [d]$ so that the event that $\mathcal{A}(G) = a$ is equivalent to the event where

$$\hat{T}_j > g_j(G) \text{ and } f_{r(j),j}(G) + \nu_{r(j),j} \geq \hat{T}_j \text{ for each } j \in [d]. \tag{2}$$

Let $p(\cdot)$ and $q(\cdot)$ denote the density functions of the random vectors consisting of $\hat{T}(j)$ and $\nu_{r(j),j}$ for each $j \in [d]$. Thus,

$$\Pr[\mathcal{A}(G) = a] = \Pr[\hat{T}_j \in (f_{r(j),j}(G) + \nu_{r(j),j}, g_j(G)) \; \forall j \in [d]]$$

Using v'_j to denote the value of $\nu_{r(j),j}$ and t'_j to denote the value of $\hat{T}_j$,

$$\Pr[\mathcal{A}(G) = a] = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} p(\vec{t'}) q(\vec{v'}) \cdot \mathbb{1}\{t'_j \in (f_{r(j),j}(G) + v'_j, g_j(G)) \; \forall j \in [d]\} \, d\vec{v'} \, d\vec{t'} \tag{3}$$

Now, we make a change of variables from $\vec{v'}$ to $\vec{v}$ and $\vec{t'}$ to $\vec{t}$, with the change given by

$$v'_j = v_j + g_j(G) - g_j(G') + f_{r(j),j}(G') - f_{r(j),j}(G)$$
$$t'_j = t_j + g_j(G) - g_j(G')$$

for each $j \in [d]$. We have that $d\vec{t'}_j = d\vec{t}_j$, and $d\vec{v'}_j = d\vec{v}_j$ since the new variables differ from the old ones by a constant that is independent of $\vec{v}$ and $\vec{t}$. The term inside the indicator function in the integral now becomes

$$t_j + g_j(G) - g_j(G') \in (f_{r(j),j}(G) + v_j + g_j(G) - g_j(G') + f_{r(j),j}(G') - f_{r(j),j}(G), g_j(G)) \forall j \in [d]$$

which on subtracting $g_j(G) - g_j(G')$ gives

$$t_j \in (f_{r(j),j}(G') + v_j, g_j(G')) \forall j \in [d]$$

Thus, we can apply the change of variables to rewrite (3) as

$$\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} p(\vec{t'}) q(\vec{v'}) \mathbb{1}\{t_j \in (f_{r(j),j}(G') + v_j, g_j(G')) \; \forall j \in [d]\} \, d\vec{v} \, d\vec{t}.$$

Since we have conditioned on the randomness of $\nu_{i,j}$ for each $j \in [d]$ and $i < r(j)$, we have $\|\vec{v'} - \vec{v}\|_1 \leq 2D$ and $\|\vec{t'} - \vec{t}\|_1 \leq D$ since the vectors $f_{i\cdot}(G)$ have sensitivity D, implying that the vectors $g_{\cdot}(G)$ also have sensitivity D. This implies that $p(\vec{t'}) \leq \exp(\varepsilon/2) \cdot p(\vec{t})$ and $q(\vec{v'}) \leq \exp(\varepsilon/2) \cdot q(\vec{v})$ by the properties of the Laplace distribution. We can thus upper bound the above integral by

$$\exp(\varepsilon) \cdot \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} p(\vec{t}) q(\vec{v}) \mathbb{1}\{t_j \in (f_{r(j),j}(G') + v_j, g_j(G'))\ \forall j \in [d]\}\, d\vec{v}\, d\vec{t}$$

Rewriting the integral as the probability of an event, we get

$$\exp(\varepsilon) \cdot \Pr[\hat{T}_j \in (f_{r(j),j}(G') + \nu_{r(j),j}, g_j(G'))\ \forall j \in [d]].$$

Finally, by our observation in (2), this is equal to

$$\exp(\varepsilon) \cdot \Pr[\mathcal{A}(G') = a].$$

Putting the inequalities together, we get

$$\Pr[\mathcal{A}(G) = a] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(G') = a].$$

Since G, G' were arbitrary edge-neighboring graphs, this completes the proof. ◀

4.1 LEDP Multidimensional AboveThreshold (MAT) Mechanism

In our algorithms, we apply the multidimensional AboveThreshold mechanism in the special case where the queries $\vec{f_i}$ have $d = n$ coordinates, and each coordinate of the query corresponds to a node u in the graph G and only depends on the edges $e = (u, v)$ for $v \in V - \{u\}$. In other words, we have an instance of the AboveThreshold on each node $u \in V$, where the data used for the AboveThreshold instance is only the local (edge) data of u. We will show that in this setting, the multidimensional AboveThreshold mechanism can be implemented locally to satisfy local edge-differential privacy.

For clarity of notation, we index the coordinates of the queries and threshold vector by nodes $v \in V$ instead of indices $j \in [n]$. That is, each query consists of $\vec{f}_{i,v}$ for $v \in V$ and the threshold vector consists of $\vec{T_v}$ for $v \in V$. We will now present the changes needed for the local implementation. In Lines 1–3, let each node $u \in V$ compute and store its noisy threshold $\hat{T}_u$ using the public threshold $\vec{T}_u$. Then for each query $\vec{f_i}$ in lines 5–15, each node $u \in V$ can sample its own noise $\nu_{i,u}$ and check the condition $f_{i,u}(G) + \nu_{i,u} \geq \hat{T}_u$. The pseudocode is given in Algorithm 2, where actions performed by node v in the algorithm are performed their corresponding local randomizers, and we now show that this is locally edge-differentially private.

▶ **Theorem 19.** *Algorithm 2 is ε-locally edge differentially private.*

Proof. First, we need to show that the local randomizers are in fact edge-differentially private. This can be seen because each randomizers' output is a subset of the output of Algorithm 1 and it is only computed based on the incident edges of each vertex. In other words, the output is a post-processing of Algorithm 1, so the privacy guarantees follow from Theorem 18 and Lemma 15. Next, we argue that the transcript is also ε-edge differentially private. Recall that the transcript consists of the set of vertices chosen, the set of local randomizers assigned, the set of privacy parameters assigned, and the set of outputs. In the algorithm, the set of vertices chosen, the set of local randomizers assigned, and the set of privacy parameters assigned are all functions of the outputs from the previous rounds and the public information. Thus, it suffices to prove that the outputs are differentially private by post-processing (Lemma 15). But this was proven in Theorem 18, so we are done. ◀

Algorithm 2 Local Multidimensional AboveThreshold.

```
1 Input: Private graph $G$, adaptive queries $\{\vec{f_i}\}$, threshold vector $\vec{T}$, privacy $\varepsilon$,
   sensitivity $D$.
2 Output: A sequence of responses $\{\vec{a}_i\}$ where $a_{i,j}$ indicates if $f_{i,j}(G) \geq \vec{T}_j$
 1: for $v \in V$ do
 2:     Node $v$ computes $\hat{T}_v \leftarrow \vec{T}_v + \mathrm{Lap}(2D/\varepsilon)$ and stores it
 3: end for
 4:
 5: for each query $\vec{f_i}$ do
 6:     for $v \in V$ do
 7:         Node $v$ samples $\nu_{i,v} \leftarrow \mathrm{Lap}(4D/\varepsilon)$
 8:         if $f_{i,v}(G) + \nu_{i,v} \geq \hat{T}_v$ then
 9:             Node $v$ outputs $a_{i,j} = \top$
10:             Node $v$ outputs STOP, and stops answering queries
11:         else
12:             Node $v$ outputs $a_{i,v} = \perp$
13:         end if
14:     end for
15: end for
```

5 LEDP k-Core Decomposition

In this section, we present one application of MAT by giving an improved algorithm for differentially private k-core decomposition. We first present a variant of the classical (non-private) algorithm for k-core decomposition in Section 5.1, and then show how to make it private in Section 5.2. We defer our other proofs to the full version.

5.1 A Variation of the Classical Algorithm

The classical peeling algorithm begins with the full vertex set V, which is the 0-core of the graph. Given the $(k-1)$-core, the algorithm computes the k-core via an iterative peeling process: the algorithm repeatedly removes all nodes v for which the induced degree is less than k, and labels the nodes which remain as being part of the k-core. Running this for k from 1 up to n gives the full algorithm, the pseudocode of which is given in Algorithm 3.

▶ **Theorem 20.** *For all $v \in V$, the output $\hat{k}(v)$ given by Algorithm 3 is the core number of v.*

Proof. We will inductively show that the algorithm recovers the k-core of the graph. The base case of $k = 0$ is easy. Now, assume that the algorithm finds the true $(k-1)$-core. Let $V(k)$ denote the subset of nodes which aren't removed in the iterative process for $k-1$ in Line 2. We have that each node $v \in V(k)$ has induced degree at least k in $V(k)$, or else it would have been removed. Thus, we know that the core numbers $k(v)$ is at least k for each $v \in V(k)$, so we have that $V(k)$ is a subset of the true k-core. Now, let K denote the true k-core. Since the k-core is always a subset of the $(k-1)$-core by definition, each node $v \in K$ is in V_t at the beginning of the iterations. Furthermore, we know that $v \in K$ is never removed from V_t since the induced degree is always at least k since $K \subseteq V_t$. Thus, we have that $V(k)$ is a superset of the true k-core as well. Thus, $V(k)$ is the true k-core for each k, so all nodes are labelled correctly. ◀

Algorithm 3 Threshold-Based k-core Decomposition Algorithm.

```
1 Input: Graph $G = (V, E)$.
2 Output: $k$-core value of each node $v \in V$
   1: Initialize $V_0 \leftarrow V$, $t \leftarrow 0$, $\hat{k}(v) \leftarrow 0$ for all $v \in V$
   2: for $k = 1, \ldots, n$ do
   3:   repeat
   4:     $t \leftarrow t + 1$, $V_t \leftarrow V_{t-1}$
   5:     for $v \in V_{t-1}$ do
   6:       if $d_{V_{t-1}}(v) < k$ then
   7:         $V_t \leftarrow V_t - \{v\}$
   8:       end if
   9:     end for
  10:   until $V_{t-1} - V_t = \emptyset$
  11:
  12:   Update the core numbers $\hat{k}(v) \leftarrow k$ for all nodes $v \in V_t$
  13: end for
```

5.2 Private Implementation of the Algorithm

It is difficult to turn the classical algorithm into a differentially private one because it has $\Omega(n)$ iterations, which causes us to incur $\tilde{\Omega}(n)$ additive error when using basic composition (Lemma 14). In fact, this was cited in [20] as the motivation for basing their algorithms on parallel/distributed algorithms for k-core decomposition, since those algorithms often have $\operatorname{poly}\log(n)$ round-complexity. Our main observation is that the private version of Algorithm 3 can be analyzed as a special case of the Multidimensional AboveThreshold mechanism, so it doesn't need to incur the $\Omega(n)$ additive error due to composition.

▶ **Theorem 21.** *Algorithm 4 is ε-edge differentially private.*

Proof. We will show that Algorithm 4 is an instance of the multidimensional AboveThreshold mechanism, implying that it is ε-edge differentially private. Specifically, we will show that its output can be obtained by post-processing the output of an instance of Algorithm 1. Indeed, consider the instance of the multidimensional AboveThreshold mechanism where the input graph is G, the privacy parameter is ε, and the threshold vector $\vec{T} = \vec{0}$ is the zero vector. We will now inductively (and adaptively) define the queries.

For each iteration t, the t^{th} query consists of the vector of $k - d_{V_{t-1}}(v)$ for each $v \in V$, where $V_0 = V$ as in the algorithm; note that this matches with the queries on Line 2 of the algorithm. First, observe that the sensitivity of the vector queries of $k - d_{V_{t-1}}(v)$ for each $v \in V$ is $D = 2$ since one edge change will only affect two coordinates of the query (each by 1), as needed in the privacy analysis. Next, observe that we can construct V_t from V_{t-1} using only the outputs $a_t(v)$ of the queries at the current iteration: given $v \in V_{t-1}$, we include v in V_t as well if and only if $a_t(v) = \perp$, which is equivalent to the condition in Line 2 of $d_{V_{t-1}}(v) + \nu_{t,v} \leq k + \tilde{\ell}(v)$[3]. Thus, this is a feasible sequence of adaptive queries for the AboveThreshold mechanism.

[3] Here, we implicitly assume that the randomness used by the algorithm and the AboveThreshold mechanism are the same. This is justified by a coupling of the random variables in the two algorithms. Specifically, we couple the noise added in Line 4 of the algorithm with the noise added in Line 2 of the AboveThreshold mechanism and couple the noise $\nu_{t,v}$ with $\nu_{i,j}$ in the AboveThreshold mechanism. It

Algorithm 4 ε-Differentially Private k-Core Decomposition.

```
1 Input: Graph $G = (V, E)$, privacy parameter $\varepsilon > 0$.
2 Output: An $(1, 120\log(n)/\varepsilon)$-approximate $k$-core value of each node $v \in V$
 1: $V_0 \leftarrow V$, $t \leftarrow 0$, $k = 60\log n/\varepsilon$.
 2: Initialize $\hat{k}(v) \leftarrow 0$ for all $v \in V$
 3: for $v \in V$ do
 4:     $\tilde{\ell}(v) \leftarrow \mathrm{Lap}(4/\varepsilon)$
 5: end for
 6:
 7: while $k \leq n$ do
 8:     repeat
 9:         $t \leftarrow t + 1$, $V_t \leftarrow V_{t-1}$
10:         for $v \in V_{t-1}$ do
11:             $\nu_{t,v} \leftarrow \mathrm{Lap}(8/\varepsilon)$
12:             if $d_{V_{t-1}}(v) + \nu_{t,v} \leq k + \tilde{\ell}(v)$ then
13:                 $V_t \leftarrow V_t - \{v\}$
14:             end if
15:         end for
16:     until $V_{t-1} - V_t = \emptyset$
17:
18:     Update the core numbers $\hat{k}(v) \leftarrow k$ for all nodes $v \in V_t$
19:     $k \leftarrow k + 60\log n/\varepsilon$
20: end while
```

Since the sequence of subsets $\{V_t\}$ are obtained by post-processing the outputs $a_t(v)$ of the AboveThreshold mechanism, they are ε-differentially private by Theorem 18 and Lemma 15. By applying post-processing again, the sequence of pairs (V_t, k_t) for each iteration t is also ε-differentially private since the sequence k_t is public. Given the pairs (V_t, k_t) for each iteration t, we can now recover the approximate core numbers which the algorithm outputs by setting the core numbers as $\hat{k}(v) = k_t$ for each node in $V_t - V_{t-1}$. It is easily verified that this gives the exact same output as Algorithm 4, so we have ε-differential privacy for the algorithm by applying post-processing (Lemma 15) once again. ◀

▶ **Theorem 22.** *Algorithm 4 outputs* $(1, \frac{120\log n}{\varepsilon})$*-approximate core numbers* $\hat{k}(v)$ *with probability* $1 - O(\frac{1}{n^2})$.

Proof. By the density function of the Laplace distribution, we know that we have $|\nu_{t,u}| \leq \frac{40\log n}{\varepsilon}$ and $|\tilde{\ell}(u)| \leq \frac{20\log n}{\varepsilon}$ each with probability $\geq 1 - \frac{1}{n^5}$. Since there are at most $O(n^3)$ such random variables, taking a union bound over all nodes $u \in V$ and all iterations of the loops gives the above guarantee with probability $\geq 1 - O\left(\frac{1}{n^2}\right)$. We condition on the event that the above inequalities hold true for all $t \in [T]$ and $u \in V$ for the remainder of the proof.

Fix an arbitrary iteration k of the while loop starting on Line 7. Let H be the set of nodes remaining in V_t at the end of the while loop in Lines 8–16. We claim that (i) all nodes $u \in H$ have core number at least $k - \frac{60\log n}{\varepsilon}$ and (ii) all nodes $u \notin H$ have core number at most $k + \frac{60\log n}{\varepsilon}$. To see (i), consider the subgraph H; we claim that each node $u \in H$ has

is easy to see that for $\Delta = 2$, the random variables are exactly the same so the privacy guarantees of multidimensional AboveThreshold translates to privacy guarantees for our algorithm.

induced degree at least $k - \frac{60 \log n}{\varepsilon}$ in H. Indeed, since each node in H was not removed in the final iteration of the while loop in Lines 8–16, we have that $d_{H_t}(u) + \nu_{t,u} \geq k + \tilde{\ell}(u)$ for each $u \in H$. But since we have assumed $|\nu_{t,u}| \leq \frac{40 \log n}{\varepsilon}$ and $|\tilde{\ell}(u)| \leq \frac{20 \log n}{\varepsilon}$, our desired bounds follow directly by the triangle inequality. To see (ii), let's suppose for contradiction that the k-core value of u is $k(u) > \ell + \frac{60 \log n}{\varepsilon'}$. Then there exists a subgraph $u \in K \subseteq V$ where the induced degree of each node $v \in K$ is $d_K(v) \geq k(u)$. But for such a subgraph K, the condition in Line 12 will always be true (again, because $|\nu_{t,u}| \leq \frac{40 \log n}{\varepsilon}$ and $|\tilde{\ell}(u)| \leq \frac{20 \log n}{\varepsilon}$) so $K \subseteq H$. But since $u \in K$, this contradicts the fact that $u \notin H$.

Using the above bounds, we can now prove our desired results. First, consider an arbitrary node $u \in V$ labeled $\hat{k}(u)$ within the while loop in Lines 7–20 and not relabeled later. Since u was labeled $\hat{k}(u)$ in the current iteration, we have that $k(u) \geq \hat{k}(u) - \frac{60 \log n}{\varepsilon'}$. Since u was not relabelled in the next iteration where the threshold was $k = \hat{k}(u) + \frac{60 \log n}{\varepsilon}$, the node u was removed from V_t at that iteration. Consequently, we have $k(u) \leq k + \frac{60 \log n}{\varepsilon} = \hat{k}(u) + \frac{120 \log n}{\varepsilon}$ by what we just proved above. Since the output of our algorithm is $\hat{k}(u)$, the desired bounds follow directly. Now, let's consider an arbitrary node $u \in V$ labeled 0 at Line 2 at the beginning of the algorithm and not relabeled later. Since the node u was not relabelled at the first iteration where $k = \frac{60 \log n}{\varepsilon}$, it was removed from V_t at that iteration, implying that $k(u) \leq \frac{120 \log n}{\varepsilon}$. Hence, we have the desired approximation guarantees for all nodes. ◀

References

1. J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, 2005.
2. Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '23, pages 141–153, New York, NY, USA, 2023. Association for Computing Machinery. `doi:10.1145/3584372.3588681`.
3. Bahman Bahmani, Ashish Goel, and Kamesh Munagala. Efficient primal-dual graph algorithms for MapReduce. In *International Workshop on Algorithms and Models for the Web Graph (WAW)*, volume 8882, pages 59–78, 2014. `doi:10.1007/978-3-319-13123-8_6`.
4. Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proc. VLDB Endow.*, 5(5):454–465, 2012. `doi:10.14778/2140436.2140442`.
5. Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Annual International Cryptology Conference*, pages 451–468, 2008. `doi:10.1007/978-3-540-85174-5_25`.
6. Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 11:1–11:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPICS.ICALP.2020.11`.
7. Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. Counting subgraphs in degenerate graphs. *J. ACM*, 69(3), June 2022. `doi:10.1145/3520240`.
8. Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 38:1–38:20, 2020. `doi:10.4230/LIPICS.ITCS.2020.38`.
9. Edvin Berglin and Gerth Stølting Brodal. A simple greedy algorithm for dynamic graph orientation. *Algorithmica*, 82(2):245–259, 2020. `doi:10.1007/S00453-018-0528-0`.

10 Anup Bhattacharya, Arijit Bishnu, Gopinath Mishra, and Anannya Upasana. Even the easiest(?) graph coloring problem is not easy in streaming! In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ITCS.2021.15.

11 Sayan Bhattacharya, Fabrizio Grandoni, Janardhan Kulkarni, Quanquan C. Liu, and Shay Solomon. Fully dynamic $(\Delta + 1)$-coloring in $O(1)$ update time. *ACM Trans. Algorithms*, 18(2):10:1–10:25, 2022. doi:10.1145/3494539.

12 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *ACM Symposium on Theory of Computing (STOC)*, pages 173–182, 2015. doi:10.1145/2746539.2746592.

13 Chandra Chekuri, Aleksander Bjørn Christiansen, Jacob Holm, Ivor van der Hoog, Kent Quanrud, Eva Rotenberg, and Chris Schwiegelshohn. Adaptive out-orientations with applications. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3062–3088. SIAM, 2024.

14 Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. Densest subgraph: Supermodularity, iterative peeling, and flow. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1555, 2022. doi:10.1137/1.9781611977073.64.

15 Aleksander BG Christiansen, Eva Rotenberg, Teresa Anna Steiner, and Juliette Vlieghe. Private graph colouring with limited defectiveness. *arXiv preprint arXiv:2404.18692*, 2024.

16 Luciano da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana, and Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011.

17 Anindya De. Lower bounds in differential privacy. In *Theory of Cryptography Conference, TCC*, pages 321–338. Springer, 2012. doi:10.1007/978-3-642-28914-9_18.

18 Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Julienne: A framework for parallel graph algorithms using work-efficient bucketing. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 293–304, 2017. doi:10.1145/3087556.3087580.

19 Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Theoretically efficient parallel graph algorithms can be fast and scalable. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

20 Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 754–765. IEEE, 2022. doi:10.1109/FOCS54457.2022.00077.

21 Michael Dinitz, Satyen Kale, Silvio Lattanzi, and Sergei Vassilvitskii. Almost tight bounds for differentially private densest subgraph. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 2908–2950. SIAM, 2025. doi:10.1137/1.9781611978322.94.

22 Michael Dinitz, George Z. Li, Quanquan C. Liu, and Felix Zhou. Differentially private matchings, 2025. doi:10.48550/arXiv.2501.00926.

23 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, pages 265–284, 2006. doi:10.1007/11681878_14.

24 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *ACM Symposium on Theory of Computing (STOC)*, pages 381–390, 2009. doi:10.1145/1536414.1536467.

25 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. `doi:10.1561/0400000042`.

26 Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. Triangle counting with local edge differential privacy. In *International Colloquium on Automata, Languages, and Programming, ICALP*, pages 52:1–52:21, 2023. `doi:10.4230/LIPICS.ICALP.2023.52`.

27 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 397–399, 2016. `doi:10.1145/2935764.2935813`.

28 Hossein Esfandiari, Silvio Lattanzi, and Vahab Mirrokni. Parallel and streaming algorithms for k-core decomposition. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1397–1406, 2018.

29 Alireza Farhadi, Mohammad Taghi Hajiaghai, and Elaine Shi. Differentially private densest subgraph. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151, pages 11581–11597, 2022. URL: `https://proceedings.mlr.press/v151/farhadi22a.html`.

30 Mohsen Ghaffari and Fabian Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1009–1020. IEEE, 2022.

31 Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrović. Improved parallel algorithms for density-based network clustering. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2201–2210, 2019. URL: `http://proceedings.mlr.press/v97/ghaffari19a.html`.

32 Mohsen Ghaffari and Christiana Lymouri. Simple and near-optimal distributed coloring for sparse graphs. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPICS.DISC.2017.20`.

33 Meng He, Ganggui Tang, and Norbert Zeh. Orienting dynamic graphs, with applications to maximal matchings and adjacency queries. In *International Symposium on Algorithms and Computation*, pages 128–140. Springer, 2014. `doi:10.1007/978-3-319-13075-0_11`.

34 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Explicit and implicit dynamic coloring of graphs with bounded arboricity. *CoRR*, abs/2002.10142, 2020. `arXiv:2002.10142`.

35 Monika Henzinger and Pan Peng. Constant-time dynamic $(\Delta+1)$-coloring. *ACM Trans. Algorithms*, 18(2):16:1–16:21, 2022. `doi:10.1145/3501403`.

36 Justin Hsu, Zhiyi Huang, Aaron Roth, Tim Roughgarden, and Zhiwei Steven Wu. Private matchings and allocations. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 21–30, 2014. `doi:10.1145/2591796.2591826`.

37 H. Kabir and K. Madduri. Parallel k-core decomposition on multicore platforms. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1482–1491, 2017.

38 Łukasz Kowalik. Approximation scheme for lowest outdegree orientation and graph density measures. In *International Symposium on Algorithms and Computation*, pages 557–566. Springer, 2006.

39 Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *arXiv preprint arXiv:2303.14467*, 2023. `doi:10.48550/arXiv.2303.14467`.

40 Yang Li, Michael Purcell, Thierry Rakotoarivelo, David Smith, Thilina Ranbaduge, and Kee Siong Ng. Private graph data release: A survey. *ACM Computing Surveys*, 55(11):1–39, 2023. `doi:10.1145/3569085`.

41 Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309, 2019. `doi:10.1145/3313276.3316377`.

42 Quanquan C. Liu, Jessica Shi, Shangdi Yu, Laxman Dhulipala, and Julian Shun. Parallel batch-dynamic algorithms for k-core decomposition and related graph problems. In *34th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 191–204, 2022. `doi:10.1145/3490148.3538569`.

43 Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, 29:61–92, 2020. `doi:10.1007/S00778-019-00587-4`.

44 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, July 1983. `doi:10.1145/2402.322385`.

45 Yannic Maus. Distributed graph coloring made easy. *ACM Trans. Parallel Comput.*, 10(4), December 2023. `doi:10.1145/3605896`.

46 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T Vu. Densest subgraph in dynamic graph streams. In *International Symposium on Mathematical Foundations of Computer Science*, pages 472–482. Springer, 2015. `doi:10.1007/978-3-662-48054-0_39`.

47 Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Transactions on Algorithms (TALG)*, 12(1):1–15, 2015. `doi:10.1145/2700206`.

48 Dung Nguyen and Anil Vullikanti. Differentially private densest subgraph detection. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8140–8151, 2021. URL: `http://proceedings.mlr.press/v139/nguyen21i.html`.

49 Ahmet Erdem Sariyüce and Ali Pinar. Fast hierarchy construction for dense subgraphs. *Proceedings of the VLDB Endowment*, 10(3):97–108, 2016. `doi:10.14778/3021924.3021927`.

50 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 181–193, 2020. `doi:10.1145/3357713.3384327`.

51 Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 469–478. IEEE, 2016. `doi:10.1109/ICDM.2016.0058`.

52 Shay Solomon and Nicole Wein. Improved dynamic graph coloring. *ACM Trans. Algorithms*, 16(3), June 2020. `doi:10.1145/3392724`.

53 Hsin-Hao Su and Hoa T. Vu. Distributed Dense Subgraph Detection and Low Outdegree Orientation. In *34th International Symposium on Distributed Computing*, pages 15:1–15:18, 2020. `doi:10.4230/LIPICS.DISC.2020.15`.

54 Salil P. Vadhan and Tianhao Wang. Concurrent composition of differential privacy. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 582–604. Springer, 2021. `doi:10.1007/978-3-030-90453-1_20`.