Discover Internet of Things

Research

Determining critical nodes in optimal cost attacks on networked infrastructures

Ishfaq Ahmad[1] · Addison Clark[1] · Muhammad Ali[1] · Hansheng Lei[2] · David Ferris[3] · Alex Aved[3]

Received: 27 April 2023 / Accepted: 29 November 2023
Published online: 18 January 2024

Abstract
A wide range of critical infrastructures are connected via wide area networks as well as the Internet-of-Thing (IoT). Apart from natural disasters, these infrastructures, providing services such as electricity, water, gas, and Internet, are vulnerable to terrorist attacks. Clearly, damages to these infrastructures can have dire consequences on economics, health services, security and safety, and various business sectors. An infrastructure network can be represented as a directed graph in which nodes and edges denote operation entities and dependencies between entities, respectively. A knowledgeable attacker who plans to harm the system would aim to use the minimum amount of effort, cost, or resources to yield the maximum amount of damage. Their best strategy would be to attack the most critical nodes of the infrastructure. From the defender's side, the strategy would be to minimize the potential damage by investing resources in bolstering the security of the critical nodes. Thus, in the struggle between the attacker and defender, it becomes important for both the attacker and defender to identify which nodes are most critically significant to the system. Identifying critical nodes is a complex optimization problem. In this paper, we first present the problem model and then propose a solution for computing the optimal cost attack while considering the failure propagation. The proposed model represents one or multiple interconnected infrastructures. While considering the attack cost of each node, the proposed method computes the optimal attack that a rational attacker would make. Our problem model simulates one of two goals: maximizing the damage for a given attack budget or minimizing the cost for a given amount of damage. Our technique obtains solutions to optimize the objective functions by utilizing integer-linear programming while observing the constraints for each of the specified goals. The paper reports an extensive set of experiments using various graphs. The results show the efficacy of our technique in terms of its ability to obtain solutions with fast turnaround times.

Keywords Security · Critical Infrastructures · Optimization · Graph Analysis

1 Introduction

The safety and security of IoT-enabled infrastructure encompass more than cyber-attacks. Real dangers can arise from natural calamities and terrorist attacks, causing not just functional damage but also physical damage [23, 25, 27, 30]. Large-scale IoT utility infrastructures are made up of individual entities that are often connected via various IoT and other networks. Often, multiple IoT networks, such as electricity, gas, water, and Internet services are interdependent

✉ Ishfaq Ahmad, iahmad@cse.uta.edu; Addison Clark, addison.clark@mavs.uta.edu; Muhammad Ali, muhammad.ali3@mavs.uta.edu; Hansheng Lei, Hansheng.Lei@utrgv.edu; David Ferris, david.ferris.3@us.af.mil; Alex Aved, alexander.aved@us.af.mil | [1]Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX, USA. [2]The University of Texas at Rio Grande, Brownsville, TX, USA. [3]Air Force Research Labs, Rome, NY, USA.

[26]. An entity within one infrastructure may have a subset of critical entities for continued operation and also other entities that may depend on it. Thus, when a single entity fails, its failure may propagate along the network causing a portion of the system, or the entire system, to be disrupted. Due to the dependence on other entities and the physical characteristics of each entity, certain nodes may be more important to the system than others. For instance, in an electric power grid, the fuel store is a critical entity. Other entities that may critically depend on the electric power grid may be within other infrastructures such as water and gas [22, 25, 27, 30, 31]. Due to the interconnection and interdependency of these infrastructures, the failure of some entities in one system may even shut down a subset of entities in another system. There have been numerous examples of damage propagation including several massive blackouts worldwide. One of the prime examples is the electrical blackout that occurred in a large portion of Italy in 2003 [1]. The catastrophe was of enormous proportions. The disabled power stations triggered the shutdown of service points in the internet communication network [22, 25, 27]. This caused the damage to spread to controllers of other entities that relied on the internet, which caused more power stations to shut down. Another notable episode is the vast blackout of July 2012, which affected more than 600 million people in India [2]. Because of the propagated damage and avalanche effect infrastructures are more vulnerable to attack and sabotage. An informed adversary that plans to attack the infrastructure system likely aims to cause the maximum amount of damage with the least effort possible. Therefore, the most rational strategy would be to attack the most critical entities in order to cause the most damage. In contrast, a system operator could work to minimize damage by protecting the most critical entities of the system [21, 24, 29, 31].

An infrastructure network can be represented as a directed graph in which nodes and edges denote operation entities and dependencies between entities, respectively. Critical nodes are the set of nodes that a rational attacker is likely to attack because they maximize the damage to the system. Damage is defined as the number of nodes in the system that are disabled. Thus, it is important for both the defender and attacker to identify the most critical nodes of the network. Identifying the critical nodes in a graph is a complex problem because out of all possible attacks, the one that is optimal must be found.

The cost of a node is the weight representing the number of resources, such as money, ammunition, or another metric relevant to the specific problem, that would be required to disable that node. The graph of an infrastructure network must be augmented with an attack cost for each node. Next, one can define the budget of the attack, which is the total amount of resources that the attacker can use to disable nodes. Given this maximum budget for an attack, the optimal attack would aim to yield the most damage. A simplistic approach would be to compute all possible attacks, filtering those that disable the entire network, and then selecting the one that costs the minimum. The runtime complexity of such an approach is exponential in terms of the number of nodes. Critical infrastructures typically have nodes ranging from hundreds to hundreds of thousands, making this simplistic approach infeasible. Therefore, it is important to design techniques that yield solutions in a fast turnaround time. A special case of this problem is when there are no cycles in the network, implying the graph is a Directed Acyclic Graph (DAG). When an attacker aims to disable the entire network, one can simply determine the nodes with zero in-degree, that is, the sources in the network. Attacking the sources directly is the only way to disable them. Once all sources are disabled, their failure propagates to the dependent nodes, which, in turn, will disable all the nodes with in-degree edges from the sources. Hence, attacking the sources is the minimum cost attack for a DAG. This approach is also very fast because only the nodes with zero in-degree need to be identified. However, general real-world networks may not be represented as DAGs since most networks contain redundant links to enhance fault tolerance and robustness.

This paper presents a method of computing the optimal cost attack on an infrastructure network that considers failure propagation. Given a graph that represents one or multiple infrastructure networks and the cost to attack each node, the proposed method computes the optimal attack that a rational attacker is likely to make. Our technique uses integer-linear programming to obtain the solution, given the objective functions and set of constraints for each of the specified goals. That is, maximizing the damage for a given attack budget or minimizing the cost for a given amount of damage. Our problem formulation is simple and intuitive, making it easy to understand and apply to some real-world environments. Most importantly, as demonstrated in the experimental results, the proposed solution is very fast regarding runtime complexity.

The rest of this paper is organized as follows: The next section discusses related work, including graph theory and other defense and attack models for networks. Section 3 presents the problem formulation. Section 4 describes the experimental setup and results. Section 5 provides some conclusions and the discussion on future work.

2 Related work

Various researchers have addressed the attacker-defender problem in various contexts and angles. Here, we address the physical attacks and not cyber-attacks. For the latter, see various kinds of attacks and their comparison in [3]. The work reported in [4] defines the resilience of large infrastructures in terms of the operation of each individual component that makes up the system. It suggests that quantitative models can facilitate infrastructure operators with three significant features: First, to analyze the overall resilience of an entire infrastructure system. Second, to identify critical vulnerabilities that may threaten the system's operation in times of stress. Third, to provide policymakers of the infrastructure with suggestions for allocating new resources best to improve the system's resilience.

Many researchers have long used graph theory for effectively modeling real-world networking problems. In 1847, Kirchhoff, one of the pioneers of graph theory, applied it to solve electrical circuits [5]. Graphs can capture the important details of complex networking systems, reducing them to elegant and simple graph theoretical equivalents without losing critical information. One example of recent work is [4], which demonstrates the efficacy of using graphical models to improve the resiliency of critical infrastructures. Several additional examples of modeling failure propagation in graphs are [6–10]. However, most of these research works assume factors that may reduce problem complexity, which loses some of the realistic aspects of the system. The work in [6] compares its model with the uniform model reported in [7] and the small clusters model reported in [10]. The results indicate that the uniform model is too simplistic to apply to all systems due to the underlying assumptions it makes. In order to solve these problems, the authors propose the so-called HINT model, another failure propagation formulation for interdependent power and communication networks. The research work reported in [11] expands upon this model, again focusing on interconnected and interdependent networks. The heterogeneity of the different networks is considered in failure propagation, unlike in many generic models where all nodes are considered to be the same. Additionally, feature selection is used to identify the most important features categorizing the critical nodes of the networks.

The work reported in [9] considers smart grids for evaluating their robustness in the presence of propagating damage. The proposed technique employs the percolation theory to simulate cascading failures. It does so by guessing the set of nodes that remain functional after the process is complete. The paper reports an important observation that in smart grid networks, there exists a certain threshold for the proportion of faulty nodes with respect to that of the working nodes in the system, beyond which the whole system is guaranteed to go dysfunctional. The paper also provides a relationship between the level of robustness that the operator of the network can ensure and the costs they are likely to incur in order to control and monitor their system. The work in [7] presents a framework for modeling robustness in interdependent networks that are subject to cascading failures. It also proposes a scheme to solve the problem of identifying critical nodes in interacting networks. The critical fraction of nodes, when removed, leads to a failure propagation situation, causing a complete fragmentation of the two interdependent networks. The work also compared the proposed models for interdependent networks against isolated single networks, showing that interdependent networks are more vulnerable to random failures as the degree distribution gets broader.

The paper [12] proposes a game theoretical defense model, called the defender-attacker-defender (DAD) model. Effectively, it is a three-stage sequential game model and consists of two players. The first is a *defender* who starts by making budget-limited improvements and takes other necessary measures to maximize the resilience of an infrastructure system and minimize the damage that an attack could cause. The second is an *attacker*, which observes the prior actions of the *defender* and makes an intelligent attack on the system that maximizes damage. [13] devises a strategy to mitigate the propagation of failures in a generic network. Again, specific nodes are selected for protection. However, there is very little knowledge required of the network as a whole in order for the algorithm presented in [13] to protect local portions of the network from cascading failure.

Much of the above work relies solely on graph theory to analyze the presented techniques [21, 25, 28, 29, 31, 32]. Conversely, [14] uses both graph theory and simulation to analyze critical nodes in power grids. Other work in [15] analyzes network graphs to determine the best way to repair a network post-disaster. The allocation of limited network resources is modeled as a non-linear programming optimization problem. The graph theory here is similar to other works listed above, but the purpose of network repair and continued emphasis on critical nodes for the best network coverage are important considerations for infrastructure analysis.

Most of the above-mentioned models are specific to a single system and are designed to capture the dependencies between different networks, often the power and communication networks. In contrast, our model is designed to be generic and applicable to a wider range of networks and allows each node to be a real-world entity. The producers and consumers of all resources can be included in the graph as nodes without losing generality. This model is therefore applicable to more problems and scenarios than the previous models.

3 Problem formulation

Given a graph G = (V, E) and the attribute *w(u)*, the attack cost of each node *u*, we have to compute the optimal attack; that is, the attack that achieves *Damage Maximization*. The graph is constructed such that the infrastructure entities are represented as nodes and the dependencies between entities are arcs between the nodes as suggested by the *k-n* dependency model [9]. If there is an edge from node *u* to node *v*, *(u, v)*, this means node *v* is dependent on node *u*. Node *u* has an attack cost of *w(u)* which is determined by the threat potential and arranged security. This is different for each node. The *total attack cost* of an attack is the sum of the attack cost of all nodes that were targeted. *Damage* to the system is defined by the total number of disabled nodes in the network. Table 1 lists the notation used in this paper.

An adversary planning an attack on a single real-world entity has two options for their strategy–they can either attack the entity directly or prevent access to all of the entity's supplying nodes, such as fuel in the case of the power grid example. To model these two options, we say that node *u* is disabled if it is attacked directly or if all of the nodes on which *u* depends are disabled in any way. This recursive definition of node failure allows damage to propagate throughout the graph to simulate failure propagation. We formulate the problem as a *Mixed Integer Linear Programming Problem (MILP)*.

3.1 Damage maximization

Each node is assigned its cost property during the generation of the graph. For our example graphs, costs are assigned randomly using uniform, normal, or exponential distributions. These cost properties represent how difficult or expensive it would be to attack a particular node. These weights are assigned before the attacker has done anything. Figure 1a shows an example graph with nodes, edges, and node costs.

If the attacker has a fixed budget, then their goal is *Damage Maximization*; that is, to cause the maximum amount of damage to the network without exceeding their budget. Figure 1b shows this example where the attacker has a budget of B = 1. The cost of this attack is $C = \sum w(u) \times \alpha_u$, i.e., the sum of the weights of the attacked nodes. By attacking nodes 1 and 3, we have a total cost of $C = 0.65 + 0.19 = 0.84 \leq 1$. Node 0 is disabled via failure propagation since it is depended upon node 1 and no other nodes. Nodes 2 and 4 remain functional as their attack costs are too high to fit within the specified budget.

If the attackers instead have a desired amount of damage, or degradation, they will want to minimize the cost of the attack while still achieving the desired damage. Figure 1c shows this scenario with a desired degradation of D = 1, that is, the entire network is disabled. The cost of this attack is $C = \sum w(u) \times \alpha_u$. In this case, it is C = 0.92 + 0.19 = 1.11. This is the minimum cost to disable the entire network, which is a degradation of D = 1. Nodes 3 and 4 are attacked and the failure propagates to all other nodes.

Table 1 Notation used

Symbol	Meaning
V	Set of nodes
E	Set of edges
V_u^{in}	Set of nodes that node *u* is dependent upon
$deg_{in}(u)$	The cardinality of V_u^{in}
$w(u)$	Cost to attack node *u*
α_u	*Binary variable*: 1 if node *u* is attacked and 0 otherwise
β_u	*Binary variable*: 1 if node *u* is functional and 0 otherwise
y_u	*Binary variable*: 1 if any node in V_u^{in} is enabled and 0 otherwise
B	Attack budget
D	Desired degradation between 0 and 1
C	The total cost of the attack

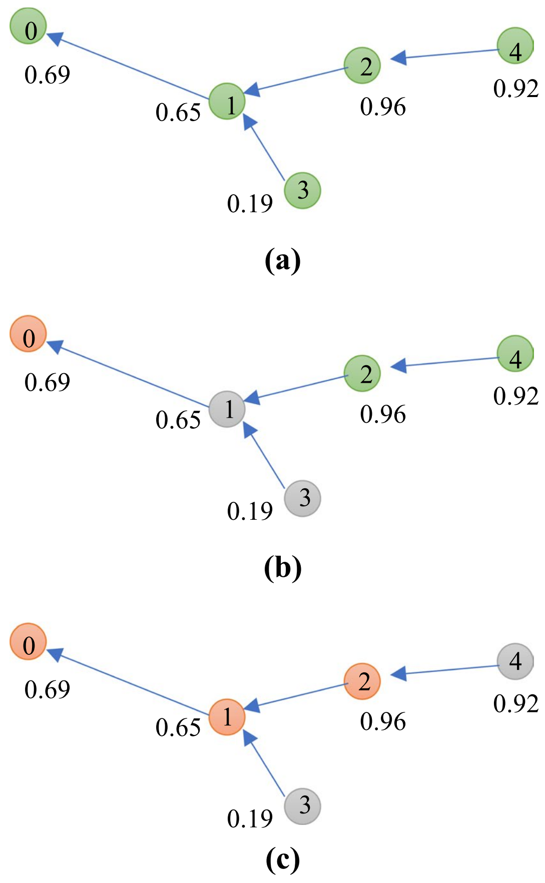

Fig. 1 An illustrative example showing a graph (**a**) before an attack and (**b**) after an attack that maximizes the damage with a fixed budget. Gray nodes are the nodes that were attacked. Red nodes are ones disabled by failure propagation. Green nodes are still functional. (**c**) shows the graph after an attack that minimizes the cost with fixed damage output

In the case of damage maximization, the attacker plans to attack the system with the constraint of a fixed budget B. The total attack cost, i.e., the sum of attack costs of all targeted nodes, cannot exceed B. The attack that achieves maximum damage measured in disabled nodes without surpassing this constraint is selected as the result.

The case where the attack budget B is excessively large is not of much interest. If the budget is larger than the cost of attacking every node in the network, the attacker can simply do just that. Realistically, the attack budget B will be lower than the cost of attacking every node in the infrastructure system. In this case, we must determine the set of nodes that maximize the damage while utilizing up to, but not exceeding, the entire attack budget.

For example, when an attacker uses explosives to damage the power grid, they do not need an intelligent algorithm if they have enough explosives to attack every entity in the system. In this case, the system operators would simply have to defend every node. Such an algorithm is only needed if the attackers have a limited supply of explosives and need to utilize their fixed resources to cause maximum damage.

3.2 Cost minimization

The other possibility is that the attacker planning to harm the system might have a motive to cause a desired degree of damage to the system. In our formulation, we refer to this desired degree of damage as *desired degradation* and denote it with D. The desired degradation D is a fraction between 0 and 1 that specifies the desired proportion of disabled nodes to the total nodes in the network after the attack has been executed. An attacker with such a motive is looking for a smart way to utilize their resources such that they minimize their spending in order to bring the system down to the desired degradation.

Consider an attacker who aims to bring down every generator in a power grid. Since there is an attack cost associated with each node in the system, for such a motive, the attacker would like to minimize the total attack cost or the sum of the costs of attacked nodes to achieve this desired degree of damage. In most cases, we expect the attacker to go for D = 1.0, i.e. disabling the whole network but in theory, D can be any number between 0 and 1 and our solution accommodates for that.

Table 2 The Objectivce Functions of Damage Maximization

	Equation	Explanation
a)	$max \sum_{u\epsilon v} 1 - \beta_u$	Maximize the number of disabled nodes for a certain B
b)	$min \sum_{u\epsilon v} w(u) \times \alpha_u$	Minimize the attack cost for a certain D

Table 3 The Objective Function Subject to these Constraints

	Equation	Case
1.1)	$\beta_u = 1 - \alpha_u$	if $deg_{in}(u) = 0$
1.2)	$\beta_u \leq 1 - \alpha_u$	if $deg_{in}(u) > 0$
2)	$\beta_u \leq y_u$	if $deg_{in}(u) > 0$
3)	$\beta_u \geq y_u - \alpha_u$	if $deg_{in}(u) > 0$
4)	$deg_{in}(u)y_u - \sum_{v\epsilon V_u^{in}} \beta_v \geq 0$	if $deg_{in}(u) > 0$
5)	$deg_{in}(u)y_u - \sum_{v\epsilon V_u^{in}} \beta_v \leq deg_{in}(u) - 1$	if $deg_{in}(u) > 0$
6.a)	$\sum_{u\epsilon V} w(u) \times \alpha_u \leq B$	if $deg_{in}(u) > 0$
6.b)	$\frac{\sum_{u\epsilon V}(1-\beta_u)}{\lvert V \rvert} \geq D$	if $deg_{in}(u) > 0$

Tables 2, 3 show the mathematical equations that comprise our implementation of the Integer Programming solution. Table 2 contains the objective functions and Table 3 contains the list of constraints.

The objective is described as follows:

a) This function is used if the goal is to maximize damage. The maximization function in this objective is the sum of the inverse of all the functionality variables β of all the nodes in the graph or the set *V*. The function evaluates the number of disabled nodes. This objective sets the goal of the formulation to maximization of the count of disabled nodes in the system once the attack has been executed.

b) This function is used if the goal is to minimize cost. The minimization function in this objective is the weighted sum of all the attack variables α_u for all nodes where the weights *w(u)* are the attack costs of the individual nodes. The expression evaluates the *total attack cost* of the attack. This objective sets the goal of the formulation to minimization of the *total attack cost* of the attack.

Table 3 describes the detailed explanation for each of the constraints.

1.1) If $deg_{in}(u) = 0$, i.e., there are no edges coming into *u*, then *u* is not dependent on any node. In this case, β_u should be exactly equal to $1-\alpha_u$.

The equation states that $\beta_u = 0$ if $\alpha_u = 1$ and $\beta_u = 1$ if $\alpha_u = 0$, meaning if node *u* is attacked it is not functional and if it is not attacked, it is functional. Since *u* is not dependent on any set of nodes, the only way to disable is to attack it directly.

1.2) If $deg_{in}(u) > 0$, i.e., the set of nodes which *u* is dependent on is not empty, then β_u is less than or equal to $1-\alpha_u$. The equation states that if $\alpha_u = 1$ then $\beta_u = 0$ but if $\alpha_u = 0$ then β_u can take up a value of either 0 or 1 which means if *u* is attacked then it is definitely disabled but if it is not attacked then it can either be functional or disabled depending on if any node that *u* is dependent upon is functional.

2) If $deg_{in}(u) > 0$, i.e., the set of nodes which *u* is dependent on is not empty, then β_u is less than or equal to y_u. The equation states that if $y_u = 0$ then $\beta_u = 0$ but if $y_u = 1$ then β_u can take up a value of either 0 or 1 which means if all the nodes that *u* is dependent on are disabled then *u* is definitely disabled but if that is not the case i.e. there is some node still functional that *u* is dependent upon, then *u* can either be functional or disabled depending on if it is directly attacked.

3) If $deg_{in}(u) > 0$ i.e. the set of nodes which *u* is dependent on is not empty, then β_u is greater than or equal to $y_u - \alpha_u$. The equation states that if $y_u = 1$ and $\alpha_u = 0$ then $\beta_u = 1$ otherwise β_u can take up a value of either 0 or 1 which means if *u* is not attacked and there is at least one functional node in the set of nodes that *u* is dependent on then *u* is definitely functional; otherwise it can either be functional or disabled depending on the values of the variables y_u and α_u. The constraint makes sure that β_u is not free.

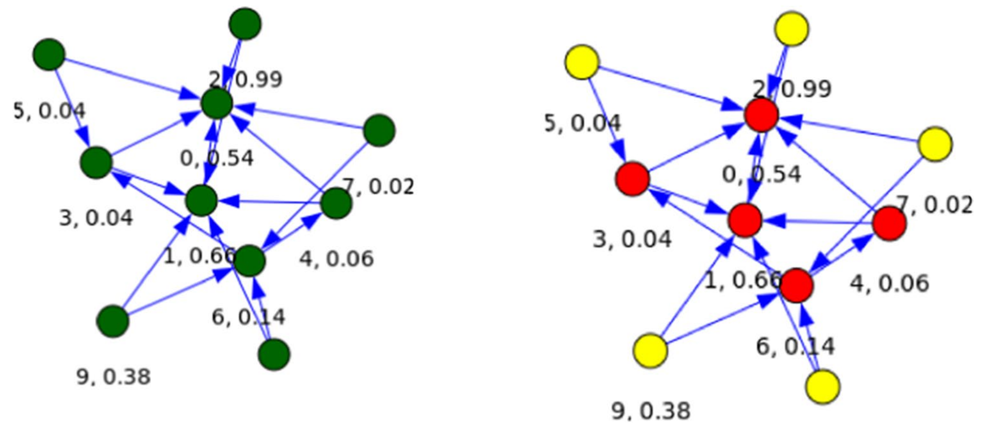

Fig. 2 Example graphs (left) and solution attack (right) for Barabási–Albert scale-free graphs using uniform distributions for weights

4–5) Since y_u is 1 if any node in V_u^{in} i.e. the set of nodes that u is dependent upon is functional and 0 otherwise, therefore the value of y_u is equal to the logical and of the functionality variables β of all the nodes in V_u^{in}.

$y_u = \bigcap_{v \epsilon V_u^{in}} \beta_v$

Constraints 4 and 5 are combined from the arithmetic equivalence of the logical and operation. A convex optimization formulation requires arithmetic definitions, not logical ones, so constraints 4 and 5 are used instead.

6. a) This function is used if the goal is to maximize damage. The left-hand side in this constraint is the weighted sum of all the attack variables α_u for all nodes where the weights *w(u)* are the attack costs of the individual nodes. The expression evaluates the total attack cost of the attack. The constraint ensures that this total attack cost does not exceed the attack budget B.
6. b) This function is used if the goal is to minimize the cost. The left side of this equation is a fraction where the numerator is the sum of the inverse of functionality variables β for all the nodes. Thus, it evaluates the number of disabled nodes. The denominator of this fraction is the cardinality of the set of all nodes V, and hence is the total number of nodes. The fraction as a whole represents the ratio of the disabled nodes to the total number of nodes in the graph. The constraint ensures that the requirement of the desired degradation is met and the ratio on the left-hand side is greater than or equal to the desired degradation.

Figure 2 illustrates some graphical examples for an optimal attack to disable a whole system. These examples are for a system of 10 nodes. The average node degree ranges from 2 to 10.

In Fig. 2, the graph on the left shows the state of the network before the attack is executed while the whole network is operational. The green color in the figure on the left represents the functional nodes in the network. The graph on the right shows the state after the attack is executed. The yellow color represents the attacked nodes and red color represents the nodes disabled due to failure propagation. The sum of the attack costs of the yellow nodes represents the total attack cost. Out of all possible attacks, the solution shows the attack with the minimum total attack cost.

Figure 3 illustrates comparatively larger graphs with the same color coding, but generated using Erdős–Rényi random graph generation model.

4 Experimental setup and results

In this section, we discuss the performance and results of our algorithm. We used Python's igraph package [16] to generate and draw graphs. We implement our formulation in GAMS, which is able to interface with many different optimization problem solvers. Currently, we use CPLEX [17] as the backend solver for our algorithm. Our GAMS implementation was run on all generated graphs to evaluate the solution of each graph.

4.1 Data generation

The graph generators take the following inputs: *number of nodes, average node degree, node weights,* and *budget*. Of these inputs, the *number of nodes* and *average node degree* are varied deterministically to generate different-sized scale-free

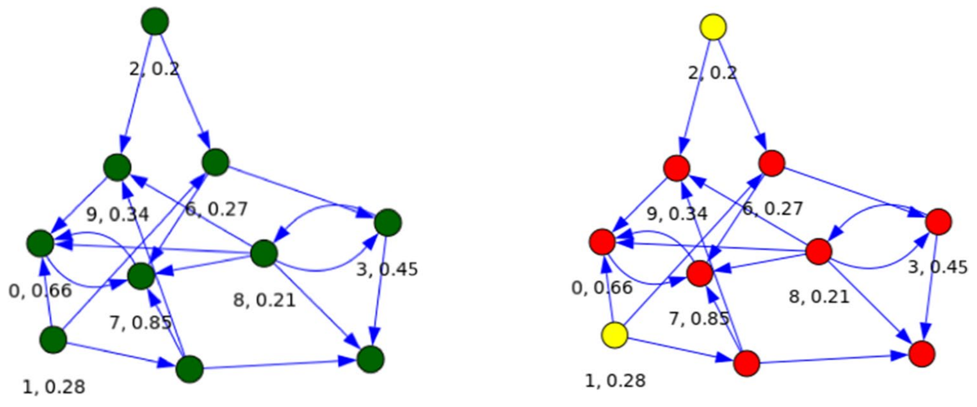

Fig. 3 Example graphs (left) and solution attack (right) for Erdős–Rényi graphs using uniform distributions for weights

random graphs using Barabási–Albert model [18]. Scale-free networks are ones in which the degree follows the power law P(k) ~ ck-γ, due to which they have highly connected hubs. Scale-free graphs can model several real-world networks that have been observed to have this property, such as social and infrastructure networks [19, 20]. Scale-free graphs were chosen along with random graphs as they can be important to analyze due to their real-world applications.

The *node weights* are randomly assigned to each node at graph generation. They are set as a number between 0 and 1. The *budget* of the attack is fixed at 1.0 throughout each experiment. Using the above parameters, we generated 25 random graphs with the number of nodes ranging from 60 to 300. For each different number of nodes, five separate graphs are generated with varying average node degree, ranging from 20 to 100% of the total number of nodes. For each test graph, the algorithm was run to determine which nodes should be attacked in order to either maximize the damage with a fixed budget or minimize cost while disabling the entire network. Table 4 lists the input parameters used for graph generation and how they were varied to generate multiple graphs for experimentation.

4.2 Experimental results

Tables 5, 6 include the results obtained from our runs on the generated graphs described above. We do not list the set of nodes attacked in each run, but only the number of nodes in that attacked set. For these experiments, the weights of nodes were randomly generated using the uniform distribution.

Tables 5, 6 display the results of experiments performed on Barabási–Albert scale free graphs. It can be noted that the number of attacked nodes decreases as the number of edges per nodes increases. This is because scale-free graphs contain highly connected hub nodes; as the number of edges increases, so do the connections to those hubs. When the graph is highly connected, attacking a single hub can propagate failure through the entire network. It can also be observed that the execution time for discovering the optimal attack increases with the number of edges in the graph.

Tables 7, 8 are the results for the Erdős–Rényi random graphs. Because they do not contain the hubs of scale-free graphs, the number of attacked nodes does not necessarily decrease as the network becomes more connected. The execution time also does not increase in the same way as it does for analyzing scale-free graphs. It can also be noted that for damage maximization, the number of attacked nodes is often smaller than for cost minimization. In many of the

Table 4 System Input Parameters

Parameters	Variation
Number of Nodes	60 to 300 with steps of 60
Average Node Degree	20% to 100% of the total number of nodes with steps of 20%
Node Weights	A uniformly random number between 0 and 1, both inclusive
Budget	Fixed at 1.0
Desired Degradation	Fixed at 1.0

Table 5 Experimental Results for Cost Minimzation based on fixed Degradation performed on Barabási–Albert scale free graphs using uniform distribution for weights

Number of Nodes	Number of edges per node	Number of attacked nodes	Execution time (s)
60	12	20	0.003
60	24	15	0.006
60	36	11	0.004
60	48	6	0.011
60	60	1	0.008
120	24	44	0.010
120	48	25	0.011
120	72	21	0.016
120	96	10	0.023
120	120	1	0.015
180	36	65	0.014
180	72	48	0.020
180	108	25	0.030
180	144	17	0.027
180	180	1	0.030
240	48	75	0.032
240	96	59	0.053
240	144	39	0.069
240	192	22	0.050
240	240	1	0.053
300	60	95	0.031
300	120	68	0.052
300	180	48	0.090
300	240	22	0.077
300	300	1	0.127

cases for damage maximization, the budget prevents the attacker from disabling the entire network and they are only able to attack enough nodes to damage part of it.

As shown in Fig. 4, as the average node degree increases for scale-free graphs, the number of nodes in an optimal attack to disable the entire network decreases. When the incoming edges to a node increase, that means the node has more supporting nodes, so more nodes will have to be disabled to propagate failure to this particular node. However, when outgoing edges in a node increase, this means that many nodes are dependent on this node. If a node with many outgoing edges is attacked, it is likely to affect many nodes. Further, these Barabási–Albert scale-free networks are generated with a linear preferential attachment [18]. In our context, this means that if a node has many dependencies, then a new node is more likely to also be dependent on this node. This results in densely connected hub nodes as we move towards the center of the graph. Thus, the decrease in attacked nodes could be due to the higher probability of new edges being added as outgoing edges rather than incoming edges.

Figures 4, 5, 6, 7, present the results of the damage maximization objective within a fixed budget. For each generated graph referenced in these figures, the attack cost of each node was randomly generated, and the budget was fixed at 1.

Because the node weights are randomly generated, the function is sometimes able to attack only a few nodes without exceeding the budget. This is why there are fluctuations ate the beginnings of the graphs. With fewer edges, the attacker may not be disabling the entire network and is using as much of the budget as possible for maximum damage. When the node degree is sufficiently high, the attacker can disable the entire network with fewer attacked nodes (1 in each of these cases) due to the scale-free property of these graphs.

In Fig. 5, the time for maximizing damage increases as the average node degree increases since this increases the total number of equations that must be solved for. Specifically, dependencies (4) and (5) increase with the number of edges. As the average edges per node increase, so do the number of equations that must be solved and therefore the execution time.

Table 6 Experimental Results for damage maximization based on a fixed budget performed on Barabási–Albert scale free graphs using uniform distribution for weights

Number of nodes	Number of edges per node	Number of attacked nodes	Execution time (s)
60	12	23	0.097
60	24	18	0.102
60	36	10	0.053
60	48	13	0.076
60	60	1	0.072
120	24	18	0.080
120	48	23	0.082
120	72	21	0.083
120	96	19	0.114
120	120	1	0.163
180	36	18	0.101
180	72	24	0.118
180	108	23	0.126
180	144	28	0.121
180	180	1	0.337
240	48	26	0.127
240	96	28	0.152
240	144	31	0.156
240	192	27	0.173
240	240	1	0.614
300	60	33	0.152
300	120	33	0.182
300	180	26	0.216
300	240	29	0.245
300	300	1	0.867

Figure 7 shows the execution time of the damage maximization function on Erdős–Rényi graphs. It can be noted that the values are more random than the scale-free graphs and that the execution time is not necessarily increasing for higher average node degree.

Figures 8, 9, 10, 11 present the results of the objective of minimizing the cost of an attack that causes a desired amount of damage. For each generated graph referenced in these figures, the attack cost of each node was randomly generated, and the degradation was fixed at 1.

As shown in Fig. 8, the number of nodes needed to be attacked to disable the entire network decreases linearly with an increase in the average node degree. When incoming edges in a node increase, it means the node now has more supporting nodes, that is, more nodes have to be disabled to propagate failure to this node. In contrast, when outgoing edges in a node increase, this means that more nodes are now dependent on this node and if this node goes down, it is likely to affect more nodes than before. Further, the graph generation model that we are using, the Barabási model, generates scale-free networks with a linear preferential attachment [18]. In our context, it means that if a node has a lot of dependencies, a new node is more likely to be dependent on this node. As a result, there are more densely connected hubs as we move towards the center of the graph. So, the reason of this linear decrease could be the higher probability of new edges being added as an outgoing edge as compared to an incoming edge.

Figure 9 indicates that as the graph gets denser, the execution time increases. This increase is the result of the increase in the number of equations with an increase in the number of edges. As the number of edges increases, the dependencies between the nodes increase and hence the equations capturing these dependencies (Eq. 4–5) increase.

As Fig. 10 illustrates, the attacked nodes do not follow a regular pattern. This is because Erdős–Rényi graphs are not scale-free like Barabási–Albert. So, a higher number of nodes does not necessarily mean more dependency on a single node. Figure 11 shows the execution time plots for Erdős–Rényi graphs.

Table 7 Experimental Results for cost minimization based on fixed degradation performed on Erdős–Rényi graphs using uniform distribution for weights

Number of nodes	Number of edges per node	Number of attacked nodes	Execution time (s)
60	12	10	0.003
60	24	6	0.003
60	36	5	0.002
60	48	12	0.004
60	60	2	0.004
120	24	10	0.007
120	48	18	0.005
120	72	11	0.006
120	96	13	0.006
120	120	20	0.006
180	36	24	0.008
180	72	25	0.004
180	108	29	0.008
180	144	25	0.003
180	180	29	0.009
240	48	30	0.005
240	96	30	0.006
240	144	32	0.013
240	192	27	0.010
240	240	21	0.005
300	60	41	0.015
300	120	41	0.010
300	180	40	0.006
300	240	44	0.006
300	300	52	0.012

5 Conclusions and future work

This paper addressed the problem of identifying critical nodes in an IoT network that make up an optimal cost attack on that network. It provided an effective and minimalistic integer linear programming solution to the problem. The experimental results show that our scheme is fast and scalable for large-scale infrastructure systems. It achieves one of two problem objectives: maximizing damage caused to the network within a fixed budget or minimizing the cost of an attack that causes a desired amount of damage. Both methods are tested on random graphs and scale-free graphs. There are several extensions of the proposed work. Currently, the proposed model is minimalistic and as simple as possible to ensure efficient solutions. The goal was to propose an efficient scheme that is easy to extend for specific use cases and accommodates for the needs of specific infrastructures. One possibility is to include the demand and supply parameters for each node. In our formulation, we made the assumption that a single node is sufficient to keep its dependent nodes alive. But, in the real world, not every node is able to keep its dependent nodes running all by itself. In fact, it depends on the demand of the dependent nodes and the supply that this node has. Having a demand and a supply parameter for each node solves that problem. This way we can model this real-world limitation in our formulation making a more accurate representation of the real-world IoT systems.

Additional work on including various aspects of the network requirements (e.g., specific security requirements, performance requirements, usability requirements, and cost calculation) will be worth exploring as it will add more real-world factors.

Table 8 Experimental Results for damage maximization based on a fixed budget performed on Erdős–Rényi graphs using uniform distribution for weights

Number of nodes	Number of edges per node	Number of attacked nodes	Execution time (s)
60	12	7	0.097
60	24	7	0.032
60	36	9	0.071
60	48	6	0.025
60	60	9	0.033
120	24	6	0.061
120	48	9	0.062
120	72	9	0.051
120	96	13	0.133
120	120	11	0.068
180	36	9	0.087
180	72	17	0.114
180	108	12	0.095
180	144	9	0.114
180	180	21	0.098
240	48	15	0.102
240	96	21	0.113
240	144	17	0.115
240	192	17	0.134
240	240	17	0.123
300	60	17	0.158
300	120	14	0.171
300	180	19	0.137
300	240	24	0.141
300	300	18	0.172

Another extension is to incorporate the heterogeneity of the dependencies that each node has. A real-world entity usually does not run on a single kind of resources. Instead, there are several different resources that a single entity in the system requires in order to perform its operations. In our current formulation we do not consider that, but we can model each node to have a demand for different kinds of resources. So, adding this extension allows us to capture this real-world behavior of these infrastructure systems.

The current work assumes that the channels in the system have infinite flow capacity but that is not really accurate. But we can add a capacity parameter for each edge to put a limit on its capacity and to make the model more practical and closer to reality. The current formulation, moreover, puts all nodes on equal footing in terms of importance to the system, that is, no node is more important than its peers other than the fact that it might have more dependent nodes to provide. But in the real world, some power grids, for example, may be supplying power to more sensitive or critical systems as compared to others which makes them more crucial to the system. Therefore, the system should allocate more resources to defend the nodes of more importance. We can simulate this behavior of these systems by incorporating an importance parameter.

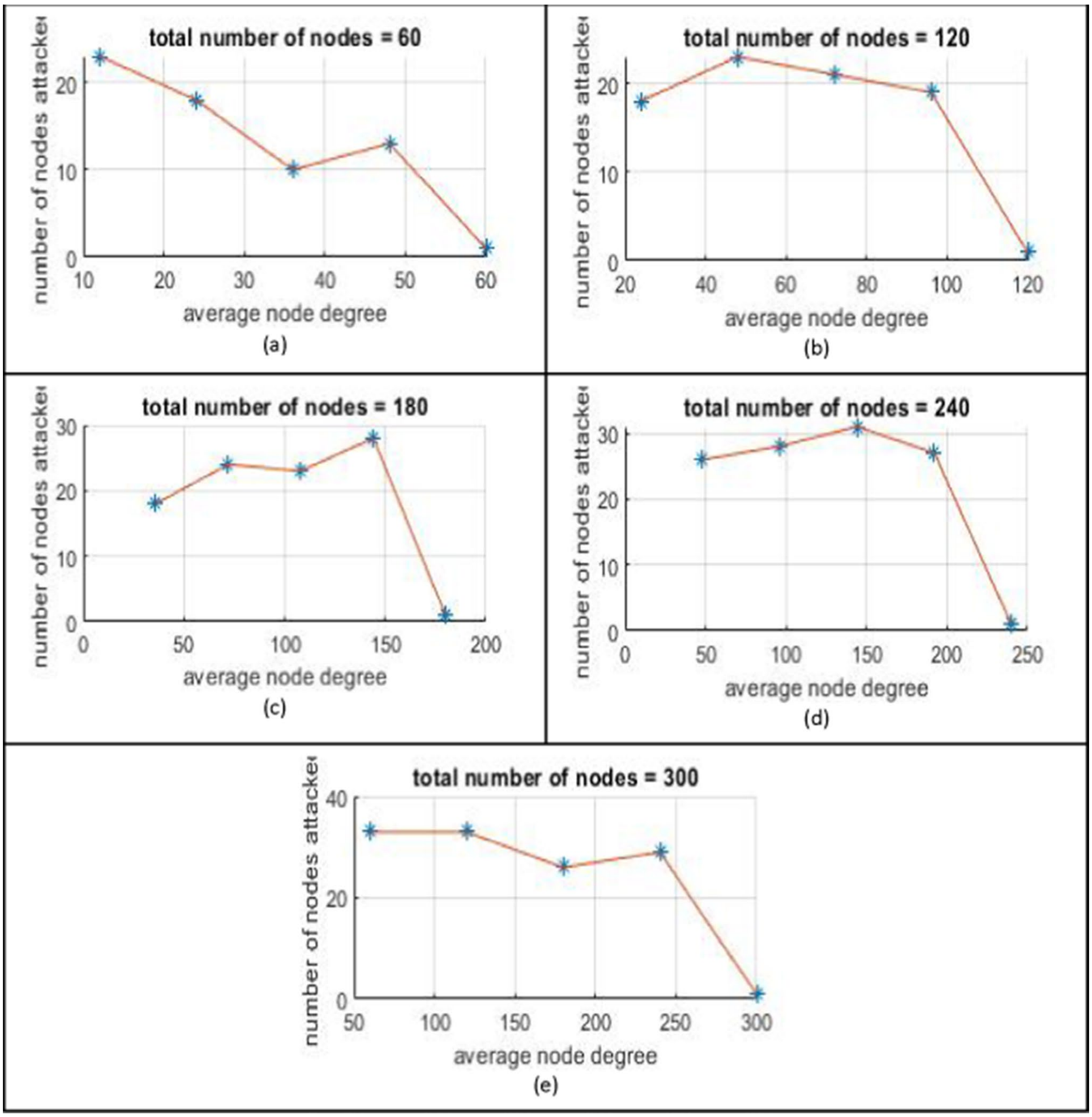

Fig. 4 Change in the number of nodes attacked w.r.t. average node degree for Barabási–Albert scale free graphs using uniform distribution for weights. Solved for maximum damage based on a fixed budget

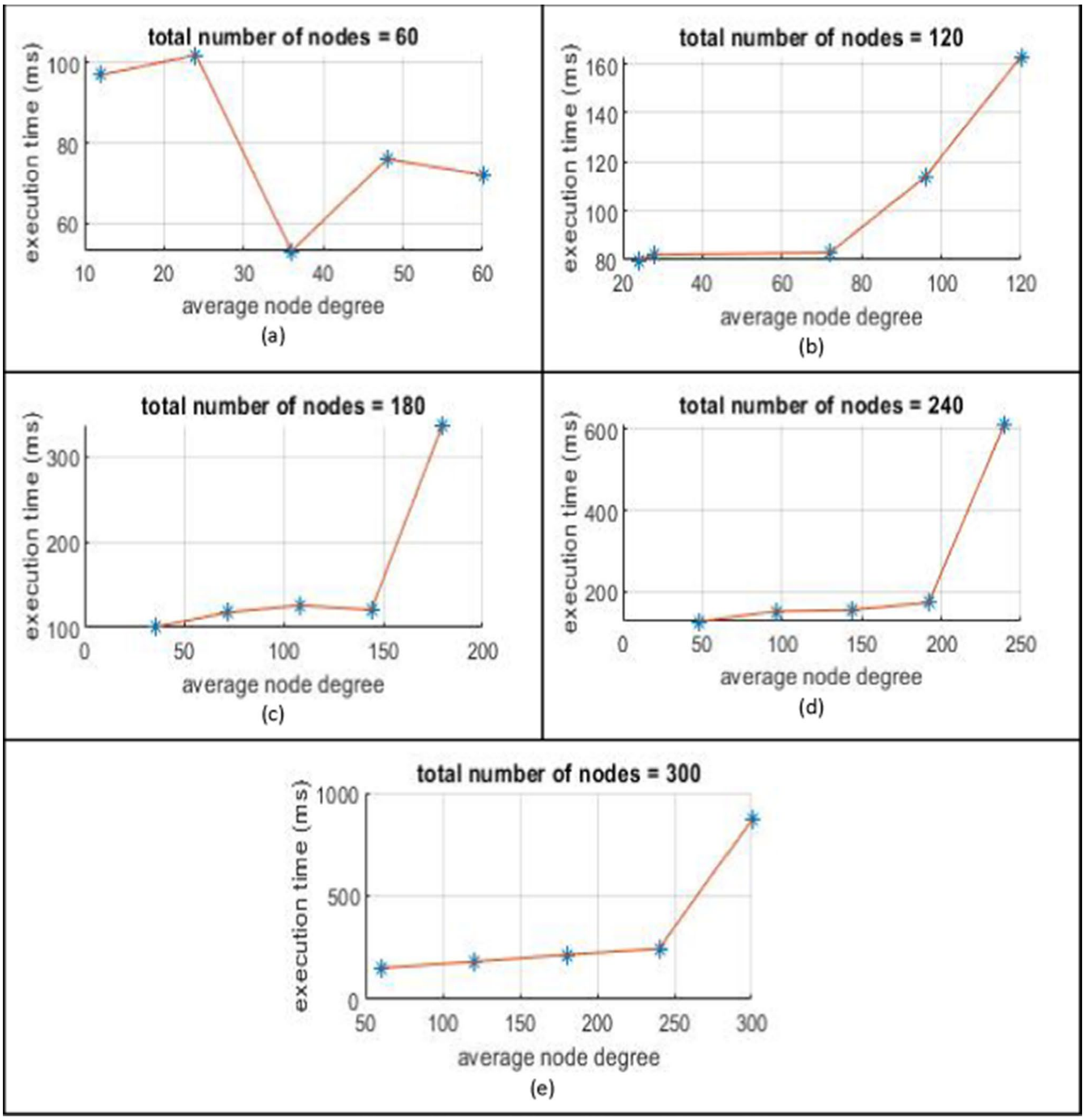

Fig. 5 Change in the execution time w.r.t. average node degree for Barabási–Albert scale free graphs using uniform distribution for weights. Solved for maximum damage based on a fixed budget

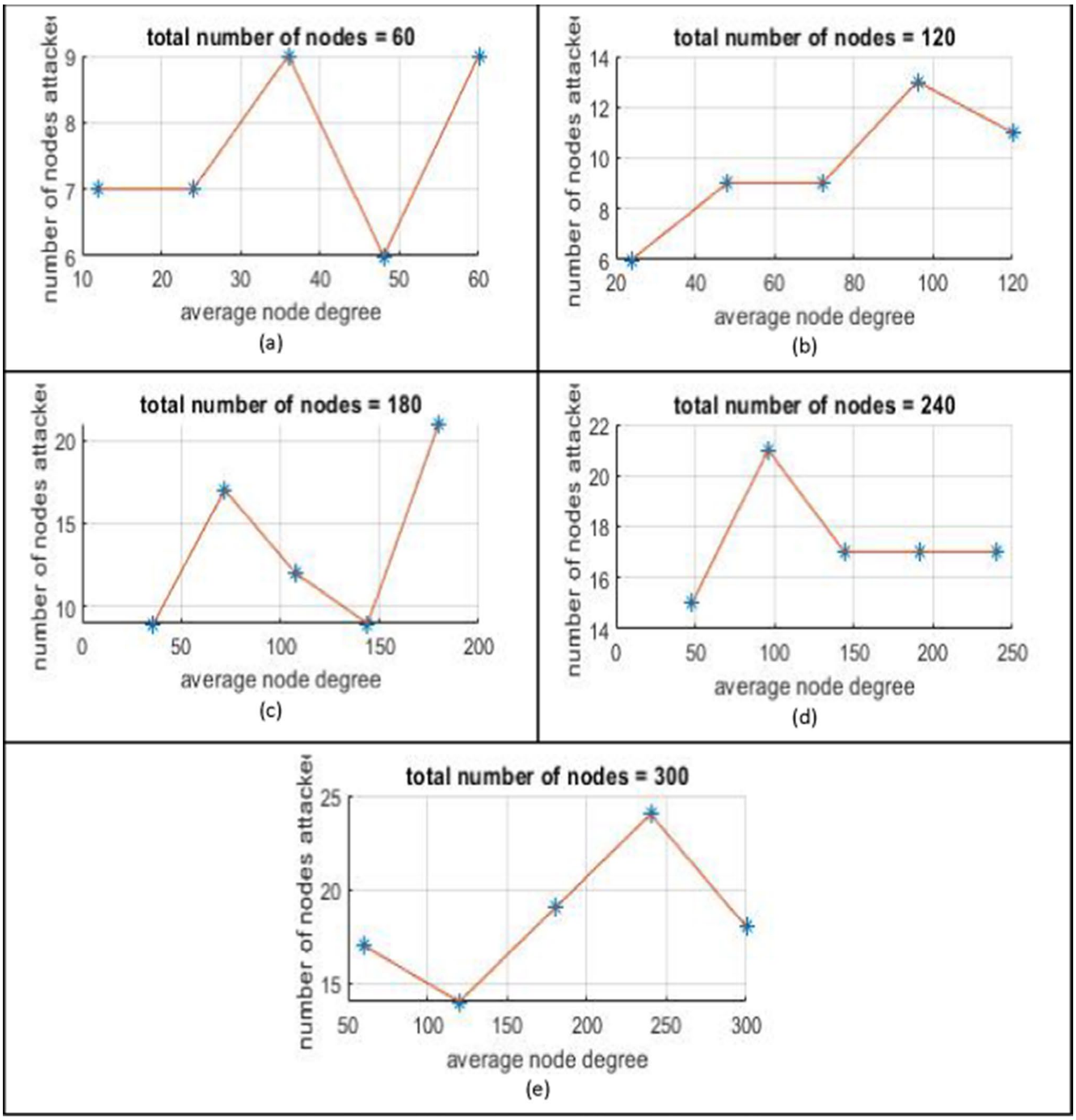

Fig. 6 Change in the number of nodes attacked w.r.t. average node degree for Erdős–Rényi graphs using uniform distribution for weights. Solved for maximum damage based on a fixed budget

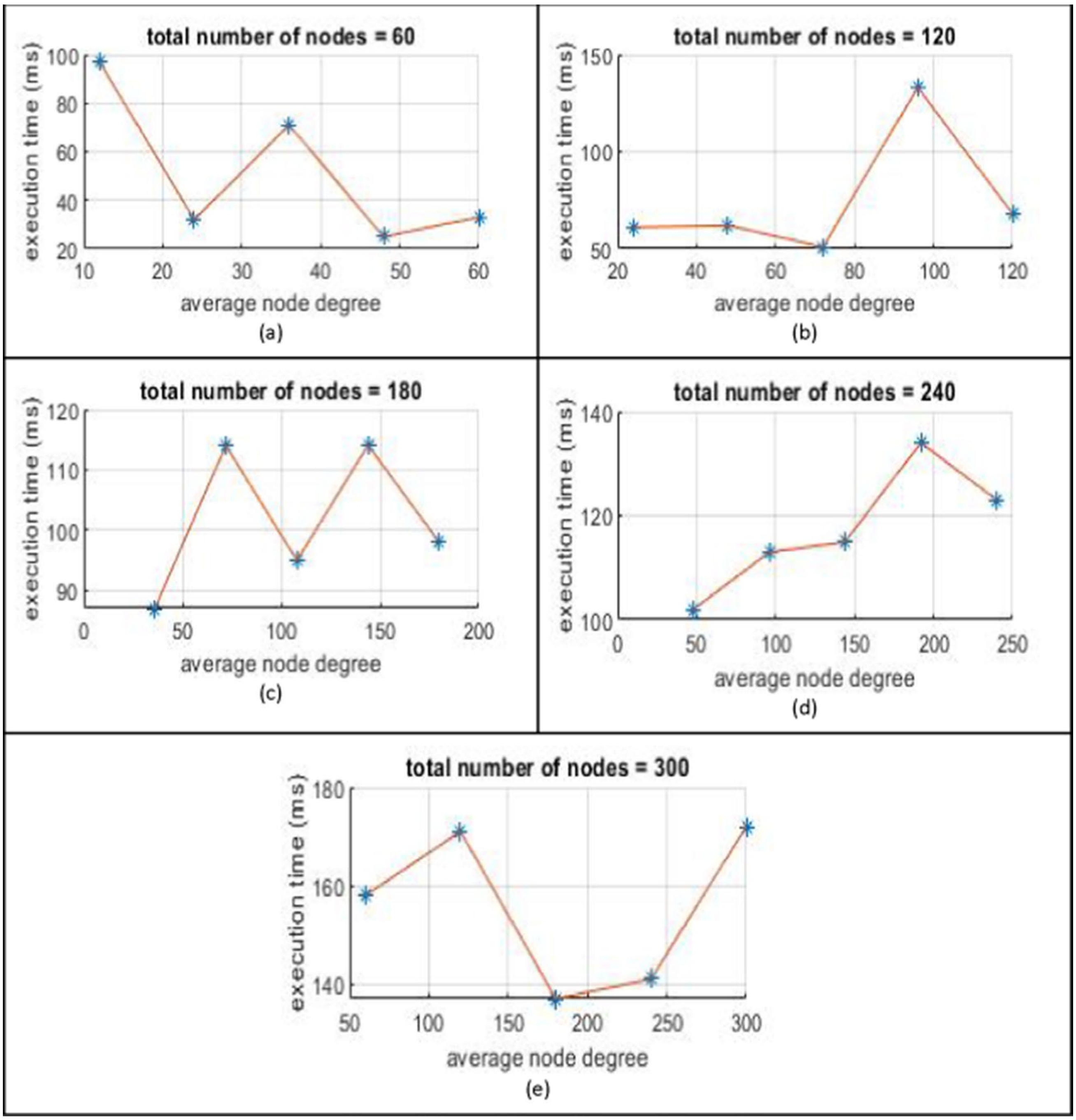

Fig. 7 Change in the execution time w.r.t. average node degree for Erdős–Rényi graphs using uniform distribution for weights. Solved for maximum damage based on a fixed budget

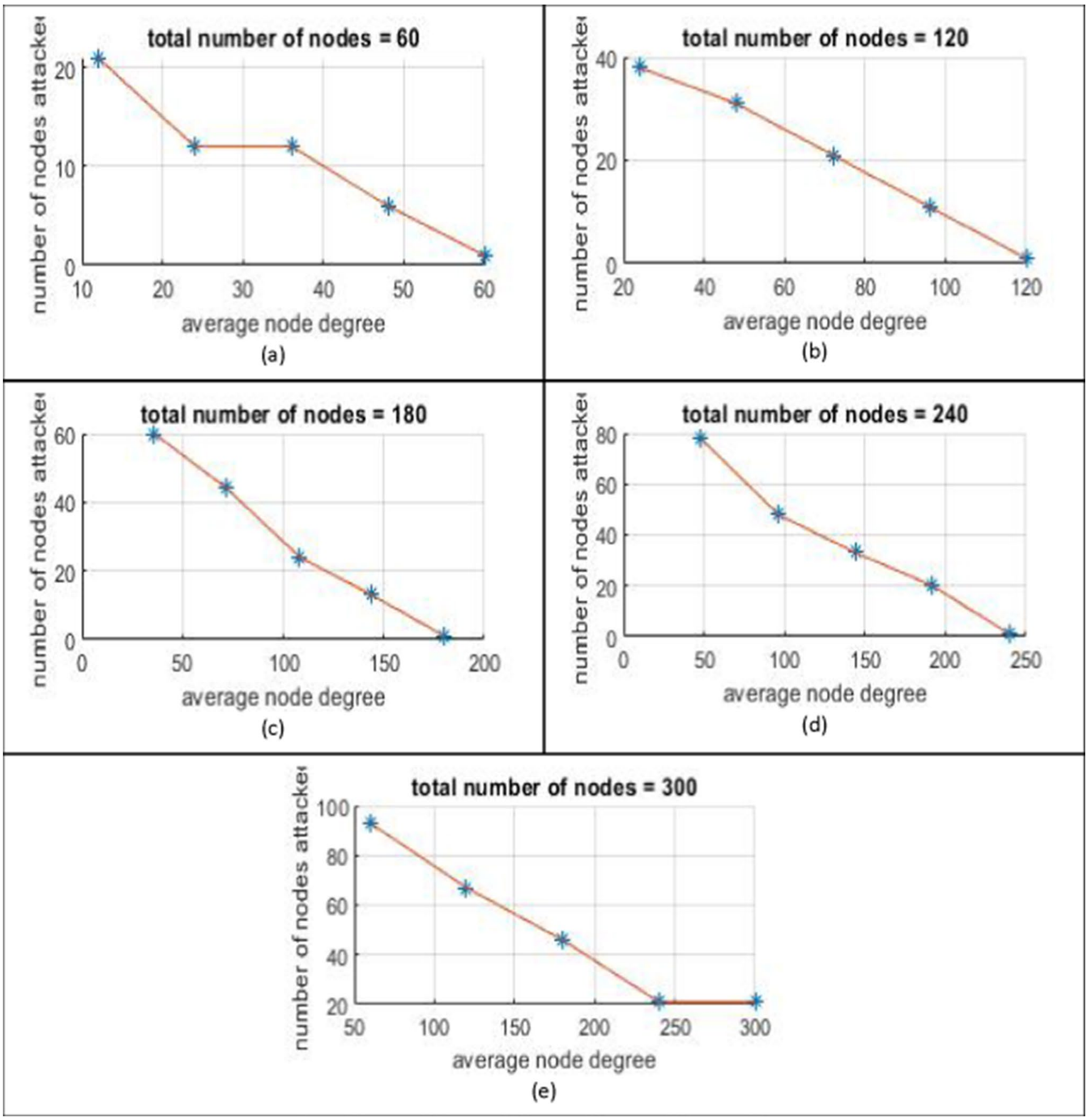

Fig. 8 Change in the number of nodes attacked w.r.t. average node degree for Barabási–Albert scale free graphs using uniform distribution for weights. Solved for minimum cost based on a fixed degradation

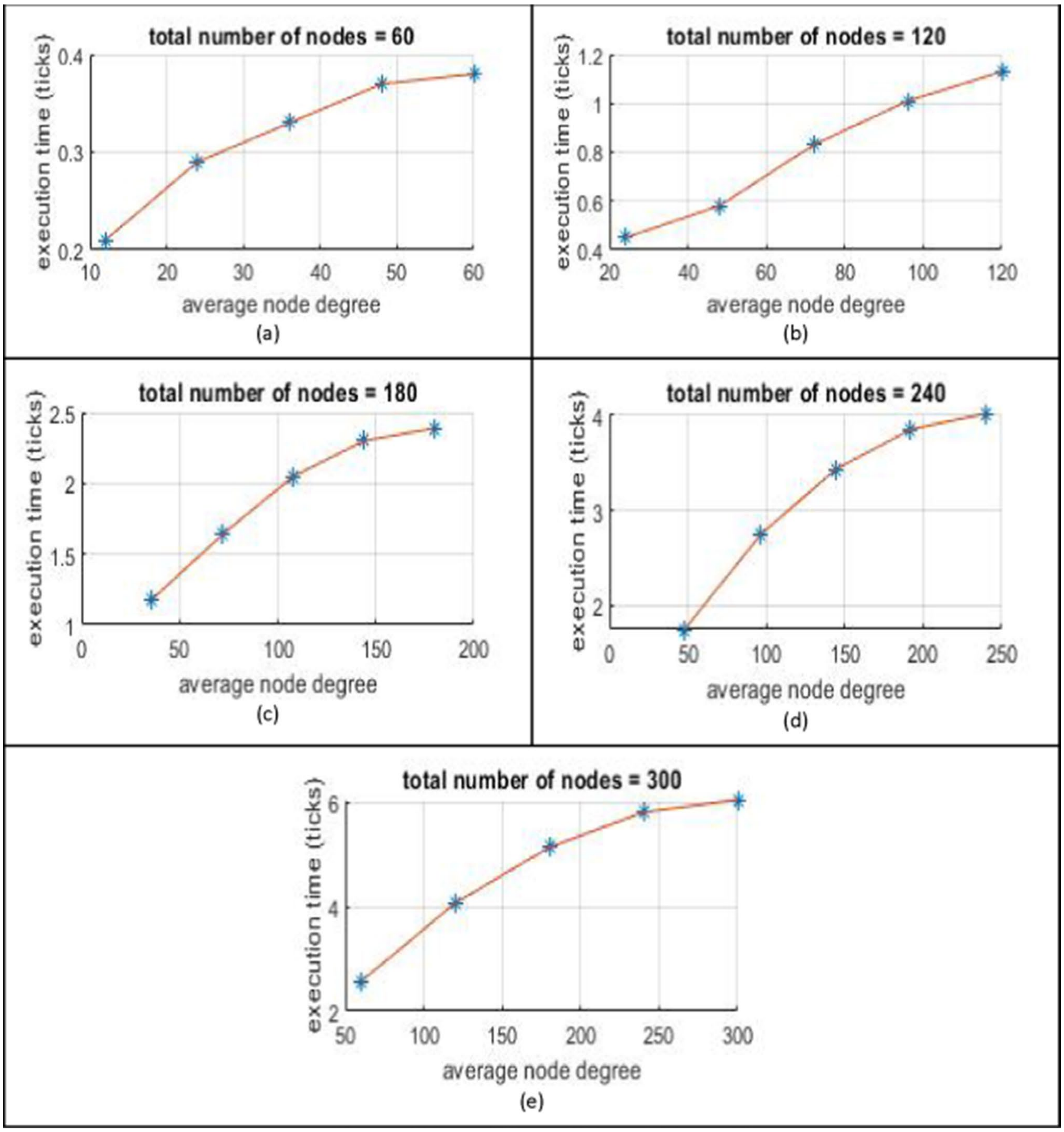

Fig. 9 Change in the execution time w.r.t. average node degree for Barabási–Albert scale free graphs using uniform distribution for weights. Solved for minimum cost based on a fixed degradation

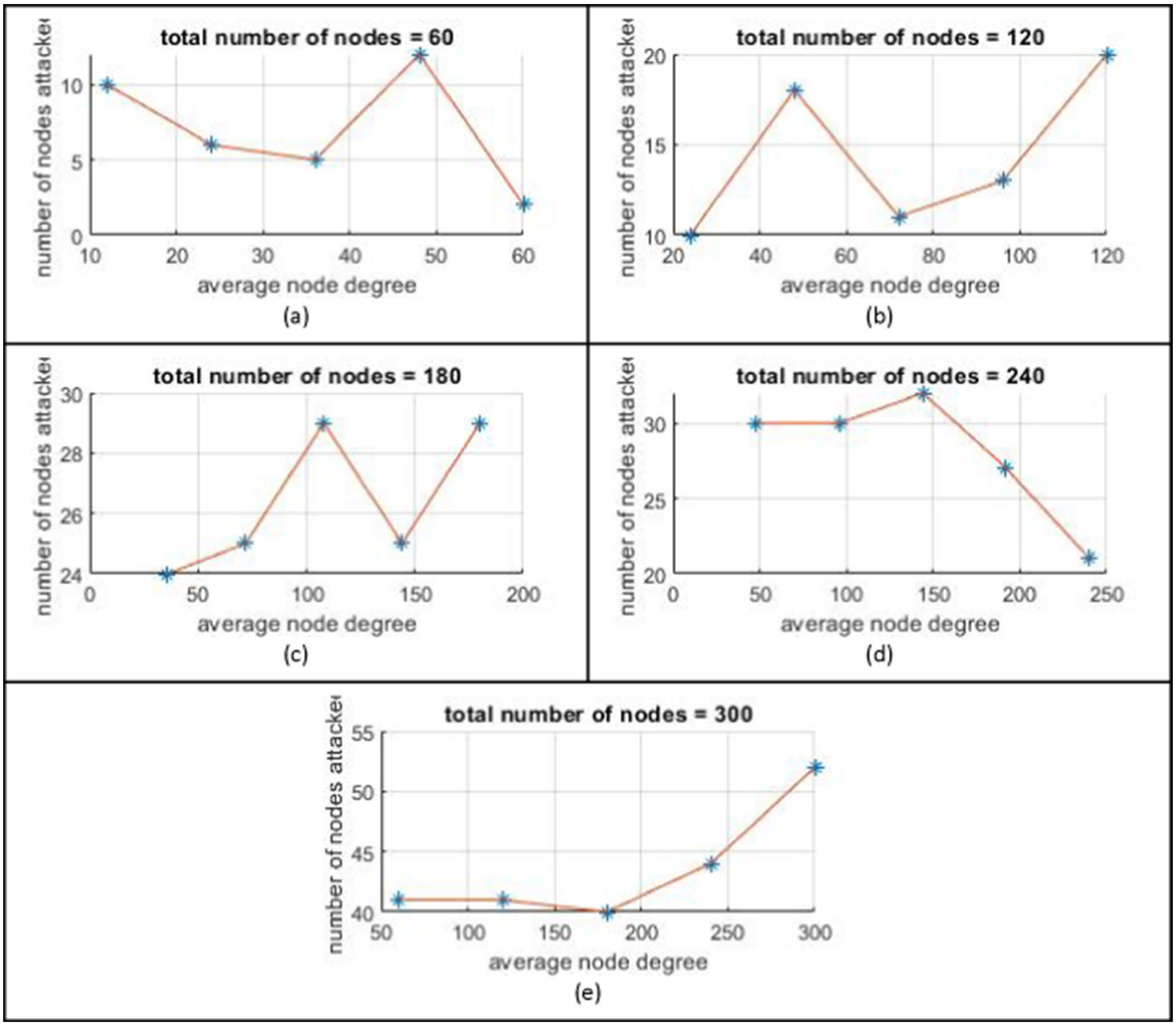

Fig. 10 Change in the number of nodes attacked w.r.t. average node degree for Erdős–Rényi graphs using uniform distribution for weights. Solved for minimum cost based on a fixed degradation

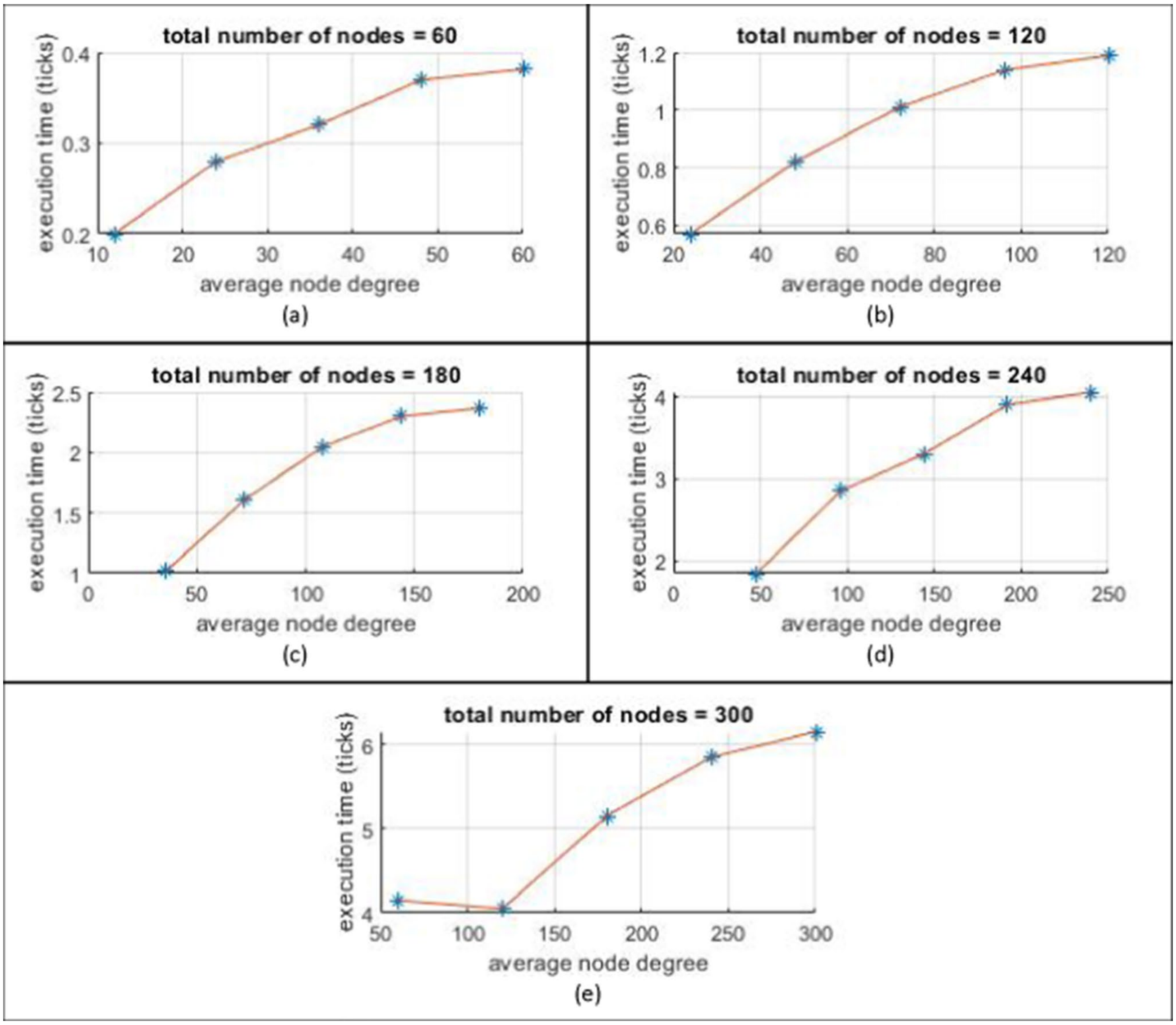

Fig. 11 Change in the execution time w.r.t. average node degree for Erdős–Rényi graphs using uniform distribution for weights. Solved for minimum cost based on a fixed degradation

Acknowledgements This work was supported by Air Force Research Labs' fellowship to Ahmad and Clark.

Author contributions All authors contributed equally and to every section

Funding Air Force Research Laboratory, SA2018-UP-025,SA2018-UP-025

Declarations

Competing interests The authors declare no competing interests.

References

1. Rosato V, Issacharoff L, Tiriticco F, Meloni S, Porcellinis S, Setola R. Modelling interdependent infrastructures using interacting dynamical models. Int J Crit Infrastruct. 2008;4(12):63–79.
2. Romero J. Lack of rain a leading cause of indian grid collapse. IEEE spectrum: technology, engineering, and science news. 2012. https://spectrum.ieee.org/energywise/energy/the-smarter-grid/disappointing-monsoon-season-wreaks-havoc-with-indias-grid.
3. Naik N, Jenkins P, Grace P, Song J. Comparing attack models for IT systems: lockheed martin's cyber kill chain, MITRE ATT&CK framework and diamond model, 2022 IEEE International Symposium on Systems Engineering (ISSE), Vienna, Austria, 2022, pp. 1–7
4. Alderson DL, Brown GG, Carlyle WM. Operational models of infrastructure resilience. Risk Anal. 2015;35(4):562–86.
5. A dictionary of physics, 6th ed. Oxford [Angleterre]: Oxford University Press, 2009.
6. Sturaro S, Silvestri M, Conti Das S. Towards a realistic model for failure propagation in interdependent networks. In: International conference on computing, networking, and Communications (ICNC); 2016. pp. 1–7.
7. Buldyrev S, Parshani R, Paul G, Stanley H, Havlin S. Catastrophic cascade of failures in interdependent networks. Nature. 2010;464(7291):1025–8.
8. Buldyrev SV, Shere NW, Cwilich GA. Interdependent networks with identical degrees of mutually dependent nodes. Phys Rev E. 2011;83(1):016112.
9. Huang Z, Wang C et al. Balancing system survivability and cost of smart grid via modeling cascading failures. In: IEEE transactions on emerging topics in computing. 2013; 1(1).
10. Huang Z,Wang C et al. Small cluster in cyber physical systems: network topology, interdependence and cascading failures. In: IEEE transactions on parallel and distributed systems. 2015; 26(8).
11. Sturaro A, Silvestri S, Conti M, Das SK. A realistic model for failure propagation in interdependent cyber-physical systems. In IEEE Transactions on Network Science and Engineering, 2020; 7(2): pp. 817–831. Doi: https://doi.org/10.1109/TNSE.2018.2872034.
12. Alderson DL, Brown GG, Carlyle WM, Wood RK. Solving defender-attacker-defender models for infrastructure defense. In: Naval postgraduate school, department of operations research, 2011.
13. Smolyak A, Levy O, Vodenska I, et al. Mitigation of cascading failures in complex networks. Sci Rep. 2020;10:16124. https://doi.org/10.1038/s41598-020-72771-4.
14. Nakarmi U, Rahnamay-Naeini M, Khamfroush H. Critical component analysis in cascading failures for power grids using community structures in interaction graphs. IEEE Transact Netw Sci Eng. 2020;7(3):1079–93. https://doi.org/10.1109/TNSE.2019.2904008.
15. Hazra K et al. A novel network architecture for resource-constrained post-disaster environments. In: International conference on communication systems & networks (COMSNETS), Bengaluru, India; 2019. pp. 328–335. https://doi.org/10.1109/COMSNETS.2019.8711166.
16. Csardi G, Nepusz T. The igraph software package for complex network research. Int J Complex Syst. 2006;1695(5):1–9.
17. User's Manual for CPLEX. International business machines corporation, 2009, p. 157.
18. Barabási A, Albert R. Emergence of scaling in random networks. Science. 1999;286(5439):509–12.
19. Venkateswaran V, Aved A, Ferris D, Siviy N. Critical node analysis on interconnected networks under cascading failures. 2017.
20. Erdös P, Rényi A. On the evolution of random graphs, the structure and dynamics of networks. Princeton: Princeton University Press; 2006. p. 38–82.
21. Arulselvan A, Commander CW, Elefteriadou L, Pardalos PM. Detecting critical nodes in sparse graphs. Comput Oper Res. 2009;36:2193–200.
22. Bichi BY, Islam SU, Kademi AM, Ahmad I. An energy-aware application module for the fog-based internet of military things, accepted for publication in discover internet-of-things. 2022.
23. British airways IT failure caused by 'Uncontrolled return of power', the guardian, 2017. https://www.theguardian.com/business/2017/may/31/ba-it-shutdown-caused-by-uncontrolled-return-of-power-after-outage.
24. Bienstock D, Verma A. The N-k problem in power grids: new models, formulations, and numerical experiments. SIAM J Optim. 2010;20(5):2352–80.
25. Nguyen DT, Shen Y, Thai MT. Detecting critical nodes in interdependent power networks for vulnerability assessment. In: IEEE transactions on smart Grid. 2013;4(1).
26. Ahmad I. Discover internet-of-things editorial, inaugural issue. Discov Internet Things 2021. https://doi.org/10.1007/s43926-021-00007-6.
27. Salmeron J, Wood K, Baldick R. Analysis of electric grid security under terrorist threat. IEEE Trans Power Syst 2004;19(2):905–12. https://doi.org/10.1109/TPWRS.2004.825888.
28. Golari M. Integer programming formulations for minimum spanning forest problems, presentation, systems and industrial engineering, University of Arizona. 2015. http://math.arizona.edu/glickenstein/math443f14/golari.pdf.
29. Albert R, Jeong H, Barabasi A-L. Error and attack tolerance of complex networks. Nature. 2000;406(6794):378–82.
30. Albert R, Albert I, Nakarado GL. Structural vulnerability of the north american power grid. Phys Rev E. 2004;69(2):025103.
31. Venkateswaran V, Bennette W. Critical node analysis (CNA) of electrical infrastructure networks, in machine intelligence and bio-inspired computation: theory and applications X. In: Misty B, Jonathan W, Hall RD, editors. Proceedings of SPIE Vol. 9850, SPIE, Bellingham, WA, 2016. https://doi.org/10.1117/12.2223499.
32. Maksimovic Z. A new mixed integer linear programming formulation for the maximum degree bounded connected subgraph problem. Publications De L'Institut, Mathmatique. Nouv Ser. 2016;99(113):99–108.