Finding Minimum Node Separators: A Markov Chain Monte Carlo Method

Joohyun Lee[a,1], Jaewook Kwak[b], Hyang-Won Lee[c,2,*], Ness B. Shroff[b,d]

[a]*Division of Electrical Engineering, Hanyang University, Ansan, Korea*
[b]*Department of Electrical and Computer Engineering, The Ohio State University*
[c]*Department of Software, Konkuk University, Seoul, Korea*
[d]*Department of Computer Science and Engineering, The Ohio State University*

Abstract

In networked systems such as communication networks or power grids, graph separation from node failures can damage the overall operation severely. One of the most important goals of network attackers is thus to separate nodes so that the sizes of connected components become small. In this work, we consider the problem of finding a minimum α-separator, that partitions the graph into connected components of sizes at most αn, where n is the number of nodes. To solve the α-separator problem, we develop a random walk algorithm based on Metropolis chain. We characterize the conditions for the first passage time (to find an optimal solution) of our algorithm. We also find an optimal cooling schedule, under which the random walk converges to an optimal solution almost surely. Furthermore, we generalize our algorithm to non-uniform node weights. We show through extensive simulations that the first passage time is less than $O(n^3)$, thereby validating our analysis. The solution found by our algorithm allows us to identify the weakest points in the network that need to be strengthened. Simulations in real topologies show that attacking a dense area is often not an efficient solution for partitioning a network into small components.

Keywords: Graph separation problem, Node attack, Markov Chain Monte Carlo, Metropolis algorithm, Hierarchical Markov chain

1. Introduction

Today's critical infrastructures are organized in the form of a network such as the communication network, the smart electrical power grid or the water distribution system. In

*H. Lee is the corresponding author. Postal address: 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Korea

Email addresses: joohyunlee@hanyang.ac.kr (Joohyun Lee), kwak.84@osu.edu (Jaewook Kwak), leehw@konkuk.ac.kr (Hyang-Won Lee), shroff.11@osu.edu (Ness B. Shroff)

[1]A preliminary version of this work was presented at DRCN (International Conference On Design Of Reliable Communication Networks) 2017 [1].

[2]Hyang-Won Lee was supported by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(2015R1A1A1A05001477).

these networks, small node failures can have a significant impact on connectivity and lead to graph separation [2, 3], where components become more reliable as their sizes increase. For example, in a power grid, there should be a sufficient amount of power generation to serve the power loads while not every bus (frequently represented as nodes in the graph modeling power grid) can have a generator. Protecting minimum separating nodes can keep the sizes of components higher than a given threshold, thereby increasing the chance of load being served. Thus, identifying the weak parts in practical networks is a major concern in general. Graph separation can also incur further cascading failures, because the amount of resource in the remaining components (e.g., the amount of generated power) becomes so reduced that the remaining components are overloaded and subsequently failed [4]. Thus, one of the most important goals of network attackers is to separate nodes such that the sizes of connected components become small. However, attackers have resource constraints as well, and they would like to inflict the greatest harm with limited cost. A defender wants to identify these weakest points in advance. In this work, we consider the problem of finding a minimum α-separator that partitions the graph into connected components of sizes smaller than αn, where n is the number of nodes in the graph. Finding a minimum α-separator is proven to be NP-hard for a general topology for $\alpha \geq \frac{2}{3}$ [5]. For topologies such as trees and cycles, the authors in [6] have developed polynomial-time algorithms, yet they require knowledge of the type of the graph topology in advance.

To tackle the minimum α-separator problem without prior knowledge of the graph topology, we apply a Markov Chain Monte Carlo (MCMC) method. The basic idea is to solve this combinatorial optimization problem by constructing a random walk over a Markov chain, which traverses through feasible solutions, where the transitions lead the system to move to states with desired objectives (smaller α-separator in our case). In other words, the stationary distribution is higher for states with better objective values (i.e., closer to the minimum). The Markov Chain Monte Carlo method has been applied in many NP-hard problems in various applications [7, 8, 9, 10, 11]. In our random walk algorithm, we additionally design a simple data structure to quickly identify the sizes of the components that vary under the random walk in each step. This allows us to further reduce the computational complexity in each step.

We then analytically characterize how long it takes to obtain the optimal solution by our algorithm. The standard metric often used in the literature is the *mixing time*, the convergence time until the Markov chain is close to its stationary distribution, and there are several existing works showing the polynomial convergence of the chain. For the independent set problem, [12, 13, 14] used a coupling technique to show polynomial convergence. However, conditions for guaranteeing polynomial convergence are limited. For example, in [12], the maximum node degree of a graph should be less than or equal to five, for polynomial convergence. Hence, in this paper we present an approximate analysis on the *first passage time*, using a hierarchical structure of the underlying Markov chain. We then provide sufficient conditions for the expected first passage time to be polynomial. As an example, we compute the expected first passage time in star-like topologies, which is $O(n^2)$, where n is the number of nodes. The conditions also include simple topologies that the sizes of minimum α-separators do not scale with the network size, e.g., line, circle, and balanced tree.

To evaluate the performance and support our analytical results, we run our random walk algorithm over various topologies including real topologies from US fiber networks of 20 Internet providers [15] and Italian power grid and Internet networks [16, 17]. In all topologies, our random walk algorithm converges to a solution within $O(n^3)$ steps, for a network with n nodes. We also compare our random walk algorithm with other heuristic algorithms including highest-degree-first and greedy algorithms, and illustrate the performance improvements we obtain. We underscore that the solution from our algorithm allows us to characterize the weakest points in the network that need to be strengthened. In real topologies, we find that attacking a dense area may not be an efficient way to partition a network graph which can lead to large cascading effects. Finally, we discuss some future directions to apply our general results to defense problems and practical networks such as power grids and water distribution systems in consideration of physical dynamics. Even though our graphical model is simple to model physical dynamics in practical systems, our work can infer initial weak points in these complex systems. As a future work, we will work on developing an algorithm to find weak points in networks with complex physical dynamics (e.g., correlated failures).

2. Related work

We classify previous work into several categories and summarize them as follows.

Network Vulnerability and Reliability: In the network science literature, a famous paper by Paolo et al. [18] analyzed the size of the largest component after node removals and studied the ability to resist failures depending on the type of attacks (i.e., random or targeted node attacks) and the type of networks (e.g., Erdos-Renyi random graph and the Barabasi-Albert scale-free power-law network). They showed that the power-law graph that appears commonly in practice[3] is resilient to random attacks, but is vulnerable to targeted attacks. Balthrop et al. [21] studied the spread of viruses in different types of networks.

There have been several studies to provide a quantitative assessment of how failures affect the network structure. In [22], authors proposed a new measure, *pairwise connectivity*, to assess the vulnerability of networks. They showed that computing the pairwise connectivity is NP-hard, and proposed pseudo-approximation algorithms to obtain the pairwise connectivity. Zhang et al. [23] proposed a reliability metric that describes the average reliability between every pair of nodes in a network. They formulated and solved the critical component detection problem to identify the critical components that affect the reliability metric. In [24], Kabadurmus et al. proposed a new metric that combines network reliability with network resilience is presented to measure reliability/survivability effectively for capacitated networks. Authors in [25] studied a measure of the resilience of these structures based on community similarity before and after disruption and applied the approach to an electric power network. In [26], Li et al. compared connectivity reliability (CR) and topological controllability (TC) metrics, and developped a controllability index and a controllability-based node importance metric. In [27], authors presented a new methodology for quantifying the

[3]Several papers such as [19, 20] showed that many real-life networks follow the power-law degree distribution.

reliability of complex systems, using techniques from network graph theory, and showed that many real-world systems are vulnerable to the spatially coherent hazard. Authors in [28] applied the universal generating function technique (UGFT) to determine the network reliability of the mobile ad-hoc networks, which is defined as the probability that the transformed message from the source can be passed successfully without any delay. Ferrario et al. [29] proposed a multi-state hierarchical graph to evaluate the robustness of interdependent critical infrastructures using Monte Carlo simulations. Zhang et al. [30] proposed k-reliability for estimation of potential cascading failures, which is defined as the probability that at least k surviving nodes span an operating component, as the operation probability of the system and reflects the connectivity of nodes in networks. In this work, we focus on the size of the largest component as the reliability metric, and try to find the most critical nodes in a network.

Deterministic approaches and inapproximability of the α-separator problem: Mohamed et al. [6] developed polynomial-time algorithms for special topologies such as trees and cycles. Shen et al. [31] also developed polynomial-time dynamic programming algorithms on tree structures and series-parallel graphs. However, there is little known for general topologies as far as we know. They also developed a simple greedy algorithm, whose approximation ratio is $\alpha n + 1$. Proposition 7.3 in [6] also showed the maximum α-coseparator[4] problem cannot be approximated in polynomial time within a factor of $\left(\frac{1}{\alpha}\right)^{1-\epsilon}$ for any constant $\epsilon > 0$. Wachs et al. [32] developed a heuristic algorithm and studied the node separators for various α in the Internet Autonomous Systems.

Related problems of α-separator problem: If $\alpha = 1/n$, the α-separator problem becomes the minimum vertex cover problem. The case $\alpha = 2/n$ is equivalent to the minimum dissociation set problem, where the dissociation set is a subset of vertices, whose induced subgraph has the maximum degree of 1. Thus, the α-separator problem is a generalized version of these problems. Both problems are NP-hard [33]. We note that the complement of a minimum vertex cover is a maximum independent set, and the maximum independent set problem is studied in several papers using the MCMC approach [12, 13, 14]. There are several NP-hard graph separating problems with different formulations (e.g., cutting edges or removing nodes) including [34, 35].

Markov Chain Monte Carlo method in reliability engineering: The Markov Chain Monte Carlo (MCMC) method has been extensively applied to many difficult and NP-hard problems in reliability engineering. In [36], authors considered networks that consist of system components operating in a randomly changing environment, and provided a Bayesian network model to assess the network reliability. They demonstrated the efficiency of their proposed MCMC method under this model in the reliability prediction. Similarly, Authors in [37] adopted a Bayesian network model for the network failure identification problem in consideration of the fact that many important network systems such as power grids and water distribution systems exhibit nonlinear dynamics with uncertainty. Zuev et al.[38] adopted the MCMC method to compute the network failure probability. Beck et al. [39] proposed an

[4]An α-coseparator is the complement of an α-separator.

adaptive MCMC simulation approach to evaluate the integral of posterior probabilities in a full Bayesian probabilistic framework for robust system identification, where the integration is difficult because the dimension of the parameter space is usually too large for direct numerical integration. Authors in [40, 41] used MCMC to solve reliability estimation and residual life estimation problems.

3. System Model

We consider a simple graph $G = (V, E)$ with $|V| = n$. We denote $\mathcal{N}(v)$ as a set of neighbors of a node $v \in V$. We assume that the attack cost is homogeneous across nodes. Our results can be readily extended to the case of heterogeneous attack costs, which will be discussed later. *An α-separator* W for $1/n \leq \alpha < 1$ is defined as a subset of nodes such that the sizes of all components in the graph $G \setminus W$ is smaller than or equal to αn.[5] We call this an α-separating condition. $G \setminus W = (V \setminus W, E(V \setminus W))$ is obtained after removing the nodes in W and all their incident edges, where $E(V \setminus W) = \{(i, j) \in E | i \in V \setminus W, j \in V \setminus W\}$.

The α-separator problem for $1/n \leq \alpha < 1$ is to find a minimum α-separator[6] as follows:

$$\min_{W \subseteq V} |W| \quad \text{subject to} \quad f(G \setminus W) \leq \alpha n, \tag{1}$$

where $f(G)$ is the size of the largest connected component in G. We denote $m = \lfloor \alpha n \rfloor$.

In Proposition 7.3 of [6], the authors show that there is a polynomial reduction from the minimum vertex cover problem to the α-separator problem. Therefore, *the α-separator problem is NP-hard.*

We define H as the set of α-separators. In the next section, we develop a random walk algorithm over H such that the random walk is more likely to jump towards better solutions (i.e., α-separator with a smaller size). Then, we will analyze the properties of this random walk algorithm.

4. Random walk algorithm

We develop a random walk algorithm over a Metropolis chain for the α-separator problem. Our random walk algorithm takes one of three actions in each step: (1) remove v from the attack set W, (2) add a vertex v in W, and (3) stay at the current state. To compute the sizes of connected components after each action in an efficient way, we apply a simple data structure and an update mechanism, which will be explained shortly. The formal algorithm description is as follows:

A random walk algorithm for the α-separator problem

[5] From the attacker's point of view, the value of α is determined by the topology and physical properties of the network. For instance, with the same topology, power grid and water distribution networks may have different thresholds for the size of the connected component which trigger cascading failures.

[6] The minimum α-separators may not be unique.

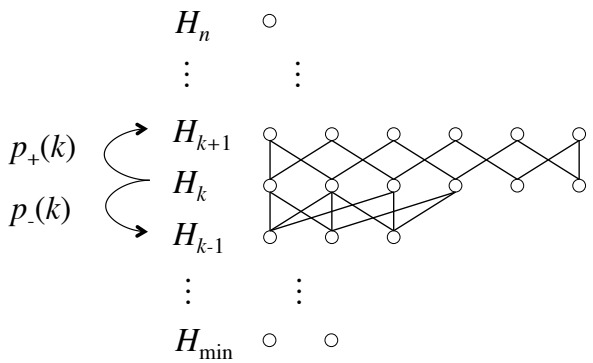


Figure 1: Illustration of a hierarchical Markov chain of α-separators. Each circle indicates a state (i.e., α-separator) and two neighboring states are connected by a line, if they can be transitioned to another state in one step with vertex addition (i.e., towards higher hierarchy) or removal (i.e., towards lower hierarchy). Note that vertex addition is accepted with probability ρ and vertex removal is accepted only if the attack set after vertex removal $W \setminus \{v\}$ satisfies the α-separating condition (i.e., the attack set is an α-separator.).

1: $W = V$; $W_{\min} = W$.

2: $\text{head}(v) = v, \forall v \in V$; $\text{cluster}(v) = \{v\}, \forall v \in V$.

3: **for** step $= 1$ **to** numStep

4: Pick a vertex $v \in V$ uniformly at random.

5: **If** $v \in W$,

6: newClusterSize $= 1 + \sum_{i \in \cup_{j \in \mathcal{N}(v) \setminus W} \text{head}(j)} |\text{cluster}(i)|$.

7: **if** newClusterSize $\leq \alpha n$,

8: $W = W \setminus \{v\}$; **if** $|W| < |W_{\min}|$, $W_{\min} = W$.

9: $\text{cluster}(v) = \cup_{i \in \cup_{j \in \mathcal{N}(v) \setminus W} \text{head}(j)} \text{cluster}(i) \cup \{v\}$.

10: **for** $i \in \text{cluster}(v)$,

11: $\text{head}(i) = v$.

12: **end for**

13: **end if**

14: **else if** $v \notin W$, with probability ρ,

15: $W = W \cup \{v\}$.

16: head$(i) = i$ for $i \in \mathcal{N}(v) \setminus W$.

17: **for** $i \in \mathcal{N}(v) \setminus W$,

18: **if** head$(i) = i$,

19: cluster$(i) \leftarrow$ graphTraverse$(i, G \setminus W)$

20: head$(j) = i, \forall j \in$ cluster(i)

21: **end if**

22: **end for**

23: **end if**

24: **end for**

Our random walk algorithm runs over an underlying Markov chain (also called a Metropolis chain specifically), as illustrated in Fig. 1 (we will explain the hierarchy later). Each α-separator is a state of the Markov chain. The stationary distribution derived from the random walk algorithm that we designed is as follows [42].

$$\pi(W) = \frac{\rho^{|W|}}{Z}, \tag{2}$$

where Z is a normalization constant. This can be easily checked from balance equations. The parameter ρ quantifies the tradeoff between the quality of the solution and the convergence speed to the steady state. For $\rho < 1$, α-separators with a smaller size have higher stationary distribution. As ρ becomes smaller, the random walk favors states with smaller sizes and the stationary distribution gets higher for optimal solutions (i.e., minimum α-separators). However, a very small value of ρ may lead to prohibitively long convergence time. An important question in a random walk algorithm over a Markov chain is how long it takes to obtain the optimal solution, e.g., the first passage time to the optimal states. We conduct the first passage time analysis in the next section. Here, we first describe the main procedures of the random walk algorithm, and explain the data structure for efficient computation of component sizes.

The random walk algorithm works as follows. We illustrate the transitions of our random walk algorithm in Fig. 2. Initially, the attack set W is set to be the set of all vertices, V (line 1). $W_{\min}$ tracks the smallest α-separator that has been found. Trivially, $W = V$ is an α-separator for any α. At each step, we first pick a vertex v randomly (line 4). There are two cases: (i) $v \in W$ and (ii) $v \notin W$. For the first case, our algorithm aims to remove v from the attack set W. The α-separating condition is checked in lines 6-7, which will be explained shortly. If $W \setminus \{v\}$ is an α-separator, this jump to the lower hierarchy (W to $W \setminus \{v\}$) is accepted. For the second case, our algorithm aims to add v in W. Since $W \cup \{v\}$

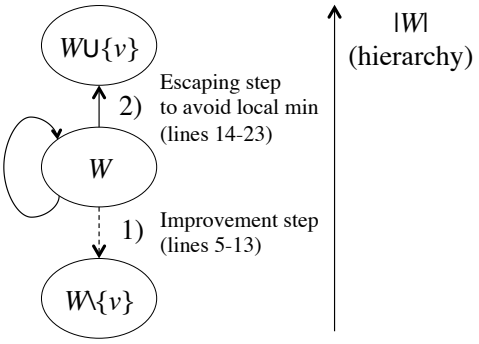
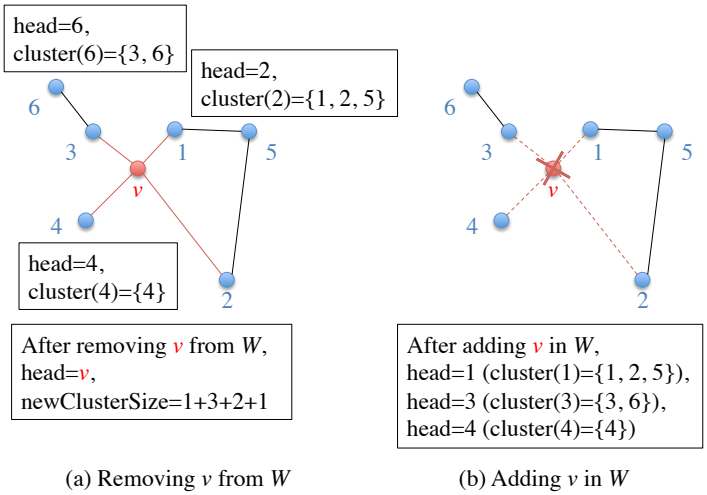


Figure 2: The transitions in the random walk algorithm.



(a) Removing v from W (b) Adding v in W

Figure 3: An example of vertex removal and addition to W and updates of heads and clusters.

is always an α-separator if W is an α-separator, we do not need to check the α-separating condition. We accept this jump with probability ρ for $\rho < 1$. This backward jump towards a higher state may slow down the random walk to optimal solutions, but it is necessary to escape local optima. This random walk algorithm leads the stationary distribution of states given in Eq. (2).

An integral step to realize our random walk algorithm is to implement a sub-module that computes the size of the largest connected component, after adding or removing a vertex from the attack set, W. This requires knowledge of the sizes of all connected components. To do this efficiently, we construct a simple data structure for each vertex and connected component, as illustrated in Fig. 3. In our data structure, each vertex v keeps track of a head vertex $\mathrm{head}(v)$, which is the head vertex of the connected component that the node v belongs to. There is only one head vertex in each connected component. The head vertex h of each connected component records all the associated vertices in $\mathrm{cluster}(h)$, which is the set of nodes in the head vertex h's connected component.

Updating this data structure after removing or adding a vertex v from W is as follows (see Fig. 3). If $v \in W$, we first compute the size of the connected component that v is associated

with. Since this component is the only one that changes its size, if the size of this component is less than or equal to αn, the vertex addition of v is acceptable. When the new connected component satisfies the condition, a vertex v is removed from W. The vertex v is assigned as a head vertex of this component and the set of associated vertices (i.e., cluster) is updated accordingly. If $v \notin W$, the vertex v is added to W with probability ρ. After v is included in W, the connected component that v was associated with can be divided into multiple components. To track this, we need to traverse from each neighboring vertex of v, until all the neighboring vertices are visited (lines 16-22). The procedure, graphTraverse$(i, G \setminus W)$, traverses the graph $G \setminus W$ starting from node i and returns the connected component (i.e., cluster) of node i. Note that if some neighboring nodes are connected to the node i (e.g., nodes 2 and 1 in Fig. 3 (b)), the head node of these nodes will be updated as i in line 20. It can be implemented as either breadth-first search or depth-first search, where the complexity of them is $O(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges in $G \setminus W$. Note that the number of graph traversals is at most the number of neighboring nodes, which is $O(n)$. The complexity in vertex removal from W is $O(|V|)$ and complexity in vertex addition to W is $O(|V| + |E|)$. Therefore, the overall complexity is $O(|V|^2 + |V||E|)\times$numStep, or $O(n^3)\times$numStep, where numStep is the number of steps to run.

In our random walk algorithm, "numStep" should be chosen appropriately to ensure that a minimum α-separator will be achieved. However, in MCMC approaches, it is hard to tell when the chain converges to equilibrium in general. One can use diagnostic tests [43] such as the Geweke diagnostic [44] to detect the convergence of a Markov chain. Still, convergence to equilibrium does not guarantee that an optimal solution is found (i.e., visited). Hence, in the next section we study how many steps our random walk algorithm takes to find an optimal solution, where numStep will be set accordingly with some margin.

4.1. Generalization to non-uniform node weights

In many practical networks, nodes generally have different weights or importance. Then, the size or proportion of a component is a weighted sum of its nodes' weights, and the problem is to find a minimum weighted separator that partitions the graph into components of sizes smaller than or equal to α fraction of the total sum weights. Accordingly, their attack costs can be also dependent on node weights possibly because of the different levels of protection. For simplicity, we use the same weight for a node's size and attack cost. To account for non-uniform node weights, one could directly apply our algorithm by changing the acceptance probability with the weighted sum cost objective. However, our analytical results in Section 5 are not applicable in the case of non-uniform node weights. The formal algorithm description for non-uniform node weights is as follows, where the node weight of node v is denoted as $\omega(v)$:

A random walk algorithm for non-uniform node weights

1: $W = V$; $W_{\min} = W$.

2: $\text{head}(v) = v, \forall v \in V$; $\text{cluster}(v) = \{v\}, \forall v \in V$.

3: **for** step $= 1$ **to** numStep

4: Pick a vertex $v \in V$ uniformly at random.

5: **If** $v \in W$,

6: newClusterSize $= \omega(v) + \sum_{i \in \cup_{j \in \mathcal{N}(v) \setminus W} \text{head}(j)} \sum_{k \in \text{cluster}(i)} \omega(k)$.

7: **if** newClusterSize $\leq \alpha \sum_{i \in V} \omega(i)$,

8: $W = W \setminus \{v\}$; **if** $\sum_{i \in W} \omega(i) < \sum_{i \in W_{\min}} \omega(i)$, $W_{\min} = W$.

9: $\text{cluster}(v) = \cup_{i \in \cup_{j \in \mathcal{N}(v) \setminus W} \text{head}(j)} \text{cluster}(i) \cup \{v\}$.

10: **for** $i \in \text{cluster}(v)$,

11: $\text{head}(i) = v$.

12: **end for**

13: **end if**

14: **else if** $v \notin W$, with probability $\rho^{\omega(v)}$,

15: $W = W \cup \{v\}$.

16: $\text{head}(i) = i$ for $i \in \mathcal{N}(v) \setminus W$.

17: **for** $i \in \mathcal{N}(v) \setminus W$,

18: **if** $\text{head}(i) = i$,

19: $\text{cluster}(i) \leftarrow \text{graphTraverse}(i, G \setminus W)$

20: $\text{head}(j) = i, \forall j \in \text{cluster}(i)$

21: **end if**

22: **end for**

23: **end if**

24: **end for**

An alternative way is to transform the network graph by duplicating nodes such that the number of copies of a node is equal to its weight (assuming integer weights). The neighboring nodes of a duplicated node is the same as the original node including all neighbors' duplicated nodes. Our random walk algorithm for uniform node weights can be applied to this transformed network, and it will find a solution to the weighted α-separator problem. This approach may suffer from increased computational complexity as the number of duplicated nodes can be large. Unlike the weighted version of random walk algorithm, our analysis using the hierarchical Markov chain model is valid for this transformation approach. We will run and compare both algorithms in Section 6.

4.2. Cooling Schedule: Convergence to Optimal States

So far, we have used a fixed parameter ρ that tradeoffs the convergence speed and the quality of the solution, for the analysis. This parameter is related to the "temperature" T in simulated annealing, where $\rho = e^{-\frac{1}{T}}$. When ρ becomes high or the temperature is high, the random walk is likely to escape local optimum points, but the stationary distribution at global optimum points becomes small, and vice versa. The simulated annealing is intended to choose the temperature parameter T (and correspondingly ρ) as a time-varying function (referred to as cooling schedule) so as to ensure the convergence of the state to the optimal points in a suitable sense. In this section, we discuss how to choose the schedule $\rho(t)$ so as to ensure the convergence of the state to the optimal points. We abuse the notation to define $\rho(t)$ as the control parameter at step t. In the following theorem in [45], the optimality of the *inverse logarithmic schedule* is guaranteed under some conditions. We will find a sufficient condition to apply this theorem to our minimum α-separator problem.

Theorem 1 (Optimality of the cooling schedule [45]) *Consider the inverse logarithmic cooling schedule defined as*

$$\rho(t) = t^{-\frac{1}{d}}, \;\; t \geq 1, \tag{3}$$

where d is the depth of the problem, which is the least number such that for any $W \in H$, there exists a path $W_0 = W, W_1, W_2, \cdots, W_p = W_{\min} \in H_{\min}$ such that $\max_{m \leq p}\{|W_m| - |W|\} \leq d$. Then, the chain converges almost surely in the Cesaro sense to the set of global minima, i.e.,

$$\lim_{\tau \to \infty} \frac{1}{\tau} \sum_{t=1}^{\tau} \mathbf{1}_{\{W_t \in H_{\min}\}} = 1 \;\; \textit{almost surely}, \tag{4}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function and W_t is the state at step t.

In our α-separator problem, $d = n - \min$ suffices for the theorem to hold, where min is the size of a minimum α-separator, because the size of an α-separator is always less than or equal to the number of vertices. We note that this cooling schedule guarantees to find optimal solutions in theory, but the convergence speed can be often extremely slow in practice. In Section 6, we will show that a cooling schedule with d smaller than $n - \min$, is enough for the underlying Markov chain to converge in cubic steps of the number of nodes, n.

5. First passage time analysis

In this section, we analytically characterize how long it takes to obtain the optimal solution, e.g., *first passage time* to an optimal state, based on a hierarchical Markov chain under uniform node weights and a fixed cooling parameter ρ. There are several existing works such as coupling techniques to the polynomial convergence of Markov Chain Monte Carlo methods [12, 13, 14]. In most problems that use coupling techniques, the transitions of a node only depend on neighboring nodes, such that they can show the speed of convergence. However, in the minimum α-separator problem, the sizes of components do not depend only on neighboring nodes, and thus we are unable to use these existing techniques. To that end, we conduct an approximate analysis on *first passage time* that can be applied to many general topologies.

For the approximate first passage time analysis, we introduce a Hierarchical Markov Chain (HMC), which groups states with the same sizes of alpha-separators into the same hierarchy. We recall that H is the state space of all α-separators. We define H_k as the k-th hierarchy in H, which is the set of states in H with k vertices, and $H = H_n \cup H_{n-1} \cup ... \cup H_{\min}$ where min is the size of a minimum α-separator. At each step of our algorithm, the hierarchy can be changed by at most one. Over this hierarchical Markov chain, we first find an approximate recursion formula for the first passage times of hierarchies. This recursion formula will be used to derive a sufficient condition for polynomial first passage time. Later, we will also analyze when the sufficient condition is satisfied.

Let X_t be the state at time t. The *first passage time* from a state in the k-th hierarchy H_k is defined as $\tau_k = \min\{t > 0 | X_0 \in H_k, X_t \in H_{\min}\}$. Trivially, $\tau_{\min} = 0$. The expected first passage time is denoted as $L_k = \mathbb{E}[\tau_k]$. For the expected first passage time, we have the following approximated recursion formula for $k > \min$:

$$L_k = p_+(k)L_{k+1} + p_-(k)L_{k-1} + (1 - p_+(k) - p_-(k))L_k + 1, \tag{5}$$

where $p_+(k) = \rho \cdot \frac{n-k}{n}$, and $p_-(k) = \frac{k}{n} \cdot \frac{|H_{k-1}| \cdot (n-k+1)}{|H_k| \cdot k}$. The value $p_+(k)$ is the probability that there is a transition from k-th hierarchy to $(k+1)$-th hierarchy, and $p_-(k)$ is the probability that there is a transition from k-th hierarchy to $(k-1)$-th hierarchy, assuming that every state in the same hierarchy has the same transition probability to the upper/lower hierarchy. For other cases (with probability $1 - p_+(k) - p_-(k)$), the random walk stays at k-th hierarchy. Note that vertex addition is proposed with probability $\frac{n-k}{n}$ (i.e., $\mathbb{P}[v \notin W]$), $W \cup \{v\}$ is always an α-separator, and it is accepted with probability ρ. Vertex removal is proposed with probability $\frac{k}{n}$ (i.e., $\mathbb{P}[v \in W]$), and the probability that $W \setminus \{v\}$ is an α-separator is $\frac{|H_{k-1}| \cdot (n-k+1)}{|H_k| \cdot k}$ which is the number of possible transitions from k-th hierarchy to $(k-1)$-th hierarchy (i.e., $|H_{k-1}| \cdot (n-k+1)$) over the number of proposed vertex removals ($|H_k| \cdot k$). For the number of possible transitions, note that it is the same as the transitions from $(k-1)$-th hierarchy to k-th hierarchy. By arranging the terms in Eq. (5), we have

$$L_k - L_{k-1} = \frac{p_+(k)}{p_-(k)}(L_{k+1} - L_k) + \frac{1}{p_-(k)}, \tag{6}$$

where $\frac{p_+(k)}{p_-(k)} = \frac{\rho(n-k)|H_k|}{(n-k+1)|H_{k-1}|}$ and $\frac{1}{p_-(k)} = \frac{n|H_k|}{(n-k+1)|H_{k-1}|}$.

We note that this is an approximation, since states in the same hierarchy can have different transition probabilities. The recursion formula is accurate when the states in a hierarchy are homogeneous in terms of transition probabilities (upwards and downwards). A star-like topology is an example instance where the homogeneous assumption is clearly true.[7]

Using the recursion formula, we find the following sufficient conditions under which the approximate first passage time is polynomial.

Proposition 1 *The approximate first passage time from Eq.* (6) *is polynomial if (i)* $|H_k|/|H_{k-1}|$ *is polynomially bounded, and (ii)* ρ^{-1} *is polynomially bounded and* $\rho^{-1} \geq |H_k|/|H_{k-1}|$.

PROOF When $k = n$, $L_n - L_{n-1} = \frac{n|H_n|}{(n-k+1)|H_{n-1}|}$ which is polynomial from $|H_n| = 1$ and $|H_{n-1}| \leq n$. As both $\frac{\rho(n-k)|H_k|}{(n-k+1)|H_{k-1}|}$ and $\frac{n|H_k|}{(n-k+1)|H_{k-1}|}$ in Eq. (6) become polynomial from the conditions, we can iteratively conclude that $L_k - L_{k-1}$ are polynomial from $k = n$ to $k = \min +1$. By summating all $L_k - L_{k-1}$ from $k = n$ to $k = \min +1$, we can conclude that L_n is polynomial.

We then investigate when the first condition is satisfied (i.e., $|H_k|/|H_{k-1}|$ is polynomially bounded) in Proposition 2. The second condition can be easily satisfied by setting ρ to a sufficiently small value.

Proposition 2 *Let* $w_T^*(n)$ *be the size of a minimum* α*-separator for a graph topology* T *of* n *nodes. Then,* $|H_k|/|H_{k-1}| = O(n^{w_T^*(n)+1})$ *for* $k \geq w_T^*(n) + 1$. *Therefore, if* $w_T^*(n)$ *is independent of* n*, or constant,* $|H_k|/|H_{k-1}|$ *is polynomially bounded.*

PROOF For simplicity, let $w^* = w_T^*(n)$ in this proof. For a minimum α-separator of size w^*, an attack set W that contains these nodes is definitely an α-separator. In the $(k-1)$-th hierarchy, the number of such α-separators is $\binom{n-w^*}{k-1-w^*}$. Therefore, the size of $|H_{k-1}|$ is lower bounded as

$$|H_{k-1}| \geq \binom{n-w^*}{k-1-w^*}.$$

Let $\bar{H}_{k-1}$ be the set of $W \subseteq V$ with size $k-1$, which is not an α-separator. Then,

$$|\bar{H}_{k-1}| = \binom{n}{k-1} - |H_{k-1}| = \binom{n}{k-1} - \binom{n-w^*}{k-1-w^*}.$$

[7]We define a star-like topology as the graph with star nodes that are connected to all nodes and all other nodes are only connected to star nodes. When there is only one star node, we have the standard star topology.

Now, we find an upper bound of $|H_k|$ using H_{k-1} and $\bar{H}_{k-1}$. For a subset or state W in H_k, there are only two cases: W^{edge} that has an edge to a state in H_{k-1} (by removing a node), and W^{noEdge} that has no edge to any state in H_{k-1}. For the first case, the maximum possible number is $(n-k+1)|H_{k-1}|$, since each state W' in H_{k-1} can be transitioned to at most $n-k+1$ states in H_k (by adding a node from $V \setminus W'$). For the second case, each state W^{noEdge} in H_k can be obtained by adding one node from a state W' in $\bar{H}_{k-1}$. Therefore, the maximum possible number of such W^{noEdge} in H_k is $(n-k+1)|\bar{H}_{k-1}|$. By summating these two cases, we have the following inequality, which concludes the proof:

$$\begin{aligned}\frac{|H_k|}{|H_{k-1}|} &\leq (n-k+1) + \frac{\left(\binom{n}{k-1} - \binom{n-w^*}{k-1-w^*}\right)(n-k+1)}{\binom{n-w^*}{k-1-w^*}} \\ &= \frac{\binom{n}{k-1}}{\binom{n-w^*}{k-1-w^*}}(n-k+1) = O(n^{w^*+1}).\end{aligned}$$

We note that in several simple topologies, the size of a minimum α-separator does not scale with the graph size, n. For a balanced tree topology such as a complete k-ary tree, the size of a minimum α-separator does not scale with the graph size since the portion of all leaf nodes of a node in depth (depth= 1 for the root node) is bounded by $\frac{1}{k^{\text{depth}}}$. Also, in line, circle, and star-like topologies, it is easy to check that the size of a minimum α-separator does not scale with the graph size. We note that Mohamed et al. [6] developed deterministic polynomial-time algorithms for trees and cycles, which exploit the special structure of graphs. While their algorithms require knowledge of the type of the topology, our random walk algorithm does not need any prior information about the graph topology.

5.1. First passage time in a star topology

As an example, we compute the expected first passage time in a star topology using our recursion formula.

Proposition 3 *For a star topology, the expected first passage time when $\rho = O(\frac{1}{n})$ is*

$$L_n = O(n^2). \tag{7}$$

PROOF Let $\rho = \frac{c}{n}$ for a constant c. In a star topology, $\min|W| = 1$ and $L_1 = 0$. The first passage time from hierarchy n to 1 is divided into two parts: (i) $L_n - L_{n-m-1}$ (from hierarchy n to $n-m-1$) and (ii) $L_{n-m-1} - L_1$ (from hierarchy $n-m-1$ to 1) where $m = \lfloor \alpha n \rfloor$.

(i) $L_n - L_{n-m-1} = O(n^2)$:

Let H_k^1 and H_k^0 be the set of states that the star node is included (1) and not included (0) in the attack set. Then, $H_k = H_k^1 \cup H_k^0$. L_k^1 and L_k^0 are the average first passage time

from H_k^1 and H_k^0, respectively. In k-th hierarchy for $n-m+1 \le k \le n$, the following holds from the transition probability:

$$L_k^1 = \frac{\rho(n-k)}{n} L_{k+1}^1 + \frac{k-1}{n} L_{k-1}^1 + \frac{1}{n} L_{k-1}^0 + \Big(1 - \frac{\rho(n-k)}{n} - \frac{k}{n}\Big) L_k^1 + 1.$$

By arranging the terms,

$$L_k^1 - L_{k-1}^1 = \frac{\rho(n-k)}{k-1}(L_{k+1}^1 - L_k^1) + \frac{1}{k-1}(L_{k-1}^0 - L_k^1) + \frac{n}{k-1}.$$

When $k = n-m$, the transition to the lower hierarchy is not accepted if the star node is chosen to be removed from the attack set. Thus,

$$L_{n-m}^1 - L_{n-m-1} = \frac{\rho \cdot m}{n-m-1}(L_{n-m+1}^1 - L_{n-m}^1) + \frac{n}{n-m-1}.$$

Now, we find an upper bound for L_k^0. We note that $L_k^1 \le L_{k+1}^1$ since the states in H_k^1 have one less number of nodes to remove than states in H_{k+1}^1. Consider the path from a state in H_k^0 to H_1. This path should have a transition from H_j^0 to H_{j+1}^1 for j such that $n-m+1 \le j \le n-1$, which adds the star node to the attack set. Since $L_{j+1}^1 \le L_n^1 = L_n$ (from $L_n^0 = \emptyset$) and it takes $\frac{n}{\rho}$ steps on average to choose the star node and add it,

$$L_k^0 \le L_n + \frac{n}{\rho}, \text{ for } n-m \le k \le n-1. \tag{8}$$

By applying the above inequality to Eq. (8),

$$L_k^1 - L_{k-1}^1 \le \frac{\rho(n-k)}{k-1}(L_{k+1}^1 - L_k^1) + \frac{1}{k-1}(L_n - L_k^1) + \frac{n}{k-1}\Big(1 + \frac{s}{\rho}\Big).$$

By summating all equations from $k = n$ to $k = n-m$, we can conclude that $L_n - L_{n-m-1} = O(n^2)$.

(ii) $L_{n-m-1} - L_1 = O(n\log(n))$:

For the hierarchy below $n-m-1$, the star node should be included in the α-separator. Then, from the recursion formula in Eq. (6),

$$L_{k+1} - L_k = \frac{\rho(n-k-1)}{k}(L_{k+2} - L_{k+1}) + \frac{n}{k}, \tag{9}$$

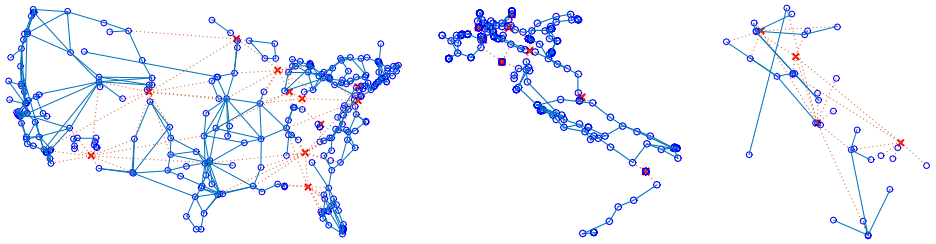
for $1 \le k \le n-m-2$. Since the transition probabilities of states in the same hierarchy are identical, this recursion formula is exact for a star-like topology. From Eq. (9), for the 1-st hierarchy,

$$L_2 - L_1 = \frac{n}{1} + \frac{c(n-2)}{n}(L_3 - L_2) \le \frac{n}{1} + \frac{cn}{2} + \frac{c}{2}(L_4 - L_3) \le n\sum_{i=1}^{\infty} \frac{c^{i-1}}{i!} \le n \cdot \frac{e^c - 1}{c}. \tag{10}$$

Similarly,

$$L_{k+1} - L_k \le \frac{n}{k} \cdot \frac{e^c - 1}{c}, \tag{11}$$

for $2 \le k \le n-m-2$. By summing up for all hierarchies and using $\sum_{i=1}^{n} \frac{1}{i} \le \log(n) + 1$, we can conclude that $L_{n-m-1} - L_1 = O(n\log(n))$.



(a) US fiber networks. (b) Italian power grid. (c) Italian Internet networks.

Figure 4: Minimum α-separators in real topologies (marked as 'x') obtained from our random walk algorithm for $\alpha = 0.25$. The dotted lines indicate the links that are attached to the minimum α-separators. (a) $\min |W| = 11$, (b) $\min |W| = 8$, and (c) $\min |W| = 4$.

6. Simulation results

In this section, we run our random walk algorithm and other heuristic algorithms over diverse topologies, including real topologies from US fiber networks [15] and Italian power grid [16, 17]. We compute the *average first passage time* to a minimum α-separator, and validate our analytical results. We also compare our random walk algorithm with other heuristic algorithms: highest-degree-first and greedy algorithms from the perspective of the largest component for a given budget (i.e., the same number of attacked nodes), and show that our random walk algorithm performs better than these algorithms.

6.1. Baseline algorithms

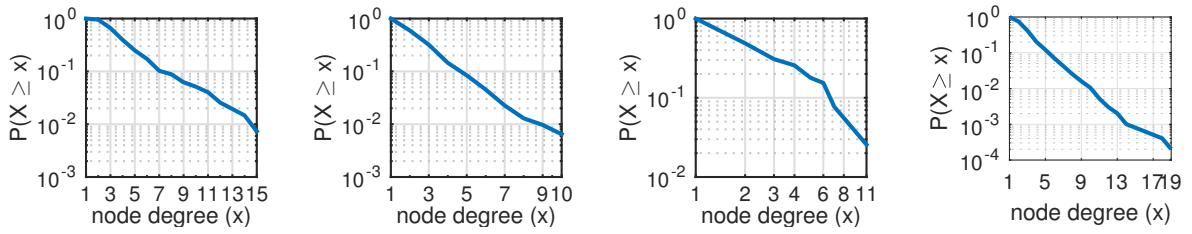
We compare our random walk algorithm with exhaustive search up to some number of vertices and two simple heuristic algorithms: highest-degree-first and greedy. The attack set is initialized to an empty set (i.e., $W = \emptyset$) for both algorithms.

Highest-degree-first: Attack vertices are chosen in the order of their node degree, i.e., the number of neighboring vertices. If several vertices have the same degree, one of them is chosen randomly to break the tie.

Greedy: In each iteration, for each candidate vertex that is not in W, it computes the size of the largest connected component after adding the vertex. Then, the vertex with the largest marginal decrease is chosen. If multiple vertices have the same marginal decrease, one of them is chosen randomly.

6.2. Graph topologies

We test the algorithms over various real and generated topologies. For real topologies, we use US fiber networks of 20 Internet providers [15] and Italian power grid and Internet network [16, 17], as depicted in Fig. 4. The US fiber networks have 273 nodes and 542 edges. The Italian power grid has 310 nodes (100 junctions, 113 generator units and 97 loads) and 347 edges. Italian Internet networks, dedicated to linking Italian universities and research institutions, have 39 nodes and 50 edges. We also test our algorithm over a large-scale power grid in the Western US with 4941 nodes and 6594 edges [46, 47]. We find that US



(a) US fiber networks. (b) Italian power grid. (c) Italian Internet networks. (d) Western US power grid.

Figure 5: Degree distributions of real topologies. The x-axes of (a), (b), and (d) are in linear scale, and the x-axis of (c) is in log scale.

fiber networks, Italian power grid, and Western US power grid follow exponential degree distributions and Italian Internet networks follow a power-law degree distribution, as shown in Fig. 5.

We also test star, binary tree, line and circle topologies, as well as randomly generated topologies: Erdos-Renyi graph [48], random graphs from the configuration model [49, 50]. In the configuration model, each node is first assigned a degree, drawn from a degree distribution. Then, a random graph is constructed by choosing a uniformly random matching on the degree "stubs" (half-edges). For the degree distributions, we use power-law and exponential distributions with the parameters obtained from real topologies (power-law distribution from the Italian Internet graph and exponential distribution from US fiber networks).

6.3. Minimum α-separators on real topologies

We run our random walk algorithm on real topologies for $\alpha = 0.25$, which is used throughout this section. The parameter ρ is set to be $4/n$. The sizes of the minimum α-separators in US fiber networks and Italian power grid are 11 nodes and 8 nodes, respectively. By averaging 100 simulations, we find that the average first passage times are 3.16×10^5 $(= n^{2.26})$ and 3.1×10^5 $(= n^{2.2})$ steps. We depict the minimum α-separators obtained from our random walk algorithm for $\alpha = 0.25$ in Fig. 4. In US fiber networks, dense areas are in the east and west coasts, which can be shortsightedly regarded as primary targets of an attacker. However, *the minimum α-separator is not concentrated in the dense area, but in the middle of the large components.* Similarly, the α-separator in the Italian power grid cuts large balanced components. Therefore, this result shows that in real networks hub nodes are not necessarily a good target, and other non-hub nodes may be critical and should be fortified.

6.4. First passage time on various topologies

We run our random walk algorithm over various topologies with $\alpha = 0.25$, and compute the *average first passage time* to a minimum α-separator. We summarize the results in Table 1. In all test cases, our random walk algorithm converges to a minimum separator within n^3 steps, where the average first passage time depends on the graph structure. To further explore how the average first passage time increases with the graph size, we run our

Table 1: Average first passage times of our random walk algorithm and the sizes of the minimum α-separators for $\alpha = 0.25$.

Topology	n (# of nodes)	Avg. degree	min $\lvert W \rvert$	L_n
Star	273	$\frac{2(n-1)}{n}$ =1.99	1	$n^{1.72}$
Binary tree	273	$\frac{2(n-1)}{n}$ =1.99	2	$n^{1.97}$
Line	273	$\frac{2(n-1)}{n}$ =1.99	3	$n^{2.7}$
Circle	273	2	4	$n^{2.34}$
Erdos-Renyi	273	2	12	$n^{2.47}$
Configuration (exponential)†	273	2.24	14	$n^{2.41}$
Configuration (power-law)♯	39	2.56	4	$n^{1.84}$
Real (US fiber)	273	3.97	11	$n^{2.26}$
Real (Italian power)	310	2.24	8	$n^{2.2}$
Real (Italian Internet)	39	2.56	4	$n^{2.07}$
Real (West US power)	4941	2.67	18	$n^{2.24}$

†: The exponent is $\lambda = 2.24$. This parameter is chosen to have the same average degree as the US fiber network.
♯: The exponent is $\beta = 2.01$, where the probability density function $p(x) \sim x^{-\beta}$. This parameter is chosen to have the same average degree as the Italian Internet graph.

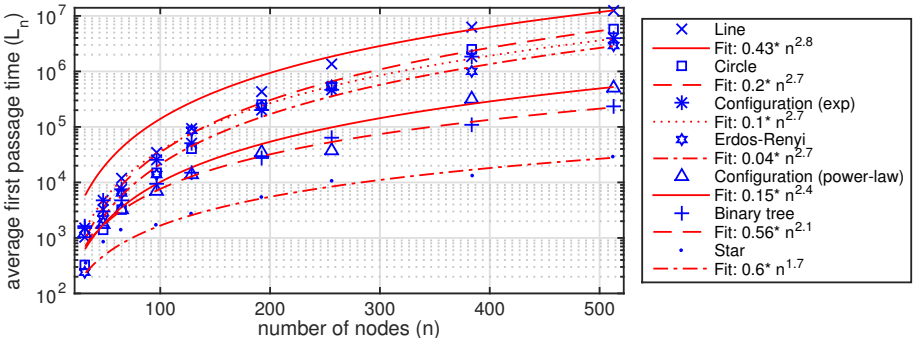


Figure 6: Average first passage times for different graph sizes, n, in generated topologies, and their fittings to a function $f(n) = a \times n^b$ that minimizes the sum of square errors.

random walk algorithm by scaling the networks up to 512 nodes for the simple topologies and randomly generated topologies, and summarize the results in Fig. 6. The best fittings to a function $f(n) = a \times n^b$ with the smallest sum of square errors show that the exponents of all tested topologies are less than 3, implying that the average first passage time is $O(n^3)$. We find that on average, the first passage time is smallest in the star topology. In randomly generated topologies, we find that it takes a shorter time to find a minimum α-separator in

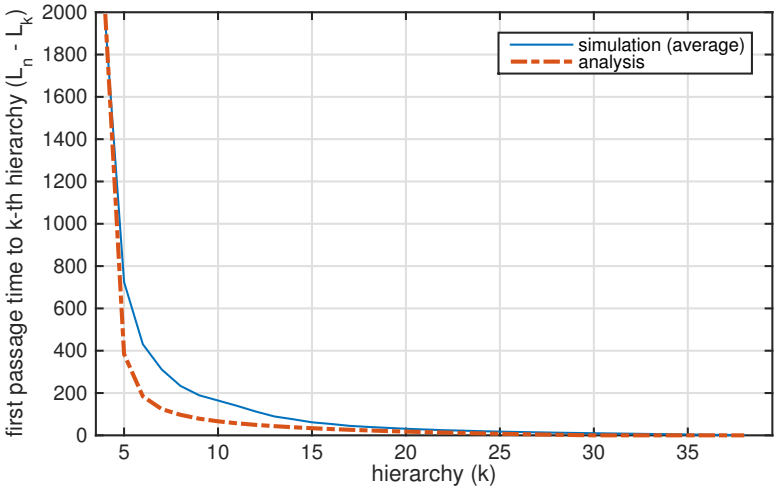


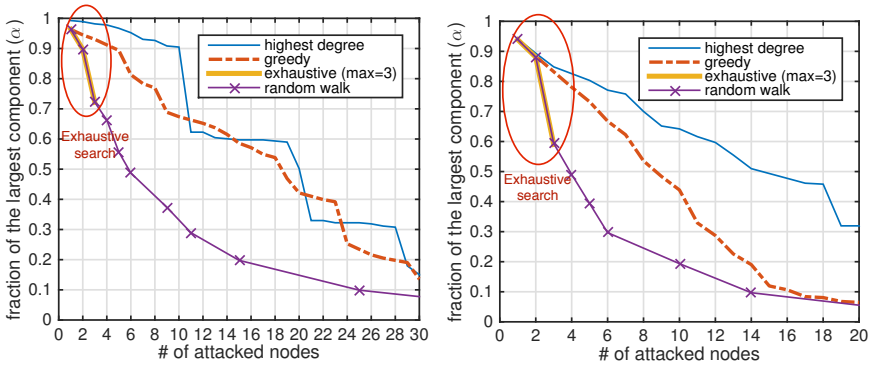
Figure 7: Average first passage time to k-th hierarchy ($L_n - L_k$) in simulation and from analytical results, in Italian Internet networks. We recall that when $k = \min |W|$, $L_k = 0$. Thus, the leftmost point indicates the average first passage time to a minimum α-separator.

the power-law graph than other graphs. There are a small number of hub nodes with high node degree in the power-law graph, which is similar to the star node in the star topology. These hub nodes are likely to be included in the small α-separator, under which our random walk algorithm can move faster to the better states without removing these nodes. This is because removing a hub node from an attack set W will be likely to break the α-separating condition.

We also numerically validate approximate first passage time in Eq. (6). We first estimate $|H_k|$ by testing the α-separating condition for random subsets with size k. By multiplying this probability with the number of possible combinations (i.e., $\binom{n}{k}$), we obtain an estimate of $|H_k|$. We then compute L_k from the recursion formula. In Fig. 7, we depict the average first passage time from simulation and our approximate first passage time, for the Italian Internet networks. As one can see, our analysis approximates the actual first passage time fairly closely. This has also been observed in other simulations we have run, which we omit for brevity. Thus, it is reasonable to use our approximation to estimate the expected first passage time from the ratios of the hierarchies. The average steps to move to the lower hierarchy typically takes much longer as the hierarchy approaches the minimum hierarchy, since the probability that α-separating conditions are satisfied in each hierarchy decreases as the hierarchy gets lower.

6.5. Comparison with other algorithms

We first explain the complexity of other compared algorithms - highest-degree-first, greedy, and exhaustive search. The highest-degree-first algorithm sorts the vertices by their degree at first. This only takes $O(n \log(n))$ computations. To obtain the sizes of the components, it runs a graph traverse algorithm at most n times. Thus, the complexity is $O(n^3)$. The greedy algorithm needs to sort the vertices in each iteration, which takes $O(n^2 \log(n))$



(a) In US fiber networks. (b) In Italian power grid.

Figure 8: Sizes of the largest components for a given budget (i.e., size of the attack set) in various algorithms.

computations, but the complexity is still $O(n^3)$ from the graph traverses. For the exhaustive search, it also requires computing the sizes of components for each subset, which takes $O(n^3)$ computations. The number of subsets with m nodes is $O(n^m)$. Finding a minimum takes $O(n^m)$ computations. Thus, the exhaustive search for subsets with m nodes takes $O(n^{m+3})$ computations. In conclusion, the highest-degree-first and greedy algorithms take much shorter computation time than our random walk algorithm (with $O(n^3)$ iterations), but we will show that their performance is much worse than our random walk algorithm. The exhaustive search is optimal, but takes a very long time unless m is small.

In Fig. 8, we compare the sizes of the largest components obtained from our random walk algorithm and other algorithms, in real topologies. For our random walk algorithm, we vary α from 0.1 to 0.95. The sizes of the largest components of our random walk algorithm are much smaller than those from other heuristic algorithms, highest-degree-first and greedy, for the same number of nodes in the attack set. Also, the solution of our random walk algorithm is the same as exhaustive search up to 3 attack nodes. Note that the complexity of exhaustive search grows exponentially, and we only run exhaustive search up to 3 nodes. Even the number of subsets with size 4 in the US fiber networks is $\binom{273}{4} = 2.3 \times 10^8$, which takes extremely long to enumerate all of them and check the α-separating conditions.

6.6. Non-uniform node attack costs

As we discussed in Section 4.1, there are two ways to generalize our framework to non-uniform node weights: (A) use our modified algorithm with the sum objective of node weights ("random walk for non-uniform weights"), or (B) virtually duplicate nodes according to their weights ("duplication"). To compare these two approaches, we use the Italian power grid network with $N = 310$ (100 junctions, 113 generator units and 97 loads) [16]. To consider non-uniform weights, we choose the node weights to be proportional to the amount of the generated power (from generator units), or the amount of the consumed power (from loads). We assume a unit node weight for the junctions. The distribution of the node weights or the amount of power generated or consumed is depicted in Fig. 9. We round the amount

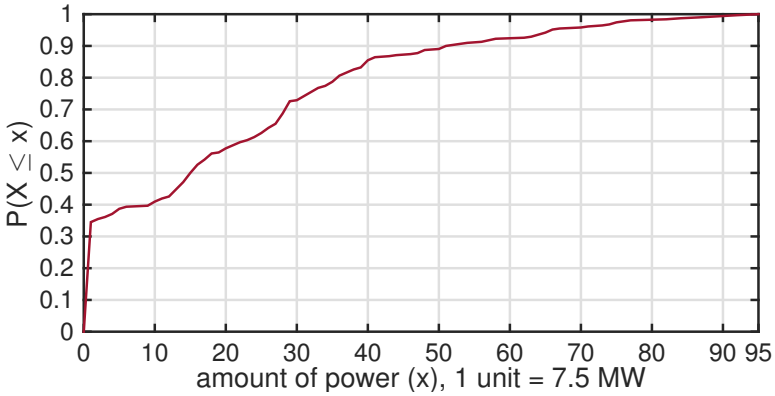


Figure 9: The CDF of the amount of generated/consumed power at nodes in the Italian power grid network [16].

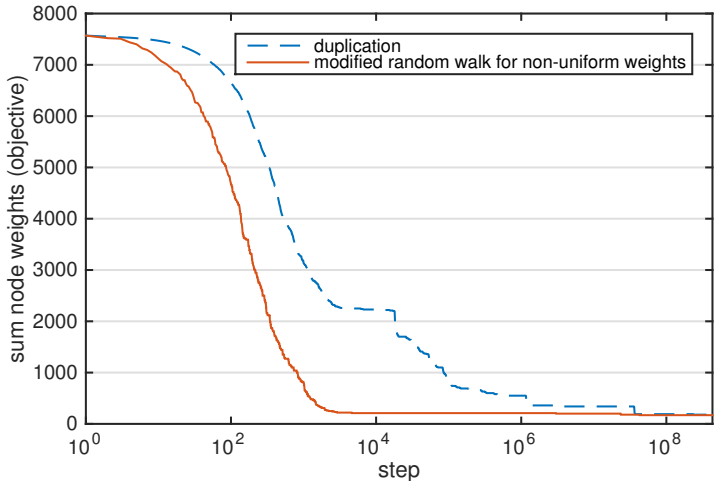


Figure 10: The sum node weights of the minimum α-separator in our modified random walk algorithm and duplication over the Italian power grid network.

of power to the nearest integer, for the duplication method. The sum node weights of the current α-separator over the time steps are depicted in Fig. 10. The modified random walk algorithm is much faster than the duplication method in finding a minimum weighted α-separator. The reason is that the duplication method takes multiple steps to remove or add a weighted node (with a node weight higher than 1) in the original graph. In other words, all duplicated nodes of the weighted node should be chosen from the random node selection to remove or add the weighted node. Also, high weight nodes are harder to be added to the attack set once they are removed, since it is more unlikely to choose all the duplicated nodes of a high weight node. These factors greatly slow down the random walk.

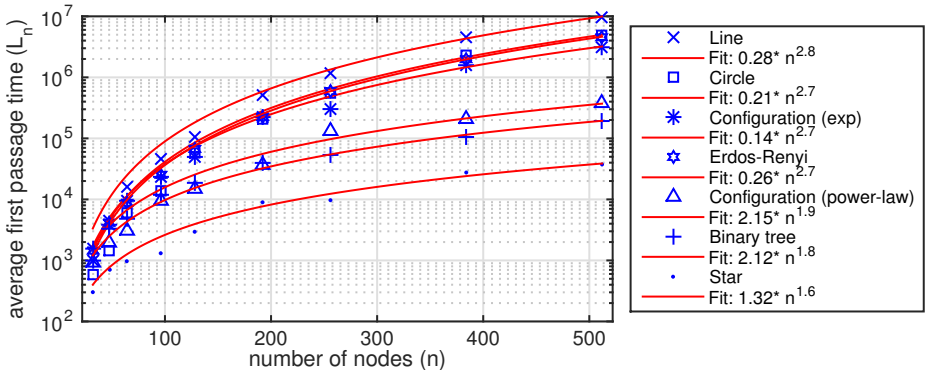


Figure 11: Average first passage times for different graph sizes, n, using the inverse logarithmic cooling $\rho(t) = t^{-\frac{1}{2.5}}$, and their fittings to a function $f(n) = a \times n^b$ that minimizes the sum of square errors, in various topologies.

6.7. Inverse logarithmic cooling schedule

As in Section 4.2, the inverse logarithmic cooling schedule with an appropriate depth parameter guarantees the optimality almost surely. In practice however, the convergence speed can be extremely slow. Thus, we set $d = 2.5$ under which the chain converges in polynomial steps, as depicted in Fig. 11. The first passage times in all topologies are similar to the case when we used fixed parameter ρ. But we note that the stationary distribution for the optimal states in the inverse logarithmic cooling schedule is higher than the fixed case after sufficiently long steps, because $\rho(t) = t^{-\frac{1}{d}}$ keeps decreasing over time.

7. Conclusion

In this paper, we developed a random walk algorithm based on a Metropolis chain to solve the minimum α-separator problem. We analyzed the first passage time of our random walk algorithm and investigated the conditions for polynomial first passage time, under the homogeneous assumption for states in the same hierarchy. The conditions include simple topologies where the sizes of minimum α-separators do not scale with the network size, e.g., star, line, circle, and balanced tree. Specifically, we showed that the first passage time in a star topology is $O(n^2)$, where n is the number of nodes. We note that unlike the existing polynomial time algorithm in [6] for trees, cycles, and stars, which requires knowledge of the topology type in advance, our random walk algorithm does not require knowledge of the topology type. Through simulations over real topologies and generated topologies, we showed that the first passage time is less than n^3 and validated our analysis for first passage time. We then show that our random walk algorithm performs better than highest-degree-first and greedy heuristic algorithms. In real topologies, we found that attacking a dense area is not always an efficient solution to partition a network graph in order to trigger cascading failures.

In this work, we primarily focused on identifying the weakest part of the network whose removal can potentially lead to disastrous network failures. There are still several problems

in this context that can be an interesting future direction. For example, one can study the network structure which is more resilient to deliberate separation attacks, i.e., networks with greater sizes of separators. Furthermore, our model in this work does not account for physical interactions in real-world networked systems. Hence, it will be interesting to incorporate physical dynamics, such as power flows in power grids, into our model and formulate the separator problem in the new model. Here, the value of a network is not simply the number of active nodes in the largest component anymore, but a function of active elements, which captures the physical dynamics among them. This problem will be able to reveal the impact of failures on network structure more precisely.

8. Acknowledgments

The work has in part been funded by: a grant from the Defense Thrust Reduction Agency (DTRA) HDTRA1-14-1-0058; grants from the Army Research Office (ARO) MURI W911NF-12-1-0385; NSF CNS-1446582. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(Ministry of Education) (No. 2018R1D1A1B07045181).

References

[1] J. Lee, J. Kwak, H. won Lee, N. B. Shroff, Finding minimum node separators: A markov chain monte carlo method, in: International Conference on the Design of Reliable Communication Networks (DRCN), 2017.

[2] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, et al., Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance, IEEE transactions on Power Systems 20 (4) (2005) 1922–1928.

[3] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (7291) (2010) 1025–1028.

[4] P. Crucitti, V. Latora, M. Marchiori, Model for cascading failures in complex networks, APS Physical Review E 69 (4).

[5] U. Feige, M. Mahdian, Finding small balanced separators, in: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, 2006, pp. 375–384.

[6] M. A. M. Sidi, K-separator problem, Ph.D. thesis, Evry, Institut national des télécommunications (2014).

[7] M. Jerrum, A. Sinclair, Approximating the permanent, SIAM journal on computing 18 (6) (1989) 1149–1178.

[8] M. Jerrum, A. Sinclair, The markov chain monte carlo method: an approach to approximate counting and integration, Approximation algorithms for NP-hard problems (1996) 482–520.

[9] S.-Y. Yun, Y. Yi, J. Shin, et al., Optimal CSMA: a survey, in: IEEE International Conference on Communication Systems (ICCS), 2012, pp. 199–204.

[10] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, A. Ganesh, Peer counting and sampling in overlay networks: random walk methods, in: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, 2006, pp. 123–132.

[11] M. Gjoka, M. Kurant, C. T. Butts, A. Markopoulou, A walk in facebook: Uniform sampling of users in online social networks, arXiv preprint arXiv:0906.0060.

[12] D. Weitz, Counting independent sets up to the tree threshold, in: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, 2006, pp. 140–149.

[13] M. Dyer, C. Greenhill, On markov chains for independent sets, Journal of Algorithms 35 (1) (2000) 17–49.
[14] M. Luby, E. Vigoda, Approximately counting up to four, in: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, 1997, pp. 682–687.
[15] R. Durairajan, P. Barford, J. Sommers, W. Willinger, Intertubes: a study of the us long-haul fiber-optic infrastructure, in: ACM SIGCOMM Computer Communication Review, Vol. 45, 2015, pp. 565–578.
[16] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, R. Setola, Modelling interdependent infrastructures using interacting dynamical models, International Journal of Critical Infrastructures 4 (1-2) (2008) 63–79.
[17] M. Parandehgheibi, E. Modiano, Robustness of interdependent networks: The case of communication networks and the power grid, in: IEEE GLOBECOM, 2013, pp. 2164–2169.
[18] R. Albert, H. Jeong, A.-L. Barabási, Error and attack tolerance of complex networks, nature 406 (6794) (2000) 378–382.
[19] A. Clauset, C. R. Shalizi, M. E. Newman, Power-law distributions in empirical data, SIAM review 51 (4) (2009) 661–703.
[20] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the internet topology, in: ACM SIGCOMM computer communication review, Vol. 29, ACM, 1999, pp. 251–262.
[21] J. Balthrop, S. Forrest, M. E. Newman, M. M. Williamson, Technological networks and the spread of computer viruses, Science 304 (5670) (2004) 527–529.
[22] T. N. Dinh, Y. Xuan, M. T. Thai, P. M. Pardalos, T. Znati, On new approaches of assessing network vulnerability: hardness and approximation, IEEE/ACM Transactions on Networking 20 (2) (2012) 609–619.
[23] C. Zhang, J. E. Ramirez-Marquez, C. M. R. Sanseverino, A holistic method for reliability performance assessment and critical components detection in complex networks, IIE Transactions 43 (9) (2011) 661–675.
[24] O. Kabadurmus, A. E. Smith, Evaluating reliability/survivability of capacitated wireless networks, IEEE Transactions on Reliability 67 (1) (2018) 26–40.
[25] J. E. Ramirez-Marquez, C. M. Rocco, K. Barker, J. Moronta, Quantifying the resilience of community structures in networks, Reliability Engineering & System Safety 169 (2018) 466–474.
[26] J. Li, L. Dueñas-Osorio, C. Chen, C. Shi, Connectivity reliability and topological controllability of infrastructure networks: A comparative assessment, Reliability Engineering & System Safety 156 (2016) 24–33.
[27] S. Dunn, S. Wilkinson, Hazard tolerance of spatially distributed complex networks, Reliability Engineering & System Safety 157 (2017) 1–12.
[28] K. Meena, T. Vasanthi, Reliability analysis of mobile ad hoc networks using universal generating function, Quality and Reliability Engineering International 32 (1) (2016) 111–122.
[29] E. Ferrario, N. Pedroni, E. Zio, Evaluation of the robustness of critical infrastructures by hierarchical graph representation, clustering and monte carlo simulation, Reliability Engineering & System Safety 155 (2016) 78–96.
[30] Z. Zhang, W. An, F. Shao, Cascading failures on reliability in cyber-physical system, IEEE Transactions on Reliability 65 (4) (2016) 1745–1754.
[31] S. Shen, J. C. Smith, Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs, Networks 60 (2) (2012) 103–119.
[32] M. Wachs, C. Grothoff, R. Thurimella, Partitioning the internet, in: International Conference on Risks and Security of Internet and Systems (CRiSIS), 2012.
[33] M. R. Garey, D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, Vol. 29, W. H. Freeman New York, 2002.
[34] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on scientific Computing 20 (1) (1998) 359–392.
[35] T. N. Bui, C. Jones, Finding good approximate vertex and edge partitions is np-hard, Information Processing Letters 42 (3) (1992) 153–159.

[36] S. Özekici, R. Soyer, Network reliability assessment in a random environment, Naval Research Logistics (NRL) 50 (6) (2003) 574–591.
[37] P. Green, K. Worden, Bayesian and markov chain monte carlo methods for identifying nonlinear systems in the presence of uncertainty, Phil. Trans. R. Soc. A 373 (2051) (2015) 20140405.
[38] K. M. Zuev, S. Wu, J. L. Beck, General network reliability problem and its efficient solution by subset simulation, Probabilistic Engineering Mechanics 40 (2015) 25–35.
[39] J. L. Beck, S.-K. Au, Bayesian updating of structural models and reliability using markov chain monte carlo simulation, Journal of engineering mechanics 128 (4) (2002) 380–391.
[40] X. Wang, N. Balakrishnan, B. Guo, Residual life estimation based on a generalized wiener degradation process, Reliability Engineering & System Safety 124 (2014) 13–23.
[41] M. Compare, P. Baraldi, I. Bani, E. Zio, D. Mc Donnell, Development of a bayesian multi-state degradation model for up-to-date reliability estimations of working industrial components, Reliability Engineering & System Safety 166 (2017) 25–40.
[42] D. A. Levin, Y. Peres, E. L. Wilmer, Markov chains and mixing times, American Mathematical Society, 2009.
[43] W. Gilks, S. Richardson, D. Spiegelhalter, Markov chain monte carlo in practice, ser. interdisciplinary statistics series (1996).
[44] J. Geweke, et al., Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments, Vol. 196, Federal Reserve Bank of Minneapolis, Research Department Minneapolis, MN, USA, 1991.
[45] V. S. Borkar, Pathwise recurrence orders and simulated annealing, Journal of applied probability 29 (02) (1992) 472–476.
[46] KONECT, Us power grid network dataset, `http://konect.uni-koblenz.de/networks/opsahl-powergrid` (2017).
[47] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world' networks, nature 393 (6684) (1998) 440–442.
[48] P. Erdös, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci 5 (17-61) (1960) 43.
[49] M. Newman, Networks: an introduction, Oxford university press, 2010.
[50] E. A. Leicht, M. E. Newman, Community structure in directed networks, Physical review letters 100 (11) (2008) 118703.