mathematics

MDPI

Article

Leveraging Minimum Nodes for Optimum Key Player Identification in Complex Networks: A Deep Reinforcement Learning Strategy with Structured Reward Shaping

Li Zeng [1,*], Changjun Fan [2] and Chao Chen [2]

[1] School of International Business and Management, Sichuan International Studies University, Chongqing 400031, China
[2] College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
* Correspondence: zengli13@sisu.edu.cn

Abstract: The problem of finding key players in a graph, also known as network dismantling, or network disintegration, aims to find an optimal removal sequence of nodes (edges, substructures) through a certain algorithm, ultimately causing functional indicators such as the largest connected component (GCC) or network pair connectivity in the graph to rapidly decline. As a typical NP-hard problem on graphs, recent methods based on reinforcement learning and graph representation learning have effectively solved such problems. However, existing reinforcement-learning-based key-player-identification algorithms often need to remove too many nodes in order to achieve the optimal effect when removing the remaining network until no connected edges remain. The use of a minimum number of nodes while maintaining or surpassing the performance of existing methods is a worthwhile research problem. To this end, a novel algorithm called MiniKey was proposed to tackle such challenges, which employs a specific deep Q-network architecture for reinforcement learning, a novel reward-shaping mechanism based on network functional indicators, and the graph-embedding technique GraphSage to transform network nodes into latent representations. Additionally, a technique dubbed 'virtual node technology' is integrated to grasp the overarching feature representation of the whole network. This innovative algorithm can be effectively trained on small-scale simulated graphs while also being scalable to large-scale real-world networks. Importantly, experiments from both six simulated datasets and six real-world datasets demonstrates that MiniKey can achieve optimal performance, striking a perfect balance between the effectiveness of key node identification and the minimization of the number of nodes that is utilized, which holds potential for real-world applications such as curbing misinformation spread in social networks, optimizing traffic in transportation systems, and identifying key targets in biological networks for targeted interventions.

Keywords: complex networks; combinatorial optimization; deep reinforcment learning; reward shaping

MSC: 90C27; 05C85

Citation: Zeng, L.; Fan, C.; Chen, C. Leveraging Minimum Nodes for Optimum Key Player Identification in Complex Networks: A Deep Reinforcement Learning Strategy with Structured Reward Shaping. *Mathematics* **2023**, *11*, 3690. https://doi.org/10.3390/math11173690

Academic Editor: Alessandro Niccolai

Received: 20 July 2023
Revised: 15 August 2023
Accepted: 24 August 2023
Published: 28 August 2023

1. Introduction

Complex networks hold substantial significance given their extensive reach and impact on diverse aspects of our lives. At the heart of complex networks lie the key players, also known as influential players [1], vital players [2], or critical players [3]. These players represent certain nodes, edges, or substructures that, when removed, can substantially degrade a network's specific functionality [4]. The importance of identifying these key players has profound implications in a variety of domains, such as epidemic control [5], drug design [6], viral marketing [7], criminal networks analysis [8] and combat network disintegration [9].

In the quest to identify key players in complex networks, many techniques and methods have been developed over the years, each with its own strengths and weaknesses. At one end of the spectrum, mathematical-programming-based methods represent one of the earliest and most fundamental approaches to this problem [10,11]. They tackle the problem by formulating it as an optimization challenge, using techniques such as integer programming, or mixed-integer programming to seek optimal solutions. However, as network size increases, these methods can quickly become computationally intensive, restricting their practical applications to larger-scale networks.

Another well-studied approach focuses on using network centrality measures such as degree centrality [12], k-core centrality [13], betweenness centrality [14], pagerank centrality [15], and collective influence (CI) [16]. These techniques identify key players based on their importance within the network, as determined by their centrality measure. Typically, before removing the next node, these algorithms adaptively recalculate the network's centrality indicators. The method that removes the node with the highest degree is termed HDA. Similarly, methods that remove nodes with the highest betweenness, pagerank, and collective influence are referred to as HBA, HPRA, and HCI, respectively. This process is repeated iteratively until there are no connected edges left in the network. However, while intuitively appealing and computationally efficient, these approaches can sometimes oversimplify the complex dynamics of the network. Additionally, they typically overlook node attributes, which can offer insights into a node's role or importance, leading to suboptimal results as they fail to account for the intricate interconnections among nodes.

Moving beyond these techniques, a wide range of heuristic-based methods have been developed, including greedy algorithms [17], acquaintance immunization [18], belief propagation dismantling [19,20], the reverse-reinsertion algorithm [21,22], and spectral partitioning [23]. These methods are designed around heuristics to select nodes for removal, with the aim of maximizing the impact on network functionality. Despite their innovative approaches, the performance of these methods can be heavily reliant on the specific heuristic that is used, and they may not always yield optimal results.

Evolutionary-algorithm-based strategies, on the other hand, have been inspired by processes found in nature. Techniques such as Tabu Search [24], genetic algorithms [25], simulated annealing [26], and artificial bee colonies [27] use these naturally inspired processes to guide the search for optimal node removal sequences. Despite their ability to handle complex problems and search a larger solution space, they are typically computationally intensive, and their probabilistic nature can impact the quality of the solutions they provide.

Continuing the exploration of these methodologies, the landscape of key player identification has been recently reshaped with the emergence of deep reinforcement learning (DRL) [28,29]. Different from traditional approaches, DRL learns from interactions within the environment and makes decisions based on the observed states, offering a more dynamic way to understand and manipulate complex networks. A representative work is the network key node identification algorithm framework called FINDER, based on deep reinforcement learning [30]. FINDER can be trained on small graphs and extended to millions of nodes, and it has a good identification performance. Since the FINDER algorithm operates on a single-objective deep reinforcement learning framework, it usually requires a large number of nodes to be removed to achieve optimal disintegration effects, even when there are no remaining edges in the network. How to achieve the best results with as few nodes as possible is an important issue that is worth studying.

In light of these issues, we introduce MiniKey (Leveraging Minimum Nodes for Optimum Key Players Identification in Complex Networks), a novel algorithm that integrates DRL with reward-shaping. MiniKey extends the original FINDER framework into a multi-objective optimization field, which can navigate the balance between functionality optimization and minimal node usage. By harnessing the power of DRL and reward shaping, it ensures a more efficient and robust learning process, resulting in the identification of fewer key players while still maintaining the network's optimal functionality.

To provide a vivid illustration of the effectiveness of MiniKey, we showcase its application in the domain of crime networks in Figure 1. This figure depicts the efficient performance of MiniKey, clearly demonstrating how it outperforms other methods, maintaining network connectivity while minimizing the number of utilized nodes. In a direct comparison using the 9/11 terrorist network as an example, MiniKey outperforms the original FINDER model, preserving network connectivity by using fewer nodes, which represents a significant step forward in this research area. This demonstration of MiniKey's superior performance emphasizes its potential in solving real-world problems, particularly in maintaining optimal network functionality while minimizing node usage.

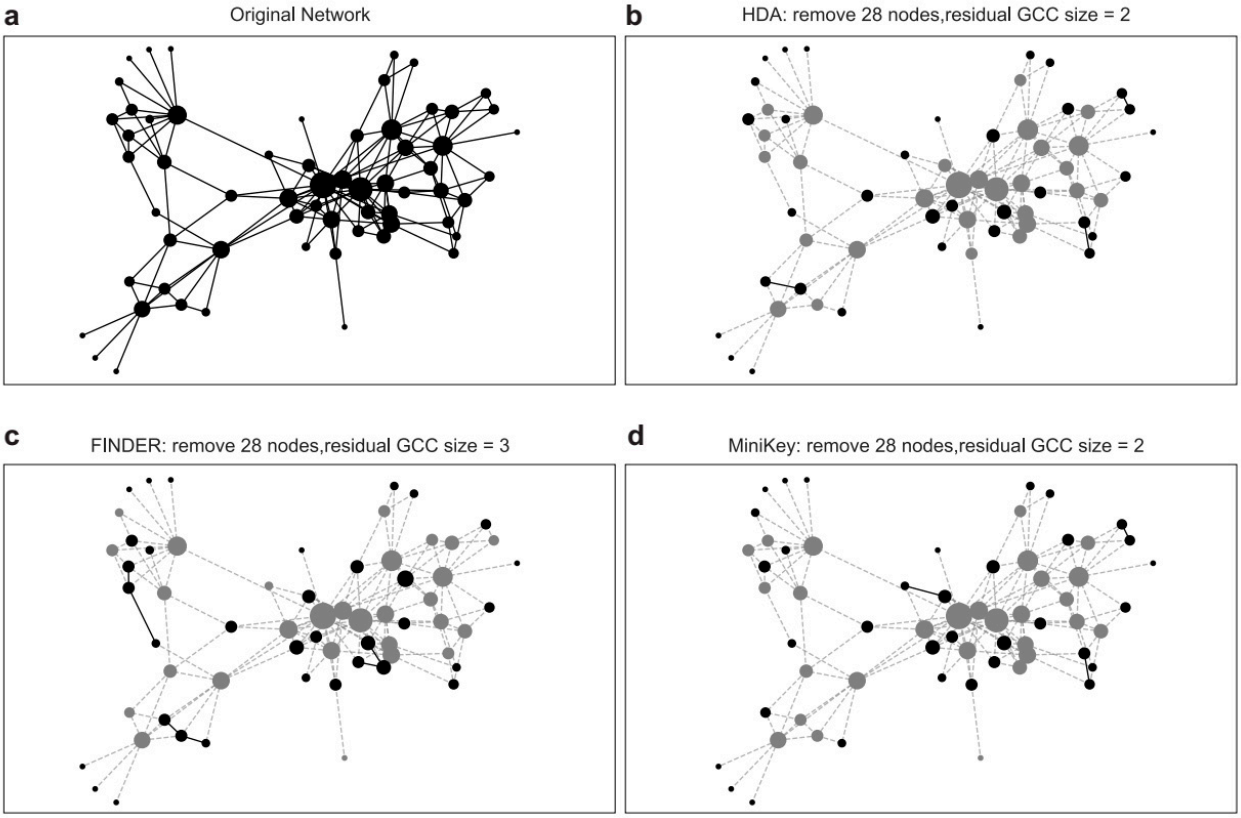

Figure 1. Leveraging minimum nodes for optimum key players' identification in complex net-works. (**a**) The 9/11 terrorist network with 62 terrorists (nodes) and 159 relations (edges). The size of a node corresponds to its degree. (**b**): HDA algorithm adaptively removes 28 nodes (grey); only 3 edges remain in the network, but this algorithm has a poor ANC score. (**c**): FINDER algorithm removes the same number of nodes as HDA, and 6 edges remain in the network. (**d**): MiniKey algorithm removes the same number of nodes as HDA; only 3 edges remain in the network and an optimal ANC score is achieved.

2. Preliminaries

2.1. Accumulated Normalize Connectivity (ANC)

Given a graph $G(V, E)$ with vertices set V, edge set E and a connectivity metric σ, where each edge E is a pair of distinct vertices $(u, v) \in V \times V$, the learning target in key players' identification is to find a optimal node removal sequence $(v_1, v_2, \cdots v_n)$ which minimizes the following accumulated normalize connectivity (ANC):

$$R(v_1, v_2, \cdots v_n) = \frac{1}{N} \sum_{k=1}^{n} \frac{\sigma(G \backslash \{v_1, v_2, \cdots v_n\})}{\sigma(G)} \tag{1}$$

In this paper, connectivity metric σ represents the size of the largest connected component remaining in the network, in which case the problem is referred to as the network

dismantling problem (NC) [31]. This problem involves the identification of a sequence of node removals that minimizes the size of the largest connected component remaining in the network. By addressing this issue, we aim to offer new insights into the process of network disintegration and propose more efficient strategies for identifying key players in complex networks.

It is worth noting that the connectivity metric σ can also represent various other network characteristics. For instance, it could represent the sum of the connectivity between nodes, referred to as the critical node problem (CN) [32]. It could also indicate the average geodesic distance [33] in the network among other possible measures.

2.2. Solution Length Ratio (SLR)

Given a graph $G(V, E)$, we define the solution length ratio (SLR) under a certain strategy as the ratio of the total number of nodes needed to disintegrate the network until no edges remain to the total number of nodes in the original network. This can be mathematically expressed as follows:

$$\text{SLR} = \frac{|V_S|}{|V|} \tag{2}$$

Here, $|V_S|$ is the number of nodes required to dismantle the network until no edges remain in the network; $|V|$ is the number of nodes in the original network $G(V, E)$. The importance of the SLR lies in its ability to measure the resource efficiency of a network-disintegration strategy. In practical scenarios, it is often preferable to achieve the network disintegration with as few node removals as possible due to resource constraints. Hence, a strategy with a lower SLR would be more desirable as it indicates less resource consumption in terms of node removals. When optimizing based solely on this metric, the problem can be transformed into the minimum vertex cover problem (MVC) [34].

2.3. Pareto Frontier

In the field of multi-objective optimization, the pareto frontier is characterized as the assortment of solutions (or, correspondingly, points within the objective function space) for which no other viable solution can be found that would decrease any given criterion without concurrently instigating an increase in at least one additional criterion.

Formally, for a problem with objectives to minimize, a solution X is said to dominate another solution Y if, and only if:

(1) For all objectives i, $1 \leq i \leq k$, the score of X on i is less than or equal to the score of Y on i;
(2) There exists at least one objective j, $1 \leq j \leq k$, such that the score of X on j is strictly less than the score of Y on j.

3. Model of MiniKey

Here, we employ a typical encoder–decoder architecture to model the key node identification problem on graphs. Simultaneously, we utilize a reinforcement learning algorithm with structured reward-shaping to train the entire MiniKey model. The design of each part of the algorithm is as follows.

3.1. Encoding Process of MiniKey

The identification of key players in complex networks heavily depends on the feature representation of elements within the network. Therefore, an effective feature-learning model can enhance the performance of the algorithm. In this work, we employ GraphSage [35] to encode the nodes within the network into a latent representation. Furthermore, we utilize a technique known as virtual node technology to capture the overall feature representation of the entire network. The detailed algorithmic framework of our graph encoding process is outlined as follows.

Algorithm 1 is the encoding process of MiniKey; given a graph $G(V, E)$, node features X_v, depth K, and learnable weight parameters $W_1 \in \mathbf{R}^{c\times p}$, $W_2 \in \mathbf{R}^{p\times(p/2)}$, $W_3 \in \mathbf{R}^{p\times(p/2)}$, the purpose of this algorithm is to produce node embeddings z_v and graph embedding z_s.

Algorithm 1: Encoding process of MiniKey

Input: Graph $G(V, E)$, node features $\{X_v \in \mathbf{R}^{1\times c}, \forall v \in V\}$, depth K, learnable weight parameters $W_1 \in \mathbf{R}^{c\times p}, W_2 \in \mathbf{R}^{p\times(p/2)}, W_3 \in \mathbf{R}^{p\times(p/2)}$
Output: Node embedding $z_v, \forall v \in V \cup \{s\}$
1. Add a virtual node s, which connects all nodes in $G(V, E)$
2. Initialize $h_v^{(0)} \leftarrow \mathrm{ReLU}(X_v \cdot W_1)$, $h_v^{(0)} \leftarrow h_v^{(0)} / \|h_v^{(0)}\|_2, \forall v \in V \cup \{s\}$
3. FOR $l = 1$ to K
4. FOR $v \in V \cup \{s\}$
5. $h_{\mathcal{N}(v)}^{(l-1)} \leftarrow \sum_{j\in\mathcal{N}(v)} h_j^{(l-1)}$
6. $h_v^{(l)} \leftarrow \mathrm{ReLU}([W_2 \cdot h_v^{(l-1)}, W_3 \cdot h_{\mathcal{N}(v)}^{(l-1)}])$
7. END FOR
8. $h_v^{(l)} \leftarrow h_v^{(l)} / \|h_v^{(l)}\|_2, \forall v \in V \cup \{s\}$
9. END FOR
10. $z_v \leftarrow h_v^{(K)}, \forall v \in V \cup \{s\}$

The algorithm starts by adding a virtual node s that connects to all nodes in the graph. This node is referred to as the graph state. The embeddings $h_v^{(0)}$ for the nodes and the virtual node are then initialized. The function Rectified Linear Unit (ReLU) is applied to the dot product of the node features X_v and the weight parameters W_1, and the result is then normalized using the L2 norm. The algorithm then enters a loop that runs K times. This loop corresponds to the K layers of the graph neural network. In each iteration, it calculates the new embeddings for each node and the virtual node. Inside this loop, there is a nested loop that runs for every node v in the graph and the virtual node. For each node v, the embedding $h_{\mathcal{N}(v)}^{(l-1)}$ is calculated by summing the embeddings $h_j^{(l-1)}$ of all nodes j in its neighborhood $\mathcal{N}(v)$. Then, the new embedding $h_v^{(l)}$ for each node v is calculated by applying the ReLU function to the concatenation of the dot product of W_2, and the previous embedding $h_v^{(l-1)}$ of the node and the dot product of W_3, and the newly calculated neighbor nodes embedding $h_{\mathcal{N}(v)}^{(l-1)}$. After all nodes have been processed, the new embeddings $h_v^{(l)}$ are normalized using the L2 norm. Finally, after K layers, the final embeddings z_v are calculated for each node and the virtual node. These are simply the final $h_v^{(K)}$ values. These embeddings can then be used for subsequent tasks, such as the identification of key players in the network.

3.2. Decoding Process of MiniKey

We use a two-layered MLPs to decode a state–action pair (s, a) to a scalar value $Q(s, a)$ that predicts the maximal rewards after taking action a in a given state s, which is defined as shown below:

$$Q(s, a) = W_5^\top \mathrm{ReLU}\left(z_a^\top \cdot z_s \cdot W_4\right) \tag{3}$$

Here, $W_4 \in \mathbf{R}^{p\times 1}$, $W_5 \in \mathbf{R}^{p\times 1}$, are learnable weight parameters, z_s and $z_a \in \mathbf{R}^{1\times p}$ are the state-embedding (Graph) and action-embedding, respectively, which are produced by Algorithm 1.

3.3. Training Algorithm of MiniKey

To leverage a minimum number nodes to find the optimum key players sets in complex networks, we formulate this problem as a Markov Decision Process (MDP) in a graph. The reinforcement learning components of MiniKey are outlined below:

States: The state, denoted as S in MiniKey, is the current configuration of the network graph G. It represents the partial solution to the key player identification problem. Each state is represented as a vector in a p-dimensional space, which is defined in Algorithm 1 as z_s.

Transitions: The transition function in MiniKey is deterministic and corresponds to the selection of a node $v \in G$ that is not yet part of the state S.

Actions: An action in MiniKey corresponds to the selection of a node v that is not currently part of the state S. The node is also represented by its p-dimensional embedding as z_v in Algorithm 1.

Rewards: The reward $r(S, v)$ in MiniKey can be be defined as ANC, i.e., the increase in ANC after selecting node v_k as the action and transitioning to the new state $S' = (S, v_k)$.Therefore, it can be expressed as:

$$r(v_k) = \frac{1}{N} \frac{\sigma(G \backslash \{v_1, v_2, \cdots v_k\})}{\sigma(G)} \tag{4}$$

The cumulative reward of a terminal state $\hat{S}$ aligns with the cost function value of $\hat{S}$:

$$R(\hat{S}) = \sum_{i=1}^{|\hat{S}|} r(S_i, v_i) = R(v_1, v_2, \cdots v_n) = \frac{1}{N} \sum_{k=1}^{n} \frac{\sigma(G \backslash \{v_1, v_2, \cdots v_n\})}{\sigma(G)} \tag{5}$$

In addition to considering the original ANC as the objective of reinforcement learning, we also incorporate an additional structural penalty as a constraint for the agent. The learning objective of MiniKey is to simultaneously optimize ANC and SLR, that is, to identify the optimal key nodes with as few nodes as possible, which could lead to multi-objective reinforcement learning [36,37].

Inspired by the network dismantling algorithm CoreHD [38], we observed that, for the network dismantling problem, star-shaped networks are more optimal for disintegration. In such cases, removing only the central node can disconnect many edges. Conversely, if there are too many nodes with degrees 1 or 2 in the largest connected component, a large number of nodes will need to be consumed in order to ensure that there are no remaining edges in the network. Nodes with a degree of 1 in a network are usually leaf nodes, and nodes with a degree of 2 are typically located in chains and cycles; the more of these two types of nodes there are, the more nodes need to be removed to ensure no edges exist in the network. In the extreme case of a chain of length N, to ensure that there are no edges, it would be necessary to remove N-1 nodes.

Based on these observations, we modified our reward function to include a penalty term, specifically, the number of nodes in the largest connected component of the remaining network with degrees 1 or 2. This design intends to minimize the presence of nodes with degrees 1 or 2 in the largest connected component while optimizing network dismantling. Experiments demonstrate that this crucial reward function design enables our MiniKey framework to identify the key nodes in the network with fewer nodes.

$$\begin{aligned} & R^{tot}(v_1, v_2, \cdots v_n) = R(v_1, v_2, \cdots v_n) + R^{penalty}(v_1, v_2, \cdots v_n) \\ & = \frac{1}{N} \sum_{k=1}^{n} \left(\frac{\sigma(G \backslash \{v_1, v_2, \cdots v_k\})}{\sigma(G)} \right) + \frac{1}{N} \sum_{k=1}^{n} \left(\frac{|N|_{1,2}^{LCC}(G \backslash \{v_1, v_2, \cdots, v_k\})}{\sigma(G)} \right) \\ & = \frac{1}{N} \sum_{k=1}^{n} \left(\frac{\sigma(G \backslash \{v_1, v_2, \cdots v_k\})}{\sigma(G)} + \frac{|N|_{1,2}^{LCC}(G \backslash \{v_1, v_2, \cdots, v_k\})}{\sigma(G)} \right) \end{aligned} \tag{6}$$

As illustrated in Equation (6), $R^{tot}(v_1, v_2, \cdots v_n)$ is the total reward of MiniKey, $R(v_1, v_2, \cdots v_n)$ is the original reward of NC, and $R^{penalty}(v_1, v_2, \cdots v_n)$ is the penalty term, where $|N|_{1,2}^{LCC}(G \backslash \{v_1, v_2, \cdots, v_k\})$ is the number of nodes with degrees of 1 or 2 in the largest connected component (LCC) of the left graph.

It is worth noting that the learning objective of MiniKey is to minimize the reward function $R^{tot}(v_1, v_2, \cdots v_n)$, and the original reward function $R(v_1, v_2, \cdots v_n)$ will inevitably

monotonically decrease as the network scale decreases. At this time, the optimal effect of the reward function $R^{tot}(v_1, v_2, \cdots v_n)$ after reward-shaping can only be achieved when the structural penalty reward $R^{penalty}(v_1, v_2, \cdots v_n)$ also decreases. That is, while reducing the number of nodes with a degree of 1 or 2 in the largest connected component, the size of the largest connected component in the remaining network decreases the fastest.

Policy: The policy in MiniKey is based on the approximated Q-function, $\hat{Q}$. A deterministic greedy policy $\pi(v|S) = \text{argmax}_{v' \in \overline{S}} \hat{Q}(z_s, v')$ is applied. When action v is taken, a node from G is added to the current partial solution, leading to a reward $r(S, v) = Q(s, a)$ which is defined in Equation (3).

Based on the aforementioned modeling, we use the DQN [39] algorithm to train MiniKey using simulated Barabási–Albert (BA) graphs as training samples. DQN is a variant of reinforcement learning where Q-Learning is combined with deep neural networks. This objective can be mathematically defined by the following cost function:

$$J(\theta) = \mathbb{E}_{(s,a,r,s') \sim D}[(r + \gamma \text{max}_{a'} Q(s', a'; \theta') - Q(s, a; \theta))^2] \quad (7)$$

In Equation (7), s represents the current state, a is the action taken, r is the reward received, and s' is the new state after taking action a. D is the experience replay memory, $Q(s, a; \theta)$ is the Q-value function approximated by the network with parameters θ, and γ is the discount factor.

4. Results

4.1. Results on Synthetic Graphs

During the training phase, we employed the Barabási Albert model (BA) with a default parameter setting: m = 4 (the number of edges attached from a new node to existing ones), and node number is uniformly chosen from the range [30, 50] (indicating that the node count varies between 30 and 50). All experiments were conducted on a platform equipped from Huawei Cloud with a Nvidia GeForce Tesla V100-32GB GPU.

In the testing phase, synthetic graphs were generated, maintaining m = 4, but with varying node numbers divided into scales: 30–50, 50–100, 100–200, 200–300, 300–400, and 400–500. For each node size category, we created 100 test graphs. Subsequently, we gauged the performance of several algorithms, namely, HDA [12], HBA [14], HCA [40], HPRA [15], FINDER [30], and MiniKey, on these generated datasets. This methodology ensured a comprehensive evaluation, permitting a detailed comparison of the methods across different graph sizes.

Figure 2 provides a comprehensive overview of the average performance (ANC and SLR) of various methods on these synthetic datasets. Importantly, we highlighted the error bars in the figure to depict the standard deviations of each method across the 100 test graphs. From Figure 2a, which shows the ANC results, we can observe that the MiniKey method generally outperforms the other techniques across all graph sizes. It achieves the lowest ANC in all categories except for the 30–50 and 50–100 range, where it is narrowly beaten by HBA. This indicates that MiniKey is more effective at identifying key nodes within the networks, as a lower ANC indicates a more effective disintegration of the largest connected component. The performance of MiniKey is especially noticeable in larger graph sizes (200–500), where it consistently outperforms the other methods. Figure 2b presents the SLR results across different graph sizes. The SLR measures the ratio of the nodes used by an algorithm to break the network completely. Here, MiniKey demonstrates an impressive performance again, consistently using fewer nodes compared to the majority of other methods across all graph sizes. This performance indicates a higher efficiency in utilizing network nodes for MiniKey. However, it is noteworthy that HDA method also showcases a competitive performance, with SLR closely following MiniKey's.

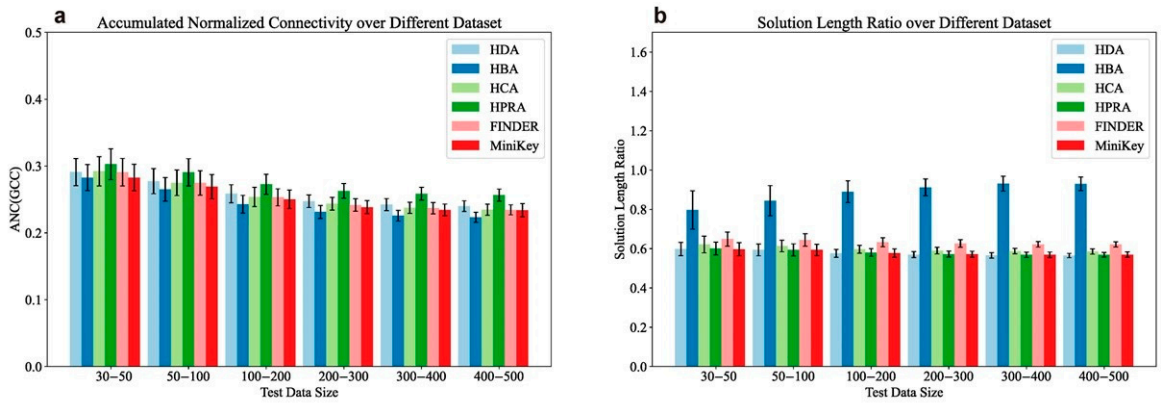

Figure 2. Accumulated Normalized Connectivity and Solution Length Ratio of different methods on six synthetic graphs. (**a**) Accumulated Normalized Connectivity of different methods on synthetic graphs; a lower value is preferable for this metric. (**b**) Solution Length Ratio of of different methods on synthetic graphs; a lower value is preferable for this metric.

Moveover, according to Figure 3, which illustrates the Pareto Front comparison of different methods on six simulated datasets, it is evident that the competitive performance of MiniKey in both identifying critical nodes within the network and efficiently using nodes to break down the network.

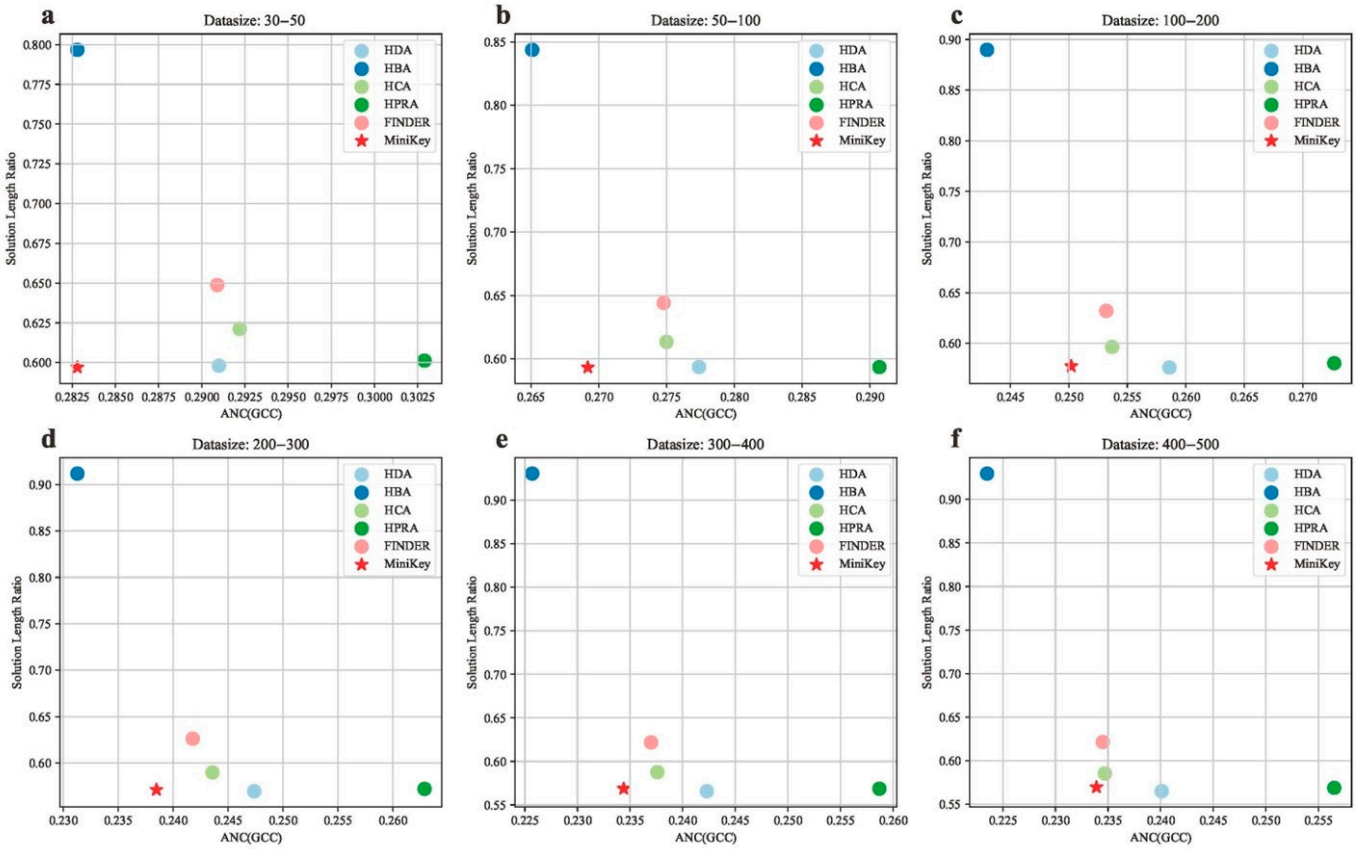

Figure 3. Comparison of pareto frontiers across different methods on six simulated datasets (**a**–**f**). MiniKey has the best performance when considering both ANC and SLR metrics simultaneously.

4.2. Results on Real-World Networks

During the testing phase on real-world graphs, we used the best model, trained on simulated BA graphs (with node range 30–50 and m = 4), as the default parameters for MiniKey. We selected six real datasets from SNAP Datasets [41]: Crime, HI-II-14, Digg,

Enron, Gnutella31, and Facebook. These datasets cover a wide range of fields, including criminal networks, biological networks, social networks, and communication networks. Table 1 presents the details of the network structures of these datasets, including node number, edge number, maximum degree, average degree, diameter, clustering coefficient and assortativity.

Table 1. Statistical analysis of network structures for six real-world datasets examined in this study.

Statistics/Dataset	Crime	HI-II-14	Digg	Enron	Gnutella31	Facebook
Node Number	829	4165	29,652	33,696	62,561	63,392
Edge Number	1473	13,087	84,781	180,811	147,878	816,831
Maximum Degree	25	286	310	1383	95	1098
Average Degree	3.55	6.28	5.72	10.73	4.73	25.77
Diameter	10	11	12	11	11	15
Clustering Coefficient	0.0058	0.0444	0.0054	0.5092	0.0055	0.2218
Assortativity	−0.1645	−0.2016	0.0027	−0.1165	−0.0927	0.1768

Furthermore, we compared MiniKey with network critical node identification algorithms that can run on large-scale networks, such as CI [16], MinSum [31], CoreHD [38], GND [23], and FINDER [30]. Table 2 and Figure 4, respectively, present the ANC score and the ANC curves of different methods on six real-world datasets. From the results, we can observe that FINDER and MiniKey consistently outperform the other methods across all datasets. Specifically, on Crime, HI-II-14, and Digg datasets, FINDER and MiniKey excel by providing the most accurate identification of key nodes. For the Gnutella31 and Facebook datasets, although BPD and MiniKey perform slightly better, FINDER still delivers results that are quite close to the top performers, indicating its effectiveness in key node identification.

Table 2. The ANC score of different methods on six real world dataset.

Method/Dataset	Crime	HI-II-14	Digg	Enron	Gnutella31	Facebook
CI	0.1243	0.0616	0.0866	0.0445	0.1174	0.2695
MinSum	0.1383	0.0652	0.0952	0.0477	0.1173	0.2725
CoreHD	0.1133	0.0606	0.0868	0.0514	0.1197	0.2747
GND	0.1381	0.0694	0.1066	0.0456	0.1257	0.2742
FINDER	0.1099	0.0554	0.0866	0.0430	0.1110	0.2684
MiniKey	0.1085	0.0566	0.0858	0.0509	0.1045	0.2810

Table 3 provides the solution length ratio (SLR), which measures the number of nodes each method uses to successfully decompose the network, a critical factor when evaluating efficiency. MiniKey exhibits an excellent performance in this respect, requiring the fewest nodes across all datasets to accomplish this task. It is particularly remarkable in the Crime and HI-II-14 datasets, significantly outperforming all other methods. FINDER also shows impressive results, especially on the Digg and Enron datasets, demonstrating high efficiency by maintaining a low SLR.

Additionally, as depicted in Figure 5, showing the pareto frontiers comparison of various methods across six real-world datasets, MiniKey evidently has a superior performance. In summary, both FINDER and MiniKey display an exceptional performance in critical node identification and the efficiency of network decomposition. MiniKey stands out due to its efficiency in maintaining a low SLR across all test datasets. This efficiency results in

resource savings and an enhanced performance, marking it as a highly effective solution for network-dismantling problems.

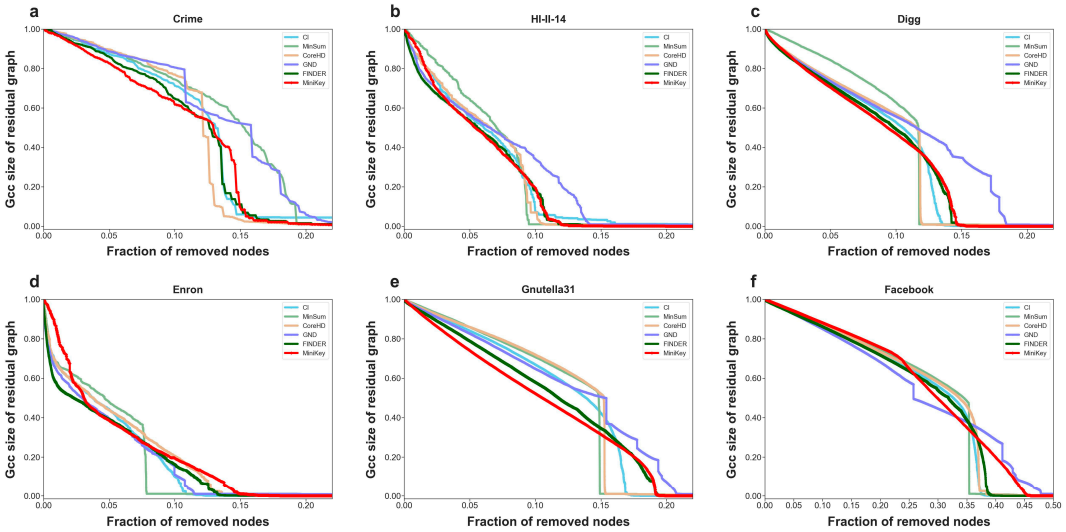

Figure 4. ANC curves for different methods on six real world datasets (**a**–**f**).

Table 3. The SLR score of different methods on six real-world datasets.

Method/Dataset	Crime	HI-II-14	Digg	Enron	Gnutella31	Facebook
CI	0.6755	0.9676	0.9985	1.0000	0.9901	0.9980
MinSum	0.6852	0.9676	0.9984	1.0000	0.9901	0.9976
CoreHD	0.6828	0.9676	0.9985	0.9619	0.9901	0.9776
GND	0.6683	0.9676	0.9985	1.0000	0.9901	0.9980
FINDER	0.5018	0.3541	0.4263	0.4901	0.2748	0.6547
MiniKey	0.4270	0.2936	0.3836	0.4626	0.2624	0.6157

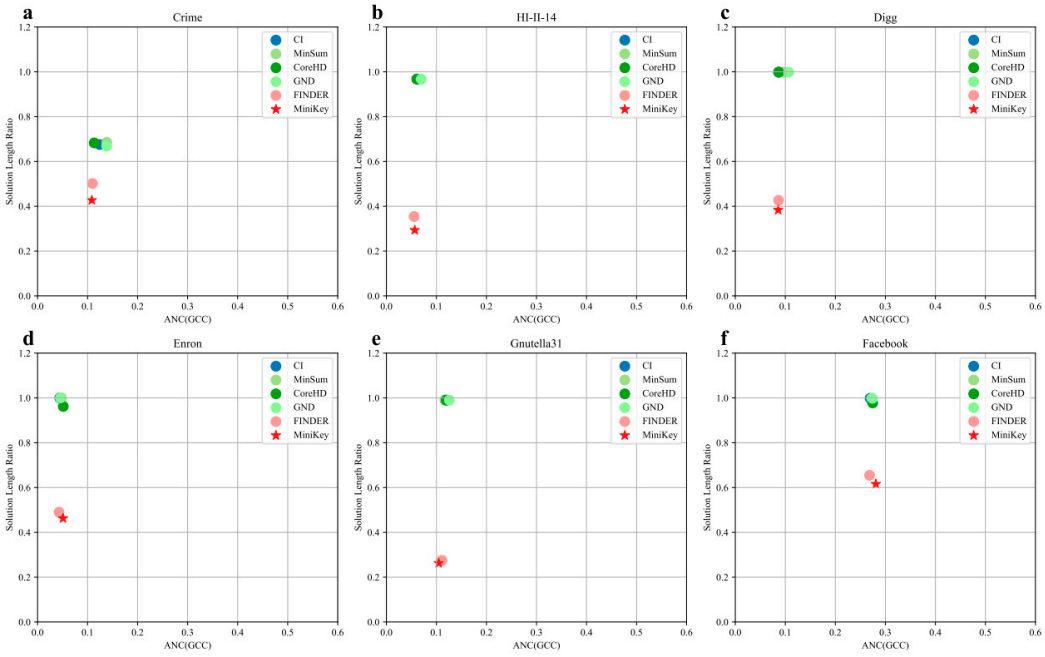

Figure 5. Comparison of pareto frontiers of different methods on six real-world datasets (**a**–**f**).

5. Discussion

The MiniKey algorithm proposed in this paper highlights a significant step forward in identifying key players in complex networks, emphasizing the use of minimal nodes. Evaluations across different datasets showcase its strength and flexibility. However, upon closer examination, clear challenges and details emerge, which are discussed as follows.

Scalability: while MiniKey exhibits notable proficiency with both small simulated graphs and extensive real-world datasets, care should be taken when navigating exceptionally large-scale networks. Networks characterized by complex structures or abundant node interactions might introduce challenges, potentially impacting the algorithm's reliability.

Dependency on Graph Representation: central to MiniKey's efficacy is its reliance on the graph representation technique, GraphSage. Nevertheless, as the landscape of graph neural networks (GNNs) continues to evolve, considering newer architectures might be beneficial. For instance, Graph Attention Network (GAT) [42], Message Passing Neural Networks (MPNN) [43], and Geometric Graph Convolutional Networks (Geom-GCN) [44] and et al. Incorporating or adapting components from these latest developments could potentially offer a more comprehensive embedding for Minikey, enhancing its capacity to tackle more intricate and diverse graph structures.

Structured Reward Shaping: our approach to reward-shaping is certainly innovative. However, it strongly relies on preset network functional markers. The absence of comprehensive markers or overlooking of pivotal network dynamics might steer the reinforcement learning agent away from the most fruitful actions.

Adaptability to Dynamic Networks: at present, MiniKey is fine-tuned to cater to static graphs. The ever-evolving landscape of dynamic networks, where node relationships fluctuate over time, presents a distinct challenge. Addressing this issue would require major changes to our current approach.

In conclusion, MiniKey presents a promising approach to optimum key players' identification with minimum nodes in complex networks; however, a comprehensive understanding and careful consideration of its limitations and intricacies are pivotal to harnessing its full potential and paving the way for future refinements in complex network analyses.

6. Conclusions

This paper presents MiniKey, an innovative approach to leveraging minimum nodes for optimum key players' identification in complex networks. Experiments conducted on a range of simulated and real-world datasets attest to the superior performance of MiniKey compared to other leading methods in terms of the Accumulated Normalized Connectivity (ANC) score and Solution Length Ratio (SLR). MiniKey outperforms existing strategies in its ability to identify and eliminate network edges, thereby proving its practical utility in various fields, including crime networks, social networks, communication networks, and bio-networks. Notably, MiniKey's unique strength lies in its efficiency, as it consistently uses fewer nodes to break network connectivity, providing resource savings and an enhanced performance. MiniKey reshapes our understanding of how to maintain or break network connectivity with minimal movements. Such insights can redefine current network analysis techniques, benefiting stakeholders in domains such as social media, urban planning, or epidemiology. Despite these promising results, the potential for future improvements and applications of MiniKey remains vast. Future research might focus on extending the approach to handling larger, more complex network structures or integrating it with other network analysis tools for more comprehensive network solutions. The adaptability and efficacy of MiniKey make it an exciting and promising frontier for network analysis.

Author Contributions: Conceptualization, methodology, formal analysis, investigation, resources, data curation, writing, L.Z. and C.F.; original draft preparation and writing, review and editing, L.Z., C.F. and C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the School-Level Scientific Research Project of Sichuan International Studies University, with the project number sisu202216.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Morone, F.; Makse, H.A. Influence Maximization in Complex Networks through Optimal Percolation. *Nature* **2015**, *524*, 65–68. [CrossRef]
2. Lü, L.; Chen, D.; Ren, X.-L.; Zhang, Q.-M.; Zhang, Y.-C.; Zhou, T. Vital Nodes Identification in Complex Networks. *Phys. Rep.* **2016**, *650*, 1–63. [CrossRef]
3. Lalou, M.; Tahraoui, M.A.; Kheddouci, H. The Critical Node Detection Problem in Networks: A Survey. *Comput. Sci. Rev.* **2018**, *28*, 92–117. [CrossRef]
4. Borgatti, S.P. Identifying Sets of Key Players in a Social Network. *Comput. Math. Organ. Theory* **2006**, *12*, 21–34. [CrossRef]
5. Pastor-Satorras, R.; Vespignanj, A. Epidemic Spreading in Scale-Free Networks. In *The Structure and Dynamics of Networks*; Princeton University Press: Princeton, NJ, USA, 2011; pp. 493–496.
6. Kuntz, I.D. Structure-Based Strategies for Drug Design and Discovery. *Science* **1992**, *257*, 1078–1082. [CrossRef] [PubMed]
7. Leskovec, J.; Adamic, L.A.; Huberman, B.A. The Dynamics of Viral Marketing. In Proceedings of the 7th ACM Conference on Electronic Commerce, Ann Arbor, MI, USA, 11–15 June 2006; ACM: New York, NY, USA, 2006.
8. Bright, D.; Greenhill, C.; Britz, T.; Ritter, A.; Morselli, C. Criminal Network Vulnerabilities and Adaptations. *Glob. Crime* **2017**, *18*, 424–441. [CrossRef]
9. Chen, L.; Wang, C.; Zeng, C.; Wang, L.; Liu, H.; Chen, J. A Novel Method of Heterogeneous Combat Network Disintegration Based on Deep Reinforcement Learning. *Front. Phys.* **2022**, *10*, 1021245. [CrossRef]
10. Walteros, J.L.; Veremyev, A.; Pardalos, P.M.; Pasiliao, E.L. Detecting Critical Node Structures on Graphs: A Mathematical Programming Approach. *Networks* **2018**, *73*, 48–88. [CrossRef]
11. Ventresca, M.; Aleman, D. A Derandomized Approximation Algorithm for the Critical Node Detection Problem. *Comput. Oper. Res.* **2014**, *43*, 261–270. [CrossRef]
12. Hooshmand, F.; Mirarabrazi, F.; MirHassani, S.A. Efficient Benders Decomposition for Distance-Based Critical Node Detection Problem. *Omega* **2020**, *93*, 102037. [CrossRef]
13. Albert, R.; Jeong, H.; Barabási, A.-L. Error and Attack Tolerance of Complex Networks. *Nature* **2000**, *406*, 378–382. [CrossRef]
14. Carmi, S.; Havlin, S.; Kirkpatrick, S.; Shavitt, Y.; Shir, E. A Model of Internet Topology Using K-Shell Decomposition. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 11150–11154. [CrossRef]
15. Wandelt, S.; Sun, X.; Feng, D.; Zanin, M.; Havlin, S. A Comparative Analysis of Approaches to Network-Dismantling. *Sci. Rep.* **2018**, *8*, 13513. [CrossRef]
16. Qin, J.; Xu, J.J.; Hu, D.; Sageman, M.; Chen, H. Analyzing Terrorist Networks: A Case Study of the Global Salafi Jihad Network. In *Intelligence and Security Informatics*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 287–304.
17. Aringhieri, R.; Grosso, A.; Hosteins, P.; Scatamacchia, R. VNS Solutions for the Critical Node Problem. *Electron. Notes Discret. Math.* **2015**, *47*, 37–44. [CrossRef]
18. Mahdavi Pajouh, F.; Boginski, V.; Pasiliao, E.L. Minimum Vertex Blocker Clique Problem. *Networks* **2014**, *64*, 48–64. [CrossRef]
19. Zhou, H.-J. Spin Glass Approach to the Feedback Vertex Set Problem. *Eur. Phys. J. B* **2013**, *86*, 455. [CrossRef]
20. Qin, S.-M.; Ren, X.-L.; Lü, L.-Y. Efficient Network Dismantling via Node Explosive Percolation. *Commun. Theor. Phys.* **2019**, *71*, 764. [CrossRef]
21. Šimon, M.; Dirgová Luptáková, I.; Huraj, L.; Hosťovecký, M.; Pospíchal, J. Combined Heuristic Attack Strategy on Complex Networks. *Math. Probl. Eng.* **2017**, *2017*, 6108563. [CrossRef]
22. Fan, C.; Zeng, L.; Feng, Y.; Xiu, B.; Huang, J.; Liu, Z. Revisiting the Power of Reinsertion for Optimal Targets of Network Attack. *J. Cloud Comput.* **2020**, *9*, 24. [CrossRef]
23. Ren, X.-L.; Gleinig, N.; Helbing, D.; Antulov-Fantulin, N. Generalized Network Dismantling. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 6554–6559. [CrossRef]
24. Deng, Y.; Wu, J.; Tan, Y. Optimal Attack Strategy of Complex Networks Based on Tabu Search. *Phys. A Stat. Mech. Its Appl.* **2016**, *442*, 74–81. [CrossRef]
25. Zhou, Y.; Hao, J.-K.; Fu, Z.-H.; Wang, Z.; Lai, X. Variable Population Memetic Search: A Case Study on the Critical Node Problem. *IEEE Trans. Evol. Comput.* **2021**, *25*, 187–200. [CrossRef]
26. Arulselvan, A.; Commander, C.W.; Elefteriadou, L.; Pardalos, P.M. Detecting Critical Nodes in Sparse Graphs. *Comput. Oper. Res.* **2009**, *36*, 2193–2200. [CrossRef]
27. Lozano, M.; García-Martínez, C.; Rodríguez, F.J.; Trujillo, H.M. Optimizing Network Attacks by Artificial Bee Colony. *Inf. Sci.* **2017**, *377*, 30–50. [CrossRef]
28. Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; Song, L. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*; NIPS: Grenada, Spain, 2017.

29. Fan, C.; Shen, M.; Nussinov, Z.; Liu, Z.; Sun, Y.; Liu, Y.-Y. Searching for Spin Glass Ground States through Deep Reinforcement Learning. *Nat. Commun.* **2023**, *14*, 725. [CrossRef]
30. Fan, C.; Zeng, L.; Sun, Y.; Liu, Y.-Y. Finding Key Players in Complex Networks through Deep Reinforcement Learning. *Nat. Mach. Intell.* **2020**, *2*, 317–324. [CrossRef] [PubMed]
31. Braunstein, A.; Dall'Asta, L.; Semerjian, G.; Zdeborová, L. Network Dismantling. *Proc. Natl. Acad. Sci. USA* **2016**, *113*, 12368–12373. [CrossRef]
32. Sarker, S.; Veremyev, A.; Boginski, V.; Arvind, S. Critical Nodes in River Networks. *Sci Rep.* **2019**, *9*, 11178. [CrossRef]
33. Holme, P.; Kim, B.J.; Yoon, C.N.; Han, S.K. Attack Vulnerability of Complex Networks. *Phys. Rev. E* **2002**, *65*, 056109. [CrossRef]
34. Dinur, I.; Safra, S. On the Hardness of Approximating Vertex Cover. *Ann. Math.* **2005**, *162*, 439–485. [CrossRef]
35. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*; NIPS: Grenada, Spain, 2017.
36. Liu, C.; Xu, X.; Hu, D. Multiobjective Reinforcement Learning: A Comprehensive Overview. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 385–398. [CrossRef]
37. Van Moffaert, K.; Nowé, A. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.* **2014**, *15*, 3483–3512.
38. Zdeborová, L.; Zhang, P.; Zhou, H.-J. Fast and Simple Decycling and Dismantling of Networks. *Sci. Rep.* **2016**, *6*, 37954. [CrossRef]
39. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
40. Bavelas, A. Communication Patterns in Task-Oriented Groups. *J. Acoust. Soc. Am.* **1950**, *22*, 725–730. [CrossRef]
41. Leskovec, J.; Krevl, A. *SNAP Datasets: Stanford Large Network Dataset Collection*; SNAP: Santa Monica, CA, USA, 2014.
42. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
43. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural Message Passing for Quantum Chemistry. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
44. Pei, H.; Wei, B.; Chang, K.; Lei, Y.; Yang, B. Geom-GCN: Geometric Graph Convolutional Networks. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.