MONA: An Efficient and Scalable Strategy for Targeted k-Nodes Collapse

Yuqian Lv, Bo Zhou, Jinhuan Wang, Shanqing Yu, and Qi Xuan, *Senior Member, IEEE*

Abstract—The concept of k-core plays an important role in measuring the cohesiveness and engagement of a network. And recent studies have shown the vulnerability of k-core under adversarial attacks. However, there are few researchers concentrating on the vulnerability of individual nodes within k-core. Therefore, in this paper, we attempt to study Targeted k-Nodes Collapse Problem (TNsCP), which focuses on removing a minimal size set of edges to make multiple target k-nodes collapse. For this purpose, we first propose a novel algorithm named MOD for candidate reduction. Then we introduce an efficient strategy named MONA, based on MOD, to address TNsCP. Extensive experiments validate the effectiveness and scalability of MONA compared to several baselines. An open-source implementation is available at *https://github.com/Yocenly/MONA*.

Index Terms—k-core decomposition, k-core robustness, Adversarial attack, Edge removal, Graph data mining.

Fig. 1. Example graph for introducing motivations.

I. INTRODUCTION

NETWORKS or graphs have become integral parts of our daily lives, playing significant roles in various complex systems, e.g., social networks, biological networks, and transport networks. Due to simplicity and efficiency, the concept of k-core has gained prominence as a crucial metric for capturing the structural engagement of networks. The k-core is a maximal induced subgraph where each node has its degree of at least k. The presence of nodes in k-core depends on their neighboring relationships in the subgraph. Thus, removing a subset of nodes or edges from the k-core may result in the detachment of other nodes within it. For example, Zhou et al. [1] discovered that the removal of a small number of edges may severely devastate the structure of the k-core. And Chen et al. [2] focused on the k-core minimization problem and proposed effective algorithms to cover it. Medya et al. [3] employed the *Shapley* value, a cooperative game-theoretic concept, to address the problem of k-core minimization.

Y. Lv, B. Zhou, J. Wang and S. Yu are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China.

B. Zhou is also with the Department of Intelligent Control, Zhejiang Institute of Communications, Hangzhou 311112, China.

Q. Xuan is with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China, with the PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518000, China, and also with the Utron Technology Co., Ltd. (as Hangzhou Qianjiang Distinguished Expert), Hangzhou 310056, China (e-mail: xuanqi@zjut.edu.cn).
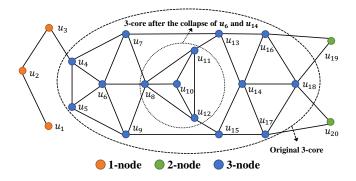
Motivations. Despite the effectiveness of above mentioned methods, they all attack the k-core from a global perspective. They pay little attention to the perturbations that affect individual nodes in the k-core. According to the research of Zhang et al. [4], the removal of critical nodes in k-core may significantly break down network engagement. However, the perturbations of removing nodes are infeasible in practice and may result in detectable consequences. Thus, for the sake of concealment and feasibility, we make the attempt to identify the critical edges whose removals will trigger the detachment of given target nodes from the k-core. In other words, given a set of target nodes in k-core, our objective is to remove a minimal-size set of edges such that all target nodes are absent from k-core. We refer to this problem as Targeted k-Nodes Collapse Problem (TNsCP).

Example 1. *We assume a social network in Figure. 1, where nodes and edges represent the users and their relationships. Users with a high significance usually exert great attraction to others. In the k-core model, the significance of users is assessed by the k-core they belong to, e.g., users in 3-core are usually more important than those just in 2-core. It can be realized that the absence of u_6 and u_{14} from 3-core will greatly reduce the number of users in 3-core, from 15 users to 4 users. With prior knowledge, we find that the breakdown of only two relationships, for example (u_4, u_7) and (u_{16}, u_{18}), will truly achieve this purpose.*

Research on TNsCP contributes to the understanding of node vulnerability in the k-core model and enables maintainers to safeguard critical relationships among users. Furthermore, this study helps to diminish the importance of confidential users, making them less noticeable to potential attackers.

Contributions. In this paper, we provide the general definition of TNsCP with proof of its NP-hardness. Then, we introduce a novel algorithm named MOD for effective candidate

reduction. Furthermore, we propose a novel heuristic algorithm named MONA to address TNsCP. We exhibit the effectiveness of MONA compared with several baselines. Finally, we show the high scalability of MONA compared with those global attack methods.

II. PRELIMINARIES AND PROBLEM STATEMENT

In this paper, we denote a graph as $G = (V, E)$, where V and E represent the sets of nodes and edges respectively. Note that we only focus on unweighted and undirected graphs without self-loops. We use $d_{(u,G)}$ and $N_{(u,G)}$ to represent the degree and neighbors of u in G, respectively. In this section, we introduce some essential definitions and concepts.

Definition 1. *k-core. Given a graph G and a positive integer k, its k-core, denoted as $G_k = (V_k, E_k)$ where $V_k \subseteq V$ and $E_k \subseteq E$, means the maximal induced subgraph where each node $u \in G_k$ occupies at least k neighbors.*

Definition 2. *Core Number. Given a graph G and a node $u \in G$, the core number of u, denoted as $C_{(u,G)}$, is the largest k, such that $u \in G_k$ while $u \notin G_{k+1}$.*

In this work, nodes with the same core number of k are called k-**nodes**. And if the core number of a node u decreases after edge removal, we refer to this event as the **collapse** of u. Generally, once a set of edges $\mathcal{E}$ is removed from G, it may lead to the collapse of multiple nodes in G due to the domino phenomenon of the k-core [5]. These nodes are called followers of $\mathcal{E}$ in G, denoted as $\mathcal{F}(\mathcal{E}, G)$. Assuming $G' = G \setminus \mathcal{E} = (V', E')$, we obtain $\mathcal{F}(\mathcal{E}, G) = V_k \setminus V'_k$.

Problem Statement. Given a graph G and a set of target k-nodes $\mathcal{T} \in V_k \setminus V_{k+1}$, the Targeted k-Nodes Collapse Problem (TNsCP) aims to remove a minimal-size set of edges $\mathcal{E} \in E$ to make all nodes in $\mathcal{T}$ collapse, i.e., $\mathcal{T} \subseteq \mathcal{F}(\mathcal{E}, G)$.

Theorem 1. *The TNsCP is NP-hard.*

Proof. We try to reduce the Set Covering Problem (SCP) to the TNsCP. In the SCP, two sets are given: a universe set U, and a set S that contains subsets of U. The union of all subsets in S could cover U. Then the SCP concerns finding the smallest number of subsets from S to cover the entire universe [6].

Here, for a given graph G and a set of target k-nodes $\mathcal{T}$, we define $U = \mathcal{T}$ and $S = E$. It is important to note that S itself does not represent the set of subsets of U. Therefore, we utilize the follower function $\mathcal{F}(\cdot, G)$ to map S to U. For example, for an element $s \in S$, we have $\mathcal{F}(s, G) \cap U \subseteq U$. It becomes evident that $\mathcal{F}(S, G) \cap U = U$. After that, the TNsCP aims to find a minimal-size set $\mathcal{E} \subseteq S$ such that $\mathcal{F}(\mathcal{E}, G) \cap U = U$. Thus, we have successfully reduced the SCP to TNsCP. Since the NP-hardness of the SCP [6], it follows that the TNsCP is also NP-hard. □

Theorem 2. *Given a graph G and an edge $(u, v) \in E$. Removal of (u, v) could make the k-nodes collapse if and only if $min(d_{(u,G_k)}, d_{(v,G_k)}) = k$.*

Theorem 3. *Given a graph G and an edge $(u, v) \in E$. Assume that $k = C_{(u,G)} \leq C_{(v,G)}$, only those nodes with the core*

Algorithm 1: Optimal(G, $\mathcal{T}$)

  input : given graph G, target k-nodes $\mathcal{T}$;
  output: removed edges $\mathcal{E}$.

1 $\mathcal{P} \leftarrow \{(u,v) | (u,v) \in E, min(C_{(u,G)}, C_{(v,G)}) = k\}$;
2   **for** $iter \leftarrow 1$ *to* $|\mathcal{P}|$ **do**
3 $combs \leftarrow$ All combinations of size $iter$ in $\mathcal{P}$;
4   **foreach** $\mathcal{E} \in combs$ **do**
5 $\lfloor$ **if** $\mathcal{T} \subseteq \mathcal{F}(\mathcal{E}, G)$ **then return** $\mathcal{E}$;

---

number of k may collapse after the removal of (u, v), and their core number will decrease at most 1.

The proofs of Theorem 2 and Theorem 3 are omitted here because they have already been established by [7] and by [8], [9], respectively. Then, on the basis of Theorem 2 and Theorem 3, we have the following observation.

Observation 1. *Those edges included in $\mathcal{P} = \{(u,v) | (u,v) \in E, min(C_{(u,G)}, C_{(v,G)}) = k\}$ are what matter in the collapse of k-nodes.*

Proof. Given a graph G and an edge $(u, v) \in E$ that satisfies $min(d_{(u,G_k)}, d_{(v,G_k)}) = k$, the followers of (u, v) are the subset of k nodes, that is, $\mathcal{F}(\{(u, v)\}, G) \subseteq V_k \setminus V_{k+1}$. After the removal of (u, v), all these followers will absolutely collapse from k-core to $(k-1)$-core. Additionally, the collapse of $\mathcal{F}(\{(u, v)\}, G)$ will trigger the emergence of more edges that satisfy Theorem 2. Following the procedure, we can iteratively remove these compliant edges until no more edges satisfy Theorem 2 in the graph. In such a scenario, it implies that there are no nodes with the core number of k in the graph, indicating the complete collapse of the k-nodes. And all the edges removed in this procedure are part of $\mathcal{P} = \{(u,v) | (u,v) \in E, min(C_{(u,G)}, C_{(v,G)}) = k\}$. □

Based on Observation 1, we can initially narrow the candidate edges from E to $\mathcal{P}$.

III. METHODOLOGIES

A. Optimal Solution

Assuming that we have prior knowledge of $\mathcal{F}(\{e\}, G)$, the removal of $\{e\}$ is the optimal solution when our target nodes are within $\mathcal{F}(\{e\}, G)$. Therefore, a naive solution for TNsCP is to exhaustively enumerate all possible combinations of edges in $\mathcal{P}$ and select the best for removal. As shown in Algorithm 1, we explore each edge combination in $\mathcal{P}$, with its size ranging from 1 to $|\mathcal{P}|$. The time complexity of Algorithm 1 is $\mathcal{O}(\sum_{i=1}^{|\mathcal{P}|} \binom{|\mathcal{P}|}{i} |E_k|)$, which is extremely time-consuming and unscalable with increasing input size. Due to the constraints of computational resources, it is not feasible to enumerate the followers generated by all possible edge combinations. Therefore, identifying the candidate edges that may cause the collapse of the target k-nodes $\mathcal{T}$ poses a challenge.

B. Improved Candidate Reduction

Motivated by the above challenge, in this part, we present two novel algorithms to improve candidate reduction.

Algorithm 2: MOD(G, k)

---

input : given graph G, core number constraint k;
output: modified onion distribution $\mathcal{M}$.

1   $layer \leftarrow 0$; $\tilde{G} \leftarrow G_k$; $\mathcal{S} \leftarrow \{u | u \in G, C_{(u,G)} = k\}$;
2   **while** $\mathcal{S}$ *is not empty* **do**
3      $layer \leftarrow layer + 1$;
4      $equal \leftarrow \{u | u \in \tilde{G}, deg(u, \tilde{G}) = k\}$;
5      $lower \leftarrow \{u | u \in \tilde{G}, deg(u, \tilde{G}) < k\}$;
6   **if** $|lower| > 0$ **then** $equal \leftarrow \emptyset$;
7   **foreach** $v \in lower \cup equal$ **do**
8        $\mathcal{M}_v \leftarrow layer$;
9      $\mathcal{S} \leftarrow \mathcal{S} \setminus \{lower \cup equal\}$;
10     $\tilde{G} \leftarrow \tilde{G} \setminus \{lower \cup equal\}$;
11 **return** $\mathcal{M}$;

Algorithm 3: BacktrackTree(G, $\mathcal{T}$)

---

input : given graph G, target k-nodes $\mathcal{T}$;
output: backtrack tree G_{BT}.

1   $\mathcal{M} \leftarrow$ **MOD**(G, k); $queue \leftarrow \mathcal{T}$;
2   $G_{BT} \leftarrow$ a directed graph $(\mathcal{T}, \emptyset)$;
3   **foreach** $u \in queue$ **do**
4      $nodes \leftarrow \{v | v \in N(u, G_k), C_{(v, G_k)} = k\}$;
5      $neighbors \leftarrow \{v | v \in nodes, \mathcal{M}_v < \mathcal{M}_u\}$;
6      $queue \leftarrow queue \cup (neighbors \setminus V_{BT})$;
7      $G_{BT} \leftarrow G_{BT} \cup \{(u,v) | v \in neighbors\}$;
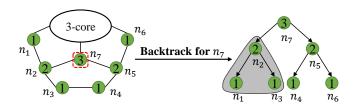8   **return** G_{BT};

Fig. 2. Example for MOD and BacktrackTree algorithms. All 2-nodes are assigned by MOD (left) and n_7 is backtracked by BacktrackTree (right).

Given a k-node u, the collapse of u may trigger the collapse of other k-nodes in its neighbors. Therefore, when our target nodes are among the collapsed neighbors of u, it is prioritized to make u collapse. This necessitates further classification of k-nodes to distinguish them with different collapse orders. Inspired by Onion Decomposition (OD) [10], we propose Modified Onion Decomposition (MOD) to divide k-nodes of G into different layers, which are illustrated in Algorithm 2.

The main difference between OD and MOD is that OD removes all nodes with degrees less than or equal to k in each iteration. For MOD, in Algorithm 2, we further divide these nodes into $equal$ and $lower$ (Lines 4-5). If there are nodes with degrees less than k, we set $equal$ to an empty set (Line 6) and then assign nodes in $lower$ with $layer$ (Lines 7-8). Otherwise, nodes in $equal$ are assigned. Then, we update $\mathcal{S}$ and $\tilde{G}$ (Lines 9-10). Finally, the modified onion distribution of k-nodes $\mathcal{M}$ is returned . The lower the layer of a k-node in $\mathcal{M}$, the higher its priority in the collapse order. For example, we execute Algorithm 2 for 2-nodes in Figure 2 (left). The number marked on each node represents its modified onion layer. In the first iteration, we remove $equal = \{n_1, n_3, n_4, n_6\}$ and assign them to $layer = 1$. Then, $lower = \{n_2, n_5\}$ and $lower = \{n_7\}$ are removed in order and assigned with $layer = 2$ and $layer = 3$, respectively.

Given a target k-node u, we could prioritize the collapse of its neighbors with higher collapse orders according to $\mathcal{M}$. Furthermore, we can iteratively backtrack on those neighbors that maintain increasing collapse orders with respect to the current node. As shown in Algorithm 3, we exploit the Breadth First Search (BFS) algorithm for this backtracking procedure. In each iteration, we extract those neighbors with higher collapse orders, denoted as $neighbors$ (Lines 4-5). Then the

BFS queue $queue$ is updated (line 6) and the edges that link the current node with its neighbors are added to G_{BT} (line 7). Note that G_{BT} is a directed graph initialized with nodes in $\mathcal{T}$ and empty edges (Line 2). For example, as shown in Figure 2 (right), we execute Algorithm 3 with the setting of n_7 as the target. We obtain G_{BT} with 7 nodes and 6 directed edges.

We can realize that those nodes in V_{BT} are most probably related to the collapse of $\mathcal{T}$. And the removal of the edges in E_{BT} will lead to the collapse of the nodes in V_{BT}. However, the edges in E_{BT} are not sufficient to cover the collapse of $\mathcal{T}$ when nodes in $\mathcal{T}$ are assigned in $equal$ during Algorithm 2. Therefore, E_{BT} is augmented by the adjacent edges of $\mathcal{T}$ in G_k. In this way, we obtain the set of candidate edges, denoted as $\mathcal{H} = E_{BT} \cup \{(u,v) | u \in \mathcal{T}, v \in N(u, G_k) \setminus V_{BT}\}$. It can be seen that $\mathcal{H} \subseteq \mathcal{P}$. Therefore, we further improve the candidates from $\mathcal{P}$ to $\mathcal{H}$.

Complexity. The time complexities of Algorithm 2 and Algorithm 3 are both $\mathcal{O}(|V_k \setminus V_{k+1}|)$. Additionally, both algorithms have a space complexity of $\mathcal{O}(|G_k|)$.

C. MONA Algorithm

Benefiting from the improved candidates $\mathcal{H}$, in this part, we propose a novel heuristic algorithm to tackle TNsCP, named Modified Onion based k-Nodes Attack (MONA).

Pruned Followers. Given an edge $e \in \mathcal{H}$, the removal of e has two main impacts. First, as discussed before, the removal of e can trigger the collapse of the nodes in G, that is, $\mathcal{F}(\{e\}, G)$. In addition, the removal of e may result in the zero-degree of nodes in G_{BT}. These nodes will no longer appear in G_{BT} during the backtracking process of Algorithm 3. This process is analogous to pruning a branch from a tree, causing all the leaves on that branch to detach from the tree. For example, as shown in Figure 2 (right), those nodes in the shadow area, i.e., n_1, n_2 and n_3, will absolutely be pruned after the removal of edge $(3, 2)$. These nodes will disappear in G_{BT} during the re-backtrack process. We integrate these two impacts and introduce the concept of pruned followers $\mathcal{F}_p(e)$ to measure the impact of removing e. The operations for obtaining $\mathcal{F}_p(e)$ is presented in Algorithm 4. Note that we bypass those zero-degree nodes contained in $\mathcal{T}$.

We exhibit the details of the MONA algorithm in Algorithm 5. In Line 2 and Line 3, we first initialize G_{BT} and $\mathcal{H}$. In Line 4 to Line 8, we iteratively remove the edge with the most pruned followers and then update G_{BT} and $\mathcal{H}$ according to the adversarial graph $G_k \setminus \mathcal{E}$.

Algorithm 4: PruneEdge(G, G_{BT}, e, $\mathcal{T}$)

   input : given graph G, backtrack tree G_{BT}, removed
             edge e, target k-nodes $\mathcal{T}$;
   output: pruned followers $\mathcal{F}_p(e)$.
1 $\tilde{G} \leftarrow G_{BT} \setminus \{e\}$; $\mathcal{F}_p \leftarrow \mathcal{F}(\{e\}, G) \cap V_{BT}$;
2 **while** *exist zero-indegree nodes in* $\tilde{V} \setminus \mathcal{T}$ **do**
3 $\mathcal{Z} \leftarrow$ All zero-indegree nodes in $\tilde{V} \setminus \mathcal{T}$;
4 $\tilde{G} \leftarrow \tilde{G} \setminus \mathcal{Z}$; $\mathcal{F}_p \leftarrow \mathcal{F}_p \cup \mathcal{Z}$;
5 **return** $\mathcal{F}_p(e)$;

Algorithm 5: MONA(G, $\mathcal{T}$)

   input : given graph G, target k-nodes $\mathcal{T}$;
   output: removed edges $\mathcal{E}$.
1 $\mathcal{E} \leftarrow \emptyset$; $G_{BT} \leftarrow$ **BacktrackTree**(G_k, $\mathcal{T}$);
2 $\mathcal{H} \leftarrow E_{BT} \cup \{(u,v) | u \in \mathcal{T}, v \in N(u, G_k) \setminus V_{BT}\}$;
3 **while** $\mathcal{T} \not\subseteq \mathcal{F}(\mathcal{E}, G)$ **do**
4   **foreach** $e \in \mathcal{H}$ **do**
5 $\mathcal{F}_p(e) \leftarrow$ **PruneEdge**($G_k \setminus \mathcal{E}$, G_{BT}, e, $\mathcal{T}$);
6 $e^\star \leftarrow$ The edge with most pruned followers;
7 $\mathcal{E} \leftarrow \mathcal{E} \cup \{e^\star\}$; Update G_{BT} and $\mathcal{H}$ with $G_k \setminus \mathcal{E}$;
8 **return** $\mathcal{E}$;

Complexity. The time complexities of Algorithm 4 and Algorithm 5 are $\mathcal{O}(|V_{BT}| + |E_k|)$ and $\mathcal{O}(|V_k \setminus V_{k+1}| + |\mathcal{H}| \cdot |V_{BT}| + |\mathcal{H}| \cdot |E_k|)$, respectively. Additionally, both algorithms have a space complexity of $\mathcal{O}(|G_k|)$.

IV. EXPERIMENTS

A. Experimental Settings

Datasets. Basic properties of used datasets are presented in Table I. All these datasets are collected from [11]. Note that all networks are converted to undirected and unweighted graphs without self-loops.

TABLE I
BASIC PROPERTIES OF USED DATASETS, WHERE d_{avg} IS THE AVERAGE DEGREE AND k_{max} IS THE MAXIMAL CORE NUMBER.

Dataset	$|V|$	$|E|$	d_{avg}	k_{max}	$|V_{k_{max}}|$	$|E_{k_{max}}|$
USAir	332	2,126	12.81	26	35	539
DeezerEU	28,281	92,752	6.56	12	71	564
Crawl	1,112,702	2,278,852	4.10	18	725	11,522
YouTube	1,134,890	2,987,624	5.27	51	845	36,363
Lastfm	1,191,805	4,519,330	7.58	70	597	35,153
Wikipedia	1,864,433	4,507,315	4.84	66	324	16,054
Roadnet	1,957,027	2,760,388	2.82	3	4,454	7,393
Talk	2,394,385	4,659,565	3.89	131	700	73,503
Patent	3,774,768	16,518,947	8.75	64	106	4,043
Livejournal	4,033,137	27,933,062	13.85	213	214	22,791

Baselines. The following baseline methods are considered for evaluations and comparisons with our proposed method.
- **Random** randomly removes an edge from $\mathcal{P}$ in each iteration until all nodes in $\mathcal{T}$ collapse.
- **Degree** removes the edge with the lowest degree (the sum of the degrees of its endpoints in G_k) in $\mathcal{P}$ in each iteration until all nodes in $\mathcal{T}$ collapse.

- **COREATTACK** [1] iteratively removes the edge with most followers in $G_{k_{max}}$ until all k_{max}-nodes collapse.
- **KC-Edge** [7] iteratively removes the edge with most followers in G_k until its perturbation budget p is fulfilled or G_k is empty.

All programs are implemented in Python 3.7.11. All experiments are carried out on a machine equipped with Intel(R) Xeon(R) Gold 5218R CPU @2.10GHz and Linux Ubuntu 20.04.4. Note that we select the **top-**b nodes with the highest degree within $G_{k_{max}}$ as our target nodes $\mathcal{T}$. And Random is performed 100 times independently for each task and the mean value is recorded.

B. Effectiveness of Improved Candidate Reduction

To evaluate the effectiveness of the improved candidate reduction proposed in Section III, we first compare the number of candidate edges utilized by Optimal and MONA, i.e. $|\mathcal{P}|$ and $|\mathcal{H}|$. We implement these two algorithms in USAir and DeezerEU with b varying from 2 to 10. As shown in Figure 3(a) and 3(b), a notable distinction is observed in their candidate sizes. Moreover, to further justify the approximation guarantee of MONA, we also compare its attack performance, i.e., the number of removed edges $|\mathcal{E}|$, with that of Optimal. The results in Figures 3(c) and 3(d) support that MONA could achieve the optimal solution. Furthermore, we also illustrate the candidate reduction for the other datasets with the setting of $b = 30$, which is shown in Figure 3(e).

(a) Candidate edges on USAir   (b) Candidate edges on DeezerEU

(c) Removed edges on USAir   (d) Removed edges on DeezerEU

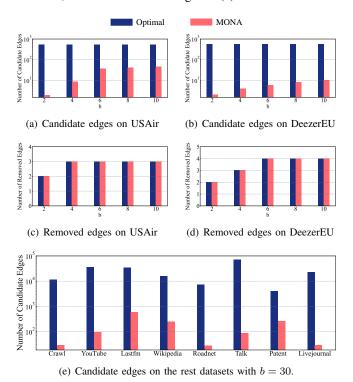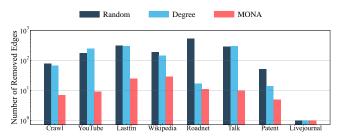(e) Candidate edges on the rest datasets with $b = 30$.

Fig. 3. Effectiveness of improved candidate reduction with different settings of hyperparameter b.

C. Effectiveness of MONA Algorithm

Furthermore, we evaluate the effectiveness of MONA algorithm with the comparisons of Random and Degree in the

setting of $b = 30$ in Figure 4. Specifically, the results in Figure 4(a) illustrate that MONA consistently removes the fewest edges for the collapse of target nodes across all tested datasets. It is worth noting that on Livejournal, all three methods end up with only one edge removed. This is due to the fact that the $G_{k_{max}}$ of Livejournal is a complete graph, where the removal of any edge results in the collapse of all k_{max}-nodes. Since Random and Degree do not traverse all candidate edges to find the best one in each iteration, it is not surprising that they have a lower time complexity than MONA in Figure 4(b). However, MONA achieves more precise results than Random and Degree in acceptable time cost. Additionally, Figure 5 presents the number of removed edges under different settings of b on Lastfm and Wikipedia. The number of removed edges clearly increases with increasing b for MONA. However, the upward trend becomes slower when b reaches 30.
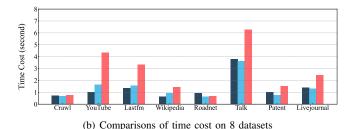

(a) Comparisons of removed edges on 8 datasets

(b) Comparisons of time cost on 8 datasets

Fig. 4. Effectiveness and efficiency of MONA compared with Random and Degree with the setting of $b = 30$.
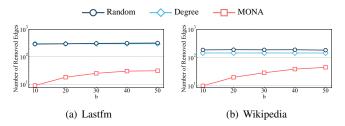

(a) Lastfm

(b) Wikipedia

Fig. 5. Removed edges on Lastfm and Wikipedia with b from 10 to 50.

D. Scalability of MONA for Global Attack

In previous works, Zhou et al. [1] and Zhang et al. [7] respectively designed global-perspective k-core attack methods, namely COREATTACK and KC-Edge. By increasing the number of target nodes, we can easily extend MONA to the realm of global attacks. To compare with COREATTACK, we set $b = |E_{k_{max}}|$. And to compare with KC-Edge, we set $b = |E_{k_{max}}|$ and terminate MONA when the number of removed edges reaches $|\mathcal{E}| = p = 10$. The comparative results are shown in Figure 6. In global settings, it is evident that

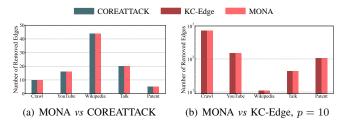(a) MONA vs COREATTACK

(b) MONA vs KC-Edge, $p = 10$

Fig. 6. The scalability of MONA for global attack with the comparisons of COREATTACK and KC-Edge.

the MONA algorithm exhibits consistent effectiveness with these global attack methods. It demonstrates the scalability of MONA for global attack.

V. CONCLUSION

In this paper, we propose and study TNsCP, which aims to remove a minimal-size set of edges for the collapse of target k-nodes. And we offer a proof of its NP-hardness. To improve time complexity, we provide a novel algorithm named MOD for candidate reduction. Furthermore, on the basis of MOD, an efficient heuristic algorithm named MONA is proposed to address TNsCP. Extensive experiments on 10 real-world datasets demonstrate the effectiveness and scalability of MONA compared to multiple baselines.

REFERENCES

[1] B. Zhou, Y. Lv, J. Wang, J. Zhang, and Q. Xuan, "Attacking the core structure of complex network," *IEEE Transactions on Computational Social Systems*, pp. 1–15, 2022.

[2] C. Chen, Q. Zhu, R. Sun, X. Wang, and Y. Wu, "Edge manipulation approaches for k-core minimization: metrics and analytics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 390–403, 2021.

[3] S. Medya, T. Ma, A. Silva, and A. Singh, "A game theoretic approach for core resilience," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 3473–3479, main track.

[4] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Finding critical users for social network engagement: The collapsed k-core problem," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[5] A. V. Goltsev, S. N. Dorogovtsev, and J. F. F. Mendes, "k-core (bootstrap) percolation on complex networks: Critical phenomena and nonlocal effects," *Physical Review E*, vol. 73, no. 5, p. 056101, 2006.

[6] T. Grossman and A. Wool, "Computational experience with approximation algorithms for the set covering problem," *European Journal of Operational Research*, vol. 101, no. 1, pp. 81–92, 1997.

[7] W. Zhu, C. Chen, X. Wang, and X. Lin, "K-core minimization: An edge manipulation approach," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1667–1670.

[8] A. E. Sarıyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and Ü. V. Çatalyürek, "Streaming algorithms for k-core decomposition," *Proceedings of the VLDB Endowment*, vol. 6, no. 6, pp. 433–444, 2013.

[9] R.-H. Li, J. X. Yu, and R. Mao, "Efficient core maintenance in large dynamic graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2453–2465, 2013.

[10] L. Hébert-Dufresne, J. A. Grochow, and A. Allard, "Multi-scale structure and topological anomaly detection via a new network statistic: The onion decomposition," *Scientific reports*, vol. 6, no. 1, pp. 1–9, 2016.

[11] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015.