

A Sample-Based Algorithm for Approximately Testing r -Robustness of a Digraph

Yuhao Yi, Yuan Wang, Xingkang He, Stacy Patterson, and Karl H. Johansson

Abstract—One of the intensely studied concepts of network robustness is r -robustness, which is a network topology property quantified by an integer r . It is required by mean subsequence reduced (MSR) algorithms and their variants to achieve resilient consensus. However, determining r -robustness is intractable for large networks. In this paper, we propose a sample-based algorithm to approximately test r -robustness of a digraph with n vertices and m edges. For a digraph with a moderate assumption on the minimum in-degree, and an error parameter $0 < \epsilon \leq 1$, the proposed algorithm distinguishes $(r + \epsilon n)$ -robust graphs from graphs which are not r -robust with probability $(1 - \delta)$. Our algorithm runs in $\exp(O((\ln \frac{1}{\epsilon \delta})/\epsilon^2)) \cdot m$ time. The running time is linear in the number of edges if ϵ is a constant.

I. INTRODUCTION

Consensus is the cornerstone of cooperative distributed systems as a mechanism to share information among agents. Due to its wide applications, the safety aspects of the problem have been considered intensively. One of the essential problems is to design consensus algorithms that tolerate a locally or globally bounded number of faulty agents or adversaries. In this context, consensus is achieved if the honest agents agree on a value which is justified by their initial values. These algorithms are also called resilient consensus algorithms.

The history of distributed systems and multi-agent systems has witnessed the development of a whole spectrum of consensus algorithms against adversaries: the seminal paper [1] and its explanatory version [2] which study the binary consensus; the paper addressing incomplete networks [3] for the binary case; more recent development on scalar consensus in incomplete networks [4], [5]; and multi-agent vector consensus [6], [7], [8]. The applications of fault tolerant consensus algorithms are beyond enumeration. Examples of recent applications include distributed optimization [9], [10], rendezvous of robots [11], hypothesis testing [12], and distributed estimation [13], [14].

Some examples in [4], [15] show that network connectivity is insufficient for resilient consensus. Therefore the new notion of r -robustness has been proposed [4] and used as

sufficient conditions for many algorithms to achieve consensus among honest agents. For example, the W-MSR [4], SW-MSR [16], and DP-MSR [17] algorithms. r -robustness imposes connectivity constraints on vertex set pairs of the network. Loosely speaking, for the MSR algorithms in an r -robust network, each honest vertex updates its state while ignoring at most $\lfloor (r - 1)/2 \rfloor$ smallest and largest values from its neighbors. Then, if each honest vertex has at most $\lfloor (r - 1)/2 \rfloor$ malicious in-neighbors, the asymptotic resilient consensus is achieved.

Despite the fact that many resilient consensus algorithms require an r -robust network, determining if a network is r -robust or not has been proven to be coNP-complete [15]. Therefore, no polynomial time algorithm exists for the problem unless P=NP. A known algorithm which solves the problem for arbitrary digraphs is proposed in [18]. It enumerates all subset pairs of the digraph, which has a running time exponential in the number of vertices n . Since then, efforts have been made to either improve the efficiency of the algorithm, or circumvent the problem. A notable work is the recent paper [19], in which the problems are formulated as integer linear programs (ILP). ILP solvers are used to improve the speed of searching. The reformulation brings practical improvement but does not give provable improvement on complexity. To bypass the problem, network construction methods are investigated to grow a network with given r [4], [20], [21]. Estimation of r is also studied for special classes of networks, such as random networks [15] and random interdependent networks [22]. In these special networks, r is bounded by spectral and structural properties of the network and can be efficiently estimated. However, we note that these estimations are not necessarily tight in arbitrary networks. To the best of our knowledge, no existing work has been done to rigorously study the approximation of r in arbitrary digraphs.

The main contribution of this paper is an algorithm for approximately testing r -robustness with provable guarantees. By setting an error bound, we study the problem of distinguishing $(r + \epsilon n)$ -robust networks from networks that are not r -robust. We devise a randomized (Monte Carlo) algorithm that solves the problem with probability $(1 - \delta)$ for an error parameter $\epsilon > 0$, and a digraph satisfying a moderate assumption about the minimum degree. We prove the performance guarantee of the proposed algorithm and show the tradeoff between precision and running time.

Our algorithm is based on random sampling of vertices and has a Enforce-and-Test flavor that is seen in graph property testing [23], [24]. Random sampling has been shown to

Y. Yi, Y. Wang, and K. H. Johansson are with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, and they are also affiliated with Digital Futures, SE-100 44 Stockholm, Sweden (email: yuhaoy@kth.se, yuanwang@kth.se, kallej@kth.se). They are supported in part by Knut & Alice Wallenberg foundation, and by Swedish Research Council.

X. He is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 46556, USA (email: xhe9@nd.edu).

S. Patterson is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA (email: sep@cs.rpi.edu).

be successful in property testing [23] and the design of approximation algorithms [25] for dense graphs. We extend the technique to approximately determining r -robustness in digraphs, which is a new type of problem compared with its traditional applications.

Outline: The remainder of the paper is structured as follows. In Section II we introduce some basic definitions. In Section III we give the definition of the considered problem. In Section IV we describe the proposed algorithm, which is analyzed in Section V. Some discussion is given in Section VI, followed by the conclusion. Discussion about practical implementation and numerical examples are shown in the appendix.

II. PRELIMINARIES

A. Concepts and Notations

A directed graph (digraph) G is defined as a pair (V, E) , where V and E are the vertex set and the edge set. We let $|V| = n$ and $|E| = m$. We let $e = (u, v)$ be the directed edge from vertex v to vertex u . We denote by \mathcal{N}_u^\downarrow the set of in-neighbors of vertex u and $|\mathcal{N}_u^\downarrow|$ the in-degree of u . For a subset of vertices $V' \subset V$, we define the subgraph supported on V' as $G[V'] = (V', E')$ where $E' = \{(u, v) \in E : u, v \in V'\}$. Undirected graphs are viewed as bidirectional digraphs.

Definition 1 (r -reachable set [4]): Given a digraph $G = (V, E)$, a nonempty set $S \subset V$, an integer $r \geq 0$, S is an r -reachable set if there exists a vertex $u \in S$ satisfying $|\mathcal{N}_u^\downarrow \setminus S| \geq r$.

Definition 2 (r -robustness [4]): A digraph is r -robust, if for every pair of nonempty, disjoint $A \subset V$ and $B \subset V$, at least one of A and B is r -reachable.

B. r -robust Graph and the Condition for Resilient Consensus

We recall the condition for resilient consensus in a time-invariant synchronous network [4]. Each honest vertex in the network updates its value using a W-MSR algorithm. Each malicious vertex is allowed to send arbitrary but the same value to its out neighbors in each time step. A set of malicious vertices is said to be F -locally bounded if any honest vertex in the network has at most F malicious in-neighbors. r -robustness is the key to attain a guarantee for a consensus among honest vertices. It has been shown that in a synchronous system with a $\lfloor (r-1)/2 \rfloor$ -locally bounded malicious set, the honest vertices in a r -robust network eventually agree on a value in the convex hull of their initial values. Then we say that the system facilitates resilient asymptotic consensus [4].

III. PROBLEM FORMULATION

In this section we formulate the problem that we consider. Recall that exactly determining the robustness of a graph is coNP-complete. In this paper we consider the following approximation problem to tradeoff precision for improvement in running time.

Problem 1: Given a digraph $G = (V, E)$, two integers $r > 0$, $0 \leq \Delta \leq n$, find an algorithm which 1) certifies

r -robustness if G is $(r + \Delta)$ -robust; 2) refutes $(r + \Delta)$ -robustness if G is NOT r -robust.

If $\Delta = 0$, Problem 1 recovers the decision problem of determining whether or not a given digraph is r -robust [18], [15], [19].

We let the algorithm output accept to certify r -robustness, and output reject to refute $(r + \Delta)$ -robustness. An algorithm solves Problem 1 if the map between input and output satisfies Fig. 1. We note that the algorithm can output either accept or reject for instances that are r -robust but not $(r + \Delta)$ -robust by definition of Problem 1. An (additive) approximation algorithm with parameters r and Δ is not required to distinguish the instances in this category from instances in the other two categories. This is the limitation of the approximation approach.

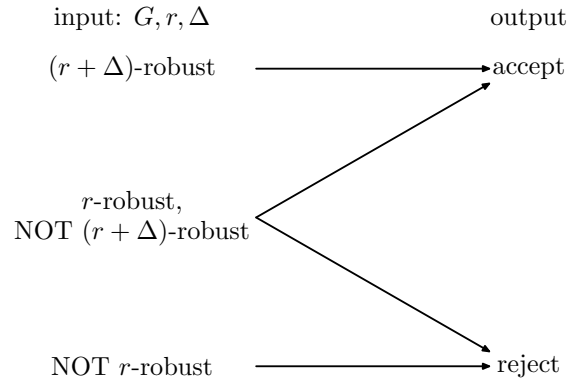


Fig. 1. The map between input and output given by an algorithm that solves Problem 1.

An algorithm that solves Problem 1 is a pessimistic testing algorithm for r -robustness in the sense that it produces no false answers but false negatives: it may reject r -robust networks but never accepts networks which are not r -robust. However it is not overly pessimistic since the rejected instances are not robust for a larger number $(r + \Delta)$. Alternatively, an algorithm that solves Problem 1 can also be viewed as an optimistic testing algorithm for $(r + \Delta)$ -robustness. Then the algorithm produces false positive answers but no false negative answers: it may accept networks that are not $(r + \Delta)$ -robust but never rejects $(r + \Delta)$ -robust networks. However it is not overly optimistic since the accepted instances are guaranteed to be robust for a smaller number r .

A Monte Carlo Algorithm: In this paper, we devise a Monte Carlo (MC) algorithm that behaves as follows: 1) certifies r -robustness, if G is $(r + \Delta)$ -robust; 2) refutes $(r + \Delta)$ -robustness with probability at least $(1 - \delta)$, if G is not r -robust.

In practice, we let $\delta = 1/3$. Then for a network that is not r -robust, it is rejected with probability at least $2/3$ in each independent run of the algorithm. We can amplify the probability of rejecting the instance to at least $(1 - \sigma)$ for

any $0 < \sigma < 1$ by running the algorithm $\lceil \frac{\ln(1/\sigma)}{\ln 3} \rceil$ times¹. If the network is rejected in any one run, then it is rejected by the MC algorithm. On the other hand, networks that are $(r + \Delta)$ -robust are always accepted by the MC algorithm.

IV. ALGORITHM

In this section we introduce an assumption for the minimum in-degree of the graph and discuss the implication of the assumption. Then we describe the approximation algorithm and provide its performance guarantee.

A. An Assumption for the Minimum In-Degree

We make an assumption for the minimum in-degree of the digraph. We show that the condition in the assumption is easy to check and is indispensable for a network to tolerate a naive attack that searches for a vertex with small in-degree and tampers with the value of half of its neighbors. Results without the assumption are left for discussion in Section VI-B.

Assumption 1: The minimum in-degree d_{\min} of the digraph G , defined as $d_{\min} \stackrel{\text{def}}{=} \min_{v \in V} \{|\mathcal{N}_v^\downarrow|\}$, is greater than $2r + \Delta$.

By checking $|\mathcal{N}_v^\downarrow|$ for all $v \in V$, an algorithm with running time² $O(m)$ determines whether or not Assumption 1 holds.

The robustness of a network can be interpreted as the minimum cost that a computationally unconstrained attacker has to pay to drive the system to undesired states. If we also consider the complexity of the problem, a computationally efficient strategy with a slightly larger cost could be in favor of the attacker. We show that if Assumption 1 does not hold, it only takes $O(m)$ running time for an attacker to find an attack strategy with a reasonable cost.

Lemma 1: Given G , r , and ϵ , if Assumption 1 does not hold, there exists an $O(m)$ -time algorithm ExamDegree which, by checking the degree of each vertex $u \in V$, finds a partition³ $(\{v\}, V \setminus \{v\})$ of G such that $\{v\}$ and $V \setminus \{v\}$ are not $(2r + \Delta)$ -reachable.

The proof of Lemma 1 is given in Appendix B.

If Assumption 1 is violated, then ExamDegree returns a vertex u with minimum in-degree. For the W-MSR algorithm discussed in [4], by attacking at least half of the in-neighbors of u , an attacker is able to prevent the honest vertex u from reaching resilient asymptotic consensus. In particular, the vertex u cannot remove all malicious messages without separating itself from all other honest vertices.

Given Lemma 1, we argue that to prevent a naive attack, it is necessary for the defender (or system designer) to ensure that Assumption 1 is satisfied. In the remainder of the paper we will assume that Assumption 1 holds. We revisit this issue and discuss arbitrary digraphs in Section VI-B.

¹The probability that all runs fail is at most $(1/3)^{\lceil \frac{\ln(1/\sigma)}{\ln 3} \rceil} \leq \sigma$. Therefore the instance is rejected in at least one run with probability at least $(1 - \sigma)$.

²For two positive functions f and g of the variable n , we denote $f = O(g)$ if there exist constants $n_0 > 0$ and $c > 0$, such that for all $n > n_0$, $f \leq c \cdot g$. We denote $f = \Omega(g)$ if $g = O(f)$.

³A 3-tuple (X, Y, Z) is said to be a partition of the graph $G = (V, E)$ if $\min\{|X|, |Y|\} \geq 1$, $X \cap Y = \emptyset$, $X \cap Z = \emptyset$, $Y \cap Z = \emptyset$, and $X \cup Y \cup Z = V$. We also denote (X, Y, \emptyset) as (X, Y) .

B. Reachability of Small Subsets

Under Assumption 1, we propose a randomized algorithm to solve Problem 1 with time complexity $\exp(\tilde{O}(1/\epsilon^2))m$. We will use the following concept of robustness in our analysis.

Definition 3 (β -close r -robustness): For a given $\beta \in [1/n, 1]$, a digraph $G = (V, E)$ is β -close to r -robustness, denoted r_β -robustness, if for every pair of nonempty, disjoint $A \subset V$ and $B \subset V$ with $\min\{|A|, |B|\} \geq \beta n$, at least one of A and B is r -reachable.

We note that the definition itself provides a weaker concept for network robustness.

To simplify notations we let $\epsilon \stackrel{\text{def}}{=} \Delta/n$. If Assumption 1 holds, we attain the following result:

Lemma 2: For a graph G that satisfies Assumption 1, the graph is r -robust if and only if it is r_ϵ -robust, where $\epsilon \stackrel{\text{def}}{=} \Delta/n$.

The proof of Lemma 2 is deferred to Appendix B.

Lemma 2 shows that under Assumption 1, sets with sizes less than ϵn are always r -reachable. Therefore to solve Problem 1, it suffices to approximately test r_ϵ -robustness of the network, by examining partitions (A, B, C) in which $\min\{|A|, |B|\} \geq \epsilon n$. In particular, an algorithm solves Problem 1 if it rejects instances which are NOT r_ϵ -robust, and accepts instances which are $(r + \Delta)_\epsilon$ -robust, where $\epsilon = \Delta/n$.

C. Algorithm Outline

Before diving into the details, we describe the overall process of the algorithm. Our algorithm is based on vertex sampling. For any fixed partition of the network, random sampling provides statistical information for the partition. If the network is not r_ϵ -robust, we seek to construct a partition (A, B, C) that violates r -robustness. We can only reconstruct it approximately. We prove that if there exists a partition (A, B, C) in which $\min\{|A|, |B|\} \geq \epsilon n$, and none of A and B is r -reachable, then the proposed algorithm finds a partition (A', B', C') in which none of A' and B' is $(r + \epsilon n)$ -reachable, with probability⁴ at least $(1 - \delta)$. The number of vertices that need to be sampled is a function of the parameters ϵ and δ , but independent of n , if ϵ and δ are positive constants. The function will be specified later in the paper.

The algorithm first randomly samples a set U of vertices from the graph. The size of U should be sufficiently large⁵ for estimating the number of vertices in \mathcal{N}_v^\downarrow that are also in subsets $A \cup C$ and $B \cup C$. For a partition (A, B, C) where A and B are not r -reachable, and $\min\{|A|, |B|\} \geq \epsilon n$, then with high probability there exists a partition $\pi(U) = (U_A, U_B, U_C)$ of U , such that $U_A \subset A$, $U_B \subset B$, $U_C \subset C$. In addition, we can estimate $|\mathcal{N}_v^\downarrow \cap (A \cup C)|$ and $|\mathcal{N}_v^\downarrow \cap (B \cup C)|$ using $|\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)|$ and $|\mathcal{N}_v^\downarrow \cap (U_B \cup U_C)|$ for all $v \in$

⁴The probability can be further amplified by repetition, as we have explained in Section III.

⁵A lower bound of $|U|$ will be later given as a function of ϵ and δ .

$(V \setminus U)$. Then there are constraints based on the sizes of the intersections that help us assign the rest of the vertices to their corresponding subsets. Specifically, if a vertex v is estimated to have a large number of in-neighbors in $(A \cup C)$, it is not likely that v belongs to B ; if a vertex v is estimated to have a large number of in-neighbors in $B \cup C$, it is not likely that v belongs to A . By utilizing these constraints we attain a partition (A', B', C') . Let $\Gamma_{A'}$ (resp. $\Gamma_{B'}$) be the set of vertices in A' (resp. B') such that each vertex in $\Gamma_{A'}$ (resp. $\Gamma_{B'}$) has the number of in-neighbors from $B' \cup C'$ (resp. $A' \cup C'$) greater or equal to a threshold of $r + O(\epsilon n)$. The attained partition (A', B', C') is constructed such that $|\Gamma_{A'}| + |\Gamma_{B'}|$ is $O(\epsilon n)$ with probability $(1 - \delta)$. Then we run one pass of updates to correct the assignments of the misclassified vertices. In this pass at most $O(\epsilon n)$ vertices are moved between A' , B' , and C' , therefore the pass does not change the number of neighbors of any vertices in A' , B' , or C' by more than $O(\epsilon n)$. Then we attain a partition (A', B', C') in which both A' and B' are not $r + O(\epsilon n)$ reachable.

Since we do not know which partition $\pi(U)$ of the sampled vertices U corresponds to the partition (A, B, C) that violates r_ϵ -robustness, we simply try all partitions of the sampled vertices.

D. The Sample-Based Algorithm

The algorithm $\text{SampledRbstTst}(G, \epsilon, \delta, r)$ outputs accept if the network is $(r + \epsilon n)$ -robust. It outputs reject with probability $(1 - \delta)$ if the network is not r -robust. The algorithm samples a set U of $t(\epsilon, \delta)$ vertices and examine all partitions of the set U . For each partition (U_A, U_B, U_C) of U , the algorithm calls 3 subroutines Restrict , Move , and TestReach . The algorithm Restrict is a one-pass algorithm that takes all vertices $V \setminus U$ and add the vertices to the partition (A', B', C') . The algorithm Move is a one-pass algorithm that refines (A', B', C') based on the partition returned by Restrict . The algorithm TestReach checks if a refutation of $(r + \epsilon n)$ -robustness is found.

The key to the approximation is the Restrict subroutine. The algorithm takes as input the graph G and all the parameters from SampledRbstTst , and a partition (U_A, U_B, U_C) of the sampled vertices U . A' , B' , and C' are initialized to be equal to U_A , U_B , and U_C respectively. Then the subroutine designates the rest of the vertices to one of A' , B' , and C' . We will analyze the algorithm in the next section.

V. ALGORITHM ANALYSIS

We show a Monte Carlo algorithm that solves Problem 1 with probability at least $(1 - \delta)$ for any $0 < \epsilon \leq 1$ and $\delta > 0$ in $\exp(\tilde{O}(1/\epsilon^2))m$ running time⁶.

Theorem 1: Given a graph G , two integers $r > 0$, $\epsilon \stackrel{\text{def}}{=} \Delta/n$ ($\epsilon \in (0, 1]$), under Assumption 1, Algorithm 1

- 1) certifies r -robustness if G is $(r + \epsilon n)$ -robust;
- 2) refutes $(r + \epsilon n)$ -robustness, with probability at least $(1 - \delta)$, if G is NOT r -robust;

⁶The notation $\tilde{O}(\cdot)$ hides factors of polynomials of $\ln(1/(\epsilon\delta))$.

Algorithm 1: $\text{SampledRbstTst}(G, \epsilon, \delta, r)$

Input : $G = (V, E)$, an error bound $\epsilon > 0$,
an error probability δ , a parameter r
Output: TestRslt: accept (a certificate for
 r -robustness) or reject (a refutation of
 $(r + \epsilon n)$ -robustness)

```

1  $p \leftarrow r/n$ ;
2 Sample a set  $U$  of size  $t(\epsilon, \delta)$  uniformly at random;
3 TestRslt  $\leftarrow$  accept;
4 for each 3-partition  $\pi(U) = (U_A, U_B, U_C)$  where
    $U_A \neq \emptyset$  and  $U_B \neq \emptyset$  do
5   # Algorithm 2
    $(A', B', C') \leftarrow \text{Restrict}(G, U_A, U_B, U_C, p, \epsilon, t)$ ;
6   # Algorithm 3
    $(A', B', C') \leftarrow \text{Move}(G, A', B', C', p, \epsilon)$ ;
7   # Algorithm 4
    $Rslt \leftarrow \text{TestReach}(G, A', B', C', p, \epsilon)$ ;
8   if  $Rslt = 0$  then
9     TestRslt  $\leftarrow$  reject;
10    Break;
11  end
12 end
```

Algorithm 2: $\text{Restrict}(G, U_A, U_B, U_C, p, \epsilon, t)$

Input : $G, U_A, U_B, U_C, p, \epsilon, t$
Output: a partition of all vertices
 $\pi'(V) = (A', B', C')$

```

1  $A' \leftarrow U_A, B' \leftarrow U_B, C' \leftarrow U_C$ ;
2 for each vertex  $v \in V \setminus U$  do
3   if  $\frac{|\mathcal{N}_v^+ \cap (U_A \cup U_C)|}{t} > (p + \epsilon/4)$  &
      $\frac{|\mathcal{N}_v^+ \cap (U_B \cup U_C)|}{t} > (p + \epsilon/4)$  then
4      $C' \leftarrow C' \cup \{v\}$ ;
5   else if  $\frac{|\mathcal{N}_v^+ \cap (U_A \cup U_C)|}{t} > (p + \epsilon/4)$  then
6      $A' \leftarrow A' \cup \{v\}$ ;
7   else if  $\frac{|\mathcal{N}_v^+ \cap (U_B \cup U_C)|}{t} > (p + \epsilon/4)$  then
8      $B' \leftarrow B' \cup \{v\}$ ;
9   else
10    add  $v$  to one of  $A'$ ,  $B'$ , or  $C'$  arbitrarily;
11 end
```

3) runs in $\exp(\tilde{O}(1/\epsilon^2)) \cdot m$ time.

Before we start proving Theorem 1, we explain the sampling method and prepare lemmas that provide guarantees to the subroutines of Algorithm 1.

Sampling: There are several ways to sample vertices from the network. One way is to assign independent Bernoulli random variables to each vertex and sample each vertex with the same probability of $O(\text{Poly}(1/\epsilon)) \frac{1}{n}$. The other two ways are to sample a fixed number of $t = O(\text{Poly}(1/\epsilon))$ vertices uniformly at random with and without replacement.

The analysis given in this paper is based on the scheme of uniformly sampling a fixed number of vertices with replace-

Algorithm 3: Move($G, A', B', C', p, \epsilon$)

Input : $G, A', B', C', p, \epsilon$, as explained
Output: a partition of all vertices
 $\pi'(V) = (A', B', C')$

```
1 ( $A'', B'', C''$ )  $\leftarrow (A', B', C')$ ;  
2 for each vertex  $v \in (V \setminus U)$  do  
3   if  $v \in A'$  &  $|\mathcal{N}_v^\downarrow \cap (B' \cup C')| > (pn + 3\epsilon n/4)$   
4     then  
5       if  $|\mathcal{N}_v^\downarrow \cap (A' \cup C')| > (pn + 3\epsilon n/4)$  then  
6          $A'' \leftarrow A'' \setminus \{v\}, C'' \leftarrow C'' \cup \{v\};$   
7       else  
8          $A'' \leftarrow A'' \setminus \{v\}, B'' \leftarrow B'' \cup \{v\}$   
9     else if  
10       $v \in B'$  &  $|\mathcal{N}_v^\downarrow \cap (A' \cup C')| > (pn + 3\epsilon n/4)$   
11        then  
12          if  $|\mathcal{N}_v^\downarrow \cap (B' \cup C')| > (pn + 3\epsilon n/4)$  then  
13             $B'' \leftarrow B'' \setminus \{v\}, C'' \leftarrow C'' \cup \{v\};$   
14          else  
15             $B'' \leftarrow B'' \setminus \{v\}, A'' \leftarrow A'' \cup \{v\}$   
16 end  
17 ( $A', B', C'$ )  $\leftarrow (A'', B'', C'');$ 
```

Algorithm 4: TestReach($G, A', B', C', p, \epsilon$)

Input : $G, A', B', C', p, \epsilon$, as explained
Output: Rslt: 1 if A' or B' are $(pn + \epsilon n)$ -reachable,
0 if both are not

```
1 Rslt  $\leftarrow$  0;  
2 for each vertex  $v \in V$  do  
3   if  $v \in A'$  &  $|\mathcal{N}_v^\downarrow \cap (B' \cup C')| \geq (pn + \epsilon n)$  then  
4     Rslt  $\leftarrow$  1;  
5   else if  $v \in B'$  &  $|\mathcal{N}_v^\downarrow \cap (A' \cup C')| \geq (pn + \epsilon n)$   
6     then  
7       Rslt  $\leftarrow$  1;  
8 end
```

ment. Technically speaking the outcomes of the sampling procedure are multisets. The intersection of the sampled multiset U and any set S is defined as a new multiset with support set $\text{support}(U) \cap S$ and occurrence number $\phi(u)$ the same as in U for any $u \in \text{support}(U) \cap S$. We ignore this point in the presentation of analysis as it is treated in the literature [23].

Approximating Number of Neighbors in Subsets: We begin by showing that $A \cup U$ and $B \cup U$ are non-empty with bounded probability:

Lemma 3: If $t \geq \frac{1}{\epsilon} \ln \frac{16}{\delta}$, with probability at least $1 - \delta/8$, for A and B with sizes $\min\{|A|, |B|\} \geq \epsilon n$, $A \cap U$ and $B \cap U$ are non-empty.

The proof of Lemma 3 is given in Appendix C.

Then we show the following result for the Restrict algorithm.

Lemma 4: Let (A, B, C) be a partition of G , $\min\{|A|, |B|\} \geq \epsilon n$, and both A and B are not r -reachable. Let U be a set of vertices sampled uniformly at

random, with $|U| \geq \frac{8}{\epsilon^2} \ln \frac{32}{\epsilon \delta}$. Let $\pi(U) = (U_A, U_B, U_C)$ be a partition of U which satisfies $U_A \subset A$, $U_B \subset B$, $U_C \subset C$. Then under Assumption 1, the Restrict algorithm, which takes as input $G, \pi(U), p, \epsilon$, and δ , outputs a partition (A', B', C') which satisfies the following property with probability $(1 - \delta)$:

(*) Let $\Gamma_{A'}$ (resp. $\Gamma_{B'}$) be the set of vertices in A' (resp. B') which consists of vertices v such that $|\mathcal{N}_v^\downarrow \cap (V \setminus A')| \geq r + 3\epsilon/4$ (resp. $|\mathcal{N}_v^\downarrow \cap (V \setminus B')| \geq r + 3\epsilon/4$), then $|\Gamma_{A'}| \leq \epsilon n/4$ (resp. $|\Gamma_{B'}| \leq \epsilon n/4$).

The proof of Lemma 4 is provided in Appendix C.

Correcting Large Violations: The Move algorithm updates (A', B', C') with the guarantee given by the following lemma.

Lemma 5: If Assumption 1 holds, then given a partition (A', B', C') which satisfies property (*) described in Lemma 4, the Move algorithm returns an updated (A', B', C') in which A' and B' are both not $(r + \epsilon n)$ -reachable.

The Proof of Lemma 5 is provided in Appendix C.

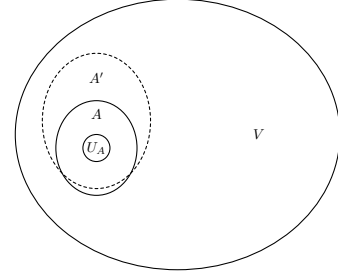


Fig. 2. The relationships between the subsets U_A , A , A' , and V .

The relationships between the subsets U_A , A and A' are shown in Fig. 2. The proposed algorithm ensures that A' and B' are disjoint. The sets $A' \cap B$ and $B' \cap A$ can be non-empty, but include $O(\epsilon n)$ vertices. The size of the set $A' \setminus A$ is not necessarily small. We do not show the details of the analysis because it is irrelevant to the correctness of the algorithm.

Complexity: Next we analyze the running time of the proposed algorithm.

Lemma 6: The running time of the SampledRbstTst algorithm is $\exp(\tilde{O}(1/\epsilon^2))m$.

The proof of Lemma 6 is shown in Appendix C.

Proof: [Proof of Theorem 1] Suppose Assumption 1 holds. By combining Lemmas 2, 3, 4, and 5, we know that if there exists a partition (A, B, C) where A and B are not r -reachable, we obtain a partition (A', B', C') in which A' and B' are not $(r + \Delta)$ -reachable with probability at least $(1 - \delta)$. Then Algorithm 4 will return 0 for such a partition (A', B', C') . Therefore Algorithm 1 returns reject for such instances with probability $(1 - \delta)$.

Algorithm 1 never rejects G that is $(r + \Delta)$ -robust because such partitions (A', B', C') in which A' and B' are not $(r + \Delta)$ -reachable does not exist. Therefore Algorithm 1 returns accept for such instances.

By combining the two properties stated above and Lemma 6, we attain Theorem 1. ■

VI. DISCUSSION

A. Estimating the Interval for Robustness

Given Algorithm 1, we can easily construct an algorithm that finds an interval of length at most $(1 + \beta)\Delta$ for any constant $\beta > 0$, which includes the maximal \bar{r} such that the digraph is \bar{r} -robust. The algorithm is a modified binary search. The lower bound $\underline{\ell}$ and upper bound $\bar{\ell}$ of the interval are initialized as 0 and $n/2$, respectively. In each round, if $\bar{\ell} - \underline{\ell} \geq (1 + \beta)\Delta$, we run Algorithm 1 with $p = (\underline{\ell} - \Delta + \bar{\ell})/2n$ and $\epsilon = \Delta/n$. If the algorithm returns accept, we let $\underline{\ell} \leftarrow (\underline{\ell} - \Delta + \bar{\ell})/2$, else we let $\bar{\ell} \leftarrow (\underline{\ell} + \Delta + \bar{\ell})/2$. We stop once $\bar{\ell} - \underline{\ell} \leq (1 + \beta)\Delta$ is satisfied. The interval $[\underline{\ell}, \bar{\ell}]$ is then returned as an estimation of the interval that includes \bar{r} . We note that the success probability of Algorithm 1 needs to be amplified to guarantee the overall success probability of the binary search. We omit the analysis since it follows straightforwardly by a union bound.

B. Without Assuming Minimum In-Degree

Throughout our analysis we assume that Assumption 1 holds. On the other hand, by combining the ExamDegree algorithm and the SampledRbstTst algorithm, we attain the following corollary for an arbitrary digraph.

Corollary 1: Given a digraph $G = (V, E)$, two integers $r > 0$, $\Delta > 0$, there exists an algorithm that runs in $\exp(\tilde{O}(1/\epsilon^2))m$ time which 1) outputs accept to certify r -robustness if G is $(2r + \Delta + 1)$ -robust; 2) outputs reject to refute $(2r + \Delta + 1)$ -robustness if G is not r -robust, with probability $(1 - \delta)$.

Proof: We first run ExamDegree to calculate d_{\min} . If $d_{\min} \leq 2r + \Delta$, we let the algorithm output reject; if $d_{\min} > 2r + \Delta$, then we let the algorithm output the result returned by SampledRbstTst($G, \epsilon \stackrel{\text{def}}{=} \Delta/n, \delta, r$). ■

C. Limitation of the Algorithm

The algorithm cannot be applied to cases where $\epsilon = 0$ regardless of the running time. In addition, if ϵ is $O(n^{-1/2})$, the running time is worse than the $\exp(O(n))m$ time exact algorithm [18]. The gain in efficiency is attained if ϵ is $\Omega(n^{-\frac{1}{2}+c})$ for a constant $c > 0$. If $\epsilon > 0$ is a fixed constant (independent of n), the algorithm is a fixed parameter algorithm with running time linear in m , although it also depends on the fixed parameter ϵ . We note that arbitrary dependency only on the parameter is allowed for fixed parameter algorithms. Similar dependency appears in property testing algorithms [23] and approximation algorithms [25] for dense graphs.

VII. CONCLUSION AND FUTURE WORK

We have proposed a sample-based algorithm to approximately test r -robustness of a network. Computational complexity of the algorithm is investigated. The algorithm shows a tradeoff between precision and running time. Future work includes improving the running time of the algorithm, discussing the impact of regularity conditions in graphs, and investigating other approaches of approximation.

REFERENCES

- [1] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, 1980.
- [2] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Progr. Lang. Sys.*, vol. 4, no. 3, pp. 382–401, 1982.
- [3] D. Dolev, "The Byzantine generals strike again," *J. Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.
- [4] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, 2013.
- [5] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative approximate Byzantine consensus in arbitrary directed graphs," in *Proc. 2012 ACM Symp. Princ. Distrib. Comput.*, 2012, p. 365–374.
- [6] N. H. Vaidya and V. K. Garg, "Byzantine vector consensus in complete graphs," in *Proc. 2013 ACM Symp. Princ. Distrib. Comput.*, 2013, p. 65–73.
- [7] L. Tseng and N. Vaidya, "Iterative approximate byzantine consensus under a generalized fault model," in *ICDCN '13: International Conference on Distributed Computing and Networking*, 2013, pp. 72–86.
- [8] N. H. Vaidya, "Iterative Byzantine vector consensus in incomplete graphs," in *ICDCN '14: International Conference on Distributed Computing and Networking*. Springer, 2014, pp. 14–28.
- [9] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proc. 2016 ACM Symp. Princ. Distrib. Comput.*, 2016, p. 425–434.
- [10] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Trans. Automat. Contr.*, vol. 64, no. 3, pp. 1063–1076, 2018.
- [11] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Trans. on robotics*, vol. 33, no. 3, pp. 565–582, 2017.
- [12] L. Su and N. H. Vaidya, "Defending non-Bayesian learning against adversarial attacks," *Distributed Computing*, vol. 32, no. 4, pp. 277–289, 2019.
- [13] A. Mitra and S. Sundaram, "Byzantine-resilient distributed observers for lti systems," *Automatica*, vol. 108, p. 108487, 2019.
- [14] L. An and G.-H. Yang, "Byzantine-resilient distributed state estimation: A min-switching approach," *Automatica*, vol. 129, p. 109664, 2021.
- [15] H. Zhang, E. Fata, and S. Sundaram, "A notion of robustness in complex networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 310–320, 2015.
- [16] D. Saldana, A. Prorok, S. Sundaram, M. F. Campos, and V. Kumar, "Resilient consensus for time-varying networks of dynamic agents," in *2017 American control conference (ACC)*. IEEE, 2017, pp. 252–258.
- [17] S. M. Dibaji and H. Ishii, "Resilient consensus of second-order agent networks: Asynchronous update rules with delays," *Automatica*, vol. 81, pp. 123–132, 2017.
- [18] H. J. LeBlanc and X. D. Koutsoukos, "Algorithms for determining network robustness," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, 2013, pp. 57–64.
- [19] J. Usevitch and D. Panagou, "Determining r - and (r,s) -robustness of digraphs using mixed integer linear programming," *Automatica*, vol. 111, p. 108586, 2020.
- [20] L. Guerrero-Bonilla, A. Prorok, and V. Kumar, "Formations for resilient robot teams," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 841–848, 2017.
- [21] L. Guerrero-Bonilla, D. Saldana, and V. Kumar, "Dense r -robust formations on lattices," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6633–6639.
- [22] E. M. Shahrivar, M. Pirani, and S. Sundaram, "Spectral and structural properties of random interdependent networks," *Automatica*, vol. 83, pp. 234–242, 2017.
- [23] O. Goldreich, S. Goldwasser, and D. Ron, "Property testing and its connection to learning and approximation," *J. ACM*, vol. 45, no. 4, pp. 653–750, 1998.
- [24] D. Ron, *Algorithmic and analysis techniques in property testing*. Now Publishers Inc., 2010.
- [25] S. Arora, D. Karger, and M. Karpinski, "Polynomial time approximation schemes for dense instances of NP-hard problems," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 193–210, 1999.
- [26] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Am. Stat. Assoc.*, pp. 13–30, 1963.

APPENDIX

A. Probabilistic Inequalities

We use Markov's inequality and additive Chernoff bounds.

Lemma 7 (Markov's Inequality): Let X be a non-negative random variable, then for all $k > 0$, $\mathbb{P}[X \geq k \cdot \mathbb{E}[X]] \leq \frac{1}{k}$.

Lemma 8 (Chernoff bound, [26], [23]): Let X_1, X_2, \dots, X_t be t independent Bernoulli random variables where $X_i \in \{0, 1\}$. Let $q \stackrel{\text{def}}{=} (1/t) \sum_i \mathbb{E}[X_i]$. Then for every $\gamma \in [0, 1]$, the following bounds hold: $\mathbb{P}[\frac{1}{t} \sum_{i=1}^t X_i > q + \gamma] < \exp(-2\gamma^2 t)$, and $\mathbb{P}[\frac{1}{t} \sum_{i=1}^t X_i < q - \gamma] < \exp(-2\gamma^2 t)$.

B. Proofs from Section IV

Proof: [Proof of Lemma 1] We note that $d_{\min} \geq r$ always holds. If Assumption 1 does not hold, then $r \leq d_{\min} \leq (2r + \Delta)$. The problem is trivial in this case because in the partition $(\{u\}, V \setminus \{u\})$, both non-empty subsets are not $(2r + \Delta)$ reachable for a vertex u with minimum degree. An $O(m)$ time algorithm finds a vertex with minimum in-degree and the corresponding partition. We refer to the algorithm as ExamDegree. ■

Proof: [Proof of Lemma 2] We first show sufficiency. Suppose the digraph G is r_ϵ -robust. For any vertex v in a non-empty subset $A \subset V$ with $|A| < \epsilon n$, $|\mathcal{N}_v^\downarrow \cap (V \setminus A)| \geq |\mathcal{N}_v^\downarrow| - (|A| - 1) > 2r + \epsilon n - \epsilon n + 1 = 2r + 1 > r$. For any A satisfying $|A| \geq \epsilon n$, $|\mathcal{N}_v^\downarrow \cap (V \setminus A)| \geq r$ holds by Definition 3. Then A is r -reachable regardless of its size. B is also r -reachable by similar analysis. Therefore G is r -robust. The necessity is straightforwardly attained from definitions of r -robustness and r_β -robustness. ■

C. Proofs from Section V

Proof: [Proof of Lemma 3] For any $\epsilon > 0$, we have $\mathbb{P}_U[A \cap U = \emptyset] \leq (1 - \epsilon)^t < \frac{\delta}{16}$, and $\mathbb{P}_U[B \cap U = \emptyset] \leq (1 - \epsilon)^t < \frac{\delta}{16}$. By a union bound, the probability that $A \cap U$ or $B \cap U$ is empty is less than $\delta/8$. ■

Proof: [Proof of Lemma 4] Each vertex u in U is sampled uniformly at random, we let $X_{u,v}$ be the indicator variable for whether or not $u \in (\mathcal{N}_v^\downarrow \cap (A \cup C))$ for a vertex v . Then $X_{u,v}$ is a Bernoulli random variable with $\mathbb{P}[X_{u,v} = 1] = |\mathcal{N}_v^\downarrow \cap (A \cup C)|/n$ and $\mathbb{P}[X_{u,v} = 0] = 1 - |\mathcal{N}_v^\downarrow \cap (A \cup C)|/n$. Then $|\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)| = \sum_{u \in U} X_{u,v}$ is a sum of independent random variables.

We let $t_0 \stackrel{\text{def}}{=} \frac{8}{\epsilon^2} \ln \frac{32}{\epsilon \delta}$, and $t \geq t_0$. We prove that if a vertex v has more than $(p + \epsilon/2)n$ neighbors in $A \cup C$, the probability that it has less than $(p + \epsilon/4)t$ neighbors in $U_A \cup U_C$ is small. It follows that

$$\begin{aligned} \mathbb{P}_U \left[\frac{|\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)|}{t} < (p + \epsilon/4) \right] \\ \leq \mathbb{P}_U \left[\frac{|\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)|}{t} < \frac{|\mathcal{N}_v^\downarrow \cap (A \cup C)|}{n} - \epsilon/4 \right] \\ < \exp(-2(\epsilon/4)^2 t) \leq \frac{\epsilon \delta}{32}. \end{aligned}$$

The first inequality is due to the fact that the latter event is a superset of the first one. The second inequality is by an additive Chernoff bound.

Similarly we prove that if a vertex v has less than pn neighbors in $A \cup C$, the probability that it has more than $(p + \epsilon/4)t$ neighbors in $U_A \cup U_C$ is small.

$$\mathbb{P}_U \left[\frac{|\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)|}{t} > (p + \epsilon/4) \right] < \frac{\epsilon \delta}{32}.$$

Similar results also hold for the sets $B \cup C$ and $(U_B \cup U_C)$. By a union bound, the probability that any of these events happen for vertex v is less than $\epsilon \delta/8$.

Let X be a random variable defined as the number of vertices that have bad estimations of their neighborhood⁷. By the linearity of expectation, $\mathbb{E}[X]$ is less or equal to $\epsilon \delta n/8$. We let $\lambda = 2/\delta$. Then we attain

$$\mathbb{P} \left[X \geq \frac{2}{\delta} \cdot \frac{\epsilon \delta n}{8} \right] \leq \mathbb{P} \left[X \geq \frac{2}{\delta} \cdot \mathbb{E}[X] \right] \leq \frac{\delta}{2}.$$

The first inequality holds because $\mathbb{E}[X] \leq \epsilon \delta n/8$; the second inequality is due to Markov's inequality. Therefore the probability that at least $\epsilon n/4$ vertices have a bad estimation of their neighborhood is less or equal to $\delta/2$. The probability that less than $\epsilon n/4$ vertices have a bad estimation is greater or equal to $(1 - \delta/2)$.

Next we analyze the Restrict Algorithm. With probability $(1 - \delta)$, for at least $(1 - \epsilon/4)n$ vertices the following inequalities hold: $||\mathcal{N}_v^\downarrow \cap (U_A \cup U_C)|/t - |\mathcal{N}_v^\downarrow \cap (A \cup C)|/n| \leq \epsilon/4$. We call these vertices with good estimations *normal* vertices.

Since the ordering of vertices for the loop does not affect the outcome of the subroutine, w.l.o.g., we assume that all *normal* vertices are added with high priority.

From Line 3 and 4 of Algorithm 2, we know that all *normal* vertices with more than $(pn + \epsilon n/2)$ in-neighbors in both $A \cup C$ and $B \cup C$ are added to C' . From Line 5 and 6, we observe that no *normal* vertices with more than $pn + \epsilon n/2$ in-neighbors in $A \cup C$ is added to B' . From Line 7 and 8, we attain that no *normal* vertices with more than $pn + \epsilon n/2$ in-neighbors in $B \cup C$ is added to A' . No *normal* vertices will fall in to the case of Line 9 and 10, because otherwise it implies that $|\mathcal{N}_v^\downarrow \cap (A \cup C)| \leq (p + \epsilon/2)n$, and $|\mathcal{N}_v^\downarrow \cap (B \cup C)| \leq (p + \epsilon/2)n$. Then we attain $|\mathcal{N}_v^\downarrow| \leq |\mathcal{N}_v^\downarrow \cap (A \cup C)| + |\mathcal{N}_v^\downarrow \cap (B \cup C)| \leq (2p + \epsilon)n$, which contradicts Assumption 1.

After we add all the *normal* vertices, the rest of the vertices are added to A' , B' , and C' arbitrarily. Therefore, after the execution of the Restrict algorithm, we let $\Gamma_{A'}^{(0)} \stackrel{\text{def}}{=} \{v \mid v \in A', |\mathcal{N}_v^\downarrow \cap (V \setminus A)| \geq pn + \epsilon n/2\}$. We know that (with probability $1 - \delta$), $|\Gamma_{A'}^{(0)}| \leq \epsilon n/4$. In addition, $|A \setminus A'| \leq \epsilon n/4$, because these vertices are not *normal*. Then we attain that $\forall v \notin \Gamma_{A'}^{(0)}, |\mathcal{N}_v^\downarrow \cap (V \setminus A')| < pn + 3\epsilon n/4$.

⁷We call v a vertex with a bad estimation of its neighborhood if any of the 4 events discussed above happen to v .

Therefore $|\Gamma_{A'}| \leq |\Gamma_{A'}^{(0)}| \leq \epsilon n/4$. A similar result holds straightforwardly for B' and $\Gamma_{B'}$. ■

Proof: [Proof of Lemma 5] After moving vertices with greater or equal to $pn + 3\epsilon n/4$ in-neighbors in $V \setminus A'$ to the outside of A' in the Move algorithm, we obtain an A' in which all $v \in A'$ satisfy $|\mathcal{N}_v^\downarrow \cap (V \setminus A')| < pn + \epsilon n$. Similarly for $v \in B'$, $|\mathcal{N}_v^\downarrow \cap (V \setminus B')| < pn + \epsilon n$. ■

Proof: [Proof of Lemma 6] There are $\exp(\tilde{O}(1/\epsilon^2))$ partitions of the sampled set U that need to be checked. The running time of Restrict, Move, and TestReach for each partition is $O(m)$. ■

D. Algorithm in Practice

We discuss approaches to further improve the efficiency of the algorithm and the quality of the result.

The first technique is to randomly assign each vertex in U to one of three subsets U_A, U_B, U_C . It would take longer for the random partition to hit the correct partition $\pi^*(U)$ that aligns with the true partition (A, B, C) , however, by making a few changes to the algorithm we can tolerate some errors in partitioning U . Hitting a partition that is close enough to $\pi^*(U)$ with high probability turns out to be more efficient than enumerating all the partitions of U , although it does not improve the upper bound $\exp(\tilde{O}(1/\epsilon^2))m$. We also note that by using random partitions, the algorithm can be readily parallelized.

To use random partitions, we need to make the following changes to the algorithm. (1) we run the for loop (Line 2 to 13) of Algorithm 3 over $v \in V$ instead of $v \in (V \setminus U)$. (2) In the for loop (Line 2 to 13) of Algorithm 3, we always check if by moving a vertex between partitions, A or B becomes empty. In that case we do not move the vertex.

The second technique is pruning. When we search for $\pi^*(U)$, we do not utilize the information contained in the subgraph $G[U]$. In fact by considering this information we can rule out some of the partitions of U . Pruning does not change the complexity of the algorithm either.

The third technique is also used to deal with the heavy dependency of the running time on $1/\epsilon$. Even with a constant ϵ , the number of partitions of U is $\exp(\tilde{O}(1/\epsilon^2))$, which can be too large from a practical point of view. In practice t can be set to a value according to the computational resource available. However, it must be used with cautions when $t < \frac{8}{\epsilon^2} \ln \frac{32}{\epsilon\delta}$. In such a case the algorithm loses the guarantee given in the analysis. We note that even without the guarantee, the algorithm is a heuristic that can be used by the attacker to find a weak partition or by the defender to check the robustness by simulating an attacker.

Lastly, we can use heuristics to further improve the results attained by Algorithm 1. Local search can be used to reduce the reachability of A' or B' without increasing the reachability of the other. Charikar's greedy algorithm for the densest subgraph problem can also be modified to design a heuristic to post-process the results. However we do not provide guarantees for these techniques.

E. Numerical Examples

We consider example digraphs with 200 vertices. We construct the digraphs such that their robustness is known by construction. Then we permute the labels of the vertices and use the digraphs as input to Algorithm 1. The algorithm then finds the hidden partition in which A and B are not R -reachable, and returns the smallest R among all the partitions that it reconstructed.

The digraphs are constructed as follows. We first partition the vertices into 3 subsets A, B , and C . We specify the sizes of A and B . For simplicity we let $|A| = |B|$, and $|C| = 200 - (|A| + |B|)$. Each subset forms a complete digraph. Then for each vertex u in A (resp. B), we find \bar{r} vertices in $B \cup C$ (resp. $A \cup C$) uniformly at random, and add edges from these vertices to u . Finally, we add all edges (u, v) where $u \in (A \cup B)$ and v is in C . When $\min\{|A|, |B|, |C|\} \geq 2\bar{r}$, the constructed digraphs are \bar{r} -robust, but not $(\bar{r} + 1)$ -robust.

We note that sizes of A, B , and C , the number \bar{r} , and the ordering of the vertices are assumed to be unknown. In real scenarios one needs to choose an r as part of the input. In our examples, we let $r = \bar{r} + 1$ to test the effectiveness of the algorithm.

To test the proposed algorithm, we let $\epsilon = 0.15$, which implies $\Delta = \epsilon n = 30$. Since we construct a digraph such that it is \bar{r} -robust but not $(\bar{r} + 1)$ -robust, the algorithm should return a pair of sets A' and B' such that both sets are not $(r + \Delta)$ -robust.

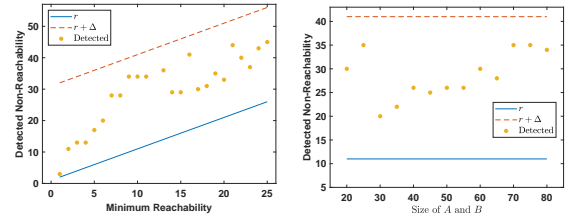


Fig. 3. Reachability of detected sets. Left: fixed $|A| = |B| = 70$ and various minimum reachability \bar{r} ; Right: fixed minimum reachability $\bar{r} = 10$ and various $|A| = |B|$. The solid lines show the optimum values; the dashed lines show the thresholds used by Algorithm 4; the data points show the values of violation detected by the algorithm.

The algorithm is implemented using Matlab 2021b. We run the algorithm using a single thread on a laptop computer with an Intel core i5-8365U CPU (1.6 GHz). We choose $t = 9$, and run 3 trails for each instance. We have discussed the issue of practically choosing t in Section D.

Fig. 3 shows the returned R in various settings. In almost all the cases the algorithm finds subsets A' and B' , of which none is R -reachable, where $R \leq r + \Delta$. The only exception is the case where $\bar{r} = 12$ in the first figure. In this failed case the algorithm returns a partition in which both A' and B' are not 82 reachable.

Most of the instances are rejected within 30 seconds. The 3 trails of the failed case take total time of less than 10^3 seconds to finish. It is of interest to compare with exact testing algorithms given in [18], [19]. However, since these comparisons are time costly for large networks, we leave them to future work.