Distributed Approximate k-Core Decomposition and Min-Max Edge Orientation: Breaking the Diameter Barrier

T-H Hubert Chan, Mauro Sozio, Bintao Sun

Distributed Approximate k-Core Decomposition and Min-Max Edge Orientation: Breaking the Diameter Barrier

T-H. Hubert Chan
The University of Hong Kong

Mauro Sozio
Telecom ParisTech University

Bintao Sun
The University of Hong Kong

***Abstract*—We design distributed algorithms to compute approximate solutions for several related graph optimization problems. All our algorithms have round complexity being logarithmic in the number of nodes of the underlying graph and in particular independent of the graph diameter. By using a primal-dual approach, we develop a $2(1+\epsilon)$-approximation algorithm for computing the coreness values of the nodes in the underlying graph, as well as a $2(1+\epsilon)$-approximation algorithm for the min-max edge orientation problem, where the goal is to orient the edges so as to minimize the maximum weighted in-degree. We provide lower bounds showing that the aforementioned algorithms are tight both in terms of the approximation guarantee and the round complexity. Finally, motivated by the fact that the densest subset problem has an inherent dependency on the diameter of the graph, we study a weaker version that does not suffer from the same limitation.**

***Index Terms*—distributed algorithms, coreness, round complexity**

I. Introduction

The k-core decomposition and algorithms for finding densest subgraphs have proved to be a valuable tool in graph mining and data analysis, with applications encompassing sociology, bioinformatics as well as graph visualization.

Coreness. Formally, a node v in a (weighted) graph G is said to have coreness value k, if k is the largest number such that v belongs to a subgraph of G with minimum (weighted) degree equal to k [25].

Intuitively, nodes belonging to well-connected communities tend to have high coreness values. Besides, several definitions of density have been proposed in the literature. In our work, we focus on the *average degree density* which is defined as the ratio between the number of edges and the number of nodes in a graph [8]. One of the appealing properties of such a definition is that a densest subgraph (in terms of average degree density) can be computed in polynomial time. Such an optimization problem is often referred to as the *densest subset problem*. Recently, the diminishingly-dense decomposition has been introduced and studied [11], [19], [26], which elegantly merges the concepts of density and k-core decomposition. Such a decomposition assigns to each node a real number, which we refer to as *maximal density value*. We also study the *min-max edge orientation problem* [27] which turns out to be related to the previous problems. The goal is to assign an orientation to every edge such that the maximum weighted in-degree is minimized.

In our work, we develop distributed algorithms for approximating the coreness values, the local density values, as well as to find an approximate solution for the *min-max edge orientation problem*. Moreover, we study a weaker version of the densest subset problem. We envision the following applications for our work. Our distributed algorithms can be executed by agents in a social network or a P2P network, for example, so as to collect relevant statistics of the agents with respect to the underlying graph. Users in a social network with large coreness value are known to have "good spreading" properties in epidemiological studies [20]. Therefore, the coreness value (or an approximation) can be leveraged to maximize the spreading of a diffusion protocol. Communities in a social network consist of set of users sharing similar interests, such as hiking, traveling or photography. The density of a given subgraph can be used to measure how likely the corresponding users belong to a same community [29]. Our work allows to approximate such a metric in a distributed fashion. Our distributed algorithms can also be used in distributed graph processing systems [22] to process very large graphs not fitting into the main memory of one single machine.

We are given an undirected edge-weighted graph $G = (V, E, w)$. We consider the classical Local model, in which every node can directly communicate only with its neighbors in synchronous rounds. Moreover, we assume each node knows (an upper bound on) the number $n = |V|$ of nodes. The *hop-diameter* D (or diameter) of a graph G is at most ℓ if every pair of nodes in G can be connected with a path consisting of at most ℓ edges. We use the convention that the approximation ratio γ is at least 1.[1] There is often a tradeoff between the number of rounds and the approximation guarantees of a distributed algorithm. Figure I.1 shows that approximating the coreness values or the min-max edge orientation problem within a factor strictly less than 2 requires

T-H. Hubert Chan was partially supported by the Hong Kong RGC under the grants 17200418.

[1]For minimization problems, this means the solution has value at most γ times the optimal value. For maximization problems, the solution has value at least $\frac{1}{\gamma}$ fraction of the optimal value. For computation problems, the solution is within a multiplicative factor of γ from its true value.

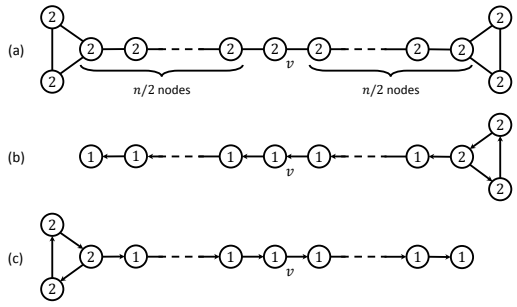

Fig. I.1. Example graphs showing that the we cannot beat 2-approximation for coreness values and min-max edge orientation problem unless the number of rounds is at least $\Omega(n)$. All graphs have unit edge weights, and the nodes are labeled with their coreness values. The coreness of v is 2 in (a), but is 1 in (b) and (c). The arrows in (b) and (c) indicate an optimal orientation, where the maximum in-degree is 1; any other orientation of edges incident on v will result in a maximum in-degree of at least 2. It takes $\Omega(n)$ rounds for node v to distinguish between the different graphs.

$\Omega(D)$ communication rounds. Similarly, for a node to be aware of whether it is included in an approximate densest subgraph, $\Omega(D)$ communication rounds are required. This follows from the fact that such a node has to verify whether a subgraph with higher density (possibly many hops away) is included in the graph or not. Hence, it is natural to investigate the following question: *Can we devise distributed approximation algorithms for the aforementioned problems, while requiring a number of communication rounds independent of the diameter?*

Although some of the problems discussed above have been studied in a distributed environment, there is no work focusing on breaking the diameter barrier, to the best of our knowledge. All our algorithms require a number of communication rounds logarithmic in n, while we provide tight lower bounds expressing the tradeoff between communication rounds and approximation ratio.

A. Our Results and Contribution

Coreness Values. Just like almost every work on these related problems [4], [6], [8], [12], [13], [15], [17], [23], [28], we consider an elimination procedure that repeatedly "peels off" nodes with small (weighted) degrees. In particular, we follow the interpretation by Montresor et al. [23]. Specifically, given a threshold value b, in each round, nodes with degree less than b in the remaining graph are removed. Using a *compact* representation, we can imagine that the elimination procedure for all possible thresholds are run in parallel, where after each iteration, each node v just remembers the largest threshold b_v (that we call the *surviving* number), for which it still survives.

It is known that after each iteration, the surviving number of each node v is at least its coreness $c(v)$; moreover, after n rounds, the surviving number will reach the coreness value[2]. However, to the best of our knowledge, so far there is no approximation analysis for the process in terms of the number of iterations.

[2]In general, $\Omega(n)$ rounds are needed, even if the diameter is constant.

Densest Subset. It is interesting that a variant of the elimination process was considered by Bahmani et al. [4] to give a $2(1+\epsilon)$-approximation for the densest subset streaming model. In each iteration or pass, the threshold is chosen to be $2(1+\epsilon)$ times the density of the current subset of surviving nodes. Then, the process terminates in $O(\log_{1+\epsilon} n)$ iterations, and the subgraph from one of the iterations gives a $2(1+\epsilon)$-densest subset. This inspires us that density can provide the right tool to design and analyze distributed approximation algorithms for coreness values. However, an immediate issue is that for every node to know the (approximate) density of the current subgraph, we already need $\Omega(D)$ rounds.

Local Notion of Density. We observe that it is not necessary to use the global density of the (sub)graph, because in some sense, coreness value measures the local density of a node. Recently, Tatti and Gionis [26] considered a so-called *locally-dense graph decomposition*, which has been further studied by Danisch et al. [11], who defined a quantity known as *maximal density* for each node. This decomposition has actually been considered in passing by Khuller and Saha [19] in the context of finding densest subsets with large sizes. Intuitively, the decomposition works by repeatedly peeling off maximal densest subsets as follows. Given a weighted graph G, the maximal densest subset B forms the first layer; every node $u \in B$ has *maximal density* equal to $\rho_G(B)$, even though no special importance is given to this value in [19]. Next, remove the nodes in B to form a *quotient graph* G', where an edge between $u \in B$ and $v \in \overline{B}$ becomes a self-loop at v in G'. Then, the procedure is recursively applied to G' until all nodes are removed. Equipped with this notion of maximal density, we can adapt previous analysis to achieve the following result.

Theorem I.1 (Gracefully Degrading Approximation Ratio for coreness values)**.** *After running the compact elimination procedure in the* Local *model for T rounds, the surviving number at each node gives a $2n^{\frac{1}{T}}$-approximation to its coreness value (and maximal density).*

In particular, if every node knows the number of nodes (or an upper bound) n and would like to achieve $2(1+\epsilon)$-approximation, then $T = \lceil \log_{1+\epsilon} n \rceil$ rounds are sufficient.

Matching Lower Bound. For $\gamma \geq 2$, by considering a γ-ary tree, we show in Section III that achieving γ-approximation requires $\Omega(\frac{\log n}{\log \gamma})$ rounds.

Min-Max Edge Orientation Problem. The centralized version of the problem was proposed by Venkateswaran [27] and has applications in telecommunication network design; it is known that the special case of unit edge weights can be solved in polynomial time. The connection of the problem with densest subsets has been explored in subsequent works [2], [3].

To the best of our knowledge, the only distributed algorithm for this problem with round complexity independent of the diameter is for unweighted graphs. Specifically, Barenboim and Elkin [5] actually studied a stronger problem, where the goal is to partition the edges of a graph into a minimum number of forests. Instead of density $\rho(S) = \frac{w(E(S))}{|S|}$, a

similar notion of *arboricity* $\mathsf{arb}(S) = \frac{w(E(S))}{|S|-1}$ is considered. In hindsight, it is not surprising that they also used a variant of the distributed elimination procedure to approximate the forest-decomposition problem in the Local model. Indeed, they showed that if the maximum arboricity is known by every node, then $O(\frac{1}{\epsilon}\log n)$ rounds are sufficient to achieve $(2+\epsilon)$-approximation. As remarked above, if every node needs to know the (approximate) maximum arboricity, then $\Omega(D)$ rounds are required. A careful study of their algorithm reveals that the first phase [5, Algorithm 3] serves a similar purpose as computing the surviving numbers as in our Theorem I.1, after which they run the second phase as if the maximum arboricity is known. This degrades the quality of the solution, and only $2(2+\epsilon)$-approximation is achieved.

Primal-Dual Approach. By observing that the LP relaxation of the min-max edge orientation problem is the dual of the densest subset LP (see Section II), we have the intuition that a procedure for approximating the maximal densities should also give information about the dual problem, without using a second phase. Indeed, we augment the distributed elimination procedure by maintaining an auxiliary subset N_v for every node v (which represents its in-neighbors) to give the same approximation ratio for the edge orientation problem. However, a very careful invariant analysis in Lemma III.11 is performed to make sure that every edge is taken care of by at least one of its end-points.

Theorem I.2 (Gracefully Degrading Approximation Ratio for Min-Max Edge Orientation Problem)**.** *After running the augmented elimination procedure (Algorithm 2) in the* Local *model for T rounds, the auxiliary subsets $\{\mathsf{N}_v : v \in V\}$ gives a $2n^{\frac{1}{T}}$-approximation to the min-max edge orientation problem.*

In particular, if every node knows n and would like to achieve $2(1+\epsilon)$-approximation, then $T = \lceil \log_{1+\epsilon} n \rceil$ rounds are sufficient.

Densest Subset Problem. As argued above, for every node to be aware of whether it should be included in an approximate densest subset, it takes $\Omega(D)$ rounds. Indeed, Sarma et al. [24] gave a distributed algorithm that gives $2(1+\epsilon)$-approximation with $O(D\log_{1+\epsilon} n)$ rounds.

We consider a weaker version of the problem (in Definition IV.1). Instead of just producing one subset, the distributed algorithm will return a collection $\{S_i : i \in \mathcal{I}\}$ of disjoint subsets such that for each $i \in \mathcal{I}$, each node will be aware of which subset (if any) it belongs to; moreover, there exists some $i \in \mathcal{I}$ such that S_i is an approximate densest subset. Using the procedure in Theorem I.1 as a subroutine, we have the following.

Theorem I.3 (Distributed (Weak) Densest Subset Problem)**.** *For any $\epsilon > 0$ and n, there exists a distributed algorithm that gives a $2(1+\epsilon)$-approximation to the (weak) densest subset problem in Definition IV.1 on any graph with n nodes using $O(\log_{1+\epsilon} n)$ rounds.*

B. Related Work

In addition to the most relevant works that have already been compared with our work, we also describe other related works.

Corenss Values. After Seidman [25] first proposed k-core decomposition, Batagelj and Zaversnik [6] presented a (centralized) $O(m)$ algorithm to compute a k-core decomposition. This notion has been extended to weighted graphs [9], [14] and directed graphs [13]. The distributed setting has been studied by Montresor et al. [23], and was further extended to dynamic graphs by Aridhi et al. [1]. The distributed algorithms have been adapted to (centralized) I/O efficient algorithms [9], [28] that were proposed to handle large graphs that cannot fit into memory.

Min-Max Edge Orientation Problem. Since Venkateswaran [27] introduced the problem, (centralized) polynomial-time algorithms are known to give optimal solutions for unweighted graphs [3], [21], [27]. However, a series of works [2], [3] showed that the weighted version is NP-hard even when all edge weights belong to the set $\{1, k\}$, where k is any fixed integer greater than 1; on the other hand, for integer edge weights, $(2-\frac{1}{k})$-approximation can be achieved, where k is the maximum weight. Gillet and Hanusse [15] considered the more general asynchronous distributed model with faults. However, their algorithm has round complexity depending on the graph diameter, and achieves $2(2+\epsilon)$-approximation.

The problem can be also be seen as a special case of a load-balancing task, where each node is a machine and each edge is a job to be assigned to one of its incident machines. From this perspective, minimizing the maximum in-degree is equivalent to minimizing the makespan. Czygrinow et al. [10] considered minimizing a slightly different objective that is the sum of the squares of the loads of the machines. In contrast, they gave a distributed 2-approximation algorithm that runs in $O(\Delta^5)$ rounds, where Δ is the maximum degree.

The Min-Max Edge Orientation Problem share some similarities with the vertex cover with hard capacities problem [16].

Densest Subsets. The densest subset problem [8] has been extensively studied and extended to directed graphs [18]. As mentioned above, even though Khuller and Saha [19] have implicitly considered locally-dense graph decompositions, this notion has not been fully studied until recently by Tatti and Gionis [26] and Danisch et al. [11]. One of its surprising applications is its usage in the computation of the (non-linear) Laplacian operator of a hypergraph [7].

II. Preliminaries

Distributed Model. The input is an edge-weighted undirected graph $G = (V, E, w)$, where $n = |V|$ and the edge weights $w : E \to \mathbb{R}_+$ are non-negative. We consider the classical Local model, where each node in V is a CPU with a unique identifier that is aware of only its incident edges (and their weights) and its neighbors; each node knows the total number n of nodes (or an upper bound). In addition to the Local model, our protocols satisfy the following.

- *Synchronous Rounds and Polynomial-Time Computation.* In each round, a node can send a message to its neighbors in G. Moreover, after receiving messages from its neighbors, a node can perform polynomial-time computation in each round.
- *Broadcast Model.* We consider protocols in which a node sends the same message to (a subset of) its neighbors in each round.
- *Message Content and Size.* In our protocols, we assume that each message contains the identity of the sender, and typically, the content of a message is a constant number of real numbers. In most useful applications, each edge weight is an integer whose size is polynomial in n. In this case, every number sent in a message in our protocols can be represented by $O(\log n)$ bits, which means the requirements of the Congest model are also satisfied. Alternatively, with arbitrary edge weights, by restricting the set of numbers to appropriate powers of $(1+\lambda)$, we can also restrict the message size.

Graph Terminology. In most cases, each edge $e \in E$ is interpreted as a 2-subset of V, although in the analysis we sometimes consider a self-loop, which is a singleton.

For each node $v \in V$, the set of neighbors of v is $N_G(v) := \{u \in V : u \neq v \wedge \{u,v\} \in E\}$, and the weighted degree of v is $\deg_G(v) := \sum_{e \in E:\, v \in e} w(e)$, i.e., the sum of the weights of the edges that contain v. When there is no ambiguity, the subscript G can be omitted.

Density. For a non-empty $S \subseteq V$, we denote $E(S) := \{e \in E : e \subseteq S\}$, and its density is $\rho_G(S) = \frac{w(E(S))}{|S|}$. A (non-empty) subset S in G is a *densest subset* if and only if S has maximum density among all subsets in G. The following fact is standard.

Fact II.1. *The maximal densest subgraph of G is unique and contains all densest subgraphs of G.*

Local Density. While a densest subset gives the densest region in a graph G, there are other notions that measure the local density around each node v. The first notion is *coreness* $c_G(v)$, which is mentioned in Section I. Another notion is *maximal density*, which is defined in terms of *quotient graph* and *diminishingly-dense decomposition* as follows.

Definition II.2 (Quotient Graph [11])**.** *Given a weighted undirected graph $G = (V, E, w)$ and a subset $B \subseteq V$, the quotient graph of G with respect to B is a graph $G \backslash B := (\widehat{V}, \widehat{E}, \widehat{w})$, where $\widehat{V} := V \backslash B$ and $\widehat{E} := \{e \cap \widehat{V} : e \in E, e \cap \widehat{V} \neq \varnothing\}$, i.e., every edge $e \in E$ not contained in B contributes towards $\widehat{E}$. Moreover, for $e' \in \widehat{E}$, $\widehat{w}(e') := \sum_{e \in E : e' = e \cap \widehat{V}} w(e)$.*

Definition II.3 (Diminishingly-Dense Decomposition and Maximal Density [11])**.** *Given a weighted undirected graph $G = (V, E, w)$, define the diminishingly-dense decomposition $\mathcal{B}$ of G as the sequence $\varnothing = B_0 \subsetneq B_1 \subsetneq \cdots \subsetneq B_k = V$ as follows.*

Initially we set $B_0 := \varnothing$ and $G_0 := G$. For $i \geq 1$, if $B_{i-1} = V$, the decomposition is fully defined. Otherwise, define $G_i := G_{i-1} \backslash B_{i-1}$ and let S_i be the maximal densest subset in G_i. Then, define $B_i := B_{i-1} \cup S_i$. Finally, for each node $v \in V$, we say that the maximal density of v is $r_G(v) := \rho_{G_i}(S_i)$ if $v \in S_i$. For simplicity, we use $r(v)$ to denote $r_G(v)$ if the context is self-evident.

Fact II.4 (Strictly Diminishing Densities)**.** *In Definition II.3, the sequence $\{\rho_{G_i}(S_i)\}_i$ is strictly decreasing in i.*

Distributed Approximation of Local Density. Our goal is to design distributed protocols such that at the end, each node v outputs some number $\beta(v)$, which is an approximation of its coreness $c(v)$ or maximal density $r(v)$. Observe that there are example graphs such that computing $c(v)$ or $r(v)$ exactly needs $\Omega(n)$ rounds. The main result of this paper is that there are protocols with $O(\log n)$ rounds that give $O(1)$-approximation for both $c(v)$ and $r(v)$ for each node v. To be precise, we use the following convention to describe approximation ratio.

Definition II.5 (Approximation Ratio)**.** *Given a non-negative real number $s \in \mathbb{R}_+$ and $\gamma \geq 1$, another number β is a γ-approximation for s, if $s \leq \beta \leq \gamma \cdot s$. In general, a function $\beta : V \to \mathbb{R}_+$ is a γ-approximation for $s : V \to \mathbb{R}_+$, if for all $v \in V$, $s(v) \leq \beta(v) \leq \gamma \cdot s(v)$.*

Min-Max Edge Orientation Problem. The input is an edge-weighted undirected graph $G = (V, E, w)$. The goal is to give an orientation to each edge such that the maximum weighted in-degree of a node is minimized. Equivalently, we wish to determine an assignment $a : E \to V$ such that each edge e is assigned to one of its end-points, and $\max_{v \in E} \sum_{e \in a^{-1}(v)} w_e$ is minimized.

Distributed Setting. After the distributed algorithm terminates, each node $v \in V$ will have computed some subset N_v of its neighbors, which represents the set of incident edges that are assigned to it. Observe that it is sufficient to enforce the condition that for every edge $\{u, v\} \in E$, we have $u \in \mathsf{N}_v$ or $v \in \mathsf{N}_u$; by means of one more round of communication, we can resolve any conflict (i.e. an edge being assigned to both its end-points).

LP Relaxation and Relationship with Densest Subset. The following is an LP relaxation of the problem together with its dual. In the primal LP, for each $e \in E$ and node $u \in e$, α_u^e is the portion of the weight w_e of edge e that is assigned to u. As observed in [11], the dual LP is exactly the densest subset LP by Charikar [8].

$$
\begin{aligned}
\min \quad & \rho & & \\
\text{s.t.} \quad & \rho \geq \sum_{e : u \in e} \alpha_u^e, & & \forall u \in V \\
& \sum_{u \in e} \alpha_u^e \geq w_e, & & \forall e \in E \\
& \alpha_u^e \geq 0, & & \forall u \in e \in E
\end{aligned}
$$

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & x_e \le y_u, && \forall u \in e \\ & \sum_{u \in V} y_u = 1, \\ & x_e, y_u \ge 0, && \forall u \in V, e \in E \end{aligned}$$

Approximation Analysis. As we shall see, a by-product of our distributed algorithm is that it gives an assignment of edges to their incident nodes. If ρ^* is the maximum density of a subset, and the sum of the edge weights assigned to every node is at most $\gamma \cdot \rho^*$, then weak duality implies that the assignment is a γ-approximation to the min-max edge orientation problem.

III. Distributed Approximation Algorithms for Coreness Values (and Min-Max Edge Orientation)

In this section, we give a distributed protocol that approximates both the coreness value and maximal density for each node in a given graph.

A. Warmup: Single Threshold

The warmup protocol is based on a well-known idea of iteratively eliminating vertices with small degree [4], [6], [8], [13], [15], [17], [23], [28], which can be easily implemented in the distributed model. Here we consider a protocol that is parameterized by some universal threshold b. In each round, each node with weighted degree less than b in the subgraph induced by surviving nodes is marked to be removed at the end of the round. It is clear that after running the protocol for n rounds, all surviving nodes have coreness at least b [13]. It is shown in [17] that running the protocol for various thresholds and $O(\log n)$ rounds can be used to give $O(1)$-approximation for the densest subgraph problem. We show that the analysis can be adapted to approximate both coreness values and maximal density.

We describe the elimination procedure in the distributed model in Algorithm 1. Each node $v \in V$ keeps a state $\sigma_v \in \{0, 1\}$ which records whether the node is present (1) or removed (0).

Algorithm 1: Elimination Procedure for a Single Threshold

Input: $G = (V, E, w)$, threshold value $b \in \mathbb{R}$, number T of rounds
Output: Each node v returns a state $\sigma_v \in \{0, 1\}$.

Initially, each node v has state $\sigma_v \leftarrow 1$.
for *each round* $t \in [1..T]$ **do**
  Each node v broadcasts its current state σ_v to all its neighbors.
  After receiving updated states from all its neighbors, each node v performs the following:
  if $\sum_{e=\{u,v\}:\sigma_u=1} w_e < b$ **then**
    $\sigma_v \leftarrow 0$
return σ_v *for each node* v.

B. Parallel Execution with Multiple Thresholds

Parallel Execution. Observe that the elimination procedure for different threshold values can be executed in parallel, but this can cause a large message size. Hence, for the purpose of analysis, we imagine that the protocol is executed for all possible thresholds in parallel and concentrate on the round complexity. Later in Section III-C, we show how parallel execution for all threshold values can be performed compactly. Depending on this parallel thought experiment, we define the *surviving number* as follows.

Definition III.1 (Surviving Number)**.** *Given a weighted undirected graph $G = (V, E, w)$, the surviving number of a node $v \in V$ after T rounds, denoted as $\beta_G^T(v)$, is defined as the maximum $b \in \mathbb{R}$ such that v survives after T rounds of the elimination procedure using threshold value b. When the context is clear, we omit G or T from the notation and simply use $\beta(v)$.*

It is known that after running the procedure for n rounds, $\beta^n(v) = c(v)$ gives the exact coreness value [23]. Our goal is to show that to get a constant approximation, it suffices to use $O(\log n)$ rounds, independent of the diameter of the graph. In particular, we shall prove that for $\epsilon > 0$, if we set $T := \lceil \log_{1+\epsilon} n \rceil$, then $\beta^T(v)$ is a $2(1+\epsilon)$-approximation for both the coreness value $c(v)$ and maximal density $r(v)$.

Lemma III.2 (Lower Bound on Surviving Number)**.** *For any node $v \in V$ and any positive integer t, $\beta^t(v) \ge c(v)$.*

Proof. For any node v with coreness $c(v)$, by the definition of coreness, suppose $S \subseteq V$ is a subset such that $v \in S$ and $\deg_{G[S]}(u) \ge c(v)$ for all $u \in S$. Then, for $b = c(v)$, all members of S will survive the elimination procedure with threshold b, no matter how many rounds of elimination are executed. In particular, v will survive. Hence, $\beta^t(v) \ge c(v)$ for all positive integers t. ∎

Lemma III.3 (Upper Bound on Surviving Number)**.** *For any positive integer T and any node $v \in V$, we have $\beta^T(v) \le 2n^{\frac{1}{T}} \cdot r(v)$. In other words, to achieve an approximation guarantee of $\gamma > 2$, it suffices to set $T = \lceil \frac{\log n}{\log(\gamma/2)} \rceil$; in particular, for $\epsilon > 0$, the special case $T := \lceil \log_{1+\epsilon} n \rceil$ gives $\beta^T(v) \le 2(1+\epsilon) \cdot r(v)$.*

Proof. The proof approach has appeared in several previous works [4], [8], [17]. For completeness, we adapt the proof from [17, Lemma 3.1].

Fix some $v \in V$. Consider the diminishingly-dense decomposition of G. Let G_i and S_i be defined as in Definition II.3 such that $v \in S_i$, which is the maximal densest subset in $G_i = (V_i, E_i)$. Then, $r(v) = \rho_{G_i}(S_i)$. Observe that $G_i = G \setminus \overline{V_i}$ is a quotient graph, which means that any edge connecting V_i to $\overline{V_i}$ becomes a self-loop in V_i, which implies that for all $v \in V_i$, for all t, $\beta_G^t(v) \le \beta_{G_i}^t(v)$.

Fix any $b > 2n^{\frac{1}{T}} \cdot r(v)$. It suffices to show that after applying T rounds of elimination with threshold b to the graph G_i, no node in V_i (which includes v) can survive.

Define $A_0 := V_i$; for $j \geq 1$, let $A_j \subseteq V_i$ be the set of nodes that survive after round j. Suppose for some $j \geq 0$, both A_j and A_{j+1} are non-empty. Then, since S_i is a densest subset in G_i, we have $r(v) = \rho_{G_i}(S_i) \geq \rho_{G_i}(A_j) = \frac{w(E(G_i[A_j]))}{|A_j|} \geq \frac{\sum_{u \in A_j} \deg_{G[A_j]}(u)}{2 \cdot |A_j|} \geq \frac{b \cdot |A_{j+1}|}{2 \cdot |A_j|} > \frac{n^{1/T} \cdot r(v) \cdot |A_{j+1}|}{|A_j|}$. Note that we use the inequality $w(E(G_i[A_j])) \geq \frac{\sum_{u \in A_j} \deg_{G[A_j]}(u)}{2}$, because G_i could contain self-loops. Hence, we have $|A_{j+1}| < \frac{|A_j|}{n^{1/T}}$. Therefore, if A_T is non-empty, we have $|A_T| < \frac{|A_0|}{n} \leq 1$, which is a contradiction. So A_T is empty, which means that no node in V_i survives after round T, as required. ∎

Lemma III.4 (Relating Maximal Density and Coreness Value)**.** *For any $v \in V$, $r(v) \leq c(v)$.*

Proof. Consider the diminishingly-dense decomposition of G. Let $\{(G_i, B_i, S_i)\}_i$ be defined as in Definition II.3. Recall that there is a unique i such that $v \in S_i \subseteq B_i$, and S_i is the maximal densest subset in G_i.

We prove by induction on $i \geq 1$ that $\deg_{G[B_i]}(v) \geq \rho_{G_i}(S_i)$ for all $v \in B_i$, which is slightly stronger than the required statement. From this, it follows that $c(v) \geq \deg_{G[B_i]}(v) \geq \rho_{G_i}(S_i) = r(v)$ for all $v \in B_i,\ i \geq 1$.

Base Case. Suppose $v \in S_1 = B_1$. Since S_1 is a densest subset in G_1, we must have $\deg_{G[B_1]}(v) \geq \rho_{G_1}(S_1)$, as required. Otherwise, the subset $S_1 \backslash \{v\}$ would have higher density than S_1 in G_1.

Induction Hypothesis. Suppose for some $i > 1$, for all $u \in B_{i-1}$, $\deg_{G[B_{i-1}]}(u) \geq \rho_{G_{i-1}}(S_{i-1})$.

Inductive Step. Consider $v \in S_i$. Note that $B_i = B_{i-1} \cup S_i$. For $u \in S_i$, since $G_i = G \setminus B_{i-1}$ is a quotient graph, it follows that $\deg_{G[B_i]}(u) = \deg_{G_i[S_i]}(u)$, because any edge connecting S_i to $B_i \backslash S_i$ will become a self-loop in G_i. Similar to the base case, because S_i is a densest subset in G_i, we have $\deg_{G_i[S_i]}(u) \geq \rho_{G_i}(S_i) = r(u)$, as required. For $u \in B_{i-1}$, we have $\deg_{G[B_i]}(u) \geq \deg_{G[B_{i-1}]}(u) \geq \rho_{G_{i-1}}(S_{i-1})$, where the second inequality follows from the induction hypothesis. By Fact II.4, the last term satisfies $\rho_{G_{i-1}}(S_{i-1}) > \rho_{G_i}(S_i)$, thereby completing the inductive step and the proof of the lemma. ∎

Theorem III.5. *Given $G = (V, E, w)$ and $\gamma > 2$, let $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$. Then, for all $v \in V$, we have $r(v) \leq c(v) \leq \beta^T(v) \leq \gamma \cdot r(v) \leq \gamma \cdot c(v)$.*

In particular, for any $\epsilon > 0$, we can set $\gamma = 2(1+\epsilon)$ and $T := \lceil \log_{1+\epsilon} n \rceil$.

Proof. This follows directly from Lemma III.2, Lemma III.3 and Lemma III.4. ∎

Corollary III.6 (Relating Coreness Value and Maximal Density)**.** *For each node v, $r(v) \leq c(v) \leq 2 \cdot r(v)$.*

C. Compact Parallel Execution

Observe that it is inefficient (or infeasible) to naively execute the elimination procedure for all threshold values in parallel. Instead, at any moment, a node v just needs to remember the maximum threshold value $\beta(v)$ for which it still survives in the corresponding elimination procedure. Hence, in each round, a node v just needs to send its current value $\beta(v)$ to all its neighbors. Indeed, this observation is made by Montresor et al. [23] to produce a compact algorithm, which we restate in Algorithm 2. However, they run the algorithm till the exact coreness value for each node is achieved, while we have already shown that for any $\gamma > 2$, $T = \lceil \frac{\log n}{\log(\gamma/2)} \rceil$ rounds are sufficient to obtain γ-approximation for both the coreness value and maximal density.

Message Size. If every edge has integer weight that is polynomial in n, then each message has size $O(\log n)$ bits. For general edge weights, to further optimize for the message size, we can restrict the sent numbers from some set Λ, thereby achieving $\log_2 |\Lambda|$ bits per message. After the receiving the current numbers from its neighbors, the node calls the subroutine Update to modify its own number, and round it down to the next number in Λ. For instance, one can set Λ to include appropriate powers of $(1+\lambda)$, for some small $\lambda > 0$. We use the convention $\lambda = 0$ to denote the case when Λ includes all real numbers.

Keeping Auxiliary Information for the Min-Max Edge Orientation Problem. One can augment the elimination procedure to approximate the min-max edge orientation problem. In this case, we consider $\Lambda = \mathbb{R}$. For each node v, in addition to the current surviving number b_v, node v also maintains a subset N_v of its neighbors for the min-max edge orientation problem. Intuitively, N_v contains the neighbors u such that edge (u, v) should be oriented towards v because u has a higher surviving number than v. The subroutine Update in Algorithm 3 is also augmented to return a subset N, which is only needed in the last round; however, in the description, we still maintain N_v after every iteration for the purpose of analysis.

For technical reasons, we need to assume that for each node v, Update is *stateful*; in particular, each node v remembers the surviving numbers of its neighbors in all past iterations, which will be used by Update in the current iteration. Specifically, the following invariants are preserved.

Definition III.7 (Maintained Invariants)**.** *For the special case $\Lambda = \mathbb{R}$, the following invariants are defined on the variables in Algorithm 2.*

- *For each node v, $\sum_{e=\{u,v\}:u \in \mathsf{N}_v} w_e \leq \mathsf{b}_v$.*
- *For each edge $\{u, v\}$, we have $u \in \mathsf{N}_v$ or $v \in \mathsf{N}_u$.*

Remark III.8. *Note that the running time of* Update *is $O(d \log d)$ due to sorting. When the graph is unweighted, similar to the original implementation in [23], the running time can be reduced to $O(d)$ with the help of a counter array of size $O(d)$.*

Fact III.9. *At the end of round T, each node v has $\mathsf{b}_v = \max\{b \in \Lambda : b \leq \beta^T(v)\}$.*

Corollary III.10. *By setting Λ to include appropriate powers of $(1+\lambda)$ and setting $T := \lceil \log_{1+\epsilon} n \rceil$, Algorithm 2 produces b_v for each node v such that*

Algorithm 2: Compact Elimination Procedure

Input: $G = (V, E, w)$, set Λ of threshold values (that are powers of $1+\lambda$), number T of rounds
Output: Each node v maintains a number $\mathsf{b}_v \in \Lambda$; for the special case $\Lambda = \mathbb{R}$, node v also returns a subset N_v of its neighbors.

1 Initially, each node v sets $\mathsf{b}_v \leftarrow +\infty$ and $\mathsf{N}_v \leftarrow N(v)$.
2 **for** *each round* $t \in [1..T]$ **do**
3 Each node v broadcasts its current number b_v to all its neighbors.
4 After receiving the updated number b_u from each neighbor $u \in N(v)$ which is connected to v with an edge of weight w_u, node v performs the following procedure defined in Algorithm 3:
5 $(\mathsf{b}_v, \mathsf{N}_v) \leftarrow v.\mathsf{Update}(\{(u, \mathsf{b}_u, w_u) : u \in N(v)\})$
6 /* N_v is auxiliary, and can be returned only in the last round. */
7 Round b_v down to the next power of $(1+\lambda)$ in Λ.
8 **return** *each node* v *has computed* b_v *(and* N_v*, for the special case* $\Lambda = \mathbb{R}$*).*

Algorithm 3: Update

Input: A sequence of tuples $\{(u_i, \mathsf{b}_i, w_i)\}_{i=1}^{d}$ $(d \geq 1)$
Output: The maximum real number b such that $\sum_{i:\, \mathsf{b}_i \geq b} w_i \geq b$, and some appropriate subset $\mathsf{N} \subseteq \{u_i : \mathsf{b}_i \geq b\}$.

1 Sort and re-index the input sequence according to the surviving numbers: $\mathsf{b}_1 \leq \cdots \leq \mathsf{b}_d$; any tie is resolved by the lexicographic order on the surviving numbers from all past iterations, where more recent iterations have higher priority. Finally,any remaining tie is resolved consistently using the node identity.
2 /* The tie resolving rule is used for analyzing the auxiliary subset N in Lemma III.11. Alternatively, each node maintains an ordering of its neighbors, and stable sorting is performed according to the current b_i's. */
3 We use the convention $\mathsf{b}_0 \leftarrow -\infty$ so that the **for** loop below will definitely terminate when i reaches 1.
4 $s \leftarrow 0$
5 **for** $i = d$ **down to** 1 **do**
6 $s \leftarrow s + w_i$ /* $s = \sum_{j=i}^{d} w_j$ */
7 **if** $s > \mathsf{b}_{i-1}$ **then**
8 $\mathsf{b} \leftarrow \mathsf{b}_i$, $\mathsf{N} \leftarrow \{u_{i+1}, \ldots, u_d\}$
9 /* Invariant: $\sum_{u_j \in \mathsf{N}} w_j \leq \mathsf{b}$ */
10 **if** $s \leq \mathsf{b}_i$ **then**
11 $\mathsf{b} \leftarrow s$, $\mathsf{N} \leftarrow \mathsf{N} \cup \{u_i\}$
12 **return** (b, N)

$\frac{r(v)}{1+\lambda} \leq \frac{c(v)}{1+\lambda} \leq \mathsf{b}_v \leq 2(1+\epsilon)r(v) \leq 2(1+\epsilon)c(v)$.

Lemma III.11 (Invariants Are Maintained)**.** *For the case* $\Lambda = \mathbb{R}$*, after the end of every round of Algorithm 2, the invariants in Definition III.7 are maintained.*

Proof. For the first invariant, consider some node v. By construction, when it calls Update in Algorithm 3, the pair $(\mathsf{b}_v, \mathsf{N}_v)$ returned satisfies $\sum_{e=\{u,v\}:u\in\mathsf{N}_v} w_e \leq \mathsf{b}_v$.

We next consider the second invariant, which is satisfied initially. Suppose t is the first iteration after which the second invariant fails for some edge $\{u, v\}$, i.e., after the iteration t, the auxiliary subsets are such that $u \notin \mathsf{N}_v^t$ and $v \notin \mathsf{N}_u^t$, where we use the superscript t to denote the states of the variables at the end of iteration t.

Same Surviving Numbers in Consecutive Iterations. We next show that $u \notin \mathsf{N}_v^t$ implies that $\mathsf{b}_u^{t-1} \leq b_v^t$. For the execution of v.Update in the t-th iteration, suppose i is the smallest index reached in the **for** loop in line 5 of Algorithm 3. For contradiction's sake, assume that $\mathsf{b}_u^{t-1} > b_v^t = \min\{\mathsf{b}_i, s\}$, where b_i and s are the local variables in line 10. If $\mathsf{b}_i < s$, then we have $\mathsf{b}_u^{t-1} > \mathsf{b}_i$, which implies that $u \in \mathsf{N}_v^t$; if $s \leq \mathsf{b}_i$, then we have $\mathsf{b}_u^{t-1} > s > \mathsf{b}_{i-1}$, which also implies that $u \in \mathsf{N}_v^t$.

The same argument gives us $v \notin \mathsf{N}_u^t \Rightarrow \mathsf{b}_v^{t-1} \leq \mathsf{b}_u^t$. However, since the surviving numbers are monotonically decreasing with the iterations, we must have $b := \mathsf{b}_u^{t-1} = \mathsf{b}_u^t = \mathsf{b}_v^{t-1} = \mathsf{b}_v^t$.

Vertex-Induced Surviving Number. We next show that for the local variables b_i and s defined above, we actually must have $\mathsf{b}_i < s$. Otherwise, we have $s \leq \mathsf{b}_i$, which means $s = \mathsf{b}_v^t = b$ and $\mathsf{b}_{i-1} \geq \mathsf{b}_u^{t-1} = b$. However, we have $s > \mathsf{b}_{i-1}$, which gives a contradiction. Therefore, we have $b = \mathsf{b}_i$.

Reaching Contradiction. Since t is the first iteration such that the second invariant is violated, without loss of generality, we assume that $u \in \mathsf{N}_v^{t-1}$. Since the first invariant holds, we must have $\sum_{x\in\mathsf{N}_v^{t-1}} w_{xv} \leq \mathsf{b}_v^{t-1} = b$. Observe that it is crucial that $\Lambda = \mathbb{R}$, because we need the upper bound b in the inequality, and a looser upper bound will not work.

Suppose in the t-th iteration, in the subroutine v.Update, the index of u among $N(v)$ is i_u after sorting. We next show that any node not in N_v^{t-1} must appear before u in this order. This is enough to get the contradiction, because we must have $\sum_{j=i_u}^{d} w_j \leq b < s$, which means u should have been included in N_v^t.

We next consider what happens in v.Update during $t-1$-st iteration; suppose i' is the smallest index reached in the **for** loop in line 5. Similar to before, consider the local variables $\widehat{\mathsf{b}}_{i'}$ and $\widehat{s}$ in line 10, where we use the widehat notation to distinguish from the local variables defined in the t-th iteration.

The first case is $\widehat{\mathsf{b}}_{i'} < \widehat{s}$, which implies that $\mathsf{b}_v^{t-1} = \widehat{\mathsf{b}}_{i'}$ and $u_{i'} \notin \mathsf{N}_v^{t-1}$. We know that the surviving number b_u^{t-2} of u can drop to $\mathsf{b}_u^{t-1} = b$. However, because of our stable sorting rule, in iteration t's v.Update, u must appear after $u_{i'}$ and any nodes not in N_v^{t-1}.

The second case is $\widehat{s} \leq \widehat{\mathsf{b}}_{i'}$, which implies that $b = \widehat{s} > \widehat{\mathsf{b}}_{i'-1}$. This means that any node not in N_v^{t-1} must have surviving numbers strictly less than b during the sorting of the t-th iteration. Hence, all these nodes must appear before u. This completes the proof. ∎

Corollary III.12 (Approximation for Min-Max Edge Orientation Problem)**.** *For* $\gamma > 2$*, after running Algorithm 2 for* $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$ *rounds, the subsets* $\{\mathsf{N}_v : v \in V\}$ *maintained by the nodes give a* γ*-approximation for the min-max edge orientation problem as defined in Section II.*

Proof. We prove in Lemma III.11 that both invariants in Definition III.7 hold. In particular, the second invariant implies

that the subsets $\{\mathsf{N}_v : v \in V\}$ form a feasible solution, i.e., for each edge $\{u, v\}$, we have $u \in \mathsf{N}_v$ or $v \in \mathsf{N}_u$.

We next prove the approximation ratio. Let ρ^* be the maximum density of a subset in the given graph G. The first invariant implies that for each node v, $\sum_{u \in \mathsf{N}_v} w_{uv} \leq \beta^T(v)$, which, by Lemma III.3, is at most $\gamma \cdot r(v)$. Finally, since the maximal density of every node is at most ρ^*, we have $\sum_{u \in \mathsf{N}_v} w_{uv} \leq \gamma \rho^*$.

By weak LP duality as mentioned in Section II, ρ^* is a lower bound on the optimal value of the min-max edge orientation problem. Therefore, γ-approximation is achieved, as required. ∎

The next lemma shows that the running times of our distributed algorithms are asymptotically tight, while the approximation factor cannot be improved without introducing a linear dependency on the diameter of the underlying graph.

Lemma III.13 (Lower Bound on the Running time)**.** *For $\gamma \geq 2$, any distributed algorithm approximating the coreness values, the maximal densities or the min-max edge orientation problem with an approximation ratio strictly smaller than γ requires $\Omega(\frac{\log n}{\log \gamma})$ communication rounds, where n is the number of nodes in the underlying graph (n sufficiently large). Moreover, any distributed algorithm for the aforementioned problems with an approximation ratio strictly smaller than 2 would require $\Omega(n)$ communication rounds.*

Proof. We prove the lemma for the case of coreness values and min-max edge orientation problem. Since coreness values and maximal densities are within a factor of 2 of each other, a lower bound for one of them implies the same lower bound for the other one.

Without loss of generality, we assume that $\gamma \geq 2$ is an integer. We construct a graph G as follows. Start with a vertex v as a root, and construct a complete γ-ary tree with at least $2\gamma + 1$ leaves. Let n be the number of nodes in the tree, while let $T := \Theta(\frac{\log n}{\log \gamma})$ be the depth of the tree. Since G is a tree, $c_G(v) = 1$, while there is an orientation of the edges with maximum degree equal to one. We next construct another graph G' obtained by planting a clique on the leaves of G. Since every node in G' has degree at least γ, we have $c_{G'}(v) \geq \gamma$. Moreover, there is no orientation of the edges with maximum in-degree $< \gamma$.

Any distributed algorithm for the k-core decomposition or the min-max assignment problem with an approximation ratio $< \gamma$ must allow the root v to distinguish between G and G'. Hence, at least T rounds are needed.

Finally, Figure I.1 shows that any distributed algorithm with an approximation ratio strictly smaller than 2 requires $\Omega(n)$ communication rounds. ∎

IV. Distributed Algorithm for Approximate Densest Subset Problem

In Section III, we gave a distributed algorithm that approximates the coreness and the maximal density of each node. We next consider a distributed algorithm to return an approximate densest subset. In the distributed setting, this means at the end, each node should know whether it is contained in the approximate solution. However, if we restrict the round complexity to be independent of the hop-diameter of the graph, then it is impossible for a node to know whether there is some other much denser subset that is many hops away. Therefore, we define the following notion of distributed approximation of the densest subset problem.

Definition IV.1 (Distributed Approximation Algorithm for (Weak) Densest Subset Problem)**.** *For $\gamma \geq 1$, a distributed algorithm achieves γ-approximation for the densest subset problem, if after the algorithm terminates, there exists a collection of disjoint subsets $\{S_i\}_{i \in \mathcal{I}}$ of nodes such that the following hold:*

- *For each $i \in \mathcal{I}$, every node in S_i knows that it belongs to S_i (and also the density of S_i). To be specific, each S_i will have some vertex $v_i \in S_i$ as its* leader*, and every node in S_i knows the identity of the leader v_i (but might not know who else is in S_i).*
- *There exists some $i \in \mathcal{I}$ such that the density $\rho(S_i) \geq \frac{\rho^*}{\gamma}$, where ρ^* is the maximum density of a subset in the input graph.*

Suppose we fix the approximation ratio $\gamma > 2$. In view of Lemma III.3, we set $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$. The distributed algorithm consists of several phases.

Phase 1: Approximating the Maximal Density. In this phase, Algorithm 2 is run for T rounds. After that, each node $v \in V$ knows some number b_v, which is a γ-approximation of its maximal density.

Phase 2: Building BFS Trees. In this phase, each node will try to identify a leader v who is the node within T hops with the largest value b_v. Moreover, for each potential leader v, a breadth-first-search (BFS) tree rooted at v with depth at most T is constructed in Algorithm 4. To resolve ties among nodes with the same b_v value, we assume that there is a global ordering $\succ$ on V that is known by every node. This induces a total ordering on $\{(v, \mathsf{b}_v) : v \in V\}$ defined by $(u, \mathsf{b}_u) \succ (v, \mathsf{b}_v)$ *iff* (i) $\mathsf{b}_u > \mathsf{b}_v$, or (ii) $\mathsf{b}_u = \mathsf{b}_v$ and $u \succ v$.

Fact IV.2. *Suppose node v has the maximum value b_v and is also the maximum under the ordering $\succ$. Then, Algorithm 4 correctly constructs the BFS tree rooted at v that includes all nodes within T hops from v in the original input graph G.*

Remark IV.3. *Observe that if $v = \mathsf{parent}[v]$, then v is the root of some BFS tree. Moreover, every node u in this BFS tree has $\mathsf{leader}[u] = (v, \mathsf{b}_v)$.*

Phase 3: Elimination Procedure within Each BFS Tree. In this phase, each node v communicates only with its $\mathsf{parent}[v]$ and $\mathsf{children}[v]$, and ignores all other nodes. Within each BFS, all nodes have a common $\mathsf{leader} = (u, \mathsf{b}_u)$. The elimination procedure in Algorithm 1 is run with threshold value b_u from the leader. However, in the augmented Algorithm 5, each node remembers its weighted degree for every iteration, which is later used to determine an approximate densest subset.

Algorithm 4: BFS Construction

Input: Each node $v \in V$ has some b_v; number T
Output: Each node v knows some potential $\mathsf{leader}[v] \in \{(u, \mathsf{b}_u) : u \in V\}$, a potential $\mathsf{parent}[v] \in V$, and a potential subset $\mathsf{children}[v] \subseteq V$.

Initially, each node $v \in V$ sets $\mathsf{leader}[v] \leftarrow (v, \mathsf{b}_v)$, $\mathsf{parent}[v] \leftarrow v$, $\mathsf{children}[v] \leftarrow \emptyset$.
for *each round* $t \in [1..T]$ **do**
  Each node v broadcasts its current $\mathsf{leader}[v]$ to all its neighbors.
  After node v receives messages from all its neighbors $N(v)$, suppose $u \in N(v)$ is the neighbor whose $\mathsf{leader}[u]$ is the maximum in the ordering $\succ$. Then, node v performs the following:
  if $\mathsf{leader}[u] \succ \mathsf{leader}[v]$ **then**
    $\mathsf{leader}[v] \leftarrow \mathsf{leader}[u]$, $\mathsf{parent}[v] \leftarrow u$
Request Parent. Each node v sends a request message containing $\mathsf{leader}[v]$ to the node $\mathsf{parent}[v]$, if $\mathsf{parent}[v] \neq v$.
Include Children. For node v, after receiving the request message from each node u, it checks whether $\mathsf{leader}[u] = \mathsf{leader}[v]$; if so, it adds $\mathsf{children}[v] \leftarrow \mathsf{children}[v] \cup \{u\}$.
Confirm Parent. If a node v has $\mathsf{parent}[v] \neq v$ and does not receive an acknowledge message, then it sets $\mathsf{parent}[v] = \perp$.
return *Each node* v *has computed* $\mathsf{leader}[v]$, $\mathsf{parent}[v]$ *and* $\mathsf{children}[v]$.

Moreover, we optimize the algorithm such that a node does not participate in the protocol once it is eliminated.

After this phase, each node v computes two arrays $\mathsf{num}_v[0..T-1]$ and $\mathsf{deg}_v[0..T-1]$. For each $t \in [0..T-1]$, $\mathsf{num}_v[t] \in \{0, 1\}$ indicates whether v survives after iteration t; if $\mathsf{num}_v[t] = 1$, then $\mathsf{deg}_v[t]$ gives the corresponding weighted degree. Lemma IV.4 gives some ideas on how these results can help to find an approximate densest subset.

Algorithm 5: Augmented Elimination Procedure

Input: Each node v has $\mathsf{leader}[v]$, neighbors $N(v) = \mathsf{parent}[v] \cup \mathsf{children}[v]$, number $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$ of rounds. Recall that all nodes in the same BFS have same value b in their leader.
Output: Each node v computes two arrays $\mathbf{num}_v[0..T-1]$ and $\mathbf{deg}_v[0..T-1]$, where $\mathbf{num}_v[t] \in \{0, 1\}$ indicates whether v survives after iteration t, and $\mathbf{deg}_v[t]$ indicates the corresponding weighted degree.

Initially, each node v is active $\sigma_v \leftarrow 1$, and initializes the arrays $\mathbf{num}_v := \mathbf{deg}_v \leftarrow \mathbf{0}$.
for *each round* $t \in [1..T]$ **do**
  Each active node v (with current $\sigma_v = 1$) broadcasts to all its neighbors $N(v)$ that it is still active.
  After hearing from all its (active) neighbors in $N(v)$, each active node v performs the following:
  $\mathbf{num}_v[t-1] \leftarrow 1$, $\mathbf{deg}_v[t-1] \leftarrow \sum_{e=\{u,v\}:\sigma_u=1} w_e$
  if $\mathbf{deg}_v[t-1] < b$ **then**
    $\sigma_v \leftarrow 0$
    Node v becomes inactive, and stops participating in the algorithm.
return *Each node* v *has computed the arrays* $\mathbf{num}_v$ *and* $\mathbf{deg}_v$.

Lemma IV.4 (Surviving Nodes in Some Iteration Give Approximate Densest Subset)**.** *Let* $\gamma > 2$*, and* $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$*. Suppose the augmented elimination procedure in Algorithm 5 is run for* T *rounds on the depth-*T *BFS rooted at some node* u *with threshold value* $\mathsf{b}_u = \beta_G^T(u)$ *(which is the value returned by running Algorithm 1 on the original graph* G *for* T *rounds).*

Suppose in Algorithm 5, for $t \in [0..T]$*,* A_t *is the set of surviving nodes at the end of round* t*. Then, there exists some* $t \in [0..T-1]$ *such that the density of* A_t *is at least* $\frac{\mathsf{b}_u}{\gamma}$*.*

Proof. The proof uses the same idea as in Lemma III.3 (which is inspired from [4], [8], [17]).

For $t \in [0..T-1]$, the density $\rho(A_t) = \frac{\sum_{v \in A_t} \mathsf{deg}_v[t]}{2 \cdot |A_t|} \geq \frac{\mathsf{b}_u \cdot |A_{t+1}|}{2 \cdot |A_t|}$.

Therefore, $\prod_{t \in [0..T-1]} \rho(A_t) \geq (\frac{\mathsf{b}_u}{2})^T \cdot \frac{|A_T|}{|A_0|}$. Observe that since threshold b_u is used, the node u still survives after round T. Therefore, we have $|A_T| \geq 1$ and $|A_0| \leq n$.

Hence, there exists some $t \in [0..T-1]$ such that $\rho(A_t) \geq \frac{\mathsf{b}_u}{2} \cdot \frac{1}{n^{1/T}} \geq \frac{\mathsf{b}_u}{\gamma}$, where the last inequality follows because $T \geq \frac{\log n}{\log(\gamma/2)}$. ∎

Phase 4: Aggregation and Finding Approximate Densest Subset. In view of Lemma IV.4, one should compute $\arg\max_{t \in [0..T-1]} \rho(A_t)$, where the density satisfies $\rho(A_t) = \frac{\sum_v \mathsf{deg}_v[t]}{2 \cdot \sum_v \mathsf{num}_v[t]}$ and the summation is over all nodes in the BFS. The details are given in Algorithm 6.

Optimizing Message Size. During aggregation, a node v sends its length-T aggregated arrays $(\mathsf{num}'_v, \mathsf{deg}'_v)$ to its parent. Since the depth of the BFS tree is at most T, the aggregation part takes T rounds, but the size of each message contains $\Theta(T)$ words. To reduce the message size, the entries of the arrays can be sent to the parent in a pipelined fashion. For instance, one entry from each of the two arrays are sent per message to the parent, and the number of rounds increases by T.

Corollary IV.5 (Correctness)**.** *Algorithm 6 gives* γ*-approximation to the distributed densest subset problem as in Definition IV.1.*

V. Conclusion and Future Directions

We have shown that the well-known elimination procedure, when implemented in a distributed setting, provides a constant approximation to both the coreness values and maximal densities. The asymptotically round complexity is tight and independent of the diameter. Moreover, by augmenting the elimination procedure, the min-max edge orientation problem can also be approximated with the same theoretical ratio.

Empirical results[3] on real-world graphs show that the approximation ratio often converges to 2 much quicker than what the worst-case analysis suggests. Are there any special properties that can explain this phenomenon? The theoretical lower bound on round complexity applies to the worst case approximation ratio over all nodes. Can one improve the

[3]Some experimental results are included in the full version.

Algorithm 6: Aggregation and Finding Approximate Densest Subset

Input: Each node v has $\mathsf{leader}[v]$, $\mathsf{parent}[v]$, $\mathsf{children}[v]$ and length-T arrays num_v and deg_v, where $T := \lceil \frac{\log n}{\log(\gamma/2)} \rceil$.

Output: Each node v computes $\sigma_v \in \{0, 1\}$. If $\sigma_v = 1$, then node v knows that it belongs to the subset with leader specified in $\mathsf{leader}[v]$.

Initially, each node v sets $\sigma_v \leftarrow 0$.

Aggregate. If a node v has $\mathsf{children}[v] = \emptyset$ and $\mathsf{parent}[v] \neq v$, then it sends its arrays $(\mathsf{num}'_v = \mathsf{num}_v, \mathsf{deg}'_v = \mathsf{deg}_v)$ to $\mathsf{parent}[v]$.

for *each node* v **do**

  After node v has received arrays from all nodes in $\mathsf{children}[v]$, it aggregates the arrays:

  $\mathsf{num}'_v \leftarrow \mathsf{num}_v + \sum_{x \in \mathsf{children}[v]} \mathsf{num}'_x$

  $\mathsf{deg}'_v \leftarrow \mathsf{deg}_v + \sum_{x \in \mathsf{children}[v]} \mathsf{deg}'_x$

  if $\mathsf{parent}[v] \neq v$ **then**

    Node v sends its aggregated arrays $(\mathsf{num}'_v, \mathsf{deg}'_v)$ to $\mathsf{parent}[v]$.

Identify Densest Subset. After a root node v (i.e., $\mathsf{parent}[v] = v$ and $\mathsf{leader}[v] = (v, \mathsf{b}_v)$ for some b_v) has computed its aggregated arrays num'_v and deg'_v, it computes $\mathsf{b}_{\max} \leftarrow \max_{t \in [0..T-1]} \frac{\mathsf{deg}'_v[t]}{2 \cdot \mathsf{num}'_v[t]}$.

if $\mathsf{b}_{\max} \geq \mathsf{b}_v$ **then**

  The root node v sets $\sigma_v \leftarrow 1$, and computes $t^* \leftarrow \arg\max_{t \in [0..T-1]} \frac{\mathsf{deg}'_v[t]}{\mathsf{num}'_v[t]}$.

  Node v sends a message containing t^* to each node in $\mathsf{children}[v]$, and terminates.

for *each node* v **do**

  Upon receiving t^* from its $\mathsf{parent}[v]$, node v performs the following:

  if $\mathsf{num}_v[t^*] = 1$ **then**

    Node v sets $\sigma_v \leftarrow 1$.

  Node v sends a message containing t^* to each node in $\mathsf{children}[v]$, and terminates.

Even if a node does not hear back from its parent, it terminates after the algorithm is run for $3T$ rounds.

return *Each node* v *has computed* σ_v.

theoretical upper/lower bounds on the round complexity when average approximation ratio over all nodes is considered?

REFERENCES

[1] Sabeur Aridhi, Martin Brugnara, Alberto Montresor, and Yannis Velegrakis. Distributed k-core decomposition and maintenance in large dynamic graphs. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16, Irvine, CA, USA, June 20 - 24, 2016*, pages 161–168, 2016.

[2] Yuichi Asahiro, Jesper Jansson, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Approximation algorithms for the graph orientation minimizing the maximum weighted outdegree. *Journal of combinatorial optimization*, 22(1):78–96, 2011.

[3] Yuichi Asahiro, Eiji Miyano, Hirotaka Ono, and Kouhei Zenmyo. Graph orientation algorithms to minimize the maximum outdegree. In *Proceedings of the 12th Computing: The Australasian Theroy Symposium-Volume 51*, pages 11–20. Australian Computer Society, Inc., 2006.

[4] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012.

[5] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.

[6] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[7] T.-H. Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *J. ACM*, 65(3):15:1–15:48, 2018.

[8] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[9] James Cheng, Yiping Ke, Shumo Chu, and M Tamer Özsu. Efficient core decomposition in massive networks. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 51–62. IEEE, 2011.

[10] Andrzej Czygrinow, Michal Hańćkowiak, Edyta Szymańska, and Wojciech Wawrzyniak. Distributed 2-approximation algorithm for the semi-matching problem. In *International Symposium on Distributed Computing*, pages 210–222. Springer, 2012.

[11] Maximilien Danisch, T.-H. Hubert Chan, and Mauro Sozio. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 233–242, 2017.

[12] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient densest subgraph computation in evolving graphs. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 300–310, 2015.

[13] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. D-cores: Measuring collaboration of directed graphs based on degeneracy. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 201–210. IEEE, 2011.

[14] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 87–93. IEEE, 2011.

[15] Noël Gillet and Nicolas Hanusse. A fully asynchronous and fault tolerant distributed algorithm to compute a minimum graph orientation. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 308–322. Springer, 2017.

[16] Fabrizio Grandoni, Jochen Könemann, Alessandro Panconesi, and Mauro Sozio. A primal-dual bicriteria distributed algorithm for capacitated vertex cover. *SIAM J. Comput.*, 38(3):825–840, 2008.

[17] Shuguang Hu, Xiaowei Wu, and T.-H. Hubert Chan. Maintaining densest subsets efficiently in evolving hypergraphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 929–938, 2017.

[18] Ravi Kannan and V Vinay. *Analyzing the structure of large graphs*. Rheinische Friedrich-Wilhelms-Universität Bonn Bonn, 1999.

[19] Samir Khuller and Barna Saha. On finding dense subgraphs. In *ICALP (1)*, volume 5555 of *Lecture Notes in Computer Science*, pages 597–608. Springer, 2009.

[20] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888, 2010.

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

[22] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[23] Alberto Montresor, Francesco De Pellegrini, and Daniele Miorandi. Distributed k-core decomposition. *IEEE Trans. Parallel Distrib. Syst.*, 24(2):288–300, 2013.

[24] Atish Das Sarma, Ashwin Lall, Danupon Nanongkai, and Amitabh Trehan. Dense subgraphs on dynamic networks. In *DISC*, volume 7611 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2012.

[25] Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[26] Nikolaj Tatti and Aristides Gionis. Density-friendly graph decomposition. In *WWW*, pages 1089–1099, 2015.

[27] Venkat Venkateswaran. Minimizing maximum indegree. *Discrete Applied Mathematics*, 143(1-3):374–378, 2004.

[28] Dong Wen, Lu Qin, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. I/O efficient core graph decomposition at web scale. In *32nd IEEE*

International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016, pages 133–144, 2016.

[29] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.*, 42(1):181–213, 2015.