# SHATTER: Searching Heterogeneous Network Attack Sequences Through Network Embedding and Reinforcement Learning

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Heterogeneous networks can effectively fuse more information than homogeneous networks. The function implementation of heterogeneous networks is also more complex. To find the key nodes and get an efficient attack sequence is our focus. Most current research on network disintegration is difficult to achieve efficiency and timeliness. We specifically proposed SHATTER (Searching Heterogeneous network ATTack sequences through network Embedding and Reinforcement learning) model, which used the induction algorithm combined with graph neural network and reinforcement learning to solve the network disintegration sequence problem. Through experiments with synthetic heterogeneous network datasets as well as real heterogeneous networks, we determined that SHATTER's performance is significantly improved over existing technologies. More importantly, SHATTER's excellent performance shows that the model framework constructed under the influence of human a priori knowledge can effectively improve the characterization of network embeddings.

## 1 Introduction and Related Work

Searching for key nodes in the network has always been the central topic of network research, which can be applied from critical infrastructure networks to biological and social systems, such as suppressing virus transmission [1], destroying terrorist networks [2], preventing rumor transmission [3] and other issues.

Geographically dispersed units can transform information technology advantages into competitive advantages through network advantages. Network disintegration has been a central topic in network research, and the study of heterogeneous network disintegration is closely related to the efficiency of its information processing and flow efficiency[4].

Distributed operations transform information technology advantages into competitive advantages through geographically dispersed units via network advantages. Network disintegration has been the central topic of network research, and the study of combat network disintegration is closely related to its information processing and flow efficiency[4]. Learning how to efficiently dismantle heterogeneous networks is important when performing task planning and coping with efficient information flow from heterogeneous networks [5].

### 1.1 Existing work

In the field of network disintegration, the search for the Achilles' heel of the network was kicked off by the article [6]. It is a NP-hard problem to find the optimal disintegration strategy to obtain the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

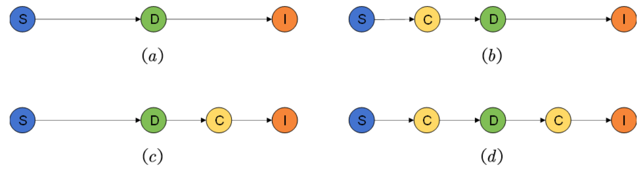



Figure 1: Schematic showing different types of operational chains.

best attack effect[7]. Heuristic and meta-heuristic algorithms are mostly used in traditional network decomposition research[1; 8; 9; 10]. Traditional heuristic or meta-heuristic algorithms are usually difficult to strike a satisfactory balance between effectiveness and efficiency.

The development of deep learning in solving combinatorial optimization problems[11; 12; 13; 14; 15; 16] has given us new ideas to solve network disintegration. Fan et al. first proposed a general and scalable deep reinforcement learning framework FINDER for finding key nodes in complex networks based on graph neural networks and reinforcement learning[17]. Zeng et al. further proposed the MINER model for the disintegration problem of multiplex networks, and the model is able to couple the association relationships between pairs of nodes in different layers[18].

### 1.2 Limitations of existing work

**Inductive Learning:** Previous algorithms based on metaheuristics are not transferable, which means that the experience of mining and exploring the network needs to be recomputed in the face of each new network and cannot be passed on.

**Network disintegration efficiency:** Previous approximation algorithms need to discover a node value evaluation method that matches the target evaluation function, which requires specialized a priori knowledge of the researcher, but this does not always result in a good attack.

### 1.3 Contributions

**Heterogeneous network processing framework.** A heterogeneous network representation is proposed in the SHATTER model to map the heterogeneous network topology and attribute information to a low-dimensional space to facilitate downstream tasks.

**Generalization capability.** SHATTER has good portability and the trained model can be adaptively applied to the tiling of heterogeneous networks with different scales, and has been experimentally validated. Compared with genetic algorithms, evolutionary algorithms and other metaheuristics, SHATTER exhibits better timeliness.

**Downstream task suitability.** SHATTER can be trained to learn feature extraction methods that are adapted to downstream tasks and have better performance than metrics based on degree and pagerank attributes, which cannot be fully adapted to downstream tasks. The experiments also verify that the SHATTER framework achieves good results in the validation set of heterogeneous networks different from the training set network model, with or without the node attack cost constraint.

## 2 Problem Formulation

In Distributed Combat network, there are usually combat units with different types and functions. These nodes with relatively single function form a combat system through information exchange. A heterogeneous network is the appropriate model for studying it.

A network contains multiple types of nodes or multiple types of links, then this is a heterogeneous network. According to IACM( information age combat model )[19] and Decker's FINC(Force, Intelligence, Networking, and C2) [20] models, combat forces on the battlefield are divided into three categories. (1) Sensor nodes. Combat units that perform early warning, detection and reconnaissance missions, referred to as $S$. (2) Decision nodes. Combat units that perform command and control tasks, referred to as $D$. (3) Influential nodes. Combat units for strike and electromagnetic jamming missions, referred to as $I$. A heterogeneous combat network (HCN) is a special kind of heterogeneous network in which different types of nodes represent sensor, decision maker, and influencer combat units and edges represent various information flows between combat units.

The HCN can be represented as $G(V,E)$, where $V = S \cup D \cup I$ represents the set of nodes, $E \subseteq V \times V$ represents the flow channel of information between operational units. The total number of nodes in the network is $N$. The node sets $S$,$D$, and $I$ represent the sensor set, decision set, and influential set, respectively. The number of sensor combat units, decision combat units, and influential combat units are denoted as $|S|$ , $|D|$, and $|I|$, respectively. Different types of functional nodes have different types of capabilities. In this study, we consider reconnaissance capabilities, decision-making capabilities and influence capabilities. The specific capabilities of combat units with the same types of capabilities are also different in strength and normalized. $CA_i(v) \in (0,1)$ ,$i \in S, D, I$ denotes the combat capability possessed by combat unit $v$ of type $i$. In this paper, we will not study the case of a single operational unit with multiple capabilities.

**Capability-oriented functional measure of HCNs.** According to the OODA (Observation-Orientation-Decision-Action) operational cycle theory, the operational process is carried out in an iterative cycle of Observation, Orientation, Decision and Action. This cyclic process involves the sensor warfare unit detecting an enemy target and passing the intelligence to the decision warfare unit. After data fusion and information analysis, the decision warfare unit makes operational decisions and orders the influence warfare unit to attack. In the process of accomplishing a specific combat mission, different combat units cooperate with each other and repeat the OODA cycle. To facilitate the expression and subsequent evaluation, the concept of OC(operational chain) is introduced to represent the collaboration between different combat units. As shown in Fig.1, the OC can be divided into basic type and general type. The basic type, as shown in Fig.1(a), consists of sensor operation unit, decision operation unit and influence operation unit directly connected. Since any combat unit can also act as an intermediary for information transfer, we extend the basic type to a general type with mediated communication nodes. Considering the timeliness of operational information, we only consider here the case where there is a one-hop intermediary in the $S-D$ and $D-I$ information transfer process, and the general type of OC is shown in Fig.1(b-d).

The combat effectiveness of the OC is determined by the collaboration of sensor combat units, decision combat units, and influence combat units. In addition, because of the time-sensitive nature of the OC, a shorter length of the OC can transmit information faster than a longer chain and has higher combat effectiveness. Since the OC is chronological and the process of reconnaissance, decision-making and influence cannot be reversed. And mediated communication nodes cannot replace these functions.

$L_G = \{l_j\}, j = 1, 2, \cdots, m$ is the set of all OCs that exist in the heterogeneous combat network $G$, where $l_k$ describes an OC containing sensor combat unit $s_j$, decision combat unit $d_j$, and influence combat unit $i_j$, and $|l_j|$ represents the length of OC $l_j$. The operational capability of OC $l_j$ can be evaluated as

$$U(l_j) = \frac{1}{|l_j|} \times CA_S(s_j) \times CA_D(d_j) \times CA_I(i_j). \tag{1}$$

Given a heterogeneous combat network containing the set of operational chains $L_G$, its operational effectiveness can be denoted as

$$\mathit{\Gamma}(G) = \sum_{l_j \in L_G} U(l_j). \tag{2}$$

The goal of the algorithm studied in this paper is to learn an optimal sequence of node removal, which makes the network performance collapse rapidly. To evaluate the merits of the disintegration strategy, we draw on the evaluation method of network robustness[21]. In our study, we use heterogeneous network operational effectiveness as an attribute metric for heterogeneous networks, and our goal is to learn an optimal sequence of node removal $(v_1, v_2, \cdots, v_N)$, which minimizes the target heterogeneous combat network accumulated normalized operational capability (ANOC)[22; 17] as follows:

$$ANOC(v_1, v_2, \cdots, v_N) = \frac{1}{N} \sum_{j=1}^{N} \frac{\mathit{\Gamma}(G \setminus \{v_1, v_2, \cdots, v_j\})}{\mathit{\Gamma}(G)}. \tag{3}$$

Here, the subscript of $v_k$ in the removal node-set $\mathcal{K} = \{v_1, v_2, \cdots, v_j\}$ marks the order in which the nodes are removed. $\Gamma\left(G \setminus \{v_1, v_2, \cdots, v_j\}\right)$ represents the network performance after sequential removal of $\mathcal{K}$.

In some scenarios, the cost of attack (cost of attack resources, cost of attack time, etc.) is different for different nodes. We define the formula of the weighted ANOC as follows.

$$ANOC_{cost}\left(v_1, v_2, \cdots, v_N\right) = \sum_{j=1}^{N} \frac{\Gamma\left(G \setminus \{v_1, v_2, \cdots, v_j\}\right)}{\Gamma\left(G\right)} c(v_j) \tag{4}$$

where $c(v_k)$ denotes the normalized attack cost of node $k$, $\sum_{j=1}^{N} c\left(v_j\right) = 1$.

## 3 Model

Our proposed model no longer uses the traditional strategy based on network attribute measures. For the problem of solving the optimal disintegration sequence for a heterogeneous combat network, we model this process as an interaction between the agent and the environment, formalized as a Markov decision process (MDP). The state is defined as the current remaining network after the last attack action, which in turn determines the best attack node to be removed, and the action reward for each step is the network combat effectiveness after the action, and the overall reward is the cumulative network combat effectiveness. Throughout the training process, the experience pool needs to be continuously updated to induce the model to learn network embedding methods that are more suitable for downstream network disintegration tasks. By learning long-term policies, a sequence of actions can be determined that allows the intelligent disintegration algorithm to obtain the minimum cumulative network combat effectiveness, which means the optimal disintegration effectiveness.

We distinguish framework SHATTER into encoding and decoding parts, where the encoding part encodes the current state of the entire network as well as the optional actions, and the decoding part decodes the rewards of the encoded state-action pairs.

**Encoding part.** The encoding part works to characterize the state of the whole network and the state of the optional attack nodes, and this part is a prerequisite for the decoding part in the next step. There are various methods to encode network states into vectors, including matrix factorization based methods[23; 24; 25], random walk based methods[26; 27], and deep learning based methods[28; 29; 30]. And graph neural networks (GNN), an emerging field of deep learning in recent years, have achieved good results on tasks related to graph data, which also shows its powerful representation capability. This approach has good portability and is more suitable for rapid decision-making in the face of heterogeneous combat networks. However, the characterization of heterogeneous networks is more complex compared to homogeneous networks. To better characterize the heterogeneous combat network OODA combat cycle properties, we used a special heterogeneous combat network characterization model for encoding. The whole coding part is divided into three processes, network layering, layer network embedding and inter-layer cross embedding. The whole encoding part is shown in Fig.2, and the specific implementation is shown in Algorithm 1, where lines 1-2 pseudocode are network layering process, lines 3-5 pseudocode are the layer network embedding process, and lines 6-11 pseudocode are the inter-layer cross embedding process.

**Algorithm 1** Heterogeneous Network Encoding

**Input:** Heterogeneous combat network $G^{[ini]}(V^{[ini]}, E^{[ini]})$, weight parameters $W_1$, $W_2$, $a_1$ , $a_2$
**Output:** Node embedding $h_v$ and network embedding $h_s$

1: Introduce virtual node $s$ in $G$, which connects all nodes in $V$ and is used to represent the state of the network
2: Generate $S-D$, $D-I$ layer networks $G_{S-D}$ and $G_{D-I}$ from the heterogeneous combat network $G$. And calculate node network attributes and generate node feature matrix $X^{[l]}$ and adjacency matrices $A^{[l]}$ for $l \in L, L = \{S-D, D-I, ini\}$
3: **for** $l \in \{S-D, D-I, ini\}$ **do**
4: $\quad h_u^{[l]} \leftarrow Graph\_Embedding^{[l]}\left(X^{[l]}, A^{[l]}\right), u \in V \cup \{s\}$
5: **end for**
6: $\alpha_l = \frac{\exp(LeakyReLU(a_1^T[h_v^{[l]} \cdot W_1 \| h_s^{[ini]} \cdot W_1]))}{\sum_{m \in L} \exp(LeakyReLU(a_1^T[h_v^{[m]} \cdot W_1 \| h_v^{[ini]} \cdot W_1]))}$
7: $h_v \leftarrow \sigma\left(h_v^{[ini]} \cdot W_1 + \sum_{l \in \{S-D, D-I\}} \alpha_l h_v^{[l]} \cdot W_1\right)$
8: $\beta_l = \frac{\exp(LeakyReLU(a_2^T[h_s^{[l]} \cdot W_2 \| h_s^{[ini]} \cdot W_2]))}{\sum_{m \in L} \exp(LeakyReLU(a_2^T[h_s^{[m]} \cdot W_2 \| h_s^{[ini]} \cdot W_2]))}$
9: $h_s \leftarrow \sigma\left(h_s^{[ini]} \cdot W_2 + \sum_{l \in \{S-D, D-I\}} \beta_l h_s^{[l]} \cdot W_2\right)$
10: $h_v \leftarrow h_v / \|h_v\|, \forall v \in V$
11: $h_s \leftarrow h_s / \|h_s\|$

In the first step of coding, we will implement a layering process for the network. Since the nodes in the model can be relayed for use as communication nodes, the initial network topology does not represent the operational chain chain relationships well. The purpose of the layering process is to enhance the representativeness of the network and consequently speed up the convergence. Based on the initial network, this step generates S-D network $G^{[S-D]}$ and D-I network $G^{[D-I]}$. The elements in the respective adjacency matrices $A^{[S-D]}$ and $A^{[D-I]}$ are shown in Eqs. (5) and (6), where $d_{ij}^{[l]}$ denotes the distance between node $i$ and node $j$ in the $l$th layer network.To represent the state information at the network layer, unlike the traditional readout mechanism of all node vectors, we introduce the virtual node $s$, which captures the network information of other nodes. It is unidirectionally connected to all nodes in the network and receives information from other nodes in Graph_Embedding without transmitting information to other nodes.

$$a_{ij}^{[S-D]} = \begin{cases} 1, & i \in S, j \in D, d_{ij}^{[S-D]} \leq 2 \\ 0, & otherwise \end{cases}. \tag{5}$$

$$a_{ij}^{[D-I]} = \begin{cases} 1, & i \in D, j \in I, d_{ij}^{[D-I]} \leq 2 \\ 0, & otherwise \end{cases}. \tag{6}$$

Then, the second step, layer network embedding process. The networks $G^{[S-D]}$ and $G^{[D-I]}$ generated in the previous process can better intuitively show the role of nodes in OODA combat cycle. The initial network $G^{[ini]}$ preserves the initial connection relationship of each node, and the functional information of the nodes in the OODA combat loop and the information of the communicating nodes as bridges are preserved. The specific details of the layer network embedding are shown in Algorithm 2. The structures of the graph neural network in different layer networks are the same, except that the propagation iteration depth $K = 2$ for $G^{[S-D]}$ and $G^{[D-I]}$, , while $K = 5$ for the initial network $G^{[ini]}$. This is because $G^{[S-D]}$ and $G^{[D-I]}$ are similar to bipartite networks and do not require a deep perceptual field, while the initial network needs to deal with a wider perceptual field due to the intermediary bridging communication task that the nodes will undertake.

The final step of the encoding part is the cross-embedding process. The network embeddings of the different layers are weighted to generate the final state encoding and action encoding, where the weights are generated through an attention mechanism. The details are shown in lines 6 to 9 in Algorithm 1.

**Algorithm 2** Graph_Embedding

**Input:** $l$-layer network $G^{[l]}(V^{[l]}, E^{[l]})$, Node feature matrix $X^{[l]}$ and adjacency matrices $A^{[l]}$, depth $K$, weight parameters $W_3^{[l]}$ , $W_4^{[l]}$, $W_5^{[l]}$

**Output:** Node embedding $h_u^{[l]}$ for $l$th layer

Initialize $h^0_{u^{[l]}} \leftarrow \mathrm{Re}LU(X^{[l]} \cdot W_3^{[l]})$, $h^0_{u^{[l]}} \leftarrow h^0_{u^{[l]}} / \|h^0_{u^{[l]}}\|$

**for** $k = 1$ to $K$ **do**

  **for** $u \in V \cup \{s\}$ **do**

    $h^k_{\mathcal{N}(u^{[l]})} \leftarrow AGGREGATE\left(\{h^{k-1}_{u'^{[l]}}, \forall u'^{[l]} \in \mathcal{N}\left(u^{[l]}\right)\}\right)$

    $h^k_{u^{[l]}} \leftarrow \mathrm{Re}LU\left(\left[h^{k-1}_{u^{[l]}} \cdot W_4^{[l]} \| h^k_{\mathcal{N}(u^{[l]})} \cdot W_5^{[l]}\right]\right)$

  **end for**

  $h^k_{u^{[l]}} \leftarrow h^k_{u^{[l]}} / \|h^k_{u^{[l]}}\|$

**end for**

$h_u^{[l]} \leftarrow h^K_{u^{[l]}}, \forall u \in V \cup \{s\}$

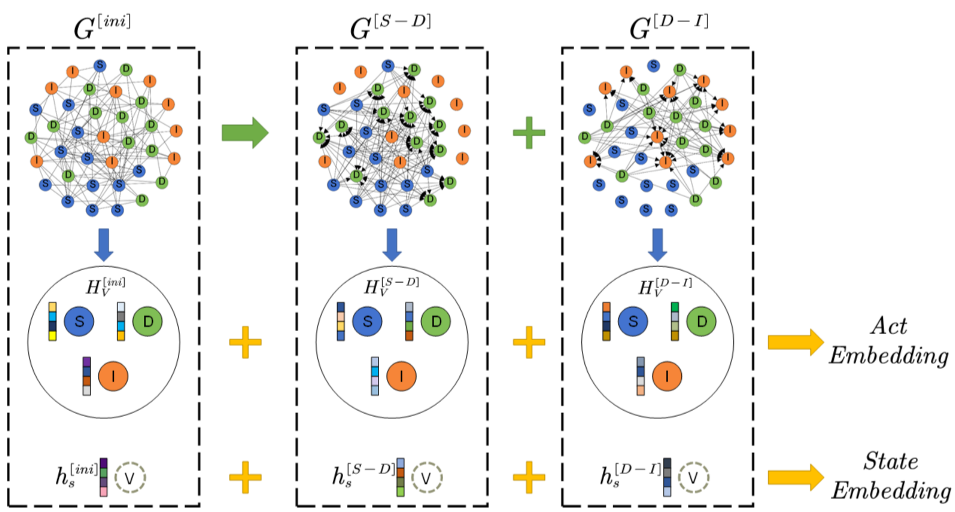



Figure 2: Overview of the SHATTER framework encoding part. The coding part of the SHATTER framework consists of three processes: network layering, layer network embedding, and inter-layer cross embedding. In the network layering process, the initial network $G^{[ini]}$ is layered to obtain $G^{[S-D]}$ and $G^{[D-I]}$. Then layer network embedding process is executed on each layer network after adding a virtual node to each layer. The layer nodes embedding after network layering process adopts the attention mechanism to perform inter-layer cross embedding process. Finally, the vector representation of each node is obtained, as well as the overall representation of the network state collected by the virtual node.

**Decoding part.** In future operations, a large number of individuals will form combat systems through networks, and even when a combat network has only a small number of nodes $n$, its solution space for performing a k-step network disintegration sequence is huge, which is $n!/k!$. Meta-heuristic algorithms are certainly a very effective approach, but due to the time-sensitive nature of combat decision-making, the use of graph neural networks combined with reinforcement learning would provide a more timeliness approach when the number of network nodes continues to increase.

Since the next state $S_{t+1}$ during disintegration is only related to the current network state $S_t$ , which is $\hat{G}$. And the selected attack action $a_t$, which is $v_j$ this process of selecting attack nodes can be mathematically formalized as a Markov decision process (MDP). In our model, the value of the reward when given $(\hat{G}, v_j)$ is $r = \frac{1}{N} \cdot \mathit{\Gamma}(\hat{G}/v_j)\left(r_{cost} = c(v_j) \cdot \mathit{\Gamma}(\hat{G}/v_j)\right)$.

In our study, we use Deep Q-Networks (DQN) to solve this Markovian decision problem. The state-action pair $(S, a)$ is embedded after the encoding part, and in the decoding part, the Q function is used to evaluate the state-action pairs. The Q function consists of a multilayer perceptron as shown below:

$$Q\left(S, a\right) = Q\left(h_s, h_v\right) = W_6 \cdot \mathrm{Re}LU\left(h_v^T \cdot h_s \cdot W_7\right) \tag{7}$$

**Offline training.** Two networks with the same structure but different parameters are used in DQN. One is the evaluation network $Q_{eval}$ and the other is the target network $Q_{tar}$, and their network structures are shown in Eq. 7. At each training step, the evaluation network continuously updates the parameters of the neural network and calculates the estimate value $Q_{pre} = Q_{eval}(S_t, a_t)$ on $S_t, a_t$. On the other hand, the target network fixes parameters and updates them at regular intervals, which are used to estimate the true value of the state-action pair $Q_{true} = r_t + \gamma \min_{a_{t+1}} Q_{tar}\left(S_{t+1}, a_{t+1}\right)$. As in a general DQN, the state-action-reward pairs $(S_t, a_t, r_t, S_{t+1})$ obtained at each step of the training process are placed in the training pool $P$. The parameters are updated by minimizing the loss function $\mathcal{L}$, as shown in Eq.8.

$$\mathcal{L} = \mathbb{E}_{(S_t, a_t, r_t, S_{t+1}) \sim U(P)}\left[\left(Q_{pre} - Q_{true}\right)^2\right] \tag{8}$$

The whole offline training process is shown in Fig.3. SHATTER is trained iteratively by drawing training samples from the training pool, and the model gradually learns the experience to shatter the heterogeneous combat networks.

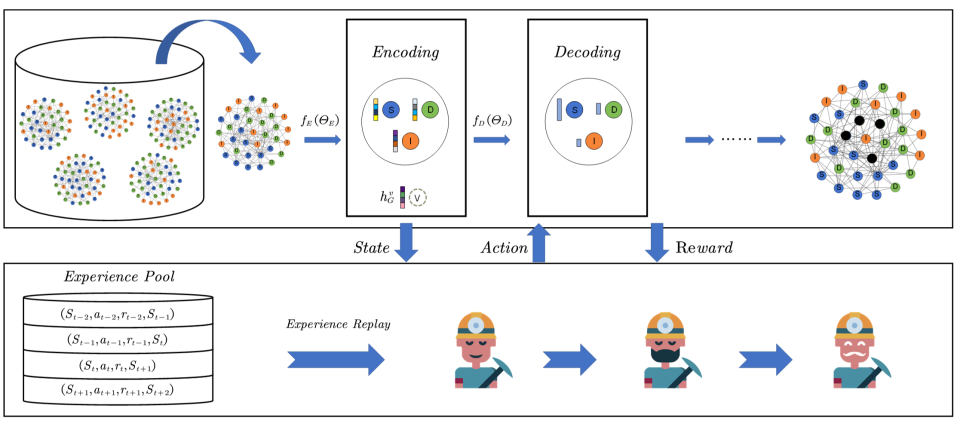



Figure 3: Offline training for SHATTER.

# 4 Experiments

**Data sets.** The heterogeneous network set for training is generated based on the scale-free network model, in which the nodes randomly acquire a capability and assign a value, while the heterogeneous network training set considering the attack cost of nodes adds a random attack cost attribute. The number of nodes in the training set is 30-60, and this setup exposes SHATTER to a smaller space of attack actions and allows SHATTER to converge faster when learning network features[18]. The sets of synthetic operational networks used for validation are based on the scale-free network model, the random network model and the small-world network model, respectively. Also, different network scales are used for each type of network validation set, with node sizes of 90-150, 150-300, 300-600 and 600-900. Such settings are used to verify the transferability of SHATTER in different network models with different network sizes. For the real network data experiments, we use FINC network data[20], which has 89 nodes, including 51 sensor nodes, 13 decision nodes, and 25 influential nodes. The nodes in the original data do not have capability attribute and attack cost values, which are taken as random numbers between 0 and 1.

**Baseline strategies for comparison.** There has been relatively little research on disintegration for heterogeneous operational networks, and we evolve some existing attack strategies against homogeneous networks so that they become baseline strategies applicable to heterogeneous operational networks. These include strategies that attack only a single type of node: HDSA (high-degree sensor node adaptive), HDDA ((high-degree decision node adaptive)) and HDIA (high-degree influential node adaptive)[31]. HDA (high degree adaptive)[6] and HPA (high PageRank adaptive)[32] are strategies that rank the networks according to their degree attributes and PageRank attributes. The original FINDER can be applied to heterogeneous network disintegration after adding the node capability property[17].

**Offline training.** All experiments were conducted on a single-core computer with 16GB RAM and 8GB NVIDIA GeForce RTX 2070 GPU. We generated small-scale scale-free networks as the training set (30-60 nodes) and the test set (60-90 nodes), where the number of networks in the training set is 5000 and the number of networks in the test set is 200. The training convergence curves are shown in Fig.4. The model was implemented in PyTorch 1.8.1 and trained with the Adam optimizer. The hyper-parameters are shown in Appendix.A.2 Tab. 5.

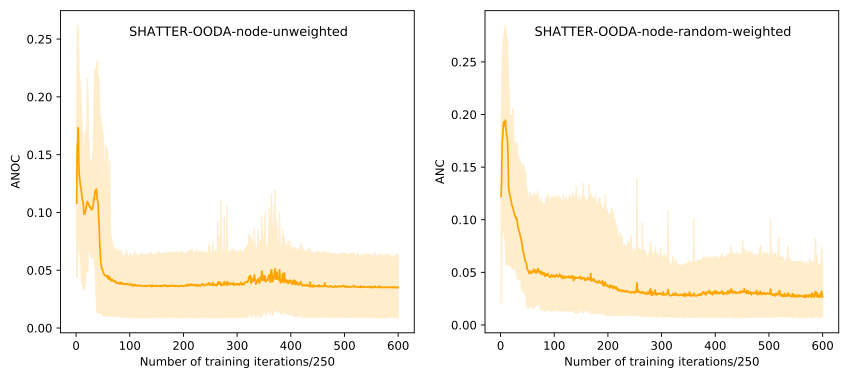



Figure 4: Training convergence of SHATTER.

**Results on synthetic heterogeneous combat networks.** The average ANOC values for each strategy in the validation set of different types and sizes of synthetic networks are displayed in Tabs. 1, 2 and 3. Figs.6, 7 and 8 in Appendix.A.3 show the disintegration performance of SHATTER on different types of synthetic heterogeneous networks (scale-free, random and small-world), which distinguish between cases where node attack cost is considered or not. It can be shown that SHATTER significantly outperforms the other baseline strategies. In the validation set of the scale-free network model, the advantage of SHATTER is not obvious due to the vulnerability of scale-free networks to deliberate attacks [6]. The connectivity of the network is the basis for forming operational chains, so the attacks based on degree and PageRank attributes work well. However, the situation becomes more complex in random and small-world networks, implying that SHATTER can merge input node features and heterogeneous network structure to extract relevant network information for downstream tasks end-to-end. The comparison with FINDER demonstrates that SHATTER is more capable of information representation in heterogeneous combat network disintegration tasks.

Table 1: Comparison of ANOC×100 obtained by different strategies on attack synthetic HCNs based on scale-free network model

| Strategies | OODA-node-unweighted | | | | OODA-node-random-weighted | | | |
|---|---|---|---|---|---|---|---|---|
| | $90-150$ | $150-300$ | $300-600$ | $600-900$ | $90-150$ | $150-300$ | $300-600$ | $600-900$ |
| **HDSA** | 8.35 | 7.83 | 8.21 | 7.47 | 8.36 | 7.85 | 8.09 | 7.50 |
| **HDDA** | 6.26 | 5.69 | 4.58 | 4.42 | 6.34 | 5.67 | 4.58 | 4.40 |
| **HDIA** | 8.88 | 8.37 | 7.57 | 7.51 | 8.90 | 8.27 | 7.61 | 7.44 |
| **HDA** | 2.77 | 1.95 | 1.41 | 1.03 | 2.83 | 1.92 | 1.39 | 1.02 |
| **HPA** | 2.80 | 1.97 | 1.43 | 1.05 | 2.86 | 1.94 | 1.40 | 1.03 |
| **FINDER** | 2.64 | 1.95 | 1.53 | 1.36 | 2.11 | **1.47** | **1.12** | **0.84** |
| **SHATTER** | **2.60** | **1.87** | **1.38** | **1.01** | **2.09** | 1.49 | 1.17 | 0.92 |

Table 2: Comparison of ANOC×100 obtained by different strategies on attack synthetic HCNs based on random network model

| Strategies | OODA-node-unweighted | | | | OODA-node-random-weighted | | | |
|---|---|---|---|---|---|---|---|---|
| | $90-150$ | $150-300$ | $300-600$ | $600-900$ | $90-150$ | $150-300$ | $300-600$ | $600-900$ |
| **HDSA** | 15.71 | 15.68 | 16.79 | 16.55 | 15.72 | 15.61 | 16.68 | 16.52 |
| **HDDA** | 14.40 | 15.66 | 16.12 | 16.68 | 14.58 | 15.70 | 16.19 | 16.70 |
| **HDIA** | 15.12 | 16.15 | 16.44 | 16.40 | 14.83 | 16.13 | 16.43 | 16.51 |
| **HDA** | 19.70 | 22.24 | 24.16 | 24.68 | 19.73 | 22.19 | 24.20 | 24.83 |
| **HPA** | 19.58 | 22.20 | 24.14 | 24.70 | 19.64 | 22.13 | 24.14 | 24.80 |
| **FINDER** | 14.36 | 17.21 | 21.70 | 23.98 | 7.82 | 8.93 | 10.03 | 10.81 |
| **SHATTER** | **11.62** | **11.40** | **11.54** | **11.19** | **7.44** | **8.50** | **8.97** | **9.22** |

Table 3: Comparison of ANOC×100 obtained by different strategies on attack synthetic HCNs based on smallworld network model

| Strategies | OODA-node-unweighted | | | | OODA-node-random-weighted | | | |
|---|---|---|---|---|---|---|---|---|
| | $90-150$ | $150-300$ | $300-600$ | $600-900$ | $90-150$ | $150-300$ | $300-600$ | $600-900$ |
| **HDSA** | 10.83 | 10.57 | 11.09 | 11.03 | 10.86 | 10.56 | 11.20 | 10.99 |
| **HDDA** | 8.99 | 9.22 | 9.18 | 9.12 | 9.00 | 9.09 | 9.18 | 9.15 |
| **HDIA** | 10.91 | 10.62 | 10.87 | 10.83 | 10.75 | 10.65 | 10.88 | 10.94 |
| **HDA** | 9.54 | 9.52 | 9.76 | 9.66 | 9.54 | 9.44 | 9.87 | 9.67 |
| **HPA** | 10.43 | 10.77 | 10.74 | 10.92 | 10.42 | 10.73 | 10.82 | 10.98 |
| **FINDER** | 8.81 | 8.84 | 8.82 | 8.97 | 6.03 | 6.21 | 5.97 | 6.49 |
| **SHATTER** | **7.31** | **7.44** | **7.43** | **7.33** | **4.50** | **4.37** | **4.47** | **4.55** |

**Results on real heterogeneous combat network.** In the real network experiment, SHATTER outperforms other baseline strategies in both cases with and without considering the cost of node attacks. Fig.5 shows specifically the degradation of the operational performance of the FINC network when facing SHATTER and other baseline strategies, and the specific ANOC values are shown in Appendix.A.4 Tab.6.

# 5 Conclusions

We propose the heterogeneous network disintegration model SHATTER, which fills the gap of heterogeneous network disintegration using graph representation and reinforcement learning by embedding representations after laying process and evaluating the value of attack actions through reinforcement learning, providing a new idea of heterogeneous network disintegration. The problem faced by previous heuristic and meta-heuristic algorithms that the disintegration effect and timeliness cannot be balanced is to some extent solved. The method exhibits better transferability and achieves better experimental results in both test sets of heterogeneous networks for network models different from the training set. Our SHATTER also has the limitation that when the number of nodes of a certain type is small, attacking this type of nodes alone works significantly better, but our model does not find this very well. It seems to be better at solving problems in complex situations.

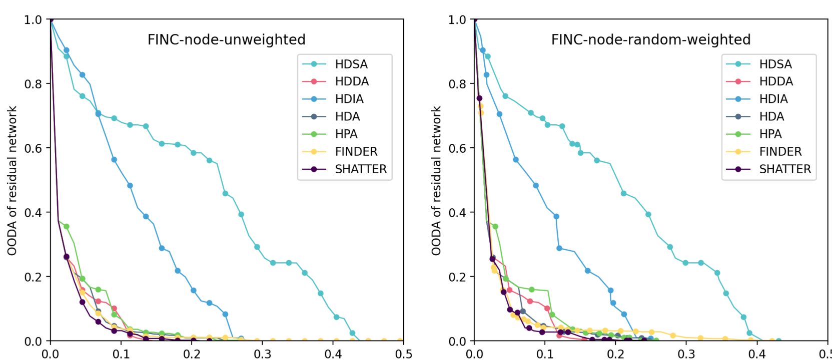


Figure 5: Performance of SHATTER on FINC heterogeneous combat network.

# References


[1] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos, "Detecting critical nodes in sparse graphs," *Computers & Operations Research*, vol. 36, no. 7, pp. 2193–2200, 2009.

[2] I. D. Kuntz, "Structure-based strategies for drug design and discovery," *Science*, vol. 257, no. 5073, pp. 1078–1082, 1992.

[3] B. Vitoriano, M. T. Ortuño, G. Tirado, and J. Montero, "A multi-criteria optimization model for humanitarian aid distribution," *Journal of Global optimization*, vol. 51, no. 2, pp. 189–208, 2011.

[4] B. Ge, K. W. Hipel, L. Fang, K. Yang, and Y. Chen, "An interactive portfolio decision analysis approach for system-of-systems architecting using the graph model for conflict resolution," *IEEE Transactions on systems, man, and cybernetics: systems*, vol. 44, no. 10, pp. 1328–1346, 2014.

[5] J. R. Cares and J. Q. Dickmann Jr, *Operations research for unmanned systems*. John Wiley & Sons, 2016.

[6] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[7] M. Lalou, M. A. Tahraoui, and H. Kheddouci, "The critical node detection problem in networks: A survey," *Computer Science Review*, vol. 28, pp. 92–117, 2018.

[8] X.-L. Ren, N. Gleinig, D. Helbing, and N. Antulov-Fantulin, "Generalized network dismantling," *Proceedings of the national academy of sciences*, vol. 116, no. 14, pp. 6554–6559, 2019.

[9] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.

[10] Y. Shen, N. P. Nguyen, Y. Xuan, and M. T. Thai, "On the discovery of critical links and nodes for assessing network vulnerability," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 963–973, 2012.

[11] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6351–6361, 2017.

[12] M. Nazari, A. Oroojlooy, M. Takáč, and L. V. Snyder, "Reinforcement learning for solving the vehicle routing problem," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9861–9871, 2018.

[13] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[14] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour dhorizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[15] J. James, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3806–3817, 2019.

[16] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," *arXiv preprint arXiv:1810.10659*, 2018.

[17] C. Fan, L. Zeng, Y. Sun, and Y.-Y. Liu, "Finding key players in complex networks through deep reinforcement learning," *Nature machine intelligence*, vol. 2, no. 6, pp. 317–324, 2020.

[18] C. Zeng, L. Lu, H. Liu, J. Chen, and Z. Zhou, "Multiplex network disintegration strategy inference based on deep network representation learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 5, p. 053109, 2022.

[19] J. R. Cares *et al.*, "An information age combat model," *Alidade, Inc., Newport, PR, USA (Produced for the Director, Net Assessment, Office of the Secretary of Defense under Contract TPD-01-C-003)*, 2004.

[20] A. H. Dekker, "C4isr, the finc methodology, and operations in urban terrain," *Journal of Battlefield Technology*, vol. 8, no. 1, pp. 25–28, 2005.

[21] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," *Physical review E*, vol. 65, no. 5, p. 056109, 2002.

[22] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[23] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[24] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, 2001.

[25] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[27] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[28] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, 2016.

[29] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[31] J. Li, J. Jiang, K. Yang, and Y. Chen, "Research on functional robustness of heterogeneous combat networks," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1487–1495, 2018.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will provide the code and experimental results in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Appendix.A.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Appendix. A.3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Section 4 Paragraph 4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Appendix

## A.1 Inference complexity analysis

In the algorithm application stage, we apply the trained SHATTER model to the given heterogeneous combat network. The time complexity of the SHATTER algorithm is determined by three parts: the encoding part, the decoding part and the action selection part. The coding part is further divided into three processes, the algorithmic complexity of the layering process is $\mathcal{O}\left(|E|\right)$, and in the layer embedding process, the algorithmic complexity is $\mathcal{O}\left(K^{[ini]} * N * \mathcal{N}^{[ini]}\left(\cdot\right) + K^{[S-D]} * N * \mathcal{N}^{[S-D]}\left(\cdot\right) + K^{[D-I]} * N * \mathcal{N}^{[D-I]}\left(\cdot\right)\right)$, where $K^{[l]}$ denotes the number of iterative layers of Graph_Embedding in $l$th layer, $N$ is the number of network nodes, and $\mathcal{N}^{[l]}\left(\cdot\right)$ is the average number of neighbors of the node in $l$th layer network. For ease of understanding, the algorithmic complexity of the process can also be expressed as $\mathcal{O}\left(K^{[ini]}\left|E^{[ini]}\right| + K^{[S-D]}\left|E^{[S-D]}\right| + K^{[D-I]}\left|E^{[D-I]}\right|\right)$, where $\left|E^{[l]}\right|$ represents the number of edges in the $l$th network. Since $\left|E^{[l]}\right|$ are proportional to the number of edges in the original network, it can be written as $\mathcal{O}\left(\left(K^{[ini]} + K^{[S-D]} + K^{[D-I]}\right)|E|\right)$. In the cross-embedding process, the algorithmic complexity of the process is $\mathcal{O}\left(N\right)$. In the decoding part, the Q values of the nodes in each layer network are calculated using Eq.7 with a time complexity of $\mathcal{O}\left(N\right)$. For the action selection part, we take the minimum Q-value of all nodes without sorting all Q-values, and the time complexity of this part is $\mathcal{O}\left(N\right)$. Since the whole process will last for $N$ rounds in the worst case. Therefore, the time complexity of the whole strategy inference process is $\mathcal{O}(N|E| + N^2)$. In Tab.4, we compare the time complexity of the SHATTER with that of other baseline methods.

Table 4: Inference complexity analysis. Among them, $N$ is the number of network nodes, $|E|$ representing the number of all edges in the network

| Method | Time complexity | Additional notes |
|---|---|---|
| HDSA | $\mathcal{O}(\|S\|^2)$ | $\|S\|$ is the number of sensor combat units |
| HDDA | $\mathcal{O}(\|D\|^2)$ | $\|D\|$ is the number of decision combat units |
| HDIA | $\mathcal{O}(\|I\|^2)$ | $\|I\|$ is the number of influential combat units |
| HDA | $\mathcal{O}(N^2)$ | |
| HPA | $\mathcal{O}(t * N^3)$ | $t$ is the number of iterations in PageRank algorithm |
| FINDER | $\mathcal{O}(K * N\|E\| + N^2)$ | $K$ denotes the number of iterative layers in network embedding |
| SHATTER | $\mathcal{O}(N\|E\| + N^2)$ | |

## A.2 Hyper-parameters in SHATTER.

Table 5: List of hyper-parameters and their values.

| Hyper-parameter | Value | Description |
|---|---|---|
| $\|P\|$ | $5 \times 10^4$ | capacity of the training pool $P$ |
| learning rate | $1 \times 10^{-4}$ | the learning rate used by Adam optimizer |
| $p, q$ | $32, 64$ | dimensions of hidden and final layer node embedding vector |
| update time | 250 | the frequency with which target network is updated |
| $K^{[ini]}, K^{[S-D]}, K^{[D-I]}$ | $5, 2, 2$ | number of neighbor-aggregation iterations |
| $\gamma$ | 0.99 | reward decay factor |
| mini-batch size | 32 | number of mini-batch training samples |

## A.3 Experiments on the synthetic HCNs

Figs.6, 7 and 8 show the disintegration performance of SHATTER on different types of synthetic heterogeneous networks (scale-free, random and small-world), which distinguish between cases where node attack cost is considered or not.

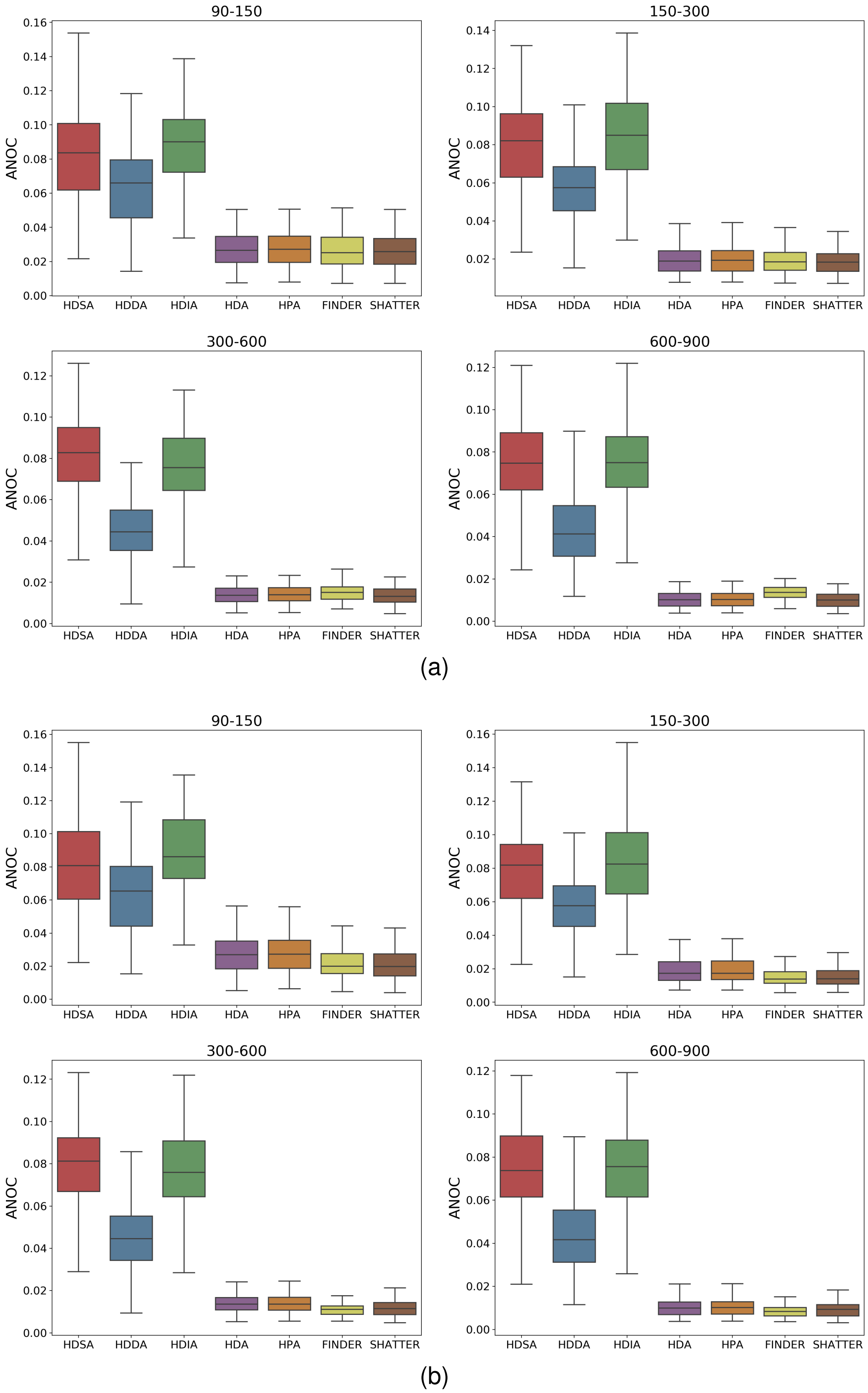
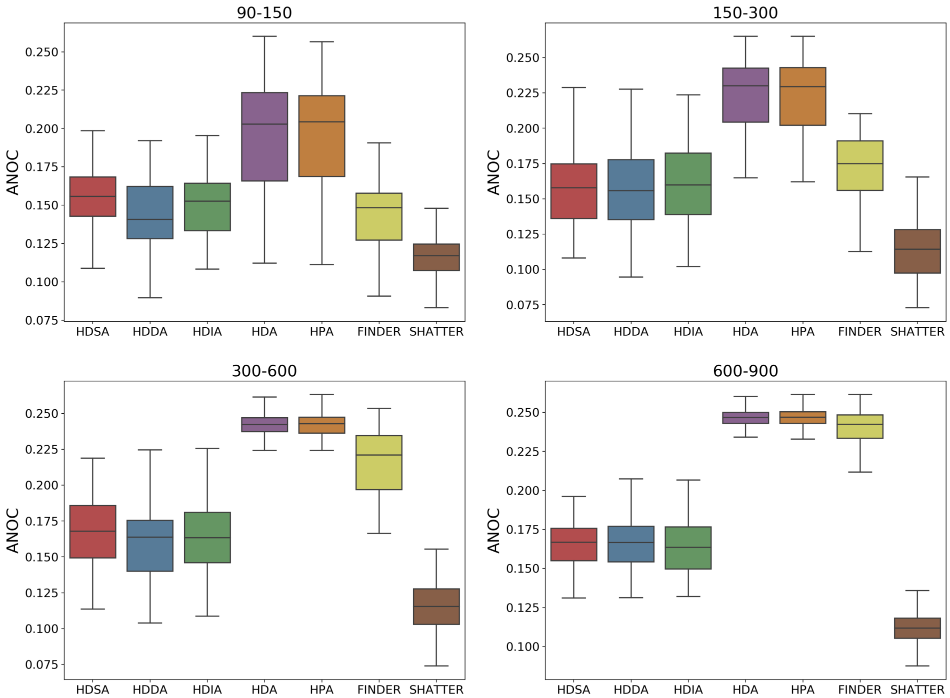
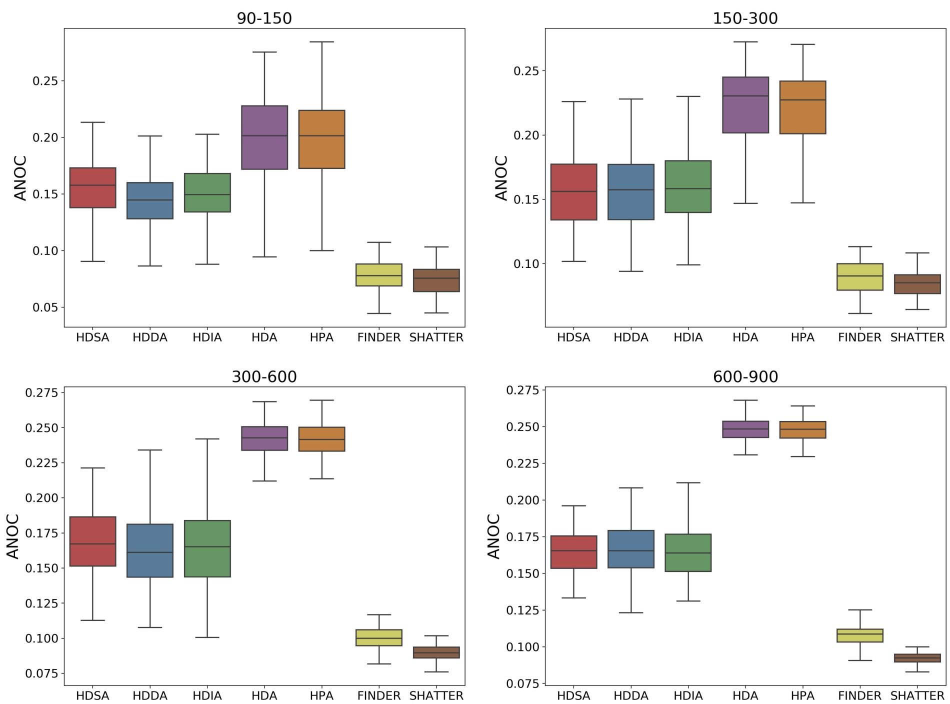



Figure 6: Comparison of SHATTER and baseline methods on synthetic HCNs based on scale-free network models. (a) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes without considering the attack's cost. (b) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes considering the attack's cost.

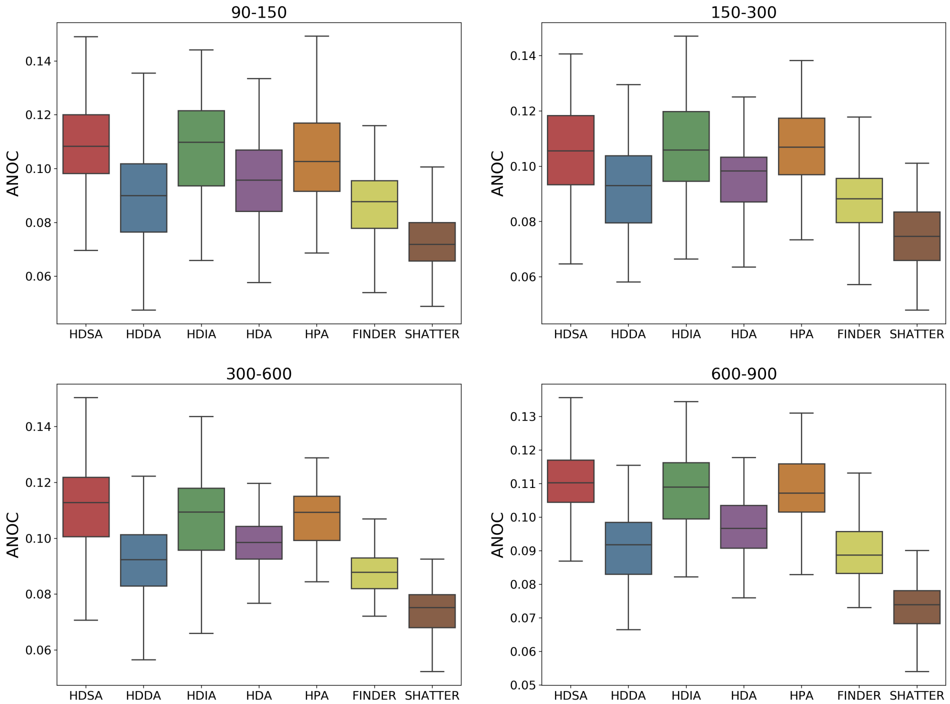



(a)

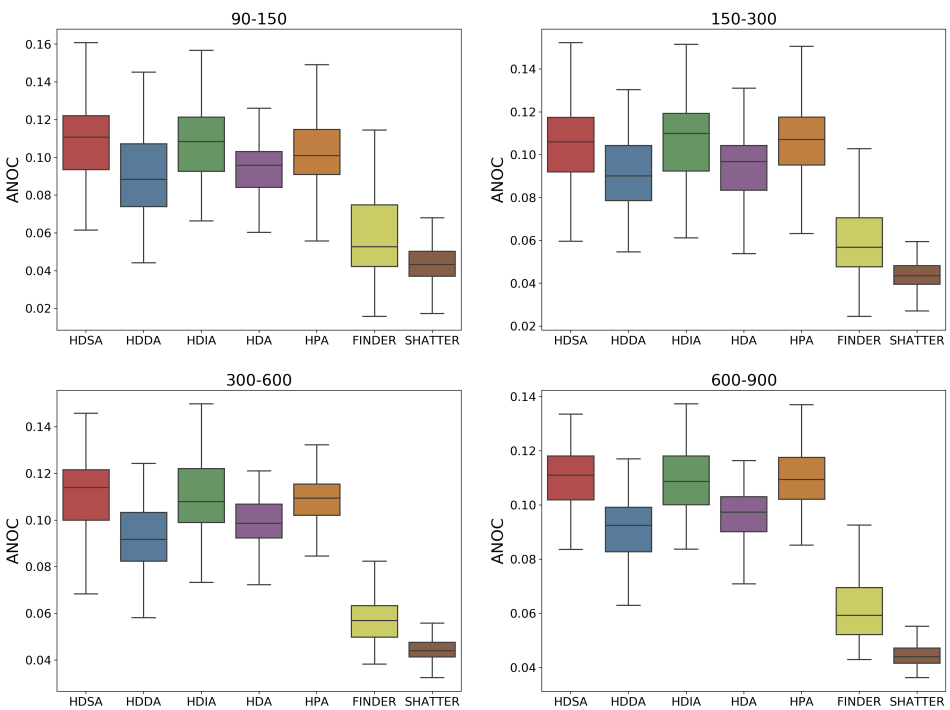



(b)

Figure 7: Comparison of SHATTER and baseline methods on synthetic HCNs based on random network models. (a) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes without considering the attack's cost. (b) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes considering the attack's cost.




(a)




(b)

Figure 8: Comparison of SHATTER and baseline methods on synthetic HCNs based on smallworld network models. (a) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes without considering the attack's cost. (b) Synthetic HCN validation sets containing 90-150, 150-300, 300-600, and 600-900 nodes considering the attack's cost.

409 ### A.4 Experiment on the FINC heterogeneous network

Table 6: Comparison of ANOC$\times 100$ obtained by different strategies on FINC heterogeneous network.

| Strategies | FINC-node-unweighted | FINC-node-random-weighted |
|---|---|---|
| **HDSA** | 2.04 | 18.50 |
| **HDDA** | 1.80 | 2.11 |
| **HDIA** | 11.24 | 8.74 |
| **HDA** | 1.86 | 2.07 |
| **HPA** | 2.36 | 2.76 |
| **FINDER** | 1.74 | 2.44 |
| **SHATTER** | **1.42** | **1.77** |