Improving the Core Resilience of Real-world Hypergraphs: Concepts, Observations, and Algorithms (Online Appendix)

[If a preview does not appear properly, please download the file]

In this document, we present additional contents for the main manuscript entitled *"Improving the Core Resilience of Real-world Hypergraphs: Concepts, Observations, and Algorithms"*. These include the empirical results of node-deletion attacks, when attackers remove a portion of the nodes along with their incident hyperedges, as opposed to removing hyperedges (presented in the main manuscript).Furthermore, we also provide the proofs for the theoretical results presented in the main manuscript.

I. Experimental results on Node deletions

While we consider attacks of hyperedge deletions in the main manuscript, another setting is when attackers deliberately remove a portion of the nodes along with their incident hyperedges. In this section, we consider such scenarios, by which we examine how different schemes of node-deletion attacks affect the core resilience of the networks and how different methods help improve the core resilience.

A. Core Resilience against Node-deletion Attacks

Attack Schemes: We introduce different schemes s of node deletion attacks $\mathsf{A}(r, s)$.

- *Random Attack*: $r\%$ of the nodes, along their incident hyperedges, are deleted randomly.
- *Degree Atttack*: Among the nodes, high-degree nodes are targeted, and the chance for a node v to be deleted is proportional to its degree.
- *Core Attack*: Among the nodes, high-core nodes are targeted, and the chance for a node v to be deleted is proportional to its core number, found by Algorithm 2.

Figure 1 shows the core resilience of real-world hypergraphs against the node-deletion attack schemes across deletion ratios. Core Attack results in the lowest core resilience per deletion ratio in most cases, while Random Attack results in the highest core resilience.

B. Empirical evaluation of CoReA

The methods introduced in the main paper are evaluated on how they improve the core resilience of the hypergraphs against node deletions. The attack scheme considered here is *Core Attack*. The budget for adding hyperedges is $5\% \times |E|$. $r\% \times |E|$ ($r = 5, 10, 15, 20, 25$) hyperedges are deleted according to *Core Attack*. We report the performance of each method in terms of the improvement of core resilience, in the same manner as the main manuscript.

The comparison of different methods in core resilience improvement across deletion ratios is in Figure 2. CoReA outperforms other methods in all datasets. This result is consistent with the result presented in the main paper, where the methods are evaluated against hyperedge-deletion attacks.

This empirically shows that CoReA is effective in improving the core resilience of real-world hypergraphs across several domains and consistently superior to the baselines in both node-deletion and hyperedge-deletion attacks.

II. Algorithms for CoReA

For the convenience of readers, we include the algorithm pseudocodes presented in the main paper:

Algorithm 1 outlines CoReA.

Algorithm 2 outlines Step 1-1 of CoReA, which follows along the core decomposition process and compute the anchor availabilities of the nodes in $\mathbf{V}$.

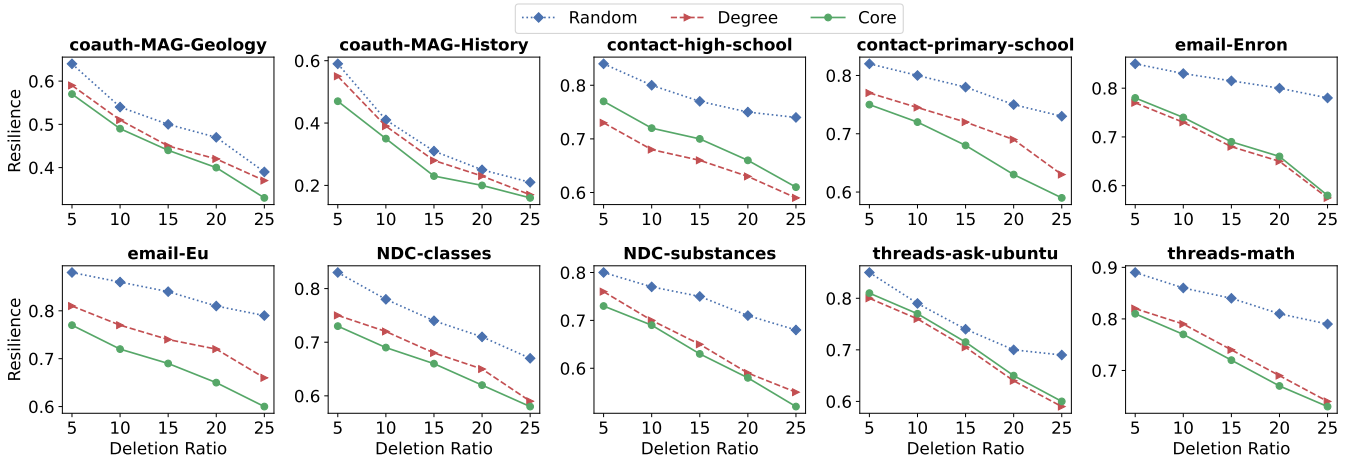

Fig. 1: Core resilience of real-world hypergraphs against node-deletion attack schemes across deletion ratios. Core Attack is the most destructive to the core resilience of most of the the hypergraphs, while Random Attack is consistently the least destructive.

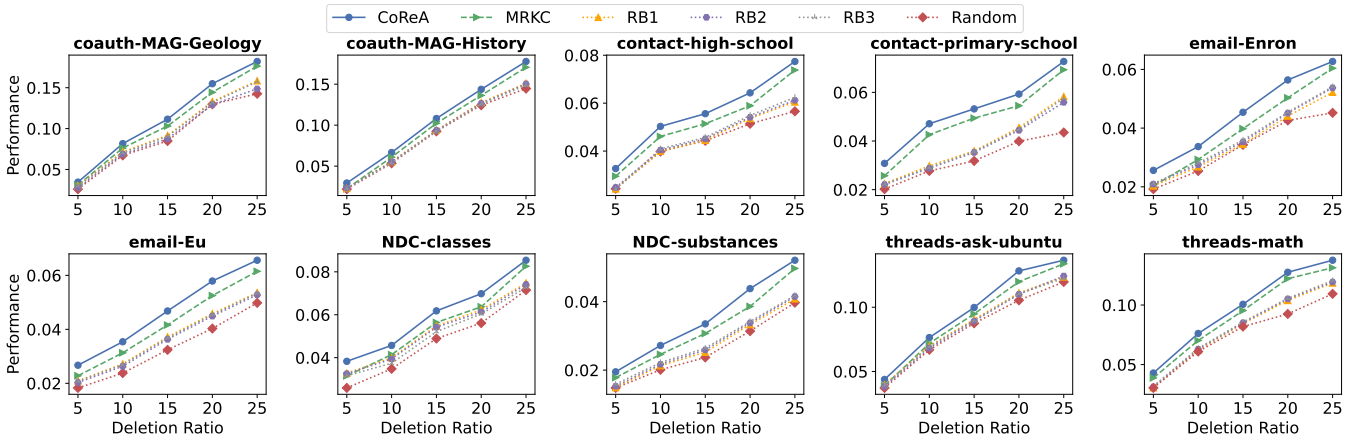

Fig. 2: CoREA consistently brings better improvement of core resilience than the others in all datasets across deletion ratios.

III. Time Complexity of CoREA

In this section, we prove the time complexity of CoREA.

Lemma 1. Given the hypergraph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with maximum hyperedge cardinality m. With budget b and the total anchor availabilities $\mathcal{C} = \sum_{v \in \mathbf{V}} c(v)$, which is constant with respect to each dataset, the time complexity of CoREA is $\mathcal{O}\left[(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e| + \mathcal{C}m^2)b\right]$.

Proof. As described in **Section 4.1**, computing the core influences of all nodes requires initializing the value 1 for each node and enumerating through each node in each hyperedge once, so the time complexity of computing core influences is $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e|)$.

Step 1-1 of CoREA, presented in **Algorithm 2**, undertakes the core decomposition process and computes the anchor availability of each node. This step requires enumerating through each node v for its removal and each hyperedge e for its removal and updating the degrees of its constituent nodes. Therefore, the time complexity of this step is $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e|)$. Computing the core strength of each node v, $N_{\mathbf{G}}(v) = k$, requires some primitive operations (subtracting k from the degree at the start of the k-core then adding 1), so the time complexity of operating this step for all nodes is $\mathcal{O}(|\mathbf{V}|)$, which is dominated by $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e|)$.

Regarding the case when the tie-breaking scheme T is $\mathcal{CS}/\mathcal{CI}$ (or $1/\mathcal{CI}$), the time complexity is still the same. The reason is that the core influences of all the nodes in the k-core can be computed by the time **Algorithm 2** completes finding the $(k-1)$-core (the core influence of v only depends on the hyperedges incident to v having lower core numbers than that of v). Also, when a node v becomes qualified for removal in **Algorithm 2**, its core strength can be updated (based on its degree at the beginning of the k-core and its core number, which is determined to be k at this point already). Therefore, computing core strengths and core influences for the scheme T does not affect the time complexity. In brief, the time complexity of Step 1-1 of CoREA is $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e|)$.

Algorithm 1: Overview of CoREA

Input: (1) input hypergraph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, no isolated nodes,
(2) budget b, (3) hypereedge size distribution D,
(4) tie-breaking scheme T, (5) sampling scheme S
Output: augmented hypergraph $\mathbf{G}' = (\mathbf{V}, \mathbf{E}')$

1 /* Step 1-1: compute anchor availabilities */
2 Given hypergraph $\mathbf{G}$ and tie-breaking scheme T,
3 obtain core numbers $\{N_{\mathbf{G}}(v)|v \in \mathbf{V}\}$ of nodes,
4 anchor availabilities $\{c(v)|v \in \mathbf{V}\}$ of nodes,
5 core numbers $\{\overline{N_{\mathbf{G}}}(e)|e \in \mathbf{E}\}$ of hyperedges,
6 removal order $\mathbb{O}$ of nodes, degeneracy $N^*_{\mathbf{G}}$ of $\mathbf{G}$,
7 and k-core $\mathbf{C}(k, \mathbf{G})$ of $\mathbf{G}$ for $k = 1, ..., N^*_{\mathbf{G}}$; ▷ Algorithm. 2
8 /* Step 1-2: build a pool P of candidate hyperedges */
9 Initialize pool of candidate hyperedges: $P \leftarrow \{\}$
10 **for** (i, v) in enumerate($\mathbb{O}$) **do**
11 **for** $j = 1, .., c(v)$ **do**
12 $e \leftarrow$ empty hyperedge, add v to e
13 Sample a hyperedge size $s \sim \mathsf{D}$
14 Sample $(s-1)$ nodes from $\mathbb{O}[i+1:]$ to fill up e by S.
15 $P \leftarrow P \cup \{e\}$
16 **end**
17 **end**
18 /* Step 2: select the best hyperedges from P */
19 $\mathbf{E}' \leftarrow \mathbf{E}, \bar{b} \leftarrow b$
20 **while** $\bar{b} > 0$ **do**
21 **for** $e \in P$ **do**
22 $s(e) \leftarrow$increase in $\sum_{v \in \mathbf{V}} \mathcal{CI}(v)\mathcal{CS}(v)$ if e is augmented
23 **end**
24 choose the hyperedge e in P of the highest score $s(e)$
25 $P \leftarrow P \setminus \{e\}, \mathbf{E}' \leftarrow \mathbf{E} \cup \{e\}$
26 update $\mathcal{CI}(v), \mathcal{CS}(v) \forall v \in \mathbf{V}$; $\bar{b} \leftarrow \bar{b} - 1$
27 **end**
28 **return** $\mathbf{G} = (\mathbf{V}, \mathbf{E}')$

In Step 1-2 of CoREA, for each node v, we need to construct $c(v)$ hyperedges anchored at v. For each hyperedge e among those $c(v)$ hyperedges, this requires sampling a hyperedge size (constant time) and sampling other nodes from $\mathbb{O}[i+1:]$ (as shown in Line 14 of **Algorithm 1**). The sampling step of other nodes to fill up e is $\mathcal{O}(m \log_2 m)$, according to [1]. Therefore the total time complexity of Step 1-2 is $\mathcal{O}(\sum_{v \in \mathbf{V}} c(v) m \log_2 m) = \mathcal{O}(\mathcal{C} m \log_2 m)$.

In Step 2, before choosing each hyperedge to augment to $\mathbf{G}$, for each candidate hyperedge e in the pool P, CoREA needs to evaluate how much augmenting e improves the term $f(\mathbf{G}) = \sum_{v \in \mathbf{V}} \mathcal{CI}(v)\mathcal{CS}(v)$. To do this, we maintain a measurement $g(v)$ for each node v, quantifying how much $f(\mathbf{G})$ increases if $\mathcal{CI}(v)$ is incremented by 1 unit. Particularly, if $\mathcal{CI}(v)$ increases by 1 unit, $f(\mathbf{G})$ increases by $g(v)$. In order to achieve this, we reverse the process of calculating all core influences. In the formula of core influence in **Section 4.1**, suppose $\mathcal{CI}(v) = 1 + \sum_{e \in \mathbf{E}_<(v)} (1 + \frac{\Delta}{N_{\mathbf{G}}(v)-1}) \left[(1 - \frac{\mathcal{CS}(t)-1}{|\mathbf{E}_=(t)|}) \mathcal{CI}(t) \right]$, for each $e \in \mathbf{E}_<(v)$, if $\mathcal{CI}(t)$ increases by 1 unit, $\mathcal{CI}(v)$ increases by $(1 + \frac{\Delta}{N_{\mathbf{G}}(v)-1}) \left[(1 - \frac{\mathcal{CS}(t)-1}{|\mathbf{E}_=(t)|}) \right]$ units. As a result, $g(t)$ needs to increase by $(1 + \frac{\Delta}{N_{\mathbf{G}}(v)-1}) \left[(1 - \frac{\mathcal{CS}(t)-1}{|\mathbf{E}_=(t)|}) \right] g(v)$ units. To compute such value $g(v)$ for each node v, we first initialize $g(v) = \mathcal{CS}(v)$, start from the nodes with the highest core number, update the values $g(.)$ until reaching the nodes with the lowest core number. The whole process requires enumerating through each node once and each node in each hyperedge once, accounting for the total time complexity of $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e|)$.

Once the values $g(.)$ are up-to-date, for each candidate hyperedge e anchored at $\{v_1, ..., v_a\}$, and the other nodes in e that are not anchors of e are $\{u_1, ...u_b\}$. Suppose adding e increases the core influences of $u_1, ...u_b$ by $\beta_1, ..., \beta_b$, respectively, which can be calculated in $\mathcal{O}(|e|^2)$ time that is upper-bounded by $\mathcal{O}(m^2)$. The contribution of e into $f(\mathbf{G})$ if augmented is then: $\sum_{i=1}^{a} \mathcal{CI}(v_i) + \sum_{j=1}^{b} \beta_j \times g(u_j)$, which can be calculated in $\mathcal{O}(m)$ time. The time complexity of calculating the score for each candidate hyperedges and choosing the best one is then $\mathcal{O}(\mathcal{C}m^2)$. Once we augment 1 hyperedge into $\mathbf{G}$, we need to update all core strengths, core influences, and the values $g(.)$, which takes $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e| + m)$ time in total. Thus, the total time complexity of 1 iteration of scoring the candidate hyperedges, selecting the best one, and updating the statistics is $\mathcal{O}(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e| + \mathcal{C}m^2)$. As CoREA needs to run b such iterations, the total time complexity of Step 2 is $\mathcal{O}\left[(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e| + \mathcal{C}m^2)b\right]$.

Summing up the time complexities of Steps 1-1, 1-2, and 2, the total time complexity of CoREA is $\mathcal{O}\left[(|\mathbf{V}| + \sum_{e \in \mathbf{E}} |e| + \mathcal{C}m^2)b\right]$.

Algorithm 2: Compute anchor availabilities

Input: (1) input hypergraph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, no isolated nodes,
(2) tie-breaking scheme T
Output: (1) core numbers $\{N_{\mathbf{G}}(v)|v \in \mathbf{V}\}$ of nodes in $\mathbf{V}$,
(2) anchor availabilities $\{c(v)|v \in \mathbf{V}\}$ of nodes in $\mathbf{V}$,
(3) core numbers of hyperedges $\{\overline{N_{\mathbf{G}}}(e)|e \in \mathbf{E}\}$ in $\mathbf{E}$,
(4) removal order $\mathbb{O}$ of nodes,
(5) degeneracy $N^*_{\mathbf{G}}$, k-core $\mathbf{C}(k, \mathbf{G})$ of $\mathbf{G}$ for $k = 1, ..., N^*_{\mathbf{G}}$

$\overline{\mathbf{V}} \leftarrow \mathbf{V}$, $\overline{\mathbf{E}} \leftarrow \mathbf{E}$, $\overline{\mathbf{G}} \leftarrow (\overline{\mathbf{V}}, \overline{\mathbf{E}})$, $C(1, G) \leftarrow \overline{\mathbf{G}}$, $\mathbb{O} \leftarrow$ empty queue, $k \leftarrow 1$
while $\overline{\mathbf{V}}$ is not empty **do**
  $\mathbb{TD} \leftarrow \{v \in \overline{\mathbf{V}} | d_{\overline{\mathbf{G}}}(v) < k + 1\}$
  while TD is not empty **do**
    pop v from $\mathbb{TD}$ by according to T, add v to $\mathbb{O}$
    $N_{\mathbf{G}}(v) \leftarrow k$, $\overline{\mathbf{V}} \leftarrow \overline{\mathbf{V}} \setminus \{v\}$, $c(v) \leftarrow k - d_{\overline{\mathbf{G}}}(v)$
    for $e \in \overline{\mathbf{E}}_{\overline{\mathbf{G}}}(v)$ **do**
      for $n \in e$ **do**
        $d_{\overline{\mathbf{G}}}(n) \leftarrow d_{\overline{\mathbf{G}}}(v) - 1$
        if $d_{\overline{\mathbf{G}}}(v) < k + 1$ **then**
          $\mathbb{TD} \leftarrow \mathbb{TD} \cup \{n\}$
        end
      end
      $\overline{N_{\mathbf{G}}}(e) \leftarrow k$
      $\overline{\mathbf{E}} \leftarrow \overline{\mathbf{E}} \setminus \{e\}$
    end
  end
  $\mathbf{C}(k, \mathbf{G}) = (\overline{\mathbf{V}}, \overline{\mathbf{E}})$
  $k \leftarrow k + 1$
end
$N^*_{\mathbf{G}} \leftarrow k - 1$
return $\{N_{\mathbf{G}}(v)|v \in \mathbf{V}\}$, $\{c(v)|v \in \mathbf{V}\}$, $\{\overline{N_{\mathbf{G}}}(e)|e \in \mathbf{E}\}$, $\mathbb{O}$, $N^*_{\mathbf{G}}$, $\{\mathbf{C}(k, \mathbf{G})|k = 1, ..., N^*_{\mathbf{G}}\}$

□

IV. Correctness of CoREA

Theorem 1 (Correctness of CoREA)**.** Step 1 of CoREA guarantees to construct a pool P of candidate hyperedges that do not change the core number of any node when they are added together to $\mathbf{G}$.

Proof. We show that for each candidate hyperedge e from P, after e is augmented to $\mathbf{G}$, the deletion order $\mathbb{O}$ in Algorithm 2 is still preserved and returns the original core numbers.

Indeed, suppose e is one among the $c(v)$ hyperedges, with $c(v)$ as the anchor availability afforded to node v of core number k by CoREA, formed by grouping v with other nodes from $\mathbb{O}[i+1:]$ (Line 14 of Algorithm 1). By this configuration, e does not affect the core numbers of the nodes that have been deleted before v in $\mathbb{O}$.

In addition, following the removal order $\mathbb{O}$ in the pruning process of obtaining the $(k+1)$-core from the k-core, without any augmentation, the degree of v prior to its removal is $k - c(v)$, so augmenting $c(v)$ hyperedges anchored at v increases the degree of v to k. Therefore, v is still qualified for removal after such augmentation. Such removal deletes v and all of its incident hyperedges, including e. As a result, augmenting e does not alter the core number k of v.

Because e is removed along with v, the augmentation of e into $\mathbf{G}$ does not affect the nodes after v in $\mathbb{O}$, or $\mathbb{O}[i+1:]$.

Therefore, the augmentation of e still preserves the deletion order $\mathbb{O}$ as well as all of the core numbers. As this claim is true for every candidate hyperedge in P, Step 1 of CoREA guarantees to construct a pool P of candidate hyperedges that do not change the core number of any node when they are added together to $\mathbf{G}$. □

Corollary 1. Assuming F is a valid augmentation of hyperedges into $\mathbf{G}$, an augmentation that preserves all core numbers, and applying F to $\mathbf{G}$, a feasible order of node deletions for the nodes in the k-core shell is: $S_k = \{a_1, ..., a_n\}$. Without F, S_k is still a feasible order of node deletions for the nodes in the k-core shell in the pruning process of obtaining the $(k+1)$-core.

Proof. Assuming the number of hyperedges augmented by F anchored at each node $a_1, ..., a_n$ are $F(a_1), ..., F(a_n)$, respectively.

Considering the hypergraph after applying the augmentation F. During the pruning process of obtaining the $(k+1)$-core, immediately prior to the deletion of node a_i, the degree of a_i must be lower than or equal to k. Denote the degree of a_i at this point as $k - c(a_i) \leq k$ with $c(a_i) \geq 0$. If the order S_k is followed without the augmentation F, the degree of a_i at this point would be: $k - c(a_i) - F(a_i) \leq k$, which qualifies a_i for deletion.

A similar argument can be generalized to all nodes in the sequence $a_1, ..., a_n$. This proves that the order S_k is still a feasible order of node deletions for the nodes in the k-core shell in the pruning process of obtaining the $(k+1)$-core without F. □

V. Invariance of CoREA

Lemma 2. Let $\mathbb{S} = \{a_1, ..., a_n\}$ be a set of n elements, $\mathcal{F}$ as the set of all subsets of S , and $t : \mathcal{F}(\mathbb{S}) \mapsto \mathbb{N}$ be a function that maps a subset of $\mathbb{S}$ to a natural number. Denote $S^{(i)} \in \mathcal{F}(\mathbb{S})$ as the set of all subsets of $\mathbb{S}$ contaning the element a_i. Then:

$$\sum_{i=2}^{n} \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s) = \sum_{s \subset S, |s| \geq 2} (|s| - 1) t(s). \tag{1}$$

Proof. It suffices to show that each term $t(s)$ for each $s \subset S, |s| \geq 2$, appears exactly $(|s| - 1)$ times on the left-hand side of the equality.

Indeed: the set $\bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})$ is the set of all subsets of $\mathbb{S}$ that contain a_i and at least 1 element among $a_1, ..., a_{i-1}$.

Let $s = \{a_{k_1}, ..., a_{k_m}\}$ with $k_1 < ... < k_m$ and $|s| = m$. For each $i = k_2, ..., k_m$, s appears exactly once in $s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})$ because for each of those $i = k_2, ..., k_m$, the set s is a subset of $\mathbb{S}$ that contains a_i and $a_{k_1} (k_1 < i)$.

For each $i \in \mathbb{S} \setminus \{k_2, ..., k_m\}$, s does not appear in $\bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})$ as s fails to contain both a_i and an element $a_j (j < i)$.

Therefore, the term $t(s)$ corresponding to the set s appears exactly $(|s| - 1)$ times on the left-hand side of equation 1. In other words, the two sides of equation 1 are equal to each other. □

Theorem 2 (Invariance of CoREA)**.** The total number of anchor availabilities $\mathcal{C} = \sum_{v \in \mathbf{V}} c(v)$ realized by CoREA is always constant.

Proof. In order to show that the total anchor availabilities realized by CoREA is the same regardless of the order $\mathbb{O}$ of deletion, it suffices to show that in the pruning process of obtaining the $(k+1)$-core from the k-core, the total anchor availabilities realized for the nodes having core number k is the same regardless of the order $\mathbb{O}$ of deletion.

Indeed: denote $S = \{a_1, .., a_n\}$ as the set of all nodes having core number k. Without loss of generality, assume a particular order in which the nodes are deleted in the pruning process of obtaining the $(k+1)$-core from the k-core is $a_1, ..., a_n$, and their respective anchor availabilities realized by CoREA are $c(a_1), ..., c(a_n)$. Denote $S^{(i)}$ as the set of all subsets of S containing a_i. For each subset $s \subset S$, let $t(s)$ be the number of hyperedges that have s as the set of anchors: $t(s) = |\{e \in \mathbf{E} | \mathbf{A_G}(e) = s\}|$.

At the k-core, the degree of node a_i is $k + R(a_i)$ with $R(a_i) \geq 0$ since the degree of each node among $\{a_1, .., a_n\}$ has to be at least k.

Assuming that the algorithm is now at the k-core and undertakes the pruning process to obtain the $(k+1)$-core while simultaneously obtaining the anchor availability for each node that has core number k. As node a_1 is the first node to delete, its degree is $\leq k$. However, since no hyperedges at the k-core have been deleted yet, the degree of a_1 at this point is $k + R(a_1) \geq k$. Therefore, $R(a_1) = 0$, and according to CoREA, the anchor availability realized for a_1 is $c(a_1) = 0$.

For each $i = 2, ..., (n-1)$, after nodes $a_1, ..., a_{i-1}$ have been removed, all of the hyperedges anchored at any of those nodes have also been removed from the network. Among those hyperedges, the ones that affect the degree of a_i are the ones co-anchored by a_i and at least 1 among $a_1, ..., a_{i-1}$. The number of such hyperedges is: $\sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s)$. Due to the removals of these hyperedges, the degree of a_i immediately prior to its deletion is $\max(0, k + R(a_i) - \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s))$. To qualify for deletion, the degree of a_i must be lower than $(k+1)$. In other words, $k + R(a_i) - \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s) \leq k$, or $-R(a_i) + \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s) \geq 0$. The anchor availability realized for node a_i is then equal to: $c(a_i) = -R(a_i) + \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s)$.

With a similar argument, for node a_n, after the deletions of $a_1, ..., a_{n-1}$, the anchor availability realized for a_n is: $c(a_n) = -R(a_n) + \sum_{s \in \bigcup_{j=1}^{n-1} (S^{(n)} \cap S^{(j)})} t(s)$.

The sum of anchor availabilities realized by CoREA for all nodes in the k-core is: $c(k) = \sum_{i=1}^{n} c(a_i) = -\sum_{j=1}^{n} R(a_j) + \sum_{i=2}^{n} \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s)$. According to Lemma 2:

$$\sum_{i=2}^{n} \sum_{s \in \bigcup_{j=1}^{i-1} (S^{(i)} \cap S^{(j)})} t(s) = \sum_{s \subset S, |s| \geq 2} (|s| - 1) t(s) \tag{2}$$

Thus, $c(k) = -\sum_{j=1}^{n} R(a_j) + \sum_{s \subset S, |s| \geq 2} (|s| - 1) t(s)$. This formula is symmetric with respect to each of $a_1, ..., a_n$, which is independent of the particular ordering of $a_1, ..., a_n$.

Therefore, the total number of anchor availabilities realized by CoREA for the nodes in the k-core is constant regardless of the order of deletions. Since the total anchor availabities realized by Algorithm 1 can be obtained by summing up all anchor availabities realized at each core level, the total number of anchor availabilities is also constant regardless of the order of deletions. □

The above proof is also for the following lemma, which states that the total anchor availabilities realized by CoREA is constant for each k-core:

Lemma 3 (INVARIANCE OF COREA in each k-core)**.** For $k = 1, ..., N^*_{\mathbf{G}}$, the total number of anchor availabilities of nodes having core number k, realized by CoREA, remains unchanged regardless of the order of nodes removed in the core decomposition.

Corollary 2. In pruning process of obtaining the $(k+1)$-core from the k-core, assume that a_n is the last node having core number k deleted, the anchor availability realized for a_n does not depend on the ordering of the other nodes having core number k.

Proof. Without loss of generality, assume that the order of deleting all nodes having core number k in the pruning process is $a_1, ..., a_n$. According to Theorem 2, the anchor availability realized for a_n is $-R(a_n) + \sum_{s \in \bigcup_{j=1}^{n-1} (S^{(n)} \cap S^{(j)})} t(s)$, which is symmetric with respect to $a_1, ...a_{n-1}$ and does not depend on the particular ordering or $a_1, ...a_{n-1}$. This demonstrates that the anchor availability realized for the last deleted node is independent of the deletion order of other nodes having core number k. □

VI. OPTIMALITY OF COREA

Lemma 4 (OPTIMALITY OF COREA in each k-core)**.** For $k = 1, ..., N^*_{\mathbf{G}}$, COREA always returns the maximum number of hyperedges that can be augmented, subject to the constraint of preserving all core numbers in the k-core of $\mathbf{G}$.

Proof. According to Theorem 2, the total anchor availabilities realized by Algorithm 1 for the nodes having core number k, is always the same regardless of the order of node deletions, and let T_k be such total number.

Assume the contradiction that T_k is not the maximum number of hyperedges, or maximum total number of anchor availabilities, for the nodes in the k-core shell. In other words, there is a valid augmentation method I that augments I_k hyperedges anchored at the nodes of the k-core shell that preserve all core numbers with $I_k \geq T_k + 1$. Without loss of generality, assume that with I, there exists an order of deletion of all nodes having core number k in the pruning process to obtain the $(k+1)$-core from the k-core as $a_1, ..., a_n$. Right before the deletion of a_i, its degree is $k - x(a_i) \leq k (x(a_i) \geq 0)$. Denote the number of hyperedges that are augmented by I and anchored at a_i as $I^{(i)}_k$, then $\sum_{i=1}^{n} I^{(i)}_k = I_k$

By Corollary 1, we know that without I, $a_1, ...a_n$ is still a feasible order of deletion in the pruning process. Without I, the pruning process still deletes nodes $a_1, ..a_n$ in this particular order to obtain the $(k+1)$-core from the k-core. The degree of a_i immediately prior to its deletion is $k - x(a_i) - I^{(i)}_k$. Algorithm 2 realizes the anchor capacity $c(a_i) = x(a_i) + I^{(i)}_k$ for node a_i. Theorem 2 proves that $T_k = \sum_{i=1}^{n} c(a_i) = \sum_{i=1}^{n} x(a_i) + I^{(i)}_k = \sum_{i=1}^{n} I^{(i)}_k + \sum_{i=1}^{n} x(a_i) = I_k + \sum_{i=1}^{n} x(a_i) \geq I_k \geq T_k + 1$, which is a contradiction.

Therefore, the initial assumption is false, which proves that Algorithm 1 returns the maxium total avaialbility of nodes having core number k.

□

Theorem 3 (OPTIMALITY OF COREA)**.** COREA always returns the maximum number of hyperedges that can be augmented, subject to the constraint of preserving all core numbers in $\mathbf{G}$.

Proof. Let D be the degeneracy of the network. Lemma 4 shows that COREA always return the maximum total number of anchor availabilities T_i of nodes having core number i for $i = 1, ..., N^*_{\mathbf{G}}$. The total anchor availabilities realized by Algorithm 1 is $T = \sum_{i=1}^{N^*_{\mathbf{G}}} T_i$.

Assume an augmentation method F augments I_k hyperedges, which can be added to the hypergraph without altering any core numbers, anchored at the nodes having core number k for $k = 1, ..., N^*_{\mathbf{G}}$. The total anchor availability realized by F is $I = \sum_{k=1}^{N^*_{\mathbf{G}}} I_k$. According to Lemma 4, $I_k \leq T_k$, so: $I = \sum_{k=1}^{D} I_k \leq \sum_{k=1}^{N^*_{\mathbf{G}}} T_k = T$. Thus, the total anchor availabity realized by F is always upperbounded by that of CoREA.

Thus, CoREA always returns the maximum total anchor availability of nodes. As the number of hyperedges constructed in the pool P is equal to the total number of anchor availabilities, CoREA indeed returns the maximum number of hyperedges that can be augmented, subject to the constraint of preserving all core numbers in $\mathbf{G}$. □

VII. Maximum anchor capacity of an individual node

Lemma 5. In the pruning process of obtaining the $(k+1)$-core from the k-core, assume that both u and v are up for removal, and the deletion order $\mathbb{O}$ chooses to remove v before u. if we obtain a feasible deletion order $\mathbb{O}'$ by switching node v with a node u behind v in a feasible deletion order $\mathbb{O}$, the anchor availability of v realized by Algorithm 1 remains the same or increases. (Note that $\mathbb{O}'$ also has to be a feasible deletion order, in which it removes one node having degree $\leq k$ at a time)

Proof. Assume that by the ordering of S, prior to its deletion, the degree of v is $\bar{d}(v)$, so Algorithm 1 realizes $c(v) = k - \bar{d}(v)$ as the anchor availability for v.

Switching v with a node u behind v in $\mathbb{O}$, we obtain $\mathbb{O}'$. The deletion of u will remove all of its incident hyperedges, some of which may also be anchored at v. As a result, the degree of v prior to its deletion is $\hat{d}(v) \leq \bar{d}(v)$, so by the ordering of S', the anchor availability realized for v is $c'(v) = k - \hat{d}(v) \geq k - \bar{d}(v) = c(v)$. This proves the claim. □

Lemma 6. In 2 different orders $\mathbb{O}$ and $\mathbb{O}'$ in which v is placed at the end or the removal of v is deferred until a point when v is the only node up for removal, the anchor availabilities of v realized by Algorithm 2 are the same.

Proof. According to Corollary 2, within a particular set of nodes, the anchor availability realized for the last node in any order does not depend on the order of the other nodes that precede it. Therefore, even though the order of the nodes preceding v are different in $\mathbb{O}$ and $\mathbb{O}'$, the anchor availabilities of v in both cases, $\mathbb{O}$ and $\mathbb{O}'$, are the same. □

Lemma 7. For a deletion order S_1 of nodes having core number k, in the pruning process of obtaining the $(k+1)$-core, which removes v as the last node or defers removing v until the a point when v is the only node up for removal, the anchor availability $c_{S_1}(v)$ realized for v is actually its maximum anchor capacity. In other words, immediately after the removal of v in the core decomposition process, if Line 14 of Algorithm 1 adds b hyperedges anchored at v to the pool P, and these b hyperedges are subsequently augmented to $\mathbf{G}$, the core number of v will change.

Proof. According to lemma 5 and 6, if the removal of v is deferred to the point when v is the only node qualified for removal, the anchor availability $c_{S_1}(v)$ realized for v by CoREA, given any particular removal order of the nodes that precede v. Therefore, in the pruning process of obtaining the $(k+1)$-core, before node v is removed, its degree is always $\geq k - a_{S_1}(v)$. If any method augments b hyperedges anchored at v at this point (at the same time as Line 14 of Algorithm 1), with $b > a_{S_1}(v)$, its degree becomes $\geq k - a_{S_1}(v) + b \geq k + 1$, which disqualifies v for the removal. Since v is the only node that is used to be qualified at this point, v will not be removed, which elevates v to a higher core number. □

Theorem 4 (Maximum anchor availability of an individual node)**.** To realize the maximum anchor availability of a node v, $N_{\mathbf{G}}(v) = k$, only delete v when v is the only node qualified for removal during the pruning process to obtain the $(k+1)$-core. In other words, at any point in this process, whenever facing multiple nodes qualified for removal including v, always defer removing v and remove another node first.

Proof. Denote S the order in which the nodes of core number k are deleted in the pruning process of obtaining the $(k+1)$-core from the k-core.

According to Lemma 5, if we keep delaying the removal of v in the sequence of node deletion by always choosing to delete another node u first, given that u is also qualified for removal, the anchor availability realized for v will either remaining the same or increase. Therefore, the anchor availability realized for v in any sequence will be upperbounded by the anchor availability as if v is removed when v is the only node qualified for removal at that point.

As Lemma 6 demonstrates, the anchor availability realized for v will be the same as long as the removal of v is deferred until a point when v is the only node up for removal, regardless of the ordering of the precedent nodes. This availability guarantees that v would retain its core number after that many hyperedges anchored at v are added to the pool P and subsequently augmented to $\mathbf{G}$, as shown in Theorem 1. Moreoever, Lemma 7 shows that further from that number of hyperedges, we cannot augment any higher number of hyperedges if the method is required to preserve all core numbers.

Therefore, deferring removing v until the point when v is the only node qualified for removal affords v its maximum anchor capacity.

□

REFERENCES

[1] P. Gupta and G. Bhattacharjee, "An efficient algorithm for random sampling without replacement," in *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 1984, pp. 435–442.