Identification of Nodes and Networks: Robustness, Immunization, and Explosive Synchronization

vorgelegt von
M.Eng.
Yang Liu

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
-Dr.-Ing.-

genehmigte Dissertation

Tag der wissenschaftlichen Aussprache: 18. March 2021

Berlin 2021

Zusammenfassung

In zahlreichen Studien hat es sich erwiesen, dass komplexe Systeme durch Netzwerke charakterisiert werden können. Diverse Probleme wie Kaskadierungsfehler, Verbreitungsdynamik und Datenverarbeitung können durch Methoden der Netzwerkwissenschaft erforscht bzw. verbessert werden. Der wesentliche Aspekt von solchen netzwerkwissenschaftlichen Studien ist die Identifikation der Kanten, der Knoten oder des gesamten Netzwerks. Deshalb erforschen wir das Problem in dieser Arbeit von zwei Aspekten aus – der eine ist die Identifikation der Knoten, der andere die Identifikation des Netzwerks.

Das Ziel dieser Arbeit in Bezug auf den ersten Aspekt ist der Vorschlag fortgeschrittener Ansätze zur Fragmentierung eines bestehenden Netzwerks. In Vergleich zu anderen Methoden, geschieht dies mit weniger zu entfernenden Knoten. Als Resultat können wir ein sichereres Netzwerk erreichen, in dem z.B. Epidemien besser eingedämmt werden können, während dieselben Ressourcen (die gleiche Anzahl von Knoten) verwendet werden. Davon profitieren Anwendungen wie die Verteilung von Impfstoffen, die Entscheidung, welche Personengruppe unter Quarantäne gestellt werden soll, die Eindämmung von Fehlinformationen in sozialen Netzwerken oder das Erkennen der Ausfallsicherheit eines netzwerkbasierten Systems unter einem gezielten Angriff. Um dies zu erreichen, integriert diese Arbeit Regeln aus der explosiven Perkolation mit Strategien aus der Graphen-Partitionierung und Ideen aus evolutionären Algorithmen. Für über 20 empirische Netzwerke sind unsere entwickelten Ansätze im Vergleich zum Stand der Technik wesentlich effektiver bei der Erfassung der Schlüsselgruppe von Knoten, die für die Fragmentierung verantwortlich sind.

In Bezug auf die Netzwerkidentifikation stellen wir uns die Frage, ob der sogenannte Ordnungsparameter – hier ein Maß der Robustheit eines Netzwerks – eine Netzwerkstruktur charakterisieren kann und untersuchen mögliche Wege die Robustheit eines bestimmten Netzwerks zu verbessern oder zu schwächen. Insbesondere haben wir beim Phänomen der explosiven Synchronisation festgestellt, dass der Ordnungsparameter in der Tat eine zentrale Rolle beim Charakterisieren der Netzwerkstruktur spielt. Für den Einfluss auf die Robustheit eines Netzwerks konnten jedoch nur indirekte Strategien gefunden werden. In dieser Arbeit wird daher die Vorhersagbarkeit und Steuerbarkeit der Netzwerkrobustheit mithilfe maschineller Lernwerkzeuge für den Datensatz aus der explosiven Synchronisation weiter erörtert.

Die Hauptbeiträge und neue Methodik dieser Arbeit lassen sich daher wie folgt zusammenfassen: die Methodik bestehend aus i) begrenzten und unbegrenzten Strategien und ii) Ausarbeitung eines evolutionären Frameworks für die Untersuchung einflussreicher Knoten; und mögliche Anwendungen umfassen iii) Identifikation der Auswirkungen der

Netzwerkrobustheit auf die explosive Synchronisation, iv) Entwicklung von Möglichkeiten zur Verbesserung der Robustheit eines Netzwerks, v) Einflüsse von Bekannten auf die Eindämmung von Epidemien und vi) Vorhersage der Hysterese bei der explosiven Synchronisation.

Abstract

Complex systems in a broad range of scientific domains have been shown to be well-characterized by networks in an increasing number of studies. Problems such as cascading failures, spreading dynamics and the extraction of leading factors from raw data through the construction of networks can all be studied within the paradigm of network science. Such problems concerning networks are usually directly or indirectly associated with the identification of edges, nodes, and sometimes the entire network. Hence, it is mainly from the two levels of network structure: nodes and networks, that we aim to study the problem of network robustness (or immunization in the context of epidemics).

On the local level of nodes, the goal of this thesis is to propose advanced approaches to fragment an existing network. Through such approaches we can achieve network fragmentation with an attack on fewer nodes than exiting methods, or alternatively, we can achieve a safer network which is more capable of containing epidemics while using the same resources (i.e., same amount of nodes). These approaches could also be applied to diverse problems such as to the distribution of a vaccine, to the decision over which group of individuals should be quarantined first, to the containment of misinformation in social networks, or to access the resilience of a network-based system under an intentional attack. To achieve that, this thesis integrates rules from explosive percolation, strategies from graph partition, and ideas from evolutionary computing. As a result, the developed approaches are much more effective at acquiring the key group of nodes responsible for the fragmentation when compared to the state-of-the-art methods.

On the level of entire networks, we attempt to ascertain whether the order parameter – here a measure of network robustness – could be used to capture the network structure, and further study potential ways that could be used to enhance or weaken the robustness of a given network regarding such a parameter. In particular, we show that for the known phenomenon of explosive synchronization, the order parameter indeed plays an important role in capturing the network structure. In regard to the network robustness, however, only indirect strategies could be found. Hence, this thesis further discusses the predictability and controllability of the network robustness by the aid of machine learning methods on the dataset from the explosive synchronization.

The main contributions of this thesis can therefore be summarized as follows: the methodology consisting of i) bounded and unbounded strategies and ii) evolutionary framework for the identification of influential nodes; and potential application include iii) effects of network robustness on explosive synchronization, iv) ways to enhance the robustness of a network, v) influences of acquaintances on the containment of epidemics, and vi) prediction of the hysteresis in explosive synchronization.

To Xi

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my exceptional supervisor at Potsdam Institute for Climate Impact Research (PIK), Prof. Dr. Jürgen Kurths, for his continuous support, help, and encouragement throughout my whole postgraduate study. I particularly appreciate his patient guidance, understanding, and warm care when I encountered difficulties in both research and life. Without his endless help, this thesis would not reach its present form. I would also like to express my sincere thanks to Prof. Dr. Klaus-Robert Müller for giving me the invaluable opportunity to pursue my PhD at TU Berlin and all the freedom to conduct my research at PIK.

Special thanks also go to my friends and colleagues at PIK, particularly to Paul Schultz, Jingfang Fan, Frank Hellmann, Xiujing Han, and Yong Zhao for inspiring and fruitful discussions. Meanwhile, I am also grateful to all my friends for filling my life with countless joys in Potsdam.

Most importantly, I would like to dedicate my greatest appreciation to my parents, my elder sister, my girlfriend, and all of my family members. Their unparalleled love, kind understanding, and endless support help me conquer all the challenges and difficulties in my PhD study.

Lastly, I would like to thank China Scholarship Council (CSC) scholarship and IRTG 1740 for generously supporting my work financially.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

# the number of 9

AcqI Acquaintance immunization 27

AP Achlioptas process 19

BA Barabási-Albert 28

BF Bohman and Frieze 21

ER Erdős and Rényi 18

ES Explosive synchronization 85

FVS Feedback vertex set 37

IIs Independent implementations 49

LCC Largest connected component (cluster) 19

ProR Product rule 21

RanS Random strategy 27

SF Scale-free 28

SIR Susceptible-Infected-Recovered 3

SumR Sum rule 21

Symbols

$\mathcal{M}$ The edges set $\mathcal{M}$. 7

$\mathcal{N}$ The node set $\mathcal{N}$. 7

m The number of edges. 7

n The number of nodes. 7

$G(\mathcal{N}, \mathcal{M})$ A network (graph) consists of all nodes in the node set $\mathcal{N}$ and all edges in the corresponding edge set $\mathcal{M}$. 7

$G(n, m)$ A network (graph) consists of n nodes and m edges. 7

e_{ij} A specific edge connecting nodes i and j. 8

A The adjacency matrix. 8

$\Gamma(i)$ The nearest neighbor set of node i. 9

k_i The degree of node i. 9

$\langle k \rangle$ The average (mean) degree. 9

p_k The degree distribution. 9

d_{ij} The length of the shortest path between node i and node j. 10

$d_{\mathbf{max}}$ The diameter of a network. 10

$\langle d \rangle$ The average shortest path of a network. 10

$d_c(i)$ The length of a cycle regarding node i. 11

CC_i The clustering coefficient of node i. 11

$\langle CC \rangle$ The average clustering coefficient. 11

λ_1 The largest eigenvalue. A2

$\mathcal{H}_i$ The score (influence) of node i. This term would be reused in several places. 12

$\mathbf{v}_i$ The i^{th} eigenvector. A2

λ_i The i^{th} eigenvalue. A2

α A temporal variable to help explain concepts or methods. This term would be reused in several places. 13

r The assortativity coefficient. 16

p The occupied probability regarding site (node) or bond (edge) percolation. 16

p_∞ The order parameter. 17

$\mathcal{M}_c(t)$ The candidate edge set at t. 20

$\mathcal{M}_o(t)$ The occupied edge set at t. 20

$\mathcal{M}_u(t)$ The unoccupied edge set at t. 20

t The time step, number of occupied edge or number of occupied nodes. 20

$\xi(\cdot)$ The local goal function regarding an edge or node. 20

$RS(\cdot, \cdot)$ The random chosen function usually following the uniform distribution. 20

n_s The number of selections or candidates. 20

t_c The critical step where a percolation transition undergoes. 21

$c(i)$ The component that node i belongs to. 21

$\mathcal{G}_p(t/n)$ The fraction of the LCC during a percolation process. 22

γ The exponent regarding SF networks. 28

Symbols

$\mathcal{N}_c(t)$	The candidate node set at t. 29
$\mathcal{N}_o(t)$	The occupied (remaining) node set at t. 29
$\mathcal{N}_u(t)$	The unoccupied (removed) node set at t. 29
q	The fraction of removed nodes. 29
$\mathcal{G}_a(q)$	The fraction of the LCC during an attack process. 29
q_c	The critical threshold. 30
η_i	The infection ratio (probability). 32
η_r	The recovery ratio (probability). A10
η_0	The basic reproductive number. 32
η_c	The epidemic threshold regarding the SIR model on networks. 33
F	The average fraction of the LCC. 36
$\mathcal{N}_{\mathbf{FVS}}$	The feedback vertex set. 38
ℓ	A control parameter regarding ACIS. 39
$\hat{\Gamma}(i)$	The distinct nearest neighbor set of node i. 40
K	A control parameter regarding AEIS. 40
r_u	A control parameter regarding ARRS and APRSrr. 45
T	A control parameter regarding ARRS and APRSrr. 45
δ_{n_s}	A control parameter regarding ARRS and APRSrr. 46
δ_{r_u}	A control parameter regarding ARRS and APRSrr. 46
$\xi_g(\cdot)$	The global goal function. 47
$D(q_c)$	The normalized q_c difference. A22
$S_p(t_1, t_1')$	A slice of a sequence S. 63
$F(S_p(t_1, t_1'))$	The local F. 64
$\hat{T}_p$	A control parameter regarding pruning strategies. 65
θ_i	The phase of oscillator i. 86
ω_i	The natural frequency of oscillator i. 86
λ	The coupling strength. 86
$\Re$	The order parameter regarding synchronization. 86
$\alpha_-(e_{ij})$	The cut procedure. 87
$\alpha_+(\sigma_{uv})$	The add procedure. 87
$\xi^\delta(g)$	The goal function regarding δ. 87
$\mathcal{J}_e$	The maximal jump size regarding the forward transition. 89
$\mathcal{J}_b$	The maximal jump size regarding the backward transition. 89
$\mathcal{S}$	The hysteresis area. 90

1 Introduction

1.1 Motivation

Network or graph as an effective approach has gained numerous attention from a large range of domains, including physics, mathematics, computer science, neuroscience, complexity science, social science, and many others [1, 2]. Usually, the reason that people focus on networks is two-fold. On the one hand, a highly interacted system can be appropriately modeled by a graph whose nodes represent the dynamic units and whose edges capture their interactions. It is much easier to study the global properties of a system from a network perspective, such as the stability of power grids [3] or the spreading dynamics of messages or epidemics in communication or contact systems [4]. On the other hand, studying the corresponding network provides a test bed for manipulation on real systems and facilitates better solutions to control, predict, optimize, or reconstruct them [5, 6]. In general, the solution of problems regarding networks converges to the identification of edges, nodes or/and the entire network, no matter using networks to capture the leading factors from raw data [7] or employing networks to study the spreading pattern of virus [8]. Hence, this thesis focuses on the identification of nodes and networks, and particularly studies problems of the network robustness (resilience) [5, 9, 10], the network immunization [11, 12], and the explosive synchronization [13].

Network robustness. Errors and failures are ubiquitous in the human world [9]. The failure of some components of our computer, like an unimportant key, would only have limited influence on our work, and we can still keep using it as usual. However, the breakdown of a critical component, such as the central processing unit, would possibly drive us crazy. Besides, the traffic perhaps becomes normal again after a short jam caused by some accident, even though that road is still blocked. But a similar incident on a different road could turn the whole system into chaos [14]. Further, climate change continuously increases the frequency and intensity of extreme events, which keeps challenging the resilience of infrastructure and boosting the global supply chain risks [15]. There is evidence that 35%-85% business losses were caused by the disruption to the transportation or electricity supplies and not by the flood itself [16]. Certainly, more roads or power lines could be built to improve their robustness, but they always corresponds to the need of a lot of resources, which is basically impossible in the short term. As an alternative, one can maybe widen some roads to increase the capacity of transportation or develop a more stable power system. But we still need to face other problems such as which road should we choose? Indeed, we cannot find exact solutions to those problems, even in the near future. But the study of those problems on networks might help us understand the principle of those problems and shine our road to the right solutions.

Usually, studies regarding the network robustness aim to answer how the structure of a network influences its robustness. For instance, decades ago, Paul Baran was assigned to design a communication system that can survive a Soviet nuclear attack, and he thought a robust network should be a network that most of its remaining nodes could still communicate with each other after some nodes failed, which guided him to suggest that a network in grid type would be robust against an intentional attack [17]. And he also concluded that the denser a network is, the more robust would it be. But decades later after Baran's suggestion was ignored, the topology of the Internet grew into a scale-free distribution [2, 9, 18]. And thus we have a network that is quite fragile under an intentional attack on hubs, though it is very robust over the random failure [9]. Shortly later, Cohen et al. accordingly gave the corresponding analytical explanations in refs. [19, 20]. And more related works could be found in refs. [1, 2], such as the effects from the clustering coefficient, the length of the average shortest path, the community structure, the assortativity, the network motifs, etc.

This thesis considers such problem, the network robustness, in a more practicable way. That is, rather than study networks generated by specific models, we directly focus on those drawn from real-world scenarios, such as email communication networks, Internet topology networks, social networks, etc. In other words, we assume that a network already exists, and aim to investigate its reactions to varied attacks, which guides us to a problem as to what the real robustness of a given network is. And apparently, a network's robustness would be bounded by the most advanced attack strategy. Hence, the considered problem is then equivalent to the search of the most advanced attack strategy, which forms one of the main goals of this thesis. Some related works that achieve state-of-the-art attacks can be found in refs. [21, 22, 23, 24, 25, 26].

Network immunization. Infectious diseases, as one of the biggest enemies to global health, could cause rapid population declines or species extinction [27]. And there is never a lack of examples, from the Black Death (probably bubonic plague) which is estimated to have caused the death of as much as one-third of the population of Europe between 1346 and 1350 [28], to nowadays COVID-19 pandemic which might result in the largest global recession in history [29], in particular, climate change keeps exacerbating the spread of diseases and increasing the probability of global epidemics [30, 31, 32]. To tackle problems regarding infectious diseases, the first thing is to figure out their transmission patterns. In tandem with the fact that most infectious diseases are transmitted through direct or indirect contacts, the network ultimately plays a key ingredient of the corresponding epidemic modelling [33, 34]. And a natural problem also arises as to the design of corresponding immunization strategies on networks, i.e., network immunization [11, 12].

Ways to contain an epidemic might include social distancing, cancellation of airlines, quarantine, and closure of shops or public areas, etc. And the aim is to curb the basic reproductive number (concept see Section 3.1.2.1) if the epidemic follows the Susceptible-Infected-Recovered (SIR) model (which most epidemics obey). Sometimes, we are lucky. The virus might vanish as, e.g., the temperature increases. Or it only has limited contagion and could easily be wiped out. But sometimes, we have to let infected individuals recover by themselves and wait for herd immunity. Or even worse, our body cannot produce an antibody, like AIDS. Immunization strategies aim to those problems and study: i) if isolation is needed, the quarantine of which group of individuals is more important to contain the spread of an epidemic, such as officials or teachers? ii) if a vaccine is found but has a limited amount, particularly in some developing countries where the resources are always limited, who should be targeted for treatments first? iii) to prevent the outbreak of an epidemic, which places, like airlines or airports, should we consider more intensively? In the context of complex networks, the solution of those problems is equivalent to the identification of a small group of key nodes that dominate the whole network. After the removal (deactivation, immunization, or isolation) of such a group, the epidemic would be contained and only have limited effects on the remaining network.

Therefore, finding a better way to immunize a network is like the other side of the same coin of searching for the most advanced attack strategy, i.e., network robustness. Intuitively, after the removal of a part of key nodes, the possible maximum spread is bounded by the size of the largest connected component (concept in Section 2.4). That means, if there is no giant component in the remaining network, then the virus would die out within only a limited spread, no matter how large the infection rate or how small the recovery rate is. Thus, the random immunization (or herd immunity) corresponds to the random failure of nodes (network robustness), which requires us to remove sufficient nodes, i.e., at least enough to fragment the entire network, to eliminate a virus.

Explosive synchronization. Explosive synchronization (ES) [13, 35, 36, 37] is a critical phenomenon, which is observed when the coupled oscillators (e.g., of the Kuramoto system [38]) are associated with a scale-free topology [13], i.e., the natural frequency of each oscillator proportionally corresponds to its number of connections and they are coupled by the related adjacency matrix. If the node degree sequence of networks is fixed, then the ES to some extent only relies on the adjacency matrix, i.e., the network structure. Thus, based on it, we could study whether the network robustness (particularly the mean of the order parameter regarding percolation) could be used as a measure to capture the network structure, which has potential applications in such as brain and climate networks [39, 40].

1.2 Contributions

The main contributions of this thesis are as follows.

Analysis of metrics, and thorough evaluation and comparison of existing approaches. Aiming at the state of the art it is important to understand the underlying theory and have unique metrics for comparisons of various methods. Hence, this thesis firstly reproduces a few results regarding percolation, explosive percolation, network robustness, SIR model, and network immunization in Sections 2.4, 2.5, and 3.1, where the connections among them are analyzed and studied too. Based on that, the reason why the critical threshold of the percolation transition and the mean of the order parameter are chosen as ultimate metrics is stated in Section 3.2.

Meanwhile, to verify the effectiveness of proposed methods more thoroughly and clearly, over 20 existing well-known approaches are brought together and compared under the same framework (i.e., the above metrics) on three rather small networks, including one scale-free network and two empirical networks. The selection of these networks considers both the density of edges and the variety of network structure. In such a manner one could thus have global views of advantages and disadvantages in regard to those mentioned methods. These comparisons can be found in Section 3.3.

Bounded and unbounded methods, and evolutionary framework. Following those comparisons, the method that on average has the best performance can be acquired. By the aid of it, this thesis then gradually studies approaches based on rules from the explosive percolation (whose order parameter usually undergoes an irreversible transition). Specifically, motivated by the fact that most percolation on regular networks are reversible (i.e., the order parameters of the forward transition (percolation) and backward transition (attack) are in principle equivalent to each other), we would like to know whether the rule leading to an explosive percolation could also be used to attack an existing network. If it does, then those rules could usually heavily delay the critical threshold, which is highly associated with the solution of the robustness and immunization problems, that is, obtaining the minimum node set whose removal would break down the entire network or whose immunization could effectively contain an epidemic. Therefore, we accordingly evaluate both bounded and unbounded rules from Section 2.5 in regard to nodes, and also propose a few strategies particularly for existing networks. As a result, 5 bounded-size strategies and 2 unbounded-size approaches are developed, which somewhat work for all kinds of networks, especially the relate-relationship strategy (ARRS). Besides, similar to the explosive percolation, unbounded methods are usually more capable of acquiring better solutions but they are also more time-consuming than bounded. More details of analysis and comparisons can be found in Section 3.4.

However, both bounded and unbounded strategies would suffer the problem of local optimum. To tackle that, the effects of the initial sequence and also the corresponding control variables in regard to ARRS are further discussed. Based on them, the evolutionary framework for identification of influential nodes is built, where selection strategies, mutation operators, and the ways to initialize and maintain a population (sequence) are studied and included in Section 3.5. Meanwhile, Section 3.6 also shows a fast scheme to suppress the order parameter.

Order parameter as a measure of the network topology. During the dynamic removal of existing nodes, the order parameter keeps tracking the size of the largest connected components. Such parameter, on the one hand, could effectively quantify the robustness of a given network under a consecutive attack, and on the other hand, shows potential ability to capture the network structure. In other words, if we view F (the mean of the order parameter F (concept in Section 3.2), a scalar of 0 to 0.5) as a measure, then what role does F play in such as brain or climate networks? To verify that, the explosive synchronization regarding the Kuramoto model is considered in Section 4.1 where effects of both F and the assortativity (concept in Section 2.3.1) as well as their combination are studied. Following that, ways to enhance a network against the most advanced attack strategy are discussed in Section 4.2, which might also play role in such as keeping the variety of network samples regarding Section 4.4 where the effectiveness of machine learning tools on the same problem of Section 4.1 is investigated. Meanwhile, based on similar ideas, this thesis also studies the influences of acquaintances on the containment of epidemics in 4.3.

1.3 Organization and Outline of the Thesis

In summary, Chapter 2 reviews basic concepts regarding complex networks and network percolation. Chapter 3 discusses the network robustness and immunization problems mainly from the perspective of the identification of nodes. And Chapter 4 follows that but in a view of the identification of networks. Those two chapters constitute our main contributions, which are summarized in Chapter 5, including the outlook.

In particular, Section 2.1 gives some basic definitions of complex networks, including ways to represent a network, which is crucial for those who want to implement the related algorithm of this thesis. From there, one can also find concepts with respect to the adjacency matrix, component, cycle, clustering coefficient, etc. Following that, a few general approaches to characterize nodes and networks are shown in Sections 2.2 and 2.3, such as PageRank and assortativity. Section 2.4 briefly introduces the concept of percolation through the example on a two-dimensional square lattice, followed by a more detailed explanation on the random network. One can also find concepts such as critical threshold, subcritical regime, supercritical regime, etc. in Section 2.4. Further, Section 2.5 reviews varied rules in regard to the explosive percolation including bounded-size rules and unbounded-size rules, which forms the basis of the proposed basic methods.

Then, Section 3.1 discusses the network robustness and immunization problems in more detail, which also gives us the goals to optimize (i.e., metrics to methods in Section 3.2). By the aid of them, Section 3.3 reviews the existing state-of-the-art methods, where comprehensive comparisons among them are also conducted. Section 3.4 shows our proposed 7 basic strategies, including 5 bounded and 2 unbounded. Basically, those strategies are suitable for all kinds of networks, and any of them could acquire better or comparable results than all of those methods in Section 3.3. Further, based on these basic methods, the evolutionary framework for the identification of influential nodes is discussed in Section 3.5. And Section 3.6 introduces a fast scheme to curb F.

Next, we move to the identification of networks. Focusing on that, Section 4.1 investigates the effects of network robustness on explosive synchronization. Section 4.2 studies a few ways to enhance the robustness of a network. Following that, Section 4.3 further discusses approaches to boost the effectiveness of immunization strategies. In Section 4.4, a few machine learning methods are also verified on data from Section 4.1 to demonstrate the predictability and controllability of a specific behaviour of such a dynamic system from Section 4.1.

Lastly, the main contributions of this thesis are summarized in Section 5.1 followed by the outlook in Section 5.2.

1.4 Relation to Previously Published Work

I thank my co-authors for allowing me to use material from our joint papers. Some of results in this thesis have already been previously published in the following papers:

[L1] Yang Liu, Xi Wang, and Jürgen Kurths. "Optimization of targeted node set in complex networks under percolation and selection". In: Physical Review E 98.1 (2018), p. 012313.

[L2] Yang Liu, Xi Wang, and Jürgen Kurths. "Framework of evolutionary algorithm for investigation of influential nodes in complex networks". In: IEEE Transactions on Evolutionary Computation 23.6 (2019), pp. 1049–1063.

[L3] Yang Liu and Jürgen Kurths. "Effects of network robustness on explosive synchronization". In: Physical Review E 100.1 (2019), p. 012312.

Additional research work not covered by the material in this thesis can be found in:

- Yong Zhao, Xiaoyan Sun, Yang Liu and Jürgen Kurths, "Phase synchronization dynamics of coupled neurons with coupling phase in the electromagnetic field". In: Nonlinear Dynamics 93.3 (2018), pp. 1315–1324.
- Xiujing Han, Yang Liu, Qinsheng Bi and Jürgen Kurths, "Frequency-truncation fast-slow analysis for parametrically and externally excited systems with two slow incommensurate excitation frequencies". In: Communications in Nonlinear Science and Numerical Simulation 72 (2019), pp. 16–25.
- Jingfang Fan, Jun Meng, Yang Liu, Abbas Ali Saberi, Jürgen Kurths and Jan Nagler, "Universal gap scaling in percolation". In: Nature Physics 16.4 (2020), pp. 455–461.

2 Complex Network Theory and Network Percolation

In this chapter we present underlying concepts regarding complex networks, attributes and measures of nodes and networks, percolation theory, and explosive percolation.

2.1 Complex Network Theory

2.1.1 Networks

A *network* (graph[1]) $G(\mathcal{N}, \mathcal{M})$ consists of a number of nodes (vertices) tied by a group of edges (links) where $\mathcal{N}$ and $\mathcal{M}$ are accordingly the node set and the edge set. Let $n = |\mathcal{N}|$ and $m = |\mathcal{M}|$ be the corresponding number of nodes and edges, respectively. Then we also refer to a network as $G(n, m)$ which indicates a network G constructed with n nodes and m edges (see Fig. 2.1 for an example). In general, the node (see Table A.1) could be an agent in a multi-agent system, an interaction in a road network, or an airport in the global airline network. Such node could also have some properties, like the size, security, and location of an airport. The edge could be a road connecting two interactions, an airline between two airports, or friendships among individuals. In practice, different scenarios could share a similar fundamental structure such as following similar macroscopic characteristics, even though their node and edge have different meanings. Besides, the structure of a network might also be far complicated, e.g., node and edge could have weights, or connections between two nodes could be multiple (see Fig. 2.1). Among them, the most fundamental and important structure in both network science and graph theory is the simple network [2, 41]. A simple network is an undirected and unweighted network without self-loops (Fig. 2.1). In this thesis, we will mainly consider simple networks and refer to a simple network as a network if there is no special explanation.

[1]The difference between network and graph is from the difference of network science and graph theory. They are actually interchangeable. Thus, we view network and graph as the same thing throughout this thesis, even though there are some subtle differences [2].

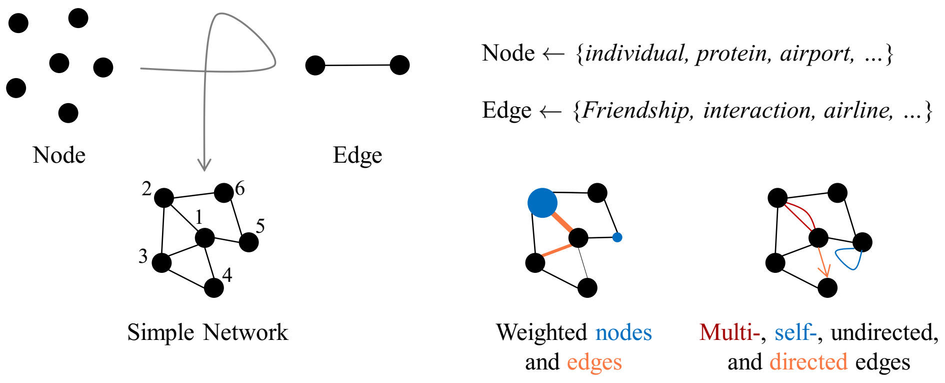

Node
Edge
Node ← {individual, protein, airport, ...}
Edge ← {Friendship, interaction, airline, ...}
Simple Network
Weighted nodes and edges
Multi-, self-, undirected, and directed edges

Figure 2.1: Example of nodes, edges, and networks. A simple network is an unweighted and undirected network without self-loops.

2.1.2 Adjacency matrix

The core to represent a network is to find an appropriate way which can fully capture those interactions among nodes. For example, we can firstly label each node from the simple network in Fig. 2.1, and then employ the following two arrays to represent that network,

$$adj = [2,3,4,5,1,3,6,1,2,4,1,3,1,6,2,5],$$

$$idx = [0,4,7,10,12,14,16].$$

In this manner, one can obtain the nodes connecting to node i through[2] $adj[idx[i]+1 : idx[i+1]]$, or get an edge e_{ij} where[3] $j = adj[idx[i]+1]$. This representation is actually very useful in storing the network or for some calculation running on the network.

However, a better such way for mathematical calculations is the *adjacency matrix* that could efficiently represent a network by the aid of a matrix. For a simple network like the one in Fig. 2.1, the element of the adjacency matrix follows

$$A_{ij} = \begin{cases} 1, & \text{if node } i \text{ and } j \text{ are connected to each other,} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Thus, the corresponding adjacency matrix A is written as

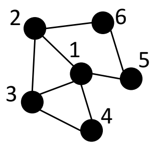
$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

[2]Here we follow the rules in computer programme but with a slight difference. That is, for instance, considering node 3 in Fig. 2.1, $idx[3] = 7$ (not 10), $idx[3]+1 : idx[3+1]$, i.e., $8:10$, denotes $[8,9,10]$, and thus we get $adj[idx[i]+1 : idx[i+1]] = [1,2,4]$. Note that we will use these symbols and rules through the whole thesis.

[3]We also use $e_{ij} \in \mathcal{M}$ to denote an edge starting from node i and ending at node j through the whole thesis.

Apparently, here A is symmetrical because the network that we considered is an undirected network. For a directed network, $A_{ij} = 1$ indicates that there is an edge starting from node i and ending at node j.

2.1.3 Essentials



$$\Gamma(1) = \{2, 3, 4, 5\}$$
$$k_1 = 4, k_2 = 3$$
$$\langle k \rangle = \frac{4+3+3+2+2+2}{6} = 16/6$$
$$p_2 = 3/6, p_3 = 2/6, p_4 = 1/6$$

Figure 2.2: Examples for the nearest neighbor set $\Gamma(i)$, node degree k_i, average degree $\langle k \rangle$ and degree distribution p_k.

2.1.3.1 Nearest neighbor

The *nearest neighbor* set $\Gamma(i)$ of a node i is a node set which contains all nodes directly connecting to node i, that is (Fig. 2.2),

$$\Gamma(i) = \{j | A_{ji} = 1, \forall j \in \mathcal{N}\}. \tag{2.2}$$

2.1.3.2 Degree, average degree, degree distribution

The *degree* k_i (see Fig. 2.2 as an example) of node i is the sum of weights on edges associated with all nodes in its nearest neighbors $\Gamma(i)$. For a simple network where weights of both node and edge are fixed to 1, the degree of i is equivalent to the number of nodes in $\Gamma(i)$, i.e.,

$$k_i = \sum_{j \in \Gamma(i)} A_{ji} = |\Gamma(i)|. \tag{2.3}$$

Obviously, the number of edges $m = \frac{1}{2} \sum_{i \in \mathcal{N}} k_i$. Besides, one can also obtain k_i through $idx[i+1] - idx[i]$.

The *average (mean) degree* $\langle k \rangle$ is defined as

$$\langle k \rangle = \frac{1}{n} \sum_{i \in \mathcal{N}} k_i = \frac{2m}{n}, \tag{2.4}$$

which corresponds to the first moment of the degree sequence.

Further, the *degree distribution* p_k represents the probability that a randomly chosen node has degree k. For a specific network, since p_k has to follow $\sum_k p_k = 1$, it is usually defined as

$$p_k = \frac{\text{\#nodes with degree } k}{n}, \tag{2.5}$$

where # means 'the number of'. With this, the average degree can also be obtained through

$$\langle k \rangle = \sum_k p_k. \tag{2.6}$$

2.1.3.3 Walks, paths, connected components, and cycles

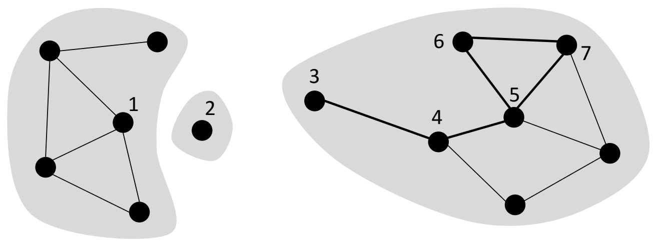


Figure 2.3: Examples of walks, paths, components and cycles. Node 6 can reach 3 through one of three walks $[6, 5, 4, 3]$, $[6, 7, 5, 4, 3]$ or $[6, 5, 4, 5, 4, 3]$, in which $[6, 5, 4, 3]$ and $[6, 7, 5, 4, 3]$ are both path and $[6, 5, 4, 3]$ is the shortest path. Besides, none of 1, 2, and 3 can be reached by each other. Thus, the network contains three connected components covered by independent gray shadows. An example of a cycle could be $[6, 5, 7, 6]$.

In graph theory, for a given network $G(\mathcal{N}, \mathcal{M})$, a *walk* from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$ is a *sequence*[4] S starting from i and ending at j, in which S comprises nodes from G and satisfies $e_{uv} \in \mathcal{M}$ if $u = S[l]$ and $v = S[l+1]$ for $\forall S[l] \neq j$ (see Fig. 2.3 as an example[5]). It is worth noting that both nodes and edges in S are repeatable. The length of the walk from i to j is the number of edges in S, or $|S| - 1$ where $|S|$ denotes the number of elements in S[6]. One can also easily obtain how many walks with length $|S| - 1$ exist between any pair of nodes as

$$A^{|S|-1},$$

where the ith-row-jth-column value is the number of walks with length $|S| - 1$ from i to j.

A *path* from i to j is a *distinct sequence*[7] S starting from i and ending at j, in which repetitions of nodes and edges are not allowed. A *shortest path* is a path with minimum $|S|$, and the corresponding length d_{ij} can be obtained through

$$d_{ij} = \min_S |S| - 1, S[1] = i, S[|S|] = j. \tag{2.7}$$

Note that it is possible to have multiple shortest paths between two nodes. The *diameter* $d_{\max}$ of a network is the longest shortest path among all node pairs,

$$d_{\max} = \max d_{ij}, \forall i \neq j. \tag{2.8}$$

Further the *average shortest path* $\langle d \rangle$ is defined as

$$\langle d \rangle = \frac{1}{n(n-1)} \sum_{i,j \in \mathcal{N}, i \neq j} d_{ij}. \tag{2.9}$$

Sometimes, it is also possible that node i cannot reach node j through existing edges, i.e., there is no path from i to j. For such a case, we denote the length of the shortest path between i and j with $d_{ij} = d_\infty = \infty$. The meaning of this might be, for instance, that a virus

[4]Sequence is a well-defined collection of ordered objects where repetitions are allowed.

[5]Through the whole thesis, we use [...] instead of regular representation (...) to denote a sequence.

[6]View occurrences of the same node as different elements if there is.

[7]Distinct sequence means that the elements in the sequence are distinct, i.e., repetitions are not allowed.

cannot transfer from an individual to another if those two do not have any direct or indirect connections. The corresponding concept from network science to indicate whether two nodes are reachable mutually is the *connected component*. A connected component in a network G is a subnetwork of G, in which every node can be reached through at least one path by others within the subnetwork but cannot be reached by those outside the subnetwork (see Fig. 2.3 as an example).

For a simple network, a *cycle* is a path ending with the start node, which means only the start node is allowed to be repeatable once. Besides, we employ $d_c(i)$ to denote the length of a cycle associated with node i,

$$d_c(i) = \#\text{edges in the cycle}. \tag{2.10}$$

In particular, $d_c(i) = 0$ means that there is no any cycle related to i in the given network.

2.1.3.4 Clustering coefficient

For a simple network, a triangle connection regarding node i is a cycle with length $d_c(i) = 3$, which characterizes three nodes are the nearest neighbors to each other. An example of a triangle connection is that one's friends are also friends to each other in such as a social network. The local clustering coefficient CC_i [42] is defined to capture the degree (ratio) of triangle connections with respect to a specific node i,

$$CC_i = \frac{\#\text{triangle connections regarding } i}{\binom{k_i}{2}}, \tag{2.11}$$

where $\binom{k_i}{2}$ means how many edges could exist among the nearest neighbors of i. Thus, we can also view the clustering coefficient as a measurement of local density: the more edges among $\{i, \Gamma(i)\}$, the larger CC_i. And apparently, $CC_i \in [0, 1]$. For the whole network, one can use the average clustering coefficient $\langle CC \rangle$ over all nodes to globally capture the status of triangle connections,

$$\langle CC \rangle = \frac{1}{n} \sum_{i \in \mathcal{N}} CC_i, \tag{2.12}$$

which could also serve as a measure of networks. Considering Fig. 2.2 as an example, we have $CC_1 = 2/6$, $CC_4 = 1/1$, $CC_5 = 0/1$, and $\langle CC \rangle = 7/18$.

2.2 Measures of Nodes

As we mentioned in Table A.1, a lot of systems can be properly modeled by networks, ranging from communications among human beings to protein interactions in a cell. And basically, as long as there are communications or interactions, there would be some individuals that are more influential than others, even for a function in which some parameters are more important than others. Thus, a problem arises as to how can we ascertain that a node is more influential than others? Indeed, this problem is associated with a lot of realistic problems. For instance, during a pandemic, which group of people should be isolated to suppress the spread instead of asking everyone to stay home? Who is the true influencer whom one can rely on to spread their information, e.g., advertising? Further, attacks on which group of proteins could effectively kill unwanted bacteria? Yet, we are not able to answer these questions in this thesis, even in the near future. But the study of these problems on networks might shine the path to the real answers. In what follows, a few measures are introduced as the purpose of gaining the basic concepts. More other methods can be found in Chapter 3 later.

2.2.1 Degree centrality

The *degree centrality* is perhaps the most straightforward and simplest way to measure a node. It identifies nodes directly through their degree. Practically, one might easily convince you that a paper is reliable if it is cited by a lot of papers. In a social network, celebrities (usually owns a lot of connections), for example, cannot feel free to post or comment on something while others with a few connections are possibly able to since they have more influences and their opinions perhaps result in disasters. From the perspective of network science, a node i is said to be more important than another one j if and only if $k_i > k_j$ under the degree centrality.

2.2.2 Eigenvector centrality

The main idea of the *eigenvector centrality* [43] is that a node connected to important nodes might also be an important node, even though sometimes it only has a few connections. Therefore, different from the degree centrality considering each node equally, the eigenvector centrality measures the influence $\mathcal{H}_i$ of a node i through summing up the centralities from its neighbors,

$$\mathcal{H}_i = \frac{1}{\lambda} \sum_{j \in \Gamma(i)} \mathcal{H}_j, \tag{2.13}$$

where λ is a constant [43]. The solution of Eq. (2.13) could be well approximated by the power method (see Appendix A.1.2). Hence, for the eigenvector centrality, a node i is more influential than another one j if and only if $\mathcal{H}_i > \mathcal{H}_j$.

2.2.3 Katz centrality

From the way in which the eigenvector centrality employs to obtain $\mathcal{H}$, we know that a node i actually gets its score $\mathcal{H}_i$ by iteratively aggregating the information from its nearest neighbors. In this manner, $\mathcal{H}_i$ could possibly contain all the information from the whole network. This is a good strategy, but there is a problem that node i under the eigenvector centrality views the information from other nodes equally, no matter whether those nodes are its nearest neighbors or some others are far away from it. The *Katz centrality* [44] can address this problem.

The Katz centrality uses a parameter α to control the magnitude of the information that it aggregates from different nodes,

$$\begin{aligned} \mathcal{H}_i &= \sum_{j \in \Gamma(i)} Z_{ji}, \\ Z &= \sum_{t=1}^{\infty} \alpha^t A^t. \end{aligned} \tag{2.14}$$

Assuming that $\alpha < \frac{1}{\lambda_1}$, then the Katz centrality could be obtained through

$$\mathcal{H} = \alpha A^{\mathrm{T}} \mathcal{H} + \mathbf{1}, \tag{2.15}$$

where $\mathcal{H}^{\mathrm{T}}$ is the transpose of $\mathcal{H}$ and $\mathbf{1}$ represents a vector $(1, 1, 1, ...)$ (detains see Appendix A.1.3).

2.2.4 PageRank

Indeed, the Katz centrality can balance the information from nodes with different distances. But still, it suffers from another problem: a node copies its centrality to all its nearest neighbors. In other words, for example, a very important node i might connect to a number of nodes in a network, and thus it makes those nodes influential. And because they get i's centrality directly, some of them would have larger centralities than other important nodes in second or further layers even though those nodes are actually unimportant. Perhaps we can overcome this problem through the adjustment of α in Eq.(2.15). But it is usually not a preferable way because one cannot know which α is the best.

To overcome that, the *PageRank* [45] is presented, which initially is developed for the ranking of web pages. Therefore, it mainly considers the problem in directed networks,

$$\mathcal{H}_i = \frac{1-\alpha}{n} + \alpha \sum_{j \in \Gamma(i)} \frac{\mathcal{H}_j}{k_j^{\mathrm{out}}}, \tag{2.16}$$

where α is a constant parameter called residual probability[8], which is usually set to 0.85, and k_j^{out} is the outdegree of node j. In this manner, the centrality $\mathcal{H}_i$ of node i is equally divided and assigned. Note that the nearest neighbor set $\Gamma(i)$ defined for undirected network corresponds to the in-neighbors here. Eq. (2.16) has a problem that a node i cannot give its score $\mathcal{H}_i$ out if $k_i^{\mathrm{out}} = 0$, which means that i would 'absorb' centralities from other nodes and make $\sum_j \mathcal{H}_j$ smaller and smaller with the increase of iteration. One way to tackle this problem is to let those nodes with $k_i^{\mathrm{out}} = 0$ connect to all other nodes in the network. Thus, we have a modified adjacency matrix, say A', and also the corresponding degree sequence k'^{out}. Then, in matrix notation, we have

$$\mathcal{H} = \alpha A'^{\mathrm{T}} D^{-1} \mathcal{H} + \frac{1-\alpha}{n} \mathbf{1}, \tag{2.17}$$

[8]Here $1-\alpha$ can be understood as that a visit might start from any pages.

in which D is the diagonal matrix of k'^{out}. Rearranging it, one can exactly get the centrality through

$$\mathcal{H} = \frac{1-\alpha}{n}(I - \alpha A'^{\mathrm{T}} D^{-1})^{-1}\mathbf{1}. \tag{2.18}$$

If letting $\mathcal{H}$ follow $\sum_i |\mathcal{H}_i| = 1$, then Eq. (2.17) can be rewritten as

$$\mathcal{H} = (\alpha A'^{\mathrm{T}} D^{-1} + \frac{1-\alpha}{n} Z)\mathcal{H}, \tag{2.19}$$

where Z is a $n \times n$ matrix with all entries equal to 1, i.e., $Z\mathcal{H} = \mathbf{1}$. Therefore, we can still employ the power method to get the PageRank centrality.

2.2.5 Closeness centrality

The basic idea of the *closeness centrality* is that an important node should be close to as many other nodes as possible. Thus, it calculates the centrality [46] through

$$\mathcal{H}_i = \frac{n}{\sum_{j \in \mathcal{N}} d_{ij}}, \tag{2.20}$$

which indicates that a node is more important if it has a smaller average length of shortest paths to other nodes. Note that Eq. (2.20) has a normalized term compared to the original definition in ref. [46], which makes it have the capability to compare two nodes from different networks.

2.2.6 Betweenness centrality

The *betweenness centrality* [47] also relies on the shortest path in a network, but is calculated by counting the number of shortest paths that a node locates at instead of the average length of shortest paths. In this way, compared to the closeness centrality, the betweenness centrality is usually more capable of identifying the importance of a node, like a node with large betweenness centrality might be associated with the 'bottleneck' of a communication system. Specifically, the betweenness centrality obtains the centrality $\mathcal{H}_i$ of a node i by

$$\mathcal{H}_i = \sum_{u,v \in \mathcal{N}, u \neq i \neq v} \frac{\#\text{shortest paths containing } i \text{ from } u \text{ to } v}{\#\text{shortest paths from } u \text{ to } v}, \tag{2.21}$$

which can be further normalized through $\mathcal{H} = \frac{\mathcal{H} - \min \mathcal{H}}{\max \mathcal{H} - \min \mathcal{H}}$.

2.3 Measure of Networks

The degree centrality tells us that a node with a large degree is more important than another one with a small degree. But what about two nodes in different networks? It is not too difficult to find an instance that two nodes with the same degree might have totally different influences in different networks, e.g., one in a dense network and the other one in a sparse network. Besides, almost all centralities in the previous section reveal that the function of a node is highly associated with the network structure. Thus, how could we globally characterize a network? Some straightforward measures could be those in Section

2.1.3, like the degree distribution, the average degree, the average shortest path, or the average clustering coefficient. For example, one can easily verify that the denser a network is, the more robust it would be [9]. More details regarding the measure of networks will be discussed later in Section 4 except for the following one since some associated definitions will be employed to explain percolation.

2.3.1 Assortativity

It is common to think that people might be more likely to develop their relationship with those who are more famous than themselves, like collaborations in the film or academic field. In a network, if we divide the nodes into two groups: one contains nodes with large degrees and the other with small degrees. One might think about a question regarding: what do the majority of edges connect (two nodes both have a large or small degree, or one with large and the other one with small)?

To answer it, we first define a conditional probability $P(k'|k)$ which characterizes the probability that a node with degree k connects to a node with degree k'. Besides, recalling that the degree distribution of a network is p_k, then we have[9]

$$P_k = \frac{kp_k}{\langle k \rangle}, \tag{2.22}$$

where P_k is the degree distribution of a node located at the end of a randomly chosen edge. Therefore, for the independent case, $P(k'|k) = P_{k'}$. But for other cases, it is usually difficult to get $P(k'|k)$ directly [48]. As an alternative, we can achieve a similar goal through

$$k_{nn}(k) = \sum_{k'} k' P(k'|k), \tag{2.23}$$

which characterizes the mean degree $k_{nn}(k)$ of nodes adjoined by nodes with degree k. Back to the independent case, $k_{nn}(k) = \sum_{k'} k' P_{k'} = \sum_{k'} k' \frac{k' p_{k'}}{\langle k \rangle} = \frac{\langle k^2 \rangle}{\langle k \rangle}$ indicates that it only depends on $\langle k^2 \rangle$ and $\langle k \rangle$. For a given network, one can estimate $k_{nn}(k)$ by

$$k_{nn}(k) = \frac{\sum_{k_i = k} \sum_{j \in \Gamma(i)} k_j}{\sum_{k_i = k} k}. \tag{2.24}$$

In this way, if $k_{nn}(k)$ increases with k for a particular network, i.e., nodes with large degrees tend to have nearest neighbors with large degrees, then the network is thought to be assortative [48]. On the contrary, a network is disassortative if $k_{nn}(k)$ decreases with k. And in a neutral network, $k_{nn}(k)$ is independent of k.

If we want to employ a single number to measure the assortativity of the network, there are two ways to achieve it. The first one employs the trend of $k_{nn}(k)$ [48],

$$k_{nn}(k) \sim k^{\alpha}, \tag{2.25}$$

[9]Supposing $P_k = \alpha k p_k$ in a view of the truth that P_k should be proportional to the node degree k and its corresponding probability p_k, one can get $\alpha = \frac{1}{\langle k \rangle}$ following the fact $\sum_k P_k = 1$.

where $\alpha > 0$, $\alpha = 0$, and $\alpha < 0$ accordingly correspond to assortative, neutral, and disassortative networks.

The other one is the assortativity coefficient r [49] defined as

$$r = \frac{\frac{1}{m}\sum_{e_{ij}\in\mathcal{M}} k_i k_j - [\frac{1}{m}\sum_{e_{ij}\in\mathcal{M}} \frac{1}{2}(k_i + k_j)]^2}{\frac{1}{m}\sum_{e_{ij}\in\mathcal{M}} \frac{1}{2}(k_i^2 + k_j^2) - [\frac{1}{m}\sum_{e_{ij}\in\mathcal{M}} \frac{1}{2}(k_i + k_j)]^2}, \tag{2.26}$$

where e_{ij} represents the edge between node i and node j. Eq. (2.26) actually gives the *Pearson correlation coefficient* of the two degree sequences associated with all edges. Thus, $r \in [-1, 1]$ and networks with $r > 0$, $r = 0$ and $r < 0$ are assortative, neutral, and disassortative, respectively.

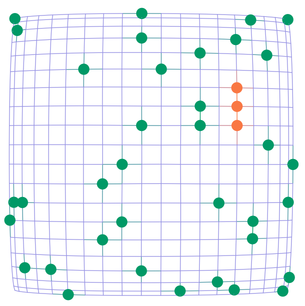
2.4 Percolation Theory

Depending on different scenarios, percolation on networks mainly includes site (node) percolation and bond (edge) percolation. In what follows, to help gain the basic concept, the percolation on a two-dimensional square lattice is briefly introduced as an example of site percolation, and the one on the random network as an instance of bond percolation. Note that even though site percolation is our main concern, bond percolation would also be presented to some extent, in particular strategies associated with the explosive percolation because the ideas from them will be used and studied in our framework.

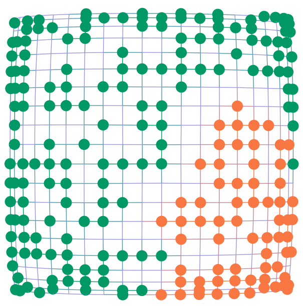
2.4.1 Percolation on two-dimensional square lattice

In a large system, a phase transition is considered to be occurring if the system undergoes an abrupt transition of its status after a tiny change of the controlling variable, such as water turning into ice or steam. In networks, like a square lattice where one site only connects to its nearest 4 neighbors (Fig. 2.4), if each site (node or vertex) is occupied with a statistically independent probability p, then at some point of p one can observe a path relying only on the occupied nodes starting (percolating) from one side of the lattice to the corresponding opposite side. More generally, with the increase of p, more and more sites merge together into some clusters in each of which one site can reach all others in the cluster through occupied nodes. For a specific p, the one with the most sites is the largest cluster. And there are always a few clusters with the same size as the largest one when p is small, but only one exists when p is large. Regarding this, people are usually interested in two problems: the expected size of the largest cluster and the average size of clusters. Apparently, both of them increase with the rise of p (Fig. 2.4). But as we mentioned, when p is small (see Fig. 2.4a), large clusters rarely exist, and they are far away from each other. Therefore, the largest cluster grows very slowly with p increases. However, if p approaches some critical point (Fig. 2.4b), say p_c where the network undergoes a percolation transition, the large clusters will be connected by a small fraction of sites. According to percolation theory [50, 2], when p is larger than p_c, the fraction of the largest cluster, namely, the probability of a randomly chosen site located in the largest (or infinite) cluster follows that

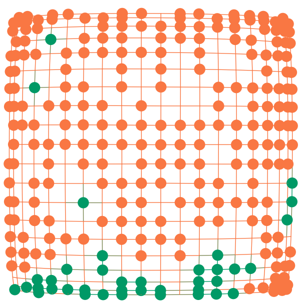
$$p_\infty \sim (p - p_c)^{\beta_p}, \tag{2.27}$$



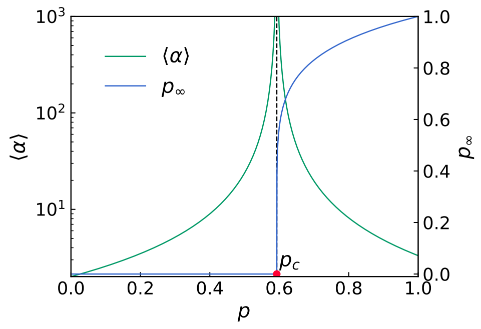
(a) $p = 0.1$



(b) $p = 0.55$



(c) $p = 0.65$



(d)

Figure 2.4: Percolation on a two-dimensional square lattice. The solid circles are occupied sites, and the largest cluster, i.e., the one with the most occupied sites, is colored orange. (a) When p is small, the majority are isolated nodes, which means all their nearest neighbors are unoccupied sites. In this case, the largest cluster is quite small. (b) The isolated nodes merge into clusters and further larger clusters as p increases. (c) Almost all occupied nodes are in the largest cluster, but still several small clusters exist. (d) The correlation length $\langle\alpha\rangle$ and the order parameter p_∞ as functions of the occupied probability p. The filled red circle and the dashed line correspond to the critical point p_c.

where p_∞ is known as the order parameter, which is zero when $p < p_c$. The average size of finite clusters is given by $\langle s\rangle \sim |p - p_c|^{-\gamma}$ [50]. Around p_c, i.e., when p approaches p_c, the percolation transition can also be characterized by the correlation length $\langle\alpha\rangle$, which is defined as the mean distance between two sites in the same finite cluster

$$\langle\alpha\rangle \sim |p - p_c|^{-\nu}. \tag{2.28}$$

Obviously (see Fig. 2.4d), when p approaches p_c from below, $\langle\alpha\rangle$ diverges, which indicates the existence of a percolation cluster. From percolation theory [50], we have $p_c \cong 0.592746$, $\beta_p = 5/36$, $\gamma = 43/18$, and $\nu = 4/3$ in the two-dimensional square lattice.

It is worth mentioning that the purpose of this part is to show some basic concepts about percolation. Hence we only introduce one of the simplest cases regarding percolation on regular graphs and give those associated conclusions directly. One can find more details and other interesting topics from ref. [50].

2.4.2 Percolation on random network

2.4.2.1 Random network

The basic concept of random graph originates from several early works finished by Gilbert and Erdős et al. [51, 52, 53, 54], and the goal is to study the probability properties over graphs controlled by some fixed variables, like the probability of observing a connected network. Usually, there are two ways to construct a random network: the Gilbert model [52] and the Erdős and Rényi (ER) model [53]. Given a network consisting of n isolated nodes, the Gilbert model, say $G(n, p)$, checks every pair of nodes and builds an edge between those two nodes with an independent probability p. Thus the expected number of edges m follows

$$m = \binom{n}{2}p = pn(n-1)/2 \tag{2.29}$$

and all graphs with n nodes and m edges can be obtained with an equal probability of $p^m(1-p)^{\binom{n}{2}-m}$, which coincides with the ER model $G(n, m)$. Note that $\binom{n}{2}$ represents the combinational factor. The ER model also considers n isolated nodes but forms the edge set by exactly choosing m edges uniformly from $\binom{n}{2}$ candidates. Studies have shown that networks obtained through those two models share many graph properties [55], in particular when $n \to \infty$.

2.4.2.2 Essentials

Since $G(n, p)$ strictly connects two nodes independently, many conclusions are derived based on this model, like the average degree follows

$$\langle k \rangle = p(n-1) \approx pn, n \to \infty, \tag{2.30}$$

which means that a node will connect to more nodes as the increase of p. As a node i has an independent probability p to build edges with any other remaining nodes, the degree distribution p_k, i.e., the probability that a randomly chosen node has degree k, follows the binomial distribution given by

$$p_k = \binom{n-1}{k}p^k(1-p)^{n-k-1}. \tag{2.31}$$

From Eq. (2.30) we know $p = \frac{\langle k \rangle}{n-1}$, thus,

$$\begin{aligned} p_k &= \binom{n-1}{k}\left(\frac{\langle k \rangle}{n-1}\right)^k\left(1-\frac{\langle k \rangle}{n-1}\right)^{n-k-1} \\ &\approx \frac{\langle k \rangle^k}{k!}e^{-\langle k \rangle} \end{aligned} \tag{2.32}$$

following the Poisson distribution when $n \to \infty$, where $k!$ denotes the factorial of k. Eq. (2.32) shows us the fundamental essential, the degree distribution, of the random network. Another property that can be directly obtained is the clustering coefficient. Since two nodes have an edge with probability p and every node has expected degree $\langle k \rangle$, the expected average clustering coefficient of a random graph is equal to the expected clustering coefficient of a randomly chosen node i,

$$\langle CC \rangle = CC_i = p = \frac{\langle k \rangle}{n-1}. \tag{2.33}$$

Besides, as we mentioned above, the network becomes denser and denser with the increase of p and finally becomes a fully connected network ($p = 1$). If considering the size of the largest connected component (cluster)[10] (LCC) p_∞, one can easily get $p_\infty \sim O(1)$ when p is small (e.g., $p = 0$) and $p_\infty \sim O(n)$ when p is large enough (e.g., $p = 1$), which indicates that p_∞ also undergoes a percolation transition as the one on the two-dimensional square lattice. To more clearly distinguish the status of p_∞, we here employ the term from ref. [54], the giant component, which is a connected component of size proportional to n. Thus, a natural question arises as to under what condition is there a giant component?

2.4.2.3 Percolation on random graph

According to refs. [54, 56, 1], the following results hold:

- if $\langle k \rangle < 1$ (recall that $\langle k \rangle = 2m/n = p(n-1)$), i.e., the subcritical regime, the network with high probability does not have connected components whose sizes are larger than $O(\log n)$;
- if $\langle k \rangle = 1$, the network with high probability has a largest connected component with a size of order $n^{2/3}$;
- if $\langle k \rangle > 1$, i.e., the supercritical regime, the network with high probability has a unique giant component whose size is proportional to n.

Therefore, there is a giant component with high probability in network $G(n, p)$ if $p > 1/(n-1)$. Full proofs regarding those conclusions refer to refs. [54, 56]. Here a straightforward explanation of the critical threshold can be found in Appendix A.1.4.

2.5 Explosive Percolation

From above we know that the Gilbert model $G(n, p)$ and the ER model $G(n, m)$ undergo a phase transition around $\langle k \rangle = 1$, recalling that $G(n, m)$ constructs a network by repeatedly adding (occupying) edges until the number of edges reaches m. And each edge is randomly chosen from all the remaining edges. But at a workshop in 2000, Dimitris Achlioptas instead asked whether it could be possible to delay the onset of the giant component by choosing an edge to add from randomly selected two based on some given function, which he called "the power of two choices"[11] [57]. From then on, this problem and the corresponding process, later called Achlioptas process (AP), have attracted a large number of researchers and grown into a large area of research [58, 59, 60, 61].

[10]Usually, for a simple network, the largest connected component (network science) and the largest cluster (graph theory) represent the same thing.

[11]Motivated by the design of hash function from computer science.

2.5.1 Basics

For the convenience of description and also the purpose of unifying the later strategies, we first define a few variables and functions as follows. Assuming that there is a distinct sequence S with random order regarding all edges in a complete network of size n, we define sets

$\mathcal{M}_c(t)$: *candidate edge set*,

$\mathcal{M}_o(t)$: *occupied edge set*, and

$\mathcal{M}_u(t)$: *unoccupied edge set*

obeying a) t is the number of edges that has been added in an empty network $G(n, \mathcal{M} = \emptyset)$, i.e., $t = |\mathcal{M}_o(t)|$; b) $\mathcal{M}_o(t) \cup \mathcal{M}_u(t) = \{S[i], \forall i\}$; and c) $\mathcal{M}_o(t) \cap \mathcal{M}_u(t) = \emptyset$. Recall that a distinct sequence represents that the elements in the sequence are distinct, i.e., repetitions are not allowed. Further, a goal function $\xi(\cdot)$ takes a node or edge as input, and a random chosen function[12] $RS(\cdot,\cdot)$ relies on a given set and integer. For example,

$$\xi(i) = k_i$$

means that the goal function returns us the degree of a node i. Similarly,

$$RS(\mathcal{M}_u(t), 2)$$

indicates that two edges are randomly selected from the unoccupied edge set $\mathcal{M}_u(t)$. With those, we have a common algorithm framework for percolation strategies in Algorithm 2.1. Taking ER network $G(n, m)$ as an example, we can construct one through Algorithm 2.1 by setting $n_s = 1$, initializing $\mathcal{M}_u(0) = \{i | S[i], \forall i\}$ and $\mathcal{M}_o(0) = \emptyset$[13], and formulating $\xi(e_{ij}) = 1$.

Algorithm 2.1: Percolation

```
   Input: $n, m, n_s$                    // $n_s$ is the number of selections
   Output: $G(n, m)$
1  Initialize $G(n, \mathcal{M} = \emptyset)$, $\mathcal{M}_u(t)$ and $\mathcal{M}_o(t)$
2  $t \leftarrow 0$
3  while $t < m$ do
4      $t \leftarrow t + 1$
5      Get candidate set, $\mathcal{M}_c(t) \leftarrow RS(\mathcal{M}_u(t), n_s)$
6      Choose the new edge, $e_{uv} \leftarrow \arg\min_{e_{ij}} \xi(e_{ij}), \forall e_{ij} \in \mathcal{M}_c(t)$
7      $\mathcal{M}_u(t) \leftarrow \mathcal{M}_u(t) \setminus \{e_{uv}\}$
8      $\mathcal{M}_o(t) \leftarrow \mathcal{M}_o(t) \cup \{e_{uv}\}$
9  $\mathcal{M} \leftarrow \mathcal{M}_o(t)$
```

[12] If there is no special explanation, it follows the uniform distribution to choose edges or nodes.

[13] $\mathcal{M}_u(t)$ and $\mathcal{M}_o(t)$ will all be initialized in this way except for an explicit explanation somewhere.

2.5.2 Bohman and Frieze's strategy

Bohman and Frieze (BF) first answered the question posted by Achlioptas with a simple strategy to show that it is possible to suppress the emergence of a giant component, which shifts the critical threshold (step) $t_c = 0.5n$ to $t_c = 0.535n$ [62]. The BF strategy is similar to the basic ER process but with an extra candidate: each time it randomly selects two edges, say e_{ij} and e_{uv}, from $\mathcal{M}_u(t)$, and chooses e_{ij} to add if both node i and node j are isolated, otherwise chooses e_{uv}. In the way of Algorithm 2.1, it corresponds to a process of $n_s = 2$ and $\xi(e_{ij}) = 2$ if both i and j are isolated, otherwise $\xi(e_{ij}) = \infty$.

The BF strategy is also referred to as a bounded-size rule [63] since it is bounded by 1, namely, it views components with size larger than 1 as the same. For a new edge, it either unites two independent components or connects two nodes in the same component. Supposing that the new edge is e_{ij}, we employ $c(i)$ to represent the component that node i belongs to, and $|c(i)|$ to denote the associated size. In addition, reformulating the goal function $\xi(e_{ij})$ as

$$\xi(e_{ij}) = \begin{cases} |c(i)|, \text{ if } c(i) = c(j) \text{ and } |c(i)| \leqslant \alpha, \\ |c(i)| + |c(j)|, \text{ if } c(i) \neq c(j) \text{ and } |c(i)| \leqslant \alpha \text{ and } |c(j)| \leqslant \alpha, \\ \infty, \text{ otherwise,} \end{cases} \tag{2.34}$$

where α is a constant or a function $\alpha(t)$ of t, then we have the bounded-size rule family that chooses the new edge leading to a smaller connected component under the constraint of α. For instance, the BF strategy corresponds to the setting of $\alpha = 1$. Further, $\alpha(t)$, e.g., can be the size of the largest connected component. It is worth noting that in the subcritical regime, i.e., $t < t_c$, it is almost impossible to have $c(i) = c(j)$. In other words, for $t < t_c$, the size of the largest connected component is of order logarithmic in n, which means $|c(i)|^2/n^2 \to 0$ when $n \to \infty$. In Eq. (2.34), both $c(i) = c(j)$ and $c(i) \neq c(j)$ are given for the purpose of an example. But for the rest of the thesis, one should know that it does not matter whether we have the term $c(i) = c(j)$ in the subcritical regime that we mainly focus on.

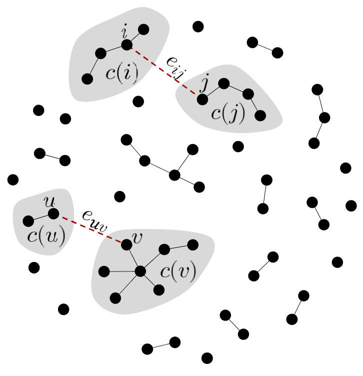
2.5.3 Product rule, sum rule, and explosive percolation

Ref. [63] proves that all bounded-size rules, i.e., Eq. (2.34), result in a continuous percolation phase transition. The first unbounded-size rule was introduced by Achlioptas et al. in ref. [58] where they call it the product rule (ProR). The ProR also considers two randomly selected edges at each step t but chooses the one with smaller product of the corresponding components (see Fig. 2.5a as an example). In the manner of Algorithm 2.1, we only need to redefine the goal function $\xi(e_{ij})$ compared to the BF strategy,

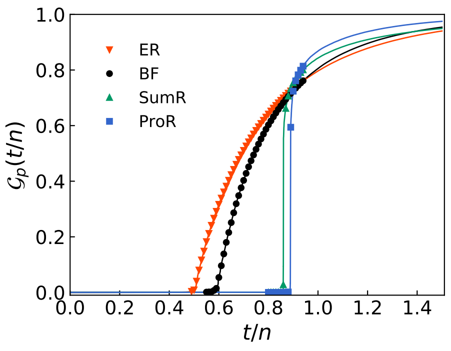
$$\xi(e_{ij}) = |c(i)||c(j)|. \tag{2.35}$$

Note again that it does not matter so much whether $c(i) = c(j)$ for the subcritical regime. Instead, the sum rule (SumR) accepts the edge having a smaller sum of the two associated components, i.e.,

$$\xi(e_{ij}) = |c(i)| + |c(j)|. \tag{2.36}$$



(a)



(b)

Figure 2.5: An example of the ProR. (a) At each step t, two candidates (edges) are selected, but only the one with a smaller product of the corresponding components is added. Here, the considered four components are covered by gray shadows, and the two dashed lines with red color are associated with the two candidates. Even though e_{ij} leads to a connect component with smaller size ($|c(i)| + |c(j)| = 4 + 4 = 8$, compared to e_{uv} having $|c(u)| + |c(v)| = 2 + 7 = 9$), the ProR will choose e_{uv} because it has smaller product, $2 \times 7 = 14$, than e_{ij}, $4 \times 4 = 16$. On the contrary, SumR, however, will choose e_{ij} which has $4 + 4 = 8$ smaller than e_{uv} with 9. Besides, in this network, the LCC is the one that node v belongs to. (b) The evolution of ER, BF, SumR, and ProR on networks with size $n = 10^6$, where $\mathcal{G}_p(t/n)$ is the fraction of the LCC. Note that the number of nodes between two marks along each specific curve is 10^4.

Fig. 2.5b shows the evolution of the classical ER model, BF strategy, SumR, and ProR, where $\mathcal{G}_p(t/n)$ denotes the fraction of the LCC of the network at step t, namely, the fraction of number of nodes in the LCC. Indeed, compared to bounded-size rules[14], SumR and ProR significantly delay the emergence of the giant component. Besides, another notable difference between the bounded-size rule and the unbounded-size rule is that, from the subcritical regime, SumR and ProR almost immediately, i.e., by the aid of a very few edges added, reach the configuration whose LCC has the similar size to the ER model and BF strategy (see the abrupt jump of $\mathcal{G}_p(t/n)$ led by 10^4 edges in Fig. 2.5b).

To analyze that abrupt jump more precisely, the scaling window $\Delta_n(a,b)$ as a function of network size n is defined in refs. [58, 60]. Specifically, during a percolation process, $\Delta_n(a,b)$ is used to count the fraction of the number of new edges from the last moment t_0 where the network has $\mathcal{G}_p(t_0/n) < n^{a-1}$ to the earliest moment t_1 where the network satisfies $\mathcal{G}_p(t_0/n) > b$, i.e.,

$$\Delta_n(a,b) = (t_1 - t_0)/n, \tag{2.37}$$

where a and b are two given parameters. Ref. [58] studied $\Delta_n(a,b)$ regarding ProR under $a = b = 1/2$ which means that how many edges are needed to have a transition of the order parameter from $n^{-1/2}$ to 0.5. However, not like the bounded-size rule, it was really difficult to get the analytical solution for an unbounded-size process. Thus, they considered networks with size up to 2^{26} and numerically got

$$\Delta_n(1/2,1/2) \propto n^{-1/3}, \tag{2.38}$$

[14]ER model could be viewed as a bounded-size rule with constraint $\alpha = 0$ in Eq. (2.34).

which, on the other hand, indicates the existence of the abrupt transition, i.e.,

$$\Delta_n(1/2, 1/2) \to 0 \tag{2.39}$$

when $n \to \infty$, and t_0 and t_1 converges to $t_c \approx 0.888n$. Compared to $\Delta_n(1/2, 1/2)$ of the ER model where they got $\Delta_n(1/2, 1/2)n > 0.193n$ linear to n, they call the transition in ProR process the *explosive percolation*. That means the ER model undergoes a second order transition and ProR undergoes a first order transition. In fact, they stated that other unbounded-size rules also yield a discontinuous transition [58], such as SumR (with $\Delta_n(1/2, 1/2) \propto n^{-0.4}$ [64]).

2.5.4 Other rules

Since then, a lot of related works have emerged, ranging from the consideration of different network models [65, 66] to the development of varied rules [64, 59, 67, 68], further to the proof of the discontinuous transition [69, 70]. The most straightforward rule is to extend SumR and ProR through increasing the number of candidates (n_s-R) [64], namely, increasing n_s in Algorithm 2.1. Ref. [64] also introduces a function $PK(t, \alpha)$ which represents the number of nodes in components with sizes equal or larger than α at time step t, i.e., t edges have been added into the network. Based on this function, they further give $\frac{1}{n}RK(t(n^a), n^{1-b})$ accounting for a fraction of nodes containing in the components with size ranging from n^{1-b} to n^a where $t(n^a)$ is the earliest moment when the network has a component with size n^a. And the set of those components is called *powder keg*. Further, one can easily observe that the number of components in the powder keg is at most n^b. Thus we have $\Delta_n(a, 1/2) \sim n^b$ if the powder keg exists, i.e., $\frac{1}{n}RK(t(n^a), n^{1-b})$ is proportional to the network size. Therefore, they conclude that the explosive percolation is guaranteed by the existence of powder keg, no matter what rules are conducted. This provides us a numerical way to detect which type of percolation is. For example, $RK(t(n^{0.5}), n^{0.4})/n$ approaches 0 in ER network while it converges to some constant under SumR.

Another strategy [69] is called dCDGM which firstly independently selects two groups with $n_s/2$ nodes and then builds an edge between two nodes i and j with the smallest $|c(i)|$ and $|c(j)|$, where i and j comes from the two groups, respectively. We can also achieve this through Algorithm 2.1 by modifying *line 5*,

$$\mathcal{M}_c(t) \leftarrow e_{ij}, \forall i, j \in RS(\mathcal{N}, n_s), i \neq j,$$

which is actually stronger to suppress the giant component than dCDGM. It is worth mentioning that $\mathcal{N}$ is the node set, and $RS(\mathcal{N}, n_s)$ means that randomly selecting n_s nodes from $\mathcal{N}$. The goal function $\xi(e_{ij})$ is the same as SumR, and $n_s = 4$ is conducted in ref. [69]. In addition, one interesting conclusion given by ref. [69] is that dCDGM (with $n_s = 4$) only leads to a continuous transition, which indicates that SumR is also continuous. Actually, a rigorous proof from ref. [70] shows that all n_s-R and dCDGM processes (i.e., AP) on random networks are continuous if n_s is fixed, but a discontinuous transitions can be observed if n_s is proportion to n, namely, $n_s \to \infty$ when $n \to \infty$.

If the process relies on the BFW model [71, 72], then one can also get a discontinuous transition. The BFW model achieves this through the suppression of the LCC. Specifically, it gradually increases a boundary regarding the LCC, and a new edge can only be added if it meets the boundary (details see Algorithm A.1). In this way, multiple large components appear simultaneously and then are merged together by a few edges so that a strongly discontinuous percolation transition is observed [72].

Another family are probability-based rules [73, 74, 59]. In the cluster aggregation model [73], an edge e_{ij} is accepted as a new edge with probability $(|c(i)||c(j)|)^{\alpha} / \sum_{\forall i,j}(|c(i)||c(j)|)^{\alpha}$, where they showed that $\alpha > 0.5$ corresponds to a continuous transition and $\alpha < 0.5$ a discontinuous transition. Another case is to control the LCC by adding a new edge if it does not increase the size of the LCC, otherwise with a probability from a Gaussian distribution,

$$\min\{1, \mathrm{e}^{-\alpha(\frac{|c(i,\mathrm{add})|-\langle |c| \rangle}{\langle |c| \rangle})^2}\}, \tag{2.40}$$

in which $|c(i, \mathrm{add})|$ represents the size of the LCC after the chosen candidate i is added, and $\langle |c| \rangle$ is the average size of all components.

3 Ways to Fragment Networks

This chapter is mainly based on results in our previously published papers [**L1**] and [**L2**].

3.1 Problems and Motivations

3.1.1 Inverse percolation and network robustness

As we mentioned in Section 1.1, errors and failures are everywhere in the human world, from a tiny chip to the entire climate system. Besides, From Section 2.1, we also learned that many real-world systems can directly or indirectly be modeled into networks, such as World Wide Web, social network, brain networks, and climate networks. In the terminology of network science, the failure of a component can be associated with the failure of a node. In networks, the failure of a single node would always have only limited damage to a network's function. The failure of several nodes (i.e., collective influence [23]), however, can bring the network to the brink of collapse. Basically, the more nodes fail, the higher probability would the network collapse, in tandem with the fact that the removal of different nodes might damage the network in different magnitude, motivating us to ask: how many or which group of nodes do we need to target to turn the network into independent components? That is, for example, what fraction of proteins should we remove from the protein-protein interaction network to fragment the network into isolated small groups so that the network would lose its function? Further, given the global airline network, which group of airports should we choose to consider more intensively so that a particular airport can be blocked from the majority. Both of those two examples, from the network science perspective, are related to the network robustness problem which this section focuses on.

3.1.1.1 Inverse percolation

For the percolation transition on the two-dimensional square lattice, one can also observe the similar behavior of the order parameter through repeated removal of nodes from a full occupied lattice. In other words, assuming that all sites are occupied in the beginning, we then check every site and remove them from the lattice with an independent probability p'. In this manner, at some value of p', one cannot find any path, which only relies on the remaining sites, from one side of the lattice to the other side. Compared to the percolation process, we can easily know that their order parameters are exactly consistent with each other at $p = 1 - p'$. Thus, this process is also called inverse percolation [2].

One can also conduct the inverse process to get an ER random network. That is: starting with a fully connected network, each edge has a probability $p' = 1 - p$ to be removed from the network. It is worth mentioning that only the classic ER model is reversible among all those processes in Sections 2.4 and 2.5. But it does not matter so much since we are mainly interested in existing networks. More details will be discussed later.

3.1.1.2 Network robustness

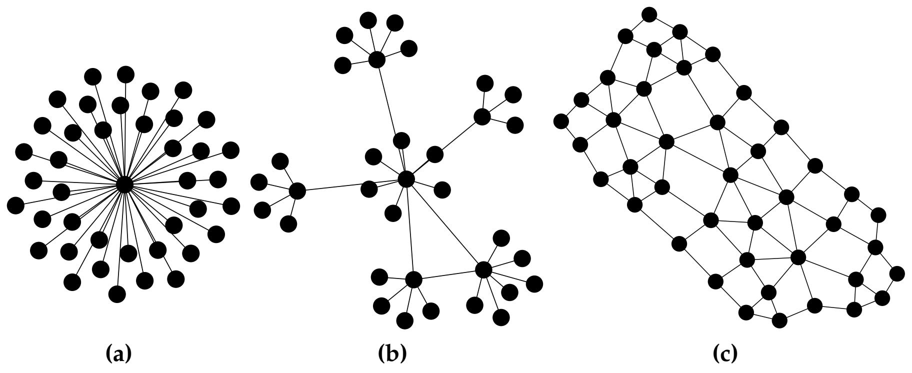


Figure 3.1: Baran's example [17]. (a) Centralized – star network. (b) Decentralized – scale-free network. (c) Distributed – grid network.

Given Baran's prototypes [17] as an example (see Fig. 3.1), obviously, the star network is extremely fragile under an intentional attack on the central node[1]. However, it is also quite robust against a random attack: the attack on leaf nodes only leads to a very limited damage, and the probability that the attack happens on the central node is $1/n$. The scale-free network can ease the damage from an intentional attack, but it would still collapse under an attack on a few nodes simultaneously. The grid network, on the contrary, is robust to both random and intentional attacks. However, compared to the star network and the scale-free network, it has a finite threshold under random attack. In what follows, we are going to focus on how a network response to those general attacks.

3.1.1.3 Overview of attack strategies

The meaning of the study on different attack strategies is two-fold. On the one hand, from the perspective of attack, an efficient attack strategy could possibly help us develop more efficient methods to, for instance, disrupt the functioning of criminal or corrupt organizations [77], or offer avenues to design drugs that kill unwanted bacteria [2]. On the

[1]In this thesis, we refer to this problem as the term robustness from refs. [9, 5] instead of survivability or reliability in ref. [17].

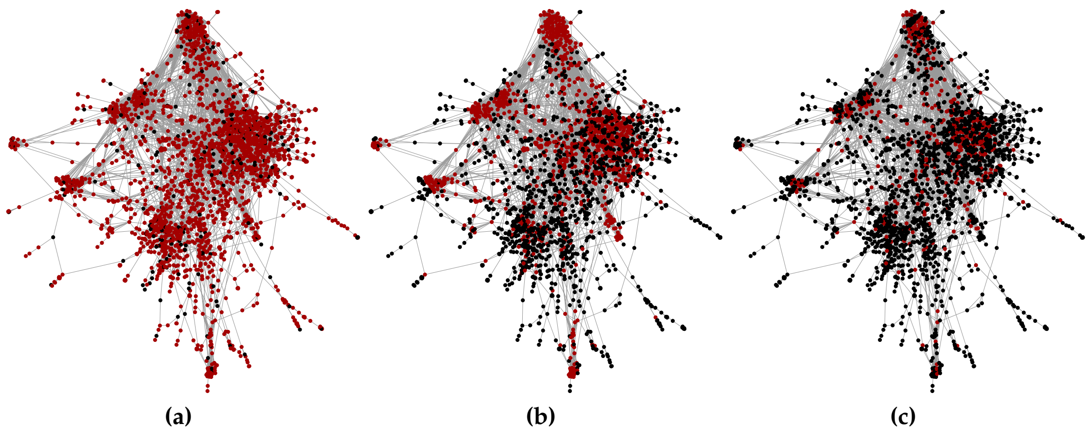


Figure 3.2: Illustrations of different attack strategies on a protein-protein interaction network with $n = 2375$ and $m = 11693$ [75, 76]. The nodes colored red are those to be removed, and the black are the remaining. After the removal of red nodes, the remaining network would only have components of size less than $1\%n$. (a) The random strategy needs to remove 90.44% nodes. (b) A basic strategy based on the degree centrality needs 43.07%. And (c) a heuristic strategy only needs 16.08%.

one hand, from the preventing side, an efficient attack strategy might provide us new insight into the solution to prevent, e.g., the spread of misinformation which has become one of the top threats to our society, or the prevalence of an epidemic [11], especially climate change exacerbates the spread of diseases [30, 31, 32].

One of the simplest attack strategies is random failure, which is widely existing in real-world systems, such as the breakdown of a router or the cancellation of a flight. Assuming that there is a distinct sequence S each of whose item is associated with a unique node in a network $G(\mathcal{N}, \mathcal{M})$, the *random strategy* (RanS) removes nodes[2] one by one from G following the order from a random permutation of S. Recall that a distinct sequence means that its elements are distinct, i.e., repetition is not allowed. If a certain demand is given, e.g., remove part of nodes to let the remaining network without a component whose size is larger than $1\%n$, then RanS coincides with an inverse site percolation to remove each node with an independent probability p.

RanS also belongs to the local strategy family. Another local strategy we would like to introduce is the acquaintance immunization (AcqI)[3] [11, 6]. AcqI firstly randomly selects a group of nodes and then remove one of their corresponding nearest neighbors. In this way, AcqI can efficiently target hubs (those nodes with a large degree in a network) without having to know precisely which individuals are hubs [11]. Considering this, a group [6] recently shows that social network fragmentation might be a good strategy to target individuals for medical treatment in low-incoming counties.

But basically, if we know the network topology information, targeted methods are always much more efficient than the local ones. For example, the scale-free network is quite fragile under intentional attack, even though its critical threshold approaches 1 against

[2]If there is no special explanation, the removal of a node means the removal of both the node and the incidental edges.

[3]A strategy is mainly designed to immunize a network. But attacking and immunizing a network are really two sides of the same coin. Thus here we also refer AcqI as an attack strategy. More details regarding immunization will be discussed later.

random failure. The basic targeted methods aim to measure the importance of nodes firstly and then based on which attack or remove the most influential ones. For example, one can identify a node based on its degree centrality, eigenvalue centrality, betweenness centrality, and so forth (see Section 2.2 for details).

More powerful strategies are mainly based on the heuristic idea (see comparisons in Fig. 3.2). One category is to design heuristic methods based on those basic strategies, like to repeatedly remove the node with the largest degree from the remaining network after the removal of the largest-degree node. More recently advanced strategies including the collective influence method [23], the explosive immunization strategy [24], and the decycling-based methods [25, 26] will be discussed in detail later.

3.1.1.4 Scale-free network

A network is said to be a scale-free (SF) network if its degree distribution follows a power-law distribution, i.e.,

$$p_k \sim k^{-\gamma}, \tag{3.1}$$

where γ is a controlling variable. One of such models to construct a network which has a power-law degree distribution is the Barabási-Albert (BA) model [18]. Given n and α, a BA network can be obtained through: i) generate a small initial network $G(\mathcal{N}, \mathcal{M})$ of size larger than α; ii) add a new node i and let $\mathcal{N} = \mathcal{N} \cup \{i\}$; iii) choose a node $j \in \mathcal{N} \setminus \{i\}$ with probability $\frac{k_j}{\sum_{u \in \mathcal{N} \setminus \{i\}} k_u}$, and then connect i and j, i.e., $\mathcal{M} = \mathcal{M} \cup \{e_{ij}\}$; iv) add other $\alpha - 1$ edges in the way of iii) (multi-edges are not allowed); v) repeat ii), iii), and iv) until $|\mathcal{N}| = n$. Note that the scale-free network from the BA model has an exponent $\gamma \approx 3$ [18].

3.1.1.5 Configuration model

Different from the ER model generating a network whose degree distribution strictly follows the Poisson distribution, the configuration model allows us to generate a random network with any degree distribution that we give beforehand [78]. Specifically, assuming that we have a degree sequence in which k_i corresponds to the degree of node i regarding a network $G(\mathcal{N}, \mathcal{M} = \emptyset)$, the configuration model constructs $\mathcal{M}$ through: i) uniformly choose a node i from node set $\{u | k_u > 0, \forall u \in \mathcal{N}\}$ and then let $k_i = k_i - 1$; ii) choose another node j following i), and then build a new edge e_{ij}, further let $\mathcal{M} = \mathcal{M} \cup \{e_{ij}\}$; iii) repeat i) and ii) until $k_i = 0, \forall i \in \mathcal{N}$. Since an edge relies on two nodes, the sum of the given degree sequence has to be even. In addition, self-loops and multi-edges are allowed in the configuration model. But it does not matter so much because their probability approaches 0 when $n \to \infty$ [1].

By the aid of the configuration model, we can study networks that have arbitrary degree distributions, such as networks following scale-free distribution with different exponents (see Eq. (3.1)). We can also construct a network based on the degree sequence from a real-world network, and then compare the difference between them. In addition, as each edge is built uniformly, the probability that there is an edge between node i and node j is equal to $2|\mathcal{M}|/n^2$, which is the fundamental for the later derivations.

3.1.1.6 Attacks on the configuration model network

We first consider the random attack on networks generated through the configuration model, i.e., remove nodes following RanS. Following Section 2.5.1, we further define

$$\begin{aligned}
&\mathcal{N}_c(t)\text{: } \textit{candidate node set},\\
&\mathcal{N}_o(t)\text{: } \textit{occupied (remaining) node set}, \text{ and}\\
&\mathcal{N}_u(t)\text{: } \textit{unoccupied (removed) node set},
\end{aligned}$$

where t is the number of occupied nodes[4], namely $t = |\mathcal{N}_o(t)|$. Apparently, $\mathcal{N}_o(t) \cup \mathcal{N}_u(t) = \mathcal{N}$ and $\mathcal{N}_o(t) \cap \mathcal{N}_u(t) = \varnothing$ hold. Since the removal of nodes indicates that both the nodes and incident edges are removed from the network, we have

$$\begin{aligned}
&\mathcal{M}_o(t) = \{e_{ij} | \forall i, j \in \mathcal{N}_o(t)\},\\
&\mathcal{M}_u(t) = \{e_{ij} | \forall i \in \mathcal{N}_u(t) \text{ or } \forall j \in \mathcal{N}_u(t)\} \text{ or } \mathcal{M} \setminus \mathcal{M}_o(t).
\end{aligned}$$

Further, letting

$$q = \frac{n-t}{n} \tag{3.2}$$

be the fraction of removed nodes and

$$\mathcal{G}_a(q) = \frac{\text{\#nodes in the LCC of } G(\mathcal{N}_o(t), \mathcal{M}_o(t))}{n} \tag{3.3}$$

the fraction of the LCC, then we can describe an attack process in a percolation way. That is,

$$\mathcal{G}_a(q) \equiv \mathcal{G}_p(t/n), \tag{3.4}$$

recalling that $\mathcal{G}_p(t/n)$ is the ratio of nodes in the LCC at step t. It is also worth mentioning the differences between the LCC and the giant component p_∞. The giant component is a component of size proportional to the network size n so that it only exists in a part of a process, while LCC exists for the whole process. In addition, the percolation process associated with an attack process is always on a finite network, which makes it no sense to talk about the giant component. In other words, it is unreasonable to scale p_∞ by a fixed n. Due to this, the LCC is usually viewed as an approximate representation of the giant component [2, 1], e.g., $\mathcal{G}_a(q) < 0.01$ corresponds to $p_\infty = 0$, when we talk about real-world networks which are always finite.

The Molloy-Reed criterion [78] gives the condition that networks generated through the configuration model with high probability have a giant component if (see Appendix A.2.1 for details)

$$\frac{\langle k^2 \rangle}{\langle k \rangle} > 2, \tag{3.5}$$

[4]Throughout this thesis, t corresponds to the number of nodes in a site percolation and the number of edges in a bond percolation.

which can help us acquire the critical threshold q_c [19, 79, 2] of a network under the random attack,

$$q_c = 1 - \frac{1}{\frac{\langle k^2 \rangle}{\langle k \rangle} - 1}, \tag{3.6}$$

where q_c represents the least fraction of nodes whose removal would result in a remaining network almost surely without a giant component (Appendix A.2.2).

3.1.1.7 Attacks on ER network

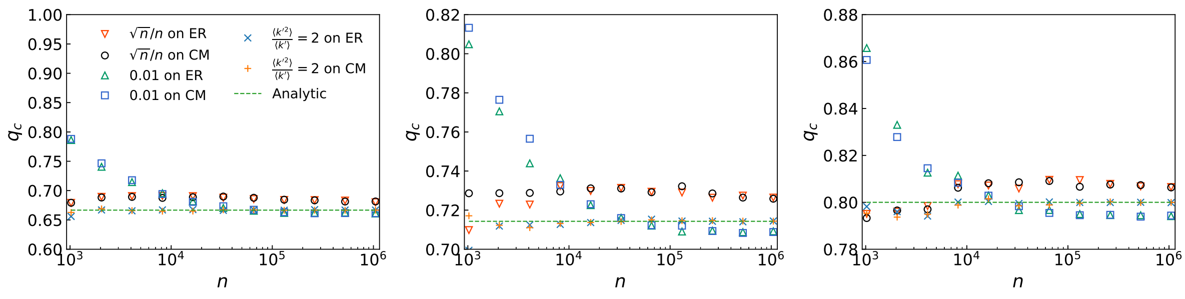


Figure 3.3: The critical threshold q_c as a function of network size n under the random attack on ER networks and the associated configuration model networks (CM). The dashed line corresponds to the analytic solution (Eq. (3.7)) regarding different average degree (a) $\langle k \rangle = 3.0$, (b) $\langle k \rangle = 3.5$, and (c) $\langle k \rangle = 5.0$.

Since an ER network follows the Poisson distribution, we have the second moment $\langle k^2 \rangle = \langle k \rangle (\langle k \rangle + 1)$. Substituting it into Eq. (3.5), one can get the condition $\langle k \rangle > 1$ for the existence of a giant component, which coincides with the previous result in Section 2.4.2.3. In addition, because a network generated through the ER model and a one through the configuration model with a Poisson distribution sequence are quite similar to each other [1] (also see Fig. 3.3), one can also easily obtain the critical threshold of an ER network under the random attack based on the result from Eq. (3.6) as

$$q_c = 1 - \frac{1}{\langle k \rangle}, \tag{3.7}$$

which indicates that the denser a network is, the more robust would it be against random failure (see Fig. 3.3). Besides, since $\langle k \rangle$ is some constant, an ER network usually has a finite critical threshold.

For the critical threshold of ER networks under the intentional attack on hubs, it has a similar behavior as the one under random failure, i.e., q_c can be obtained at $\langle k'^2 \rangle / \langle k' \rangle = 2$ (see also Fig. A.2a in Appendix A.2.3), because each node expectedly has the same degree [2, 9].

3.1.1.8 Critical threshold regarding order parameter

Fig. 3.3 shows the results of the critical threshold q_c as a function of the network size n with respect to the random attack on networks accordingly constructed through the ER model (with $G(n, m)$) and the configuration model. Specifically, given n and m, an ER network G is firstly generated. Then, a random attack process is conducted on G, during which we trace $\langle k'^2 \rangle / \langle k' \rangle$ of the corresponding subnetwork G' and obtain the threshold q_c at $\langle k'^2 \rangle / \langle k' \rangle = 2$. This attack and tracing process is also conducted on a configuration model network building upon the degree sequence same as the one of G. Each data point in Fig. 3.3 is the average of 50 independent simulations.

As we can see from Fig. 3.3, both q_c of networks generated through the ER model and the configuration model are subject to the analytic solution given by Eq. (3.7), in particular when the network size n is large. However, the Molloy-Reed criterion only applies for randomly wired networks[5].

For a network under an intentional attack, this criterion would become invalid in most cases (see Fig. A.2b in Appendix A.2.3 where we improve the robustness of an ER network against the intentional attack under the constraint of keeping the degree sequence unchanged). Besides, recall that our main goal is to trace the order parameter and obtain q_c at where the giant component disappears. Hence, here we choose to approximate the critical threshold q_c through a given parameter α,

$$q_c = \min q, \text{ s.t. } \mathcal{G}_a(q) < \alpha, \forall q. \tag{3.8}$$

Throughout this thesis, if there is no specific explanation, we employ $\alpha = 0.01$ suggested by ref. [2] to obtain q_c, which assumes that a giant component is a connected component of size at least $0.01n$. More intuitively, Fig. 3.3 shows the comparisons of q_c under varied metrics, $\alpha = 0.01$, $\alpha = \sqrt{n}/n$, and the Molloy-Reed criterion. As we can see from there, if n is small, q_c under $\alpha = 0.01$ is actually much worse than the one under $\alpha = \sqrt{n}/n$ to approximate the analytic solution. But it is better when n is large. Besides, the results regarding different average degree (see Fig. 3.3b and 3.3c) suggests that we cannot use some constant to scale q_c under $\alpha = \sqrt{n}/n$. And it is meaningless to talk about a giant component when networks are quite small. Moreover, most real-world networks we considered in this thesis have a size of $n > 10^4$. Meanwhile, since q_c would be as a criterion to quantify different attack strategies, we actually only need to know the basic trend of q_c over those compared strategies rather than the exact moment where the giant component vanishes.

3.1.1.9 Attacks on scale-free network

For a network constructed through the configuration model based on a power-law-distribution degree sequence, one can easily verify that $\langle k \rangle = \sum_k^\infty k p_k \sim \sum_k^\infty k^{1-\gamma}$ diverges when $\gamma \leqslant 2$ (p-seires). Similarly, $\langle k^2 \rangle \to \infty$ if $\gamma \leqslant 3$. In reality, most networks have γ between 2 and 3 [1], which means that $\langle k \rangle$ is finite and $\langle k^2 \rangle$ is infinite. Putting them into Eq. (3.6), we have $q_c \to 1$ for $2 < \gamma \leqslant 3$. In other words, not like a Poisson distribution network, most real-world networks have infinite critical threshold, and a random attack has to remove almost all nodes to break down a scale-free network if $n \to \infty$ (see Appendix A.2.4 for more

[5]We also tested: for a given random network G, we keep its degree sequence fixed and increase or decrease its assortativity (see Eq. (2.26), and method to tune them in Section 4.1) or robustness against the intentional attack on hubs. In those cases, q_c of those networks keeps unchanged under random attack, which arises as to another question: how to improve or weaken the network robustness against random failures if we would like to keep the degree sequence same.

details). For the intentional attack on hubs [20], the critical threshold q_c can be obtained through numerically solving

$$q_c^{(2-\gamma)/(1-\gamma)} - 2 = \frac{2-\gamma}{3-\gamma} k_{\min}(q_c^{(3-\gamma)/(1-\gamma)} - 1). \tag{3.9}$$

An example can be found in Fig. A.3 in Appendix A.2.4.

3.1.2 The connection between network robustness and immunization

3.1.2.1 Susceptible-Infected-Recovered model

The Susceptible-Infected-Recovered (SIR) model was originally introduced by Kermack and McKendrick [80]. In the basic SIR model, each individual belongs to one of the following three states: the susceptible (S) state, the infected (I) state, and the recovered or removed (R) state. A susceptible individual can be infected by those infected ones and then become infectious. Meanwhile, an infected individual could recover or die (be removed). And if that happens, then it turns into the recovered state, which means that it cannot transmit or get infected by a pathogen again. One can find the transition chain shown in Fig. 3.4.

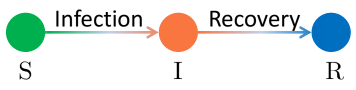


Figure 3.4: The transition chain of the SIR model. Individuals in S state can be infected and then turn into I. At the same time, infected individuals (I) would recover and then change into R.

For individuals in a fully mixed population of size n, i.e., each individual could reach all others, they undergo two phases of transition at each time step t. Assuming that $\mathrm{S}(t)$, $\mathrm{I}(t)$ and $\mathrm{R}(t)$ accordingly represent the fraction of susceptible, infected and recovered individuals at t, we then have

$$\mathrm{S}(t) + \mathrm{I}(t) + \mathrm{R}(t) = 1. \tag{3.10}$$

Further, let a be the number of individuals that per person can contact, and η_i be the probability that an infected one can successfully transmit its pathogen, that is, the number of individuals that per infectious individual can really contact and infect is $\eta_i a$. It is worth mentioning that $\eta_i a$ is viewed as a given constant in the classic SIR model. Following them, the basic reproductive number η_0 is defined as (Appendix A.2.5)

$$\eta_0 = \frac{\eta_i a}{\eta_r} > \frac{1}{\mathrm{S}(t)}. \tag{3.11}$$

At the beginning of an epidemic, $\mathrm{S}(0) \approx 1$ usually holds. Thus, an epidemic can outbreak only if $\eta_0 > 1$. An example is shown in Fig. A.4 in Appendix A.2.5. Note that η_i and η_r could be estimated based on some real data.

3.1.2.2 From SIR to percolation

The classic SIR model ignores the true connections among individuals, i.e., the assumption of a. In practice, the spread of an epidemic usually contains a lot of local patterns, that is, an individual is likely to transmit the virus to its acquaintances. In the term of network science, where a node could represent an individual and edges characterize its contacts with others, such transmission means an infectious node can only infect its nearest neighbors. For this case, a is actually the average degree $\langle k \rangle$. Thus, we have the spread of an epidemic on networks, and the classic SIR model can be thought of as a spread on a fully connected network.

Different from the classic SIR model, nodes in a network might have different degrees, which means that they possibly have different spreading capacities, or on the other hand, the different probability of getting infected. Thus, a question arises as to in what condition an epidemic can outbreak regarding a given network. Similar to the network robustness, we only have analytical solution on networks generated through the configuration model [2, 81, 82], which follows

$$\eta_c = \frac{\eta_i}{\eta_r} = \frac{1}{\frac{\langle k^2 \rangle}{\langle k \rangle} - 1}, \tag{3.12}$$

where η_c is the corresponding epidemic threshold (details refer to Appendix A.2.6).

In the above process, an infected node keeps transmitting a virus to its nearest neighbors with independent probability η_i until it recovers. Now focusing on one of its neighbors and considering that process as a Poisson process with mean η_i/η_r [81, 2], then during the period of time 0 to time t_r (the time that the infected node takes to recover), the probability that the virus is not successfully passed is $e^{-\eta_i t_r}$. On the contrary, the probability that the virus is successfully passed is $1 - e^{-\eta_i t_r}$. Refs. [4, 2, 1] show that this process actually coincides with a bond percolation on the same network G with an occupied probability $1 - e^{-\eta_i t_r}$ (see also Appendix A.2.6).

3.1.2.3 Network immunization

The outbreak of an epidemic, especially a global one, always inflicts a heavy toll and huge losses on all aspects of our lives [83, 84, 85, 86], in particular nowadays climate change exacerbates the spread of disease [32]. Ways to contain an epidemic might include social distancing, cancellation of airlines, quarantine, and closure of shops or public areas, etc. And the aim is to curb the basic reproductive number (see Eq. (3.11)) if the epidemic follows the SIR model (which most epidemics follow). Here we consider such problem on networks, i.e., network immunization.

To immune a network is actually equivalent to attack a network, i.e., network robustness (see also Section 3.1.1). Intuitively, after the removal of part nodes (corresponding to vaccination or isolation of those nodes), the possible maximum spread is bounded by the size of the LCC. That means, if there is no giant component in the remaining network, then the virus would die out within only a limited spread, no matter how large the infection rate η_i or how small the recovery rate η_r is. Thus, the random immunization (or herd immunity) corresponds to the random attack, which asks us to remove many enough nodes, i.e., at least q_c (see Eq. (3.6)), to eliminate a virus.

Now considering the way that we obtain Eq. (A.44), we can actually get a similar result if the occupation of an edge is replaced with the occupation of a node. Specifically, the probability α that a node i connect to the giant component through a particular neighbor node j also consists of two parts: i) the probability that j is removed from the network, i.e., q;

ii) the probability that node j does not belong to the giant component if j is not removed, which follows $(1-q)\alpha^{k_j-1}$. Then, we reach a formula as

$$\alpha = q + (1-q)\sum_k \frac{kp_k\alpha^{k-1}}{\langle k\rangle}. \tag{3.13}$$

Again, following ref. [1], one can get the critical threshold q_c which is exactly same as Eq. (3.6). Thus, we can use those conclusions from Section 3.1.1 to measure the behavior of the epidemic threshold η_c against immunization.

For instance, suppose that there is a random network G constructed through the configuration model, and q fraction of nodes are randomly removed from it. Here for simplification, we further assume that the removed nodes are still in the network, but the associated edges are removed (or those nodes become immune regarding the SIR model). According to Eqs. (A.19) and (A.20), one can get

$$\frac{\langle k'^2\rangle}{\langle k'\rangle} = \frac{\langle k^2\rangle}{\langle k\rangle} + (1-\frac{\langle k^2\rangle}{\langle k\rangle})q \tag{3.14}$$

of the remaining network. Supposing that $\langle k^2\rangle/\langle k\rangle$ is a constant and larger than 1 (this is the case we are interested in), then $\langle k'^2\rangle/\langle k'\rangle$ is a monotonically decreasing function of q. Adding it into Eq. (3.12), one can easily conclude that η_c' would increase as the rise of q. In other words, if η_i and η_r keep unchanged, the virus would die out once $\eta_c' > \eta_i/\eta_r$, i.e., at some point of q. For example, regarding a network whose degree sequence following the Poisson distribution, $\langle k'^2\rangle/\langle k'\rangle = (1-q)\langle k\rangle + 1$ means that we can always find some q less than 1 to have $\eta_c' > \eta_i/\eta_r$ since $\langle k\rangle$ is a constant (see Fig. 3.5).

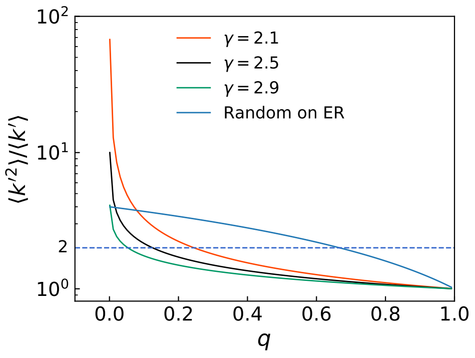


Figure 3.5: $\langle k'^2\rangle/\langle k'\rangle$ as a function of q for different γ regarding Eq. (A.32) with $k_{\min} = 1$. The dashed line corresponds to $\langle k'^2\rangle/\langle k'\rangle = 2$. Random on ER is obtained through $\langle k'^2\rangle/\langle k'\rangle = (1-q)\langle k\rangle + 1$ where $\langle k\rangle = 3.0$.

From Section 3.1.1.9 we know that scale-free networks are usually robust against random failure but quite fragile under intentional attacks, in particular for $2 < \gamma < 3$ that most real-world networks have. For a scale-free network G with $2 < \gamma < 3$, $\langle k^2\rangle/\langle k\rangle \to \infty$ indicates that $\eta_c \to 0$ (see Eqs. (A.24) (A.24) and (3.12)). That is, any kind of epidemics could outbreak on G, no matter how small η_i or how large η_r is. Rewriting Eq. (3.14) as

$$\frac{\langle k'^2\rangle}{\langle k'\rangle} = \frac{\langle k^2\rangle}{\langle k\rangle}(1-q) + q \tag{3.15}$$

and considering random immunization, obviously we need to remove almost all nodes to possibly have finite $\langle k'^2 \rangle / \langle k' \rangle$. For an intentional removal on hubs, according to Eq. (A.32) (also see Fig. 3.5), we know that $\langle k'^2 \rangle / \langle k' \rangle$ would be finite. Therefore, as the random immunization on ER networks (note that again here we view a network whose degree sequence following the Poisson distribution as an ER network), we can also find some q less than 1 to eliminate an epidemic from a scale-free network through a targeted immunization on hubs (i.e., intentional removal on hubs). Besides, Fig. 3.5 also tells us that η_c on scale-free networks increases much faster than the one on ER networks as q increases.

3.2 Metrics to Methods

As discussed in the previous sections, the removal of nodes (including the incident edges) would change the network structure. Meanwhile, different strategies would bring about different magnitude of influences on the network structure, like the differences resulted in by random and targeted removals. Further, for a specific strategy, networks constructed through different models would have different reactions to an attack. The ER network has a finite critical threshold q_c under the random failure, but it is also robust against an intentional attack. The scale-free network[6] generally has a very large q_c (approach 1) to the random failure; however, it is also quite fragile if attacks occur on its hubs. Indeed, how robust a network is depends on both the network structure and which attack strategy that we choose. But usually, the main problem that we face in reality is to protect an existing network rather than building a new one. Once considering a specific network, the influence of the degree distribution becomes useless. Further, for example, even though two networks have the same degree distribution, they might have totally different capabilities to survive from an attack (see Fig. A.2b in Appendix A.2.3). Hence, thereafter, assuming that a network is given, we mainly focus on different attack strategies to: i) unveil the true robustness that the studied network has; ii) provide some ideas about how to design a more effective way to prevent the outbreak of an epidemic; and iii) further enhance the network's robustness. But first, we should decide what criteria that we are going to employ for the fair comparison with existing methods.

In addition to the LCC [9, 79, 25, 26, 23, 87, 1], there are also other ways to qualify a network's resilience or robustness, such as the network diameter $d_{\max}$ (see Eq. (2.8)) or the average shortest path (see Eq. (2.9)) [9], the clustering coefficient (see Eq. (2.12)) [42], the assortativity (see Eq. (2.26)) [49], the redundancy level[7] [17], and the network motifs [8] [88, 89]. For a network under an attack, the average shortest path of the LCC (see also Eq. (2.28)) would firstly increase, which is due to the removal of redundant edges, and then decrease, since it becomes smaller and smaller, after the giant component vanishes as the rise of q (the

[6]We mainly consider $2 < \gamma < 3$ that most real-world networks hold.

[7]For a given network, the redundancy level one corresponds to the minimum spanning tree. If the number of edges in the network is twice as many as that of level one, then the network has the redundancy level two. Three times correspond to level three and so forth. To some extent, this measurement is actually equivalent to the average degree $\langle k \rangle$.

[8]For a given number of nodes from a network, one might connect them through a different way. Enumerating them, then each configuration corresponds to a motif. The aim is usually to count the frequency of a specific motif.

fraction of removed nodes). In other words, the change of the average shortest path can be represented by the size of the LCC. Besides, the clustering coefficient, the assortativity, the redundancy level, the network motifs, and others [87, 12] have the similar behaviors and partly or fully rely on the LCC (see Fig. A.5 in Appendix A.2.7 as an example). Hence, we here choose to still consider the most widely used metrics, i.e., the LCC.

Specifically, for a given network G, regarding $\mathcal{G}_a(q)$ (see Eq. (3.3)), on the one hand, we will verify a strategy through the performance of $\mathcal{G}_a(q)$ as a function of q. Basically, for a fixed q, a method is said to be more effective than another one if it has smaller $\mathcal{G}_a(q)$, that is, it is more capable of finding the fatal part of G or of isolating G. The other aspect of this verification is to compare the number of removed nodes for a certain $\mathcal{G}_a(q)$, which corresponds to achieve the same goal by using fewer resources, like targeting individuals for medical treatment or collapsing a criminal organization [6]. But sometimes one method works better at some q but worst at some other q. Thus, on the other hand, we also consider the whole order parameter, that is, F.

In short, we have two criteria to demonstrate the effectiveness of a peculiar method. The first one is the critical threshold q_c (see Eq. (3.8)). The second one is the average fraction of the LCC, F, which characterizes the performance of a method responding to all q,

$$F = \frac{1}{n}\sum_{q}\mathcal{G}_a(q). \tag{3.16}$$

One might find examples regarding F, such as gradually conduce a measure to curb a spread[9] or improve a network's robustness step by step. For those cases, we need to consider all q aiming at a global optimization, i.e., F.

3.3 State-of-the-art Approaches

Over the past decades, a lot of methods have already been developed to tackle the robustness or immunization problem[10] [79, 2, 87, 12, 1]. Here we give a brief review regarding part of them. And some of them have actually been introduced in previous sections (mainly see Sections 2.2 and 3.1.1.3).

3.3.1 General methods

The attack on hubs is also referred to as a strategy based on the degree centrality (see Section 2.2.1). That is, one firstly ranks the nodes in descending order based on their degree centrality, and then remove them one by one following that order. Thus, in a similar way, we can rank nodes based on the Eigenvector centrality (EigS), the Katz centrality (KatS), the PageRank (PagS), the Closeness centrality (CloS) and the Betweenness centrality (BetS) (see the related definitions in Section 2.2). Indeed, like HubS, these methods are also based on the idea that a node should be attacked or immunized if it has a large score, even though they calculate the score in different ways. For example, EigS, KatS, and PagS measure a node by repeatedly gaining information from its neighbors while CloS and BetS count on

[9]Note that spread is not limited to disease but also related to information or rumor.

[10]As we mentioned, they are equivalent to each other.

the shortest path. Another similar method that we would like to mention is the K-shell strategy (KshS) [90], which has led to another area of studies. Instead of only considering the degree of a node itself, KshS thinks that the location of that node is also very important. That is, hubs in a hubs cluster are more influential than those surrounded by small-degree nodes. Specifically, starting with the minimum degree $k_{\min}$ of a given network G and letting $a = k_{\min}$, i) iteratively remove nodes from G until the minimum degree $k'_{\min}$ of the remaining network is larger than a; ii) score those removed nodes by a; iii) considering the remaining network and letting $a = k'_{\min}$, repeat i) and ii) until all nodes are removed and scored. As ref. [90] indicates, a node with a larger a is more capable of spreading an epidemic or information. Thus, on the contrary, that kind of node should also be important for the containment of an epidemic.

Comparisons among these methods can be found in Appendix A.2.8.

3.3.2 Heuristic methods

No matter which methods, they have to identify nodes based on the network structure. But usually, the network structure would change once some nodes are removed. In other words, for example, after the node with the highest degree is removed from the given network, its nearest neighbors' degrees would decrease since the disappearance of its associated edges. For this case, some other node which has smaller degree than those neighbors in the original network possibly becomes a hub in the remaining network (see Fig. A.7), in particular for the real-world network where there are always a number of communities, and degrees among nodes have rich correlations (see Section 2.3.1). Thus, an intuitive way to improve those methods that we studied in the previous section is to recalculate the score of each node again on the remaining network after the removal of the one with the highest score. That is, starting with a given network G and letting G' is a subnetwork consisting of all nodes and edges in the LCC, we then conduct the following adaptive processes to obtain a new sequence based on a specifical method: i) score each node in G' based on the considering method; ii) remove the one with the largest score and obtain G' of the remaining network; iii) repeat the previous steps until all nodes are removed; and iv) obtain the associated sequence according to the order of removal. To distinguish from the original one, here 'A' is employed to mark the 'adaptive' new sequence, e.g., AHubS means that the new sequence is obtained based on HubS.

Comparisons among these methods refer to Appendix A.2.9.

3.3.3 Decycling-based methods

Decycling-based methods, the belief propagation-guided decimation (ABPDS) [25] and the min-sum and reverse-greedy strategy (AMSRGS) [26], are developed based on the observation on general model networks (such as the ER random network) where there are only limited number of cycles (see Sections 2.1.3.3 and 2.4). Thus, they think that one might optimize the critical threshold q_c by the aid of tackling the feedback vertex set (FVS) problem (i.e., the decycling problem in ref. [26][11]). Specifically, for a given network $G(\mathcal{N}, \mathcal{M})$, letting $\mathcal{N}_u$ and $\mathcal{M}_u$ accordingly be the unoccupied node set and edge set (see also Section 3.1.1.6), the feedback vertex set problem aims to find a minimal node set $\mathcal{N}_{\text{FVS}}$ whose removal would make the network acyclic, i.e.,

$$\mathcal{N}_{\text{FVS}} = \underset{\mathcal{N}_u}{\arg\min} |\mathcal{N}_u|, \text{ s.t. } d_c(i) = 0, \forall i \in G(\mathcal{N} \setminus \mathcal{N}_u, \mathcal{M} \setminus \mathcal{M}_u), \tag{3.17}$$

where $d_c(i) = 0$ means that there is no any cycle regarding node i (see also Eq. (2.10)). To tackle the FVS problem, which is an NP-complete problem [91], both ABPDS and AMSRGS employ a variant, but different, of the belief propagation algorithm[12] (or message-passing algorithm) to approach the optimal $\mathcal{N}_{\text{FVS}}$. After the previous process, we can easily verify that $G(\mathcal{N} \setminus \mathcal{N}_u, \mathcal{M} \setminus \mathcal{M}_u)$ would be a forest, in which some components might have size larger than the given demand $\alpha \times n$ (see Eq. (3.8)). Hence, one needs to break those components further. Indeed, breaking of them is quite a simple problem since those components are trees. But usually, rather than disintegrate the forest into a subnetwork whose LCC is a little bit smaller than $\alpha \times n$, the decycling-based methods choose to collapse it into a one whose LCC is much smaller than $\alpha \times n$. And then, they greedily reintroduce[13] the removed nodes (including those in $\mathcal{N}_{\text{FVS}}$) until meeting the demand. To sum up, the processes of the decycling-based methods are as follows[14]: i) obtain $\mathcal{N}_{\text{FVS}}$ and remove them from the given network; ii) continue to remove nodes from the remaining network to break it into a subnetwork where the LCC has a fraction of nodes α_1 much less than α, i.e., $\alpha_1 \ll \alpha$; and iii) greedily add the removed nodes back to the remaining network until we get q_c.

Performance of ABPDS and AMSRGS can be found in Appendix A.2.10.

3.3.4 Collective influence approach

The collective influence method (ACIS) is also based on the observation on general model networks but from the perspective that those kinds of networks always have a tree-like structure [23]. Thus, they think that there should be a connection between the breaking of a tree and a real-world network. Specifically, considering the site percolation (see Section 2.4.1 as an example) on a tree, ACIS holds its approximately optimal solution through repeatedly removing the node which has the largest collective influence strength in the remaining network $G'(\mathcal{N} \setminus \mathcal{N}_u, \mathcal{M} \setminus \mathcal{M}_u)$. For G', node i's collective influence strength $\text{CI}_\ell(i)$ is obtained in the following way,

$$\text{CI}_\ell(i) = (k'_i - 1) \sum_{j \in \partial\text{Ball}(i,\ell)} (k'_j - 1), \tag{3.18}$$

[11]One can find more explanation regarding the connection of the network dismantling problem and the decycling problem in ref. [26]. Note that the dismantling problem is exactly equivalent to the optimization of the critical threshold q_c.

[12]Details of this algorithm are out of the scope of this thesis.

[13]We will show this strategy later.

[14]They are the basic processes of AMSRGS. ABPDS is a little bit different but follows a similar framework.

where ℓ is the radius of the ball $\text{Ball}(i,\ell)$, k'_i is the corresponding degree of i of G' and $k'_i - 1$ accounts for the out degree (see Section 3.1.2.2), and $\partial\text{Ball}(i,\ell)$ is a node set consisting of all nodes with ℓ-length shortest path to node i (see also Eq. (2.7) where d_{ij} is equivalent to ℓ).

An example about how to calculate the collective influence strength is shown in Fig. A.11 in Appendix A.2.11, where the influence of ℓ on ACIS and the corresponding comparisons with ABetS and AHubS are also reported.

3.3.5 Percolation-based methods

ACIS can also be categorized as a percolation-based method since it is derived based on site percolation. But here we are going to mainly focus on those which share ideas from explosive percolation (see also Section 2.5), including the inverse targeting strategy (AITS) [21], the critical node detection method (ACNS) [22], and the explosive immunization method (AEIS) [24].

AITS employs a strategy similar to n_s-R with $n_s = |\mathcal{N}_u(t)|$ (see Section 2.5.4), and it is bounded by the LCC $\mathcal{G}_p(t/n)$ (see Eq. (3.4) for the connection between $\mathcal{G}_a(q)$ and $\mathcal{G}_p(t/n)$). In detail, for a given network $G(\mathcal{N},\mathcal{M})$, AITS obtains a new sequence through the following percolation process: starting with $t = 1$ and $\mathcal{G}_p(0/n) = 0$, i) at time step t, calculate

$$a_i = \mathcal{G}_p(t/n) - \mathcal{G}_p((t-1)/n) \text{ and}$$

$$b_i = k_i - k'_i$$

for each node i in the unoccupied node set $\mathcal{N}_u(t-1)$, where $\mathcal{G}_p(t/n)$ is the fraction of the LCC over the assumption that node i is occupied, k_i is the degree of node i in the full network G, and k'_i is the degree of i in $G'(\mathcal{N}_o(t),\mathcal{M}_o(t))$ assuming that i is occupied; ii) occupy node v if it is unique or randomly choose one to occupy if there are several v, given the condition

$$b_v = \min b_u, \forall u \in \{i | a_i = \min a_j, \forall j \in \mathcal{N}_u(t-1)\};$$

and iii) repeat steps i) and ii) until all nodes are occupied. In other words, AITS repeatedly occupies the node which makes the least contribution to the LCC and keeps the occupied network as dense as possible.

ACNS, instead, considers the node which leads the largest 'jump'[15] of the LCC during a percolation process. In detail, from Section 2.4.2, we know that the order parameter $\mathcal{G}_p(t/n)$ is a non-decreasing function of t. Now regarding the change of magnitude of $\mathcal{G}_p(t/n)$ at each time step, that is,

$$a(t) = \mathcal{G}_p(t/n) - \mathcal{G}_p((t-1)/n), \tag{3.19}$$

we then obtain the critical node i at t' satisfying (recall that one node is occupied at each t)

$$t' = \arg\max_t a(t). \tag{3.20}$$

[15]Usually, this 'jump' happens at the maximization of the second LCC, which is a very popular measure regarding percolation, such as refs. [14, 61].

Next, remove i from the network and repeat the previous process on the remaining network until all nodes are removed from the given network. Sometimes, one might get several nodes owning the same $a(t)$. In this case, remove the one with the largest degree, or randomly remove one if several share the same largest degree.

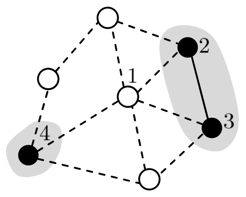


Figure 3.6: An example of the distinct nearest neighbor set $\hat{\Gamma}$. The filled circles and solid line are accordingly the occupied nodes and edge, i.e., $\mathcal{N}_o(t)$ and $\mathcal{M}_o(t)$. On the contrary, those unfilled circles and dashed lines correspond to the unoccupied, that is, $\mathcal{N}_u(t)$ and $\mathcal{M}_u(t)$, respectively. Considering node 1, we have $\hat{\Gamma}(1) = \{2, 4\}$ or $\hat{\Gamma}(1) = \{3, 4\}$ since $c(2) = c(3)$.

The third method is AEIS that conducts a similar process of AITS but with two extra restraints. One is that it uses a fixed n_s of n_s-R rather than let $n_s = |\mathcal{N}_u(t)|$. This strategy could speed up the algorithm compared to AITS. The other one is that it scores each node before the percolation process. Specifically, for a given network $G(\mathcal{N}, \mathcal{M})$, letting $c(i)$ be the component that node i belongs to and $|c(i)|$ be the corresponding size (see also Eq. (2.34)), AEIS obtains the score a_i of node i through

$$a_i = k_i^{(\text{eff})} + \sum_{j \in \hat{\Gamma}(i)} (\sqrt{|c(j)|} - 1), \tag{3.21}$$

where $\hat{\Gamma}(i)$ is the distinct nearest neighbor set of i in the occupied network $G'(\mathcal{N}_o(t), \mathcal{M}_o(t))$, that is, $\hat{\Gamma}(i)$ only contains nodes in different components (see Fig. 3.6). $k_i^{(\text{eff})}$ is used to characterize the potential spreading ability of node i [24], that is, a node with large $k_i^{(\text{eff})}$ indicates that it is more likely to transmit an epidemic out,

$$k_i^{(\text{eff})} = k_i + L_i - \alpha(\{k_j^{(\text{eff})} | j \in \Gamma(i)\}) \tag{3.22}$$

in which L_i is the number of leaves (nodes with degree 1), $\Gamma(i)$ is the i's nearest neighbor set and $\alpha(\{k_j^{(\text{eff})} | j \in \Gamma(i)\})$ is the number of strong hubs (nodes with $k_j^{(\text{eff})} \geq K$, K is a given parameter, usually $K = 6$ [24]). Note that $k_i^{(\text{eff})}$ is calculated before the percolation process. Thus all parameters in Eq. (3.22) are associated with the original network G. To sum up, AEIS gets a new sequence through: i) calculating $k_i^{(\text{eff})}$ for each node based on Eq. (3.22); ii) randomly selecting n_s nodes from $\mathcal{N}_u(t)$ and obtaining their related a_i following Eq. (3.21); iii) occupying the one with minimal a_i; iv) repeating steps ii) and iii) until all nodes are occupied.

Appendix A.2.12 gives comparison among these approaches.

3.3.6 Summary

Basically, targeted methods have better performance than local ones (see Fig. (A.3) as an example). For targeted methods, most heuristic methods surpass the corresponding general ones (see Figs. A.6 and A.8 in Appendixes A.2.8 and A.2.9). Amid those heuristic methods, decycling-based methods (ABPDS and AMSRGS) can always obtain a very small critical threshold q_c (see Fig. A.10 in Appendix A.2.10), and percolation-based methods are usually capable of getting a small F (see Fig. A.13 in Appendix A.2.12). However, all of them are less effective than ABetS in almost all cases.

3.4 Bounded and Unbounded Strategies

Though ABetS works well in most cases, it is too time-consuming to be possible for large networks. Thus, our first aim is to develop new methods that could surpass ABetS over both performance and time consumption, or at least obtain similar results using much less time[16].

3.4.1 Union-Find Algorithm

A basic algorithm that our methods rely on is the Union-Find (UF) algorithm, which is designed for the union-find set [92]. The union-find set is a data structure that can track disjoint subsets when the elements are partitioned into several groups. Roughly, the UF algorithm can help us, within nearly constant time (for ER-type networks), to merge components (Union) or to determine whether any two nodes are in the same component (Find). Following the notation in Section 2.1.2, we employ $a[i]$ to denote the root of node i. Then, we can use the Root function in Algorithm A.2 (Appendix A.2.13) to find the root of a given node i, the Find function to check whether i and j are in the same component, and the Union function to merge two components. Fig. A.15 (Appendix A.2.13) also gives an illustration regarding this algorithm.

Algorithm 3.1: Site percolation on an existing network

```
   Input: $G(\mathcal{N}, \mathcal{M})$
   Output: $S$
1  Initialize $\mathcal{N}_u(0) \leftarrow \mathcal{N}$ and $\mathcal{N}_o(0) \leftarrow \{\}$
2  $t \leftarrow 0$
3  while $t \leqslant n$ do
4      $t \leftarrow t + 1$
5      Get candidate set, $\mathcal{N}_c(t) \leftarrow RS(\mathcal{N}_u(t), n_s)$
6      Choose the new occupied node
         $i \leftarrow \arg\min_j \xi(j), \forall j \in \mathcal{N}_c(t)$
7      $\mathcal{N}_u(t) \leftarrow \mathcal{N}_u(t) \setminus \{i\}$
8      $\mathcal{N}_o(t) \leftarrow \mathcal{N}_o(t) \cup \{i\}$
9      $S[t] \leftarrow i$
```

[16]This section is mainly based on results from our previously punished paper **[L1]** where only unbounded strategies are reported. Here, the detailed routine to reach methods of ref. **[L1]** is also presented.

3.4.2 Bounded-size strategies

From Section 3.3, we know that those percolation-based methods always have good performance. And basically, both AITS and AEIS share a similar strategy to SumR (i.e., the sum rule, see also Section 2.5.3), and ACNS could also be viewed as an adaption of the rule in refs. [67, 68], where they consider the effect of a single edge on explosive percolation. Thus we are naturally to ask how about other strategies? Could they help us find better solutions?

Following Section 2.5, we start with the bounded-size rule (recall that the BF strategy is a special case of it). Since mainly considering the site percolation on existing networks, we here rewrite Algorithm 2.1 as Algorithm 3.1. For the bounded-size rule, we further rewrite Eq. (2.34) as

$$\xi(i) = \begin{cases} b, & \text{if } \sum\limits_{j\in\hat{\Gamma}(i)} |c(j)| \leqslant \alpha \\ \infty, & \text{otherwise,} \end{cases} \tag{3.23}$$

where b is a constant. Note that Eq. (3.23) is a variant of the bounded-size rule. Then, the BF strategy corresponds to the case of $\alpha = 2$. Eq. (3.23) is apparently not a good strategy to suppress either q_c or F since it views components of size less than α as same (see also Fig. A.16 in Appendix A.2.13). But it is truly much better than RanS. Besides, $n_s = 200$ and $n_s = 2000$ share similar results in both networks.

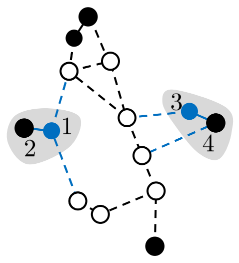


Figure 3.7: An example of external degree where black filled circles and solid edges represent occupied nodes and edges, and blue dashed and unfilled are those unoccupied, respectively. Considering candidate node 3, the corresponding component contains two nodes, i.e., $c(3) = \{3,4\}$, and its external degree is $k^{\text{out}}_{c(3)} = k_3 - k'_3 + k_4 - k'_4 = 2$. Similarly, one can have $k^{\text{out}}_{c(1)} = 2$.

Because of the low effectiveness of Eq. (3.23), we further introduce two variants regarding the bounded-size rule. The first one aims to contain the growth of external degree of components, which we name **ABonS1**. For a given network $G(\mathcal{N}, \mathcal{M})$, we define the external degree $k^{\text{out}}_{c(i)}$ of the component $c(i)$ regarding a candidate node i as

$$k^{\text{out}}_{c(i)} = \sum_{j\in c(i)} (k_j - k'_j), \tag{3.24}$$

where k_j is the degree of node j in G, and k'_j is the one in $G'(\mathcal{N}_o(t), \mathcal{M}_o(t))$ assuming that the candidate i is occupied (see Fig. 3.7 as an example). Then, starting with $\alpha = 1$, $\alpha_0 = \alpha$ and $b > 1$, ABonS1 obtains a new sequence through: i) at step t, choose node $i \in \mathcal{N}_u(t)$ to occupy if it has $k^{\text{out}}_{c(i)} \leqslant \alpha$; ii) repeat i) until there are no any unoccupied nodes having external degree less than α, i.e., $k^{\text{out}}_{c(i)} > \alpha, \forall i \in \mathcal{N}_u(t)$; iii) let $\alpha = b \times \min k^{\text{out}}_{c(i)}, \forall i \in \mathcal{N}_u(t)$ and $\alpha_0 = \alpha$ if at least one new node is occupied during steps i) and ii), otherwise, $\alpha = b \times \alpha_0$ and

$\alpha_0 = \alpha$; iv) repeat i), ii) and iii) until all nodes are occupied. Note that b can be a constant or a function of t but it has to be larger than 1 to ensure the convergence of the algorithm (see its performance in Appendix A.2.13).

The second variant (**ABonS2**) is a further extension of ABonS1, which constrains both the external degree and the component size. Supposing $b_c > 1$, $b_o > 1$, $\alpha_c = 1$ and $\alpha_2 = \alpha_c$, ABonS2 conducts the following processes to get a new sequence:

i) let $\alpha_o = \sqrt{\alpha_c}$ and $\alpha_1 = \alpha_o$;

ii) at time step t, choose node $i \in \mathcal{N}_u(t)$ to occupy if it satisfies $k^{\text{out}}_{c(i)} \leqslant \alpha_o$ and $|c(i)| \leqslant \alpha_c$;

iii) repeat ii) until $k^{\text{out}}_{c(i)} > \alpha_o$ or $|c(i)| > \alpha_c, \forall i \in \mathcal{N}_u(t)$;

iv) let $\alpha_o = b_o \times \min k^{\text{out}}_{c(i)}, \forall i \in \mathcal{N}_u(t)$ and $\alpha_1 = \alpha_o$ if at least one new node is occupied during steps ii) and iii), otherwise, $\alpha_o = b_o \times \alpha_1$ and $\alpha_1 = \alpha_o$;

v) repeat ii), iii) and iv) until $\alpha_o > \alpha_c(\langle k \rangle - 2)$;

vi) let $\alpha_c = b_c \times \min |c(i)|, \forall i \in \mathcal{N}_u(t)$ and $\alpha_2 = \alpha_c$ if at least one new node is occupied during steps ii) and iii), otherwise, $\alpha_c = b_c \times \alpha_2$ and $\alpha_2 = \alpha_c$;

vii) repeat the previous steps until $\alpha_c \geqslant n$, and the employ ABonS1 to occupy the remaining unoccupied nodes.

The effectiveness of ABonS2 verified by ABonS1 can be found in Fig. A.18 in Appendix A.2.13.

3.4.3 Sum and product rules regarding nodes

As Section 2.5.3, we further consider sum and product rules, in particular unbounded-size rule n_s-R. Following Algorithm 3.1, the sum rule corresponds to a goal function as

$$\xi(i) = \sum_{j \in \hat{\Gamma}(i)} |c(j)|, \tag{3.25}$$

where, note that again, $\hat{\Gamma}(i)$ is the distinct nearest neighbor set of i (see also Fig. 3.6) regarding the occupied network $G'(\mathcal{N}_o(t), \mathcal{M}_o(t))$. Similarly, the product rule has

$$\xi(i) = \prod_{j \in \hat{\Gamma}(i)} |c(j)|. \tag{3.26}$$

Further, the associated n_s-R simply increases the number of candidates. Here we refer n_s-R regarding the sum rule as **ASumRS**, and the product rule as **AProRS**. The corresponding performance can be found in Fig. A.19 (Appendix A.2.13).

Indeed, one can also design other strategies[17]. For example, considering ABonS2 in the previous section, at time step t, node i is directly occupied if it satisfies $k^{\text{out}}_{c(i)} \leqslant \alpha_o$ and $|c(i)| \leqslant \alpha_c$, otherwise, it is occupied following

$$\min\{1, e^{-(((\alpha_c - |c(i)|)/a_c)^2 + ((\alpha_o - k^{\text{out}}_{c(i)})/a_o)^2)}\}, \tag{3.27}$$

[17]They are important for the later evolutionary framework.

where a_c and a_o could be given constants or vary with the time step t. One can also extend ABonS1 in a similar way. The reason why we choose Eq. (3.27) can be found in ref. [59]. But it is worth mentioning that the introduction of Eq. (3.27) would heavily slow down ABonS2.

3.4.4 The power of selections over choices

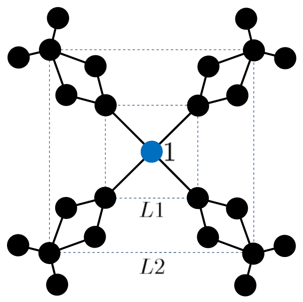


Figure 3.8: Given the demanded case of isolating every node in this network (consisting of filled circles and solid lines), almost all existing methods would remove the central node, i.e., node 1, and hence fail to get the optimal solution which is actually the eight nodes on the two dashed squares ($L1$ and $L2$). Now if we randomly remove two of those on $L1$, then, for example, both HubS and AHubS can get the optimum.

Though those strategies (especially ABonS1 and ABonS2) have similar performance to ABetS in some networks and could always obtain good solutions very fast[18], none of them can certainly surpass ABetS in the effectiveness. Thus, we might need to change our thought to search for more powerful methods. When looking into those heuristic methods, one can easily observe that almost all of them belong to the greedy family, that is, greedily remove (occupy) the most (least) important node from the remaining (occupied) network, including decycling-based methods. In other words, they only consider one-step influence among nodes. But the fact is that those impacts might also be long-term. For example, the random removal of two nodes on $L1$ of the network in Fig. 3.8, on the one hand, weakens the importance of the central node (node 1), on the other hand, relatively strengthens the influence of nodes on $L2$. One could also think that the random removal directly affects the central node and indirectly influences those on $L2$. And more importantly, such random removal could successfully help us find the optimal solution. Note that even ABetS would fail on the example in Fig. 3.8, which indeed gives us a powerful motivation, that is, there is some possibility to find a method that can overtake ABetS. And therefore, a natural question arises as to how can we model those influences?

A straightforward extension of a one-step influence is to check the influence of failures from several nodes simultaneously. For instance, considering the percolation process, we can occupy n_o nodes at each step t and check their contribution to $\mathcal{G}_p(q)$. But this strategy needs $\sim \binom{|\mathcal{N}_u|}{n_o}$ times to find the optimal configuration, which is unacceptable for most cases. Hence, as an alternative, per time n_s nodes are randomly selected from $\mathcal{N}_u$, and then one of them is chosen to be occupied based on some goal function $\xi(\cdot)$. This occupation would be repeated n_o times, which accounts for $n_s|\mathcal{N}_u|$ compared to $\sim \binom{|\mathcal{N}_u|}{n_o}$. The alternative strategy actually

[18]Actually, except for ABetS, those developed approaches surpass almost all methods that we discussed in Section 3.3, in both performance and time consumption.

coincides with ASumRS and AProRS. And again, it might converge to local optimum. Thus, we have to make some compensation for them.

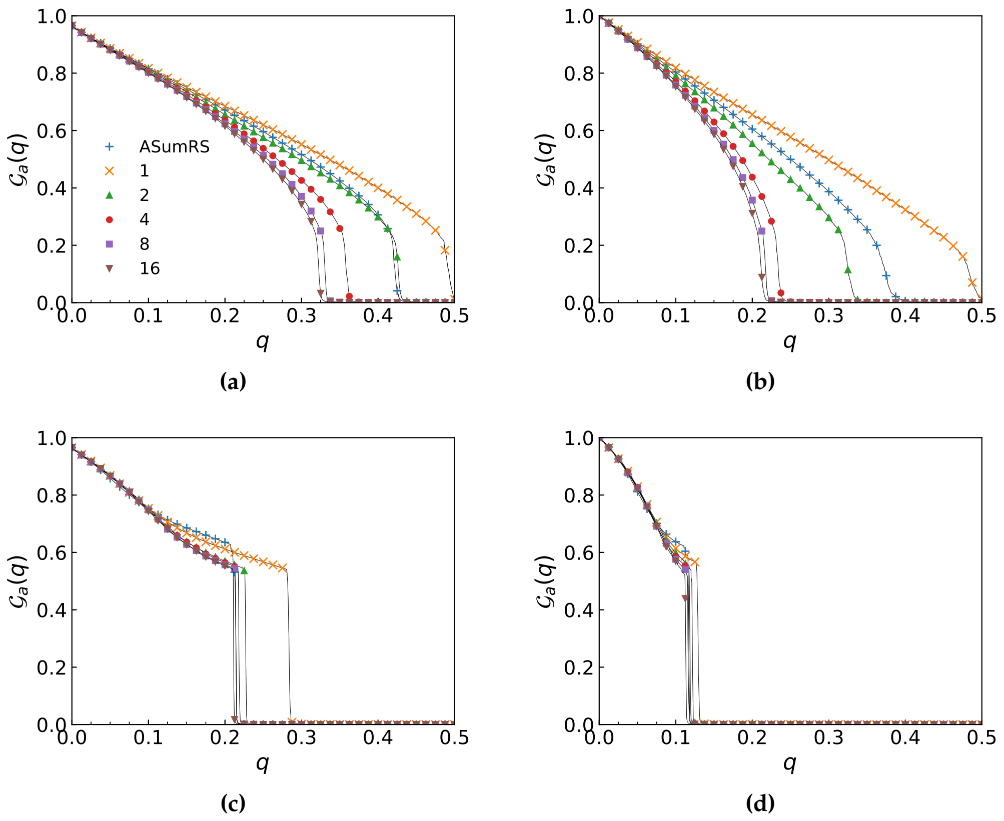


Figure 3.9: Performance of ASumRSp (with $r_u = 0.2$ and $T = 1, 2, 4, ...$) compared to ASumRS (with $n_s = 2$) on (a) and (c) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$, and (b) and (d) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$. $n_s = 2$ is conducted for ASumRSp in (a) and (b), and $n_s = 10$ in (c) and (d).

Specifically, starting with a distinct sequence S regarding a given network $G(\mathcal{N}, \mathcal{M})$, we assume that $\mathcal{N}_o(t) \equiv \{S[j] | j < t\}$ and $\mathcal{N}_u(t) \equiv \{S[j] | j \geqslant t\}$ hold for all t. Further, letting $\mathcal{N}_u^p(t)$ be a subset of $\mathcal{N}_u(t)$ and follow

$$\mathcal{N}_u^p(t) = \{S[j] | j \in [t, \min(\lfloor t + r_u \times n \rfloor, n)]\}, \tag{3.28}$$

where $r_u \in (0, 1]$ is a controlling parameter to determine which part of nodes could be candidates, then we construct the candidate set $\mathcal{N}_c(t)$ through n_s times of selections on $\mathcal{N}_u^p(t)$, i.e., modify *line 5* in Algorithm 3.1 as

$$\mathcal{N}_c(t) \leftarrow RS(\mathcal{N}_u^p(t), n_s).$$

Now let us have a glance at the function that r_u plays. Obviously, $r_u = 1$ corresponds to ASumRS or AProRS depending on which goal function $\xi(\cdot)$ is considered. $r_u < 1$ means that we construct the candidate only considering part of those unoccupied nodes. In this case, what would happen if we repeat the percolation process a number of times T under the constraint that S is reused? That is, another round of percolation process is conducted based on the new $S(T)$ from the previous round. For the ease of description, we call this process as **ASumRSp** or **AProRSp**. Indeed, as shown in Fig. 3.9, for the first round, ASumRSp is worse than ASumRS but soon becomes much better than ASumRS as the increase of T.

From Fig. 3.9, we can also conclude that the effectiveness of AProRSp relies on both n_s and r_u as well as T. And to some extent, we could say that $S(T)$ becomes more and more orderly as T increases. Considering this as an assumption, we can then further let n_s and r_u associated with T. Basically, a large n_s in tandem with a small r_u indicates an enhancement of one-step influence so that it is more likely to help find local optimizations. On the contrary, a small n_s complete with a large r_u gives more freedom to the possible configuration and hence is more likely to result in global optimums. And roughly, the time complexity of the whole procedure is $O(Tn_s n\langle k\rangle)$ which is associated with both T and n_s. Therefore, we here let

$$\begin{aligned} n_s(T) &= n_s(0) + \lfloor T\delta_{n_s} + 0.5 \rfloor \text{ and} \\ r_u(T) &= \frac{r_u(0)}{T\delta_{r_u} + 1}, \end{aligned} \tag{3.29}$$

where δ_{n_s} and δ_{r_u} represent the increase rate of $n_s(T)$ and the decrease rate of $r_u(T)$, respectively. Usually, one should start with a small $n_s(0)$ and a large $r_u(0)$, that is, firstly try to globally rank $S(T)$ and then further locally optimize it through large $n_s(T)$ and $r_u(T)$. But we still face another problem that the random selection might result in some bad solutions. To overcome this, we then measure F or q_c for each $S(T)$, say $F(S(T))$ or $q_c(S(T))$, and let $S(T) = S(T-1)$ if $F(S(T)) > F(S(T-1))$ or $q_c(S(T)) > q_c(S(T-1))$.

Algorithm 3.2: One round of ARRS

```
   Input: $G(\mathcal{N},\mathcal{M})$, $S(T-1)$
   Output: $S(T)$
1  Initialize $\mathcal{N}_u(0) \leftarrow \mathcal{N}$ and $\mathcal{N}_o(0) \leftarrow \{\}$
2  $t \leftarrow 0, S' \leftarrow S(T-1)$
3  while $t \leqslant n$ do
4      $t \leftarrow t+1$
5      Get $\mathcal{N}_u^p(t)$ based on $S'$ and $r_u(T)$
6      Get candidate set, $\mathcal{N}_c(t) \leftarrow RS(\mathcal{N}_u^p(t), n_s(T))$
7      Choose the new occupied node
         $i \leftarrow \arg\min_j \xi(j), \forall j \in \mathcal{N}_c(t)$
8      $\mathcal{N}_u(t) \leftarrow \mathcal{N}_u(t) \setminus \{i\}$
9      $\mathcal{N}_o(t) \leftarrow \mathcal{N}_o(t) \cup \{i\}$
10     $S'[t] \leftarrow i$
11 if $\xi_g(S') < \xi_g(S(T-1))$ then
12     $S(T) \leftarrow S'$
13 else
14     $S(T) \leftarrow S(T-1)$
```

We call the above processes as the relationship related method (**ARRS**). To sum up, ARRS obtains a new sequence $S(T)$ through:

i) Given $S(0)$ (could be a randomly ordered sequence), $n_s(0)$, δ_{n_s}, $r_u(0)$, δ_{r_u} and the maximal iteration $\hat{T}$;

ii) at iteration T, obtaining $S(T)$ based on Algorithm 3.2 where $\xi_g(\cdot)$ corresponds to $F(\cdot)$, $q_c(\cdot)$, or other possible global goal function;

iii) repeating ii) until $T = \hat{T}$.

With respect to the local goal function $\xi(\cdot)$, one can choose the sum rule, the product rule, or others that we have mentioned such as in Sections 3.4.2 and 3.4.3. Here we consider a hybrid of the sum and product rules. That is, if the sum rule is chosen at T and makes a positive contribution to $\xi_g(\cdot)$, i.e., $\xi_g(S(T)) < \xi_g(S(T-1))$, then $a_s(T) = a_s(T-1) + 1$, otherwise, $a_s(T) = a_s(T-1)$. Similarly, for the product rule, $a_p(T) = a_p(T-1) + 1$ if it obtains a better solution, otherwise, $a_p(T) = a_p(T-1)$. During each iteration T, either the sum rule or the product rule is conducted. The sum rule is chosen with probability $\frac{a_s(T-1)}{a_s(T-1)+a_p(T-1)}$ starting with $a_s(0) = a_p(0) = 1$, otherwise, the product rule is considered.

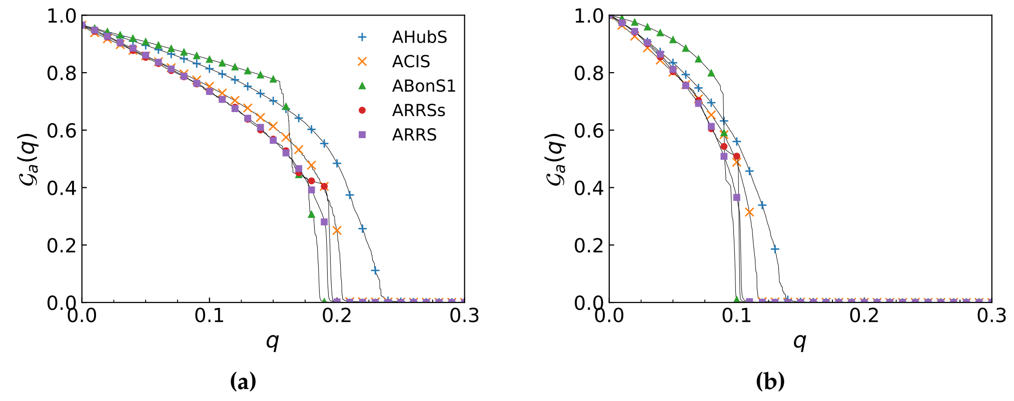


Figure 3.10: Performance of ARRSs and ARRS compared to AHubS, ACIS (with $\ell = 4$), and ABonS1 (with $b = 1.2$) on (a) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$ and (b) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$. $S(0) = \text{RanS}$, $n_s(0) = 10$, $\delta_{n_s} = 0.01$, $r_u(0) = F(\text{RanS})$, $\delta_{r_u} = 0.01$, $\hat{T} = 1000$ and $\xi_g(\cdot) = F(\cdot)$ are conducted for both ARRSs and ARRS, as well as the sum rule and the hybrid are considered for ARRSs and ARRS, respectively.

Fig. 3.10 shows the performance of **ARRSs** (only the sum rule is conducted) and ARRS on the two model networks. Apparently, both ARRSs and ARRS can obtain smaller F than AHubS, ACIS and ABonS1, in particular $(F(\text{ACIS}) - F(\text{ARRS}))/F(\text{ACIS}) = 5.29\%$ for the ER network and 6.43% for the BA network. In respect to the critical threshold q_c, ARRS with $\xi_g(\cdot) = q_c(\cdot)$ is only slightly better than ARRS with $\xi_g(\cdot) = F(\cdot)$ (in the BA network around 2.23%) and sometimes even worse (in the ER network around -0.32%), which indicates that ABonS1 still has advantage over q_c.

Here, we further extend ABonS1 by considering more information instead of viewing each node equally. Specifically, one straightforward way is to replace $k^{\text{out}}_{c(i)}$ (see also Eq. (3.24)) with

$$\mathcal{H}^{\text{out}}_{c(i)} = \sum_{j \in c(i)} \sum_{v \in \Gamma(j) \cap \mathcal{N}_u(t)} \mathcal{H}_v, \tag{3.30}$$

where $\Gamma(j) \cap \mathcal{N}_u(t)$ represent the unoccupied neighbors of node j and $\mathcal{H}$ is the node score sequence (see Fig. 3.11 for an instance). Apparently, Eq. (3.30) is equivalent to Eq. (3.24) if

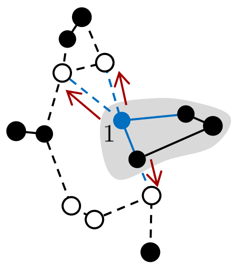


Figure 3.11: Considering the candidate node 1 colored blue, the shadow marks the component that it would lead to. For this example, $\mathcal{H}^{\text{out}}_{c(1)}$ accounts for the sum of scores from the three nodes that red arrows point to.

$\mathcal{H} = \mathbf{1}$. Here we choose to score each node based on the degree distribution, that is,

$$\mathcal{H}_i = \sum_{k \geqslant k_i} p_k, \tag{3.31}$$

which indicates that a node with a large degree would have a small score. In this manner, each occupied component would only have a few unoccupied neighbors in the subcritical regime, and thus the corresponding percolation is delayed. Indeed, the basic idea of Eq. (3.30) is to try to keep an occupied component only associated with a few influential nodes which are identified by $\mathcal{H}$. Therefore, we can also construct $\mathcal{H}$ based on, e.g., all kinds of centralities (see Section 2.2) where a potential way is to use the corresponding reciprocal of those centralities.

One can also employ the framework from ARRS by simply rewriting the local goal function $\xi(i)$ as

$$\xi(i) = \mathcal{H}^{\text{out}}_{c(i)}. \tag{3.32}$$

We here call the method relaying on Eq. (3.30) the predict relationship method (APRS). More clearly, we refer to the one integrated with ABonS1 as **APRSs1** and the one with the ARRS framework as **APRSrr**. The corresponding performance can be found in Fig. A.20 in Appendix A.2.13.

3.4.5 Applications

3.4.5.1 Data

Except for the two-type model networks, i.e., networks generated through the ER model and the BA model, the following real-world networks from a large range of domains are also considered to more thoroughly demonstrate the effectiveness of those developed methods. For the network robustness, as we mentioned in Section 3.1.1, climate change continuously increases the frequency and intensity of extreme events and keeps challenging the resilience of infrastructure and boosting the global supply chain risks. Thus, the first category is the infrastructure network [5, 14] including: a Power Grid network [42, 18] (Power) and two Road networks [93] (PAroad and Txroad). Another group of networks are constructed or extracted from communication systems, which are associated with both network robustness and immunization [2, 94, 8, 95]. For instance, considering the Internet network, one might have to find and protect the most important routers or hosts against the

Networks	n	m	Networks	n	m
Power	4941	6594	Email-Enron	36692	183831
CA-GrQc	5242	14490	p2p-Gnutella31	62586	147892
p2p-Gnutella08	6301	20777	loc-Gowalla	196591	950327
as-733	6474	12572	Email-EuAll	265214	364481
Scottish	7228	24784	com-Amazon	334863	925872
CA-AstroPh	18771	198050	web-Google	875713	4322051
CA-CondMat	23133	93439	PAroad	1088092	1541898
hep-th	27240	341923	Txroad	1379917	1921660
Cit-HepPh	34546	420877	as-Skitter	1696415	11095298

Table 3.1: Basic information of the 18 real-world networks.

breakdown of the whole system from malicious attacks. At the same time, antivirus softwares are installed to prevent or eliminate digital viruses. Here this type of networks include: two autonomous system graphs [96] (as-733 and as-Skitter), two Internet peer-to-peer networks [97, 98] (p2p-Gnutella08 and p2p-Gnutella31), three Collaboration networks [97] (ca-GrQc, ca-AstroPh and ca-CondMat), two citation networks [99, 96] (hep-th and cit-HepTh), the Scottish cattle movements network [24] (possible spread of virus), and two communication networks (email-Enron [100, 93] and email-EuAll [97]). Regarding misinformation[19], we choose one location-based online social network [101] (loc-Gowalla), the Amazon product co-purchasing network [102] (com-Amazon) and the Google web graph [93] (web-Google). Some basic information regarding these networks can be found in TABLE 3.1[20]. Note that for all networks considered here, the directed edges are simply replaced with undirected ones, and all self-loops and isolated nodes are entirely removed.

3.4.5.2 Configurations of associated methods

- RanS: average over at least 20 independent implementations (20 IIs).
- HubS: NONE.
- EigS: $\alpha = 0.1$.
- KatS: α varies from network to network and is smaller than the reciprocal of the largest eigenvalue of the adjacent matrix.
- PagS: $\alpha = 0.85$.
- CloS: NONE.
- BetS: NONE.
- KshS: NONE.
- AHubS: HubS on the whole remaining network.

[19]To some extent, it is equivalent to the immunization problem.

[20]The source data of these networks is either from `http://snap.stanford.edu/data` or `http://konect.uni-koblenz.de/networks/opsahl-powergrid`

- APagS: PagS with $\alpha = 0.85$ on the LCC.
- ABetS: BetS on the LCC.
- AMSRGS: $\alpha_1 = 3/n$ and $\alpha = 0.01$ if there is no particular explanation.
- ABPDS: $x = 12$ and $\alpha = 0.01$ if there is no particular explanation.
- ACIS: $\ell = 4$ if there is no particular explanation.
- AITS: 20 IIs.
- ACNS: 20 IIs.
- AEIS: $K = 6$ and $n_s = 2000$ if there is no particular explanation. 20 IIs.

- ABonS1: $b = 1.2$ and HubS as initial sequence if there is no particular explanation (default). If initial sequence is random, then 20 IIs.
- ABonS1q: $b = 1.2$ and HubS as initial sequence if there is no particular explanation (default). If initial sequence is random, then 20 IIs.
- ABonS2: $b_c = 1.1$, $b_o = 1.2$ and HubS as initial sequence if there is no particular explanation (default). If initial sequence is random, then 20 IIs.
- ASumRS: n_s relies on networks. 20 IIs.
- AProRS: n_s relies on networks. 20 IIs.
- ARRS: if there is no particular explanation, $S(0) = \text{HubS}$, $n_s(0) = 10$, $r_u(0) = F(\text{HubS})$, $\xi_g(\cdot) = F(\cdot)$, $\xi(\cdot)$ is a hybrid of ASumRS and AProRS with $a_s(0) = a_p(0) = 1$, $\delta_{n_s} = 0.001$ and $\delta_{r_u} = 0.01$ for $n \leqslant 10^4$, $\delta_{n_s} = 0.01$ and $\delta_{r_u} = 0.01$ for $10^4 < n \leqslant 10^5$, $\delta_{n_s} = 0.1$ and $\delta_{r_u} = 0.1$ for $10^5 < n \leqslant 10^6$, and $\delta_{n_s} = 0.5$ and $\delta_{r_u} = 0.5$ for $n > 10^6$. 20 IIs.
- APRSs1: $b = 1.2$, $\mathcal{H}$ is based on the corresponding degree distribution and HubS as initial sequence if there is no particular explanation (default). If initial sequence is random, then 20 IIs.
- APRSs1q: $b = 1.2$, $\mathcal{H}$ is based on the corresponding degree distribution and HubS as initial sequence if there is no particular explanation (default). If initial sequence is random, then 20 IIs.
- APRSrr: same as ARRS and APRSs1 if there is no particular explanation. 20 IIs.

NONE means that there is no controlling parameter regarding the corresponding method. ABonS1q and APRSs1q accordingly correspond to ABonS1 and APRSs1 with an extra constraint before q_c is reached, that is, for example, we firstly run ABonS1 bounded by α (see Eq. (3.8)) until all unoccupied node would lead to $\mathcal{G}_p(q) > \alpha$, then pure ABonS1 is conducted until all nodes are occupied.

3.4.5.3 Percolation metrics

As we mentioned in Section 3.2, we will frequently consider the order parameter $\mathcal{G}_a(q)$ as a function of q, the critical threshold q_c, and the average fraction F of the LCC as criteria to validate our developed methods. Besides, when specifically comparing two methods, e.g., $S1$ and $S2$, we say that $S1$ has an improvement of $a\%$ compared to $S2$ regarding F, simply representing that

$$\frac{F(S2) - F(S1)}{F(S2)} = a\%, \tag{3.33}$$

where a is some number. The similar way is also conducted for q_c. Note that $a\%$ would be negative if $S1$ is worse than $S2$. For this case, we refer to it as '$a\%$ regression'.

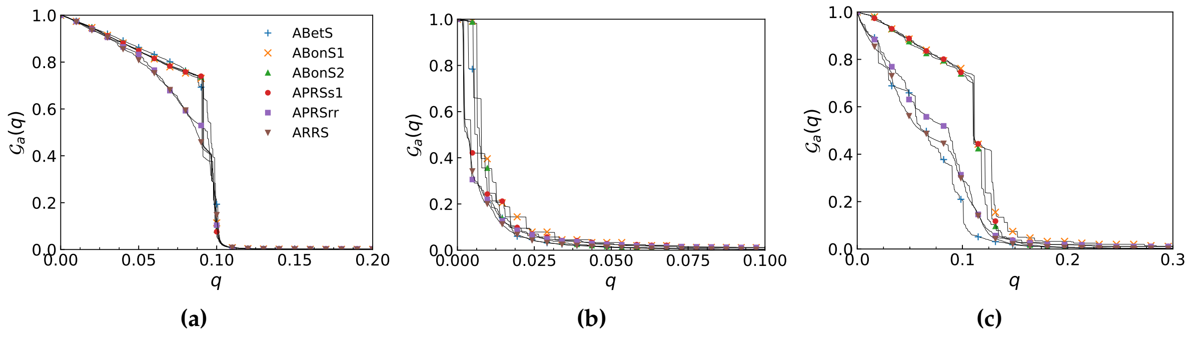
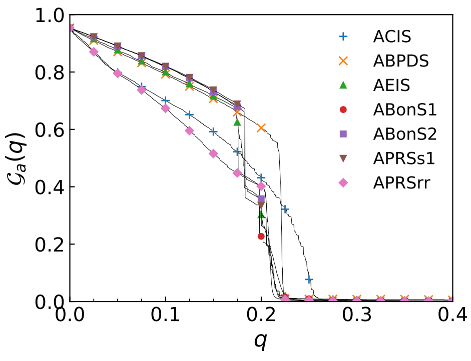


Figure 3.12: Performance of ABonS1, ABonS2, APRSs1, APRSrr, and ARRS compared to ABetS considering the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network.

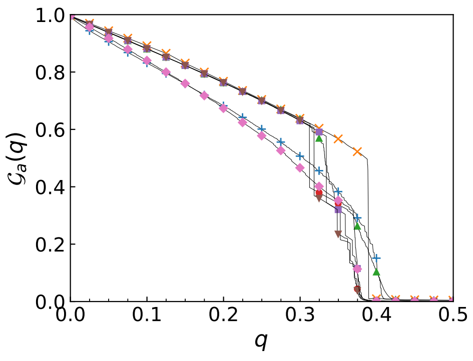
We first verify those basic methods considering the networks used in Figs. A.6, A.8, A.10, A.12, A.13 and A.14. As we can see from those results, order parameter curves are tangled with each other, which indicates that those basic methods truly have comparable performance as ABetS. More specifically, for the BA network (Fig. 3.12a), all five basic methods could obtain smaller F than ABetS, especially ARRS which is also much better than ABetS in the power grid network (21.18% improvement) but slightly worse in the yeast network (−4.95% regression). Thus, we have achieved our first goal: new methods which are capable of obtaining at least similar results to ABetS. Note that ABetS accounts for the best among those approaches that we studied in Section 3.3.

We further consider the so-called state-of-the-art strategies, mainly including ACIS, ABPDS, and AEIS. Note again that we choose ABPDS instead of AMSRGS since they have similar performance, and ABPDS is faster than AMSRGS. Even though they are, slightly or heavily, less effective than ABetS, they have the advantage of lower time complexity. Therefore, we further verify the proposed approaches through comparisons with those methods on the following networks.

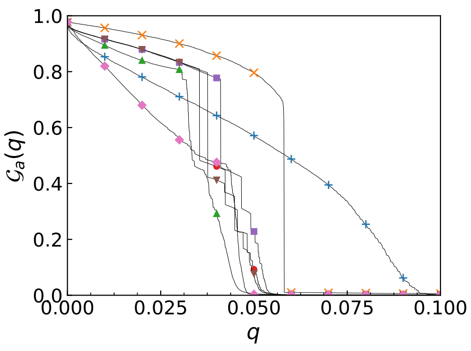
- CA-AstroPh: a collaboration network regarding Astro Physics in Arxiv [97]. Here we are interested in a question about which group of researchers promote the communication of this field. An imaginary scenario is what would happen if those researchers are held by a country. In this network, nodes represent scientists, and their collaborations (e.g., coauthor a scientific paper) are characterized by edges.



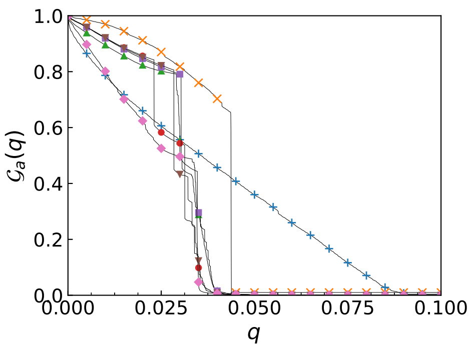
(a)



(b)



(c)



(d)

Figure 3.13: Comparisons among ACIS, ABPDS, AEIS, ABonS1, ABonS2, APRSs1, and APRSrr regarding $\mathcal{G}_a(q)$ of q on (a) the CA-AstroPh network, (b) the Cit-HepPh network, (c) the web-Google network, and (d) the as-Skitter network (where $\ell = 2$ is conducted for ACIS since it would take over one week to get the result for $\ell = 3$ in our system).

- Cit-HepPh: a citation network of Arxiv HepPh (high energy physics phenomenology) [96] where papers are denoted by nodes and a direct edge is put between two nodes if one of the corresponding papers cite the other one. For the convenience of comparisons with other methods, those directed edges are simply replaced by undirected ones. For this network, we would like to know which part of the papers are the most important ones.

- web-Google: a network of Google web where nodes represent web pages, and edges indicate hyperlinks among them [93]. Except for the robustness and immunization problem, the network is also possibly related to a problem that one should choose which group of webs to, like, advertise.

- as-Skitter: an Internet topology network on Autonomous systems by Skitter [96], which, as we mentioned, are associated with both network robustness and immunization.

Figs. 3.13 and 3.14 illustrate the performance of the basic methods compared to ACIS, ABPDS and AEIS on these networks. As we can see from Fig. 3.13, the advantages of proposed methods are not so prominent with respect to F since some methods hold smaller $\mathcal{G}_a(q)$ in the early stage (i.e., when q is small) while others have in the late. But one conclusion which could be drawn is that APRSrr is better than others, like in CA-AstroPh where APRSrr accordingly has 12.43%, 20.81%, and 13.37% improvement compared to ACIS, ABPDS, and

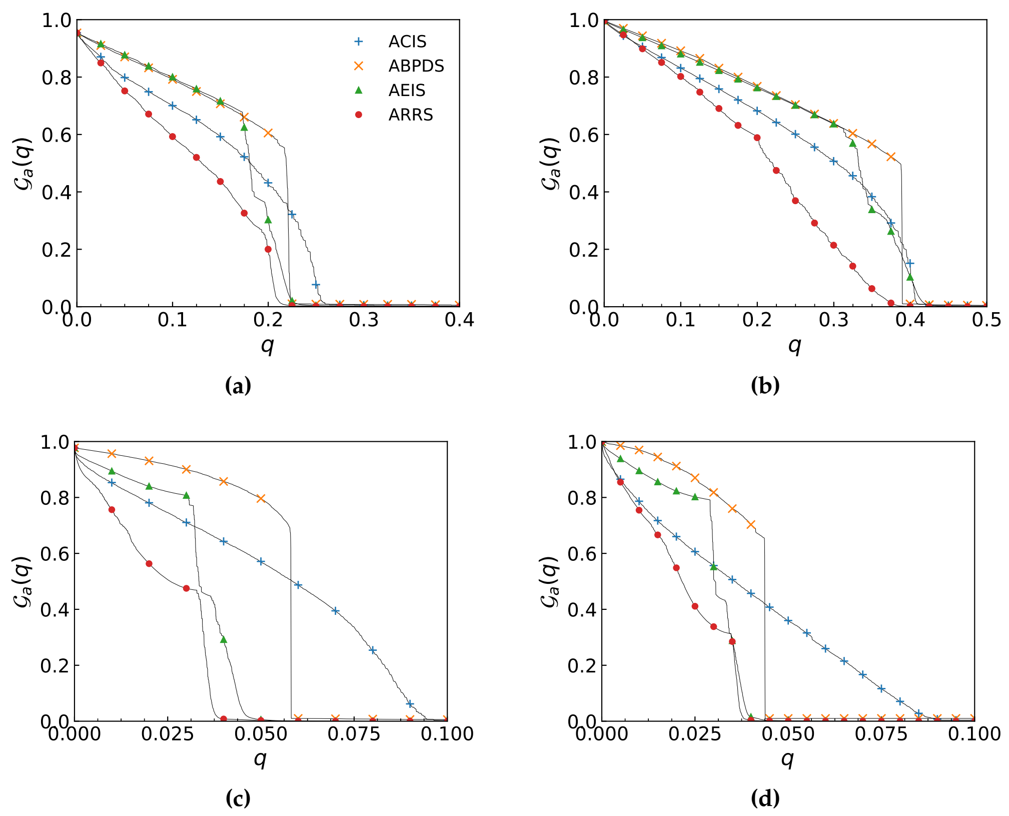


Figure 3.14: Performance of ARRS validated by ACIS, ABPDS, and AEIS on (a) the CA-AstroPh network, (b) the Cit-HepPh network, (c) the web-Google network, and (d) the as-Skitter network (ACIS with $\ell = 2$).

AEIS. ARRS pushes those advantages further, which has improvements of 23.20%, 30.55%, and 24.02%, respectively, while 12.30% over APRSrr. With regard to the critical threshold q_c, APRSrr is only slightly worse (−0.68%) than AEIS in the web-Google network and has 2.67% $\sim$ 59.00% improvement in other cases compared to ACIS, ABPDS, and AEIS. ARRS, again, is better than APRSrr in CA-AstroPh, Cit-HepPh, and web-Google but slightly worse in as-Skitter.

Significant performance of both APRSrr and ARRS further demonstrate 'the power of selections over choices'. Regarding this, another question arises as to what function the product rule plays. The reasons that we consider this are two-fold. On the one hand, it provides potential applications of those studies on the explosive percolation [62, 63, 58, 60] (see Section 2.5 for many others). On the other hand, it also demonstrates that there are some differences between real problems and explosive percolation. That is, for example, theoretically the most delayed critical threshold is obtained if all unoccupied edges are considered as candidates per step, and the system evolves following the sum rule instead of the product rule [70]. Apparently, as we can see from Fig. 3.15, the product rule accounts for the main contribution to the optimization of F. Actually, ARRS-p is usually more effective than ARRS-r in large networks but less in small networks, which is also the reason that we choose to consider the hybrid rule for ARRS.

Moreover, we also report comparisons regarding F and q_c in Tables 3.2 and 3.3, respectively. As we can see from Table 3.2, ARRS holds minimal F in all networks,

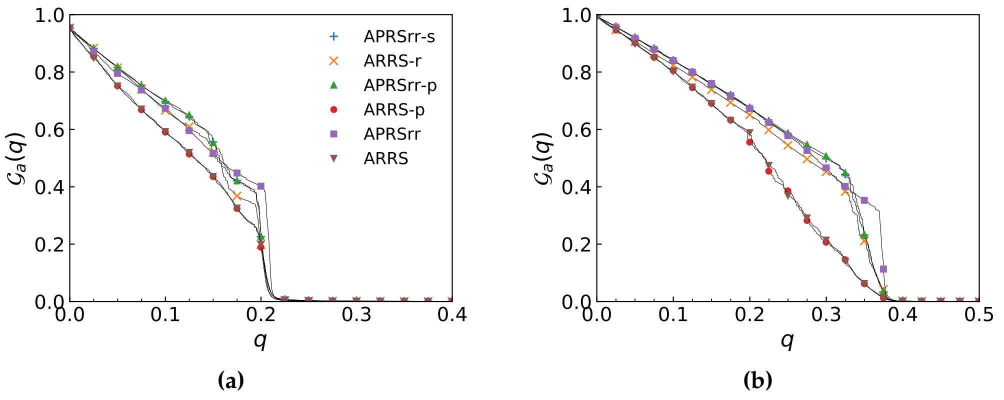


Figure 3.15: Comparisons of APRSrr-s (APRSrr with the sum rule), ARRS-r (ARRS with the sum rule), APRSrr-p (APRSrr with the product rule), ARRS-p (ARRS with the product rule), APRSrr (with the hybrid rule), and ARRS (with the hybrid rule) on (a) the CA-AstroPh network, (b) the Cit-HepPh network.

and it is certainly much better than others. For instance, compared to ACIS, ARRS has an improvement of 1.18% ∼ 97.95% (with a median of 35.00% and a mean of 44.95%). Meanwhile, it also has an advantage of 10.31% ∼ 69.57% (with a median of 25.90% and a mean of 30.70%) against AEIS. Regarding other basic methods, even the worst one among them, ABonS1, still outperforms both AHubS and ACIS in a mean of 19.53% and 24.70%, respectively. And it is only slightly worse than AEIS (−3.27% on average). Nevertheless, all ABonS1q, ABonS2, APRSs1, APRSs1q, and APRSrr are better than AHubS, ACIS, and AEIS. With respect to the critical threshold q_c (see Table 3.3), ARRS is still more effective than others in most networks, particularly in the four large networks. Other basic methods except for the two bounded-size, i.e., ABonS1 and APRSs1, are also much more effective than AHubS, ACIS, ABPDS, and AEIS. Meanwhile, a light constraint could compensate ABonS1 and APRSs1, and make them much more powerful than the compared methods, e.g., ABonS1q has improvements of 45.11%, 35.85%, 6.49%, and 10.52% compared to AHubS, ACIS, ABPDS, and AEIS, respectively.

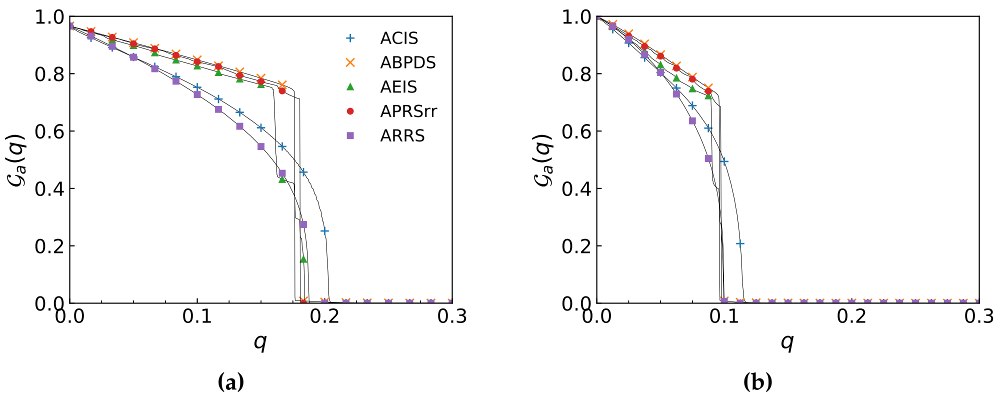


Figure 3.16: Performance of APRSrr and ARRS compared to ACIS, ABPDS, and AEIS on (a) ER networks (20 different configurations) with $\langle 3.5 \rangle$ and $n = 10^6$ and (b) BA networks (20 different configurations) with $\langle 4.0 \rangle$ and $n = 10^6$.

However, surprisingly, ABPDS has minimums of q_c in p2p-Gnutella08 and p2p-Gnutella31. As we know, ABPDS is developed based on the message-passing algorithm

Networks	AHubS	ACIS	AEIS	ABonS1	ABonS1q	ABonS2	APRSs1	APRSs1q	APRSrr	ARRS
Power	0.0524	0.0449	0.0149	0.0151	**0.0130**	**0.0107**	**0.0128**	**0.0115**	0.0154	***0.0076***
CA-GrQc	0.0685	0.0527	0.0345	0.0385	0.0361	0.0381	0.0368	0.0360	0.0356	***0.0289***
p2p-Gnutella08	0.1574	0.1414	0.1627	0.1555	0.1554	0.1542	0.1544	0.1543	0.1486	***0.1386***
as-733	0.0125	0.0150	0.0097	0.0104	0.0102	**0.0095**	0.0105	0.0102	0.0117	***0.0087***
Scottish	0.0272	0.0542	0.0259	0.0272	0.0272	**0.0256**	**0.0254**	**0.0254**	**0.0256**	***0.0231***
CA-AstroPh	0.2084	0.1562	0.1583	0.1597	0.1606	0.1602	0.1608	0.1579	**0.1368**	***0.1200***
CA-CondMat	0.1103	0.0832	0.0765	0.0776	0.0777	0.0784	0.0782	0.0782	**0.0694**	***0.0625***
hep-th	0.3048	0.2541	0.2728	0.2660	0.2650	0.2678	0.2665	0.2663	**0.2437**	***0.1915***
Cit-HepPh	0.3062	0.2645	0.2878	0.2727	0.2706	0.2781	0.2725	**0.2629**	**0.2533**	***0.2056***
Email-Enron	0.0380	0.0292	0.0316	0.0335	0.0325	0.0317	0.0324	0.0319	**0.0263**	***0.0217***
p2p-Gnutella31	0.1143	0.1015	0.1172	0.1121	0.1121	0.1112	0.1112	0.1113	0.1084	***0.1003***
loc-Gowalla	0.1142	0.0868	0.0913	0.0943	0.0932	0.0959	0.0909	0.0921	**0.0812**	***0.0625***
Email-EuAll	0.0009	0.0056	0.0018	0.0030	0.0018	0.0009	0.0009	0.0009	0.0011	***0.0008***
com-Amazon	0.1184	0.0793	0.0620	**0.0618**	**0.0610**	**0.0616**	0.0628	0.0634	**0.0583**	***0.0424***
web-Google	0.0886	0.0526	0.0312	0.0370	0.0376	0.0397	0.0375	0.0371	0.0312	***0.0227***
PAroad	0.0714	0.0417	0.0034	**0.0032**	**0.0027**	0.0039	**0.0030**	**0.0033**	**0.0019**	***0.0011***
Txroad	0.0651	0.0342	0.0023	**0.0015**	**0.0020**	**0.0022**	**0.0019**	0.0023	**0.0011**	***0.0007***
as-Skitter	0.0487	0.0394	0.0285	**0.0269**	0.0315	0.0297	**0.0280**	0.0318	**0.0239**	***0.0214***

Table 3.2: Results of F on the 18 real-world networks. CI is with $\ell = 3$ for the Email-EuAll network and $\ell = 2$ for the as-Skitter network. An item in bold represents the corresponding F is smaller than all of AHubS, ACIS, and AEIS. One in italic means that the associated method has the best performance on the related network among all those mentioned methods.

which is actually sensitive to local cycles. Thus one possible reason is that those two networks only have a few local cycles, which could be characterized by the clustering coefficient (see Section 2.1.3.4). Indeed, both p2p-Gnutella08 and p2p-Gnutella31 has a relatively small average clustering coefficient, accordingly 0.0109 and 0.0055. To further verify that, we consider networks generated through the ER model and BA model, where the number of cycles approaches a constant when $n \to \infty$ (Section 2.4.2). Fig. 3.16 shows comparisons among ACIS, ABPDS, AEIS, APRSrr, and ARRS in regard of $\mathcal{G}_a(q)$ of q. Again, APRSrr can obtain much smaller F than others even against ACIS. With respect to q_c, APRSrr outperforms both ACIS and AEIS, but it is slightly worse than ABPDS (-2.32% in ER networks and $-0.1.23\%$ in BA networks), which is in line with our previous speculation. Besides, the critical threshold q_c as a function of average degree $\langle k \rangle$ is illustrated in Fig. 3.17. Apparently,

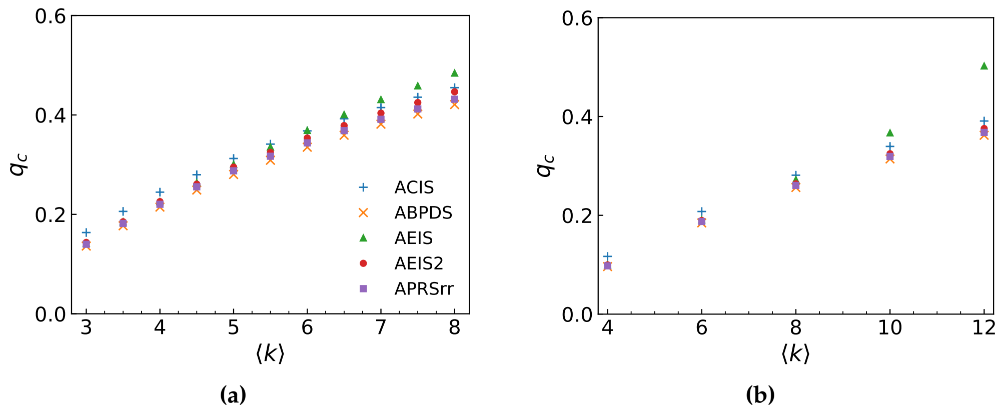


Figure 3.17: The critical threshold q_c versus the average degree $\langle k \rangle$ regarding ACIS, ABPDS, AEIS, AEIS2 (with $K = \langle k \rangle + 2$), and APRSrr on (a) ER networks with $n = 10^5$ and (b) BA networks $n = 10^5$. Each data point is drawn from 20 different networks.

Networks	AHubS	ACIS	ABPDS	AEIS	ABonS1	ABonS1q	ABonS2	APRSs1	APRSs1q	APRSrr	ARRS
Power	762	570	316	332	505	**285**	**301**	657	**292**	440.9	***282.55***
CA-GrQc	820	1760	398	423	624	**390**	502	500	***371***	**390.2**	**372.1**
p2p-Gnutella08	1584	1444	*1300*	1470	1355	1338	1344	1343	1332	1331.2	1372.55
as-733	248	192	162	168	205	**154**	**153**	209	**153**	187.8	***152.85***
Scottish	603	2036	434	465	453	441	441	454	445	***432.85***	442.7
CA-AstroPh	6274	4865	4198	4286	4445	**4102**	**4157**	4273	**4130**	**4055.6**	***4013.1***
CA-CondMat	4500	3217	2569	2680	2575	***2512***	**2516**	2571	**2552**	**2559.3**	**2534.35**
hep-th	12092	11184	10294	10959	**10162**	**10004**	**10153**	**10059**	**10032**	**9913.35**	***9732.1***
Cit-HepPh	15297	14164	13455	14405	**13191**	**13136**	**13204**	**13115**	**13089**	**13089.05**	***12982.9***
Email-Enron	4262	3074	2621	2753	2674	**2562**	**2568**	**2612**	***2553***	**2619**	**2572.9**
p2p-Gnutella31	12424	10995	*9287*	10109	9643	9630	9667	9596	9585	9811.25	10193.5
loc-Gowalla	40168	31386	26951	26851	**25982**	**25913**	**26197**	**25783**	**25654**	**25703.1**	***25015.3***
Email-EuAll	1282	1193	1064	6374	16102	***1057***	**1061**	1194	1077	1104.3	1077.2
com-Amazon	68527	42108	29572	27387	27484	27464	27444	27817	27699	28056.55	***26342.1***
web-Google	171550	82525	50861	41538	46598	46128	47412	46000	45776	**41175.95**	***33573.35***
PAroad	246270	71134	21172	16171	17222	**15223**	18169	17030	**15388**	**11150.15**	***10124.8***
Txroad	320991	82744	20873	16351	16912	**14079**	18073	16503	**14978**	**10676.5**	***9365.95***
as-Skitter	201670	151846	74286	70258	**65546**	**63344**	**69437**	**66667**	**64280**	***62059.25***	**63977.35**

Table 3.3: Results of $q_c \times n$ on the 18 real-world networks. CI is with $\ell = 3$ for the Email-EuAll network and $\ell = 2$ for the as-Skitter network. An item in bold represents the corresponding q_c is smaller than all four compared methods, i.e., AHubS, ACIS, ABPDS, and AEIS. One in italic means that the associated method has the best performance on the related network among all those mentioned methods.

ABPDS is slightly better than APRSrr, and then AEIS2 follows up. Note that, when tied by $K = 6$ (see Eq. (3.22)), AEIS performs worse and worse with the increase of $\langle k \rangle$. The reason why this happens is ascribed to the fact that $k_{v_i}^{(\mathrm{eff})}$ is harder and harder to identify nodes with similar degrees as $\langle k \rangle$ rises. In other words, more and more nodes have degrees larger than K when the network becomes dense. Thus, we also show results of AEIS2 in which we simply replace K with $K = \langle k \rangle + 2$. But one should know that our tests show that this strategy does not work for real-world networks.

3.4.5.4 FVS problem

From Section 3.3.3 we know decycling-based methods firstly tackle the feedback vertex set (FVS) problem. Here we show that those basic methods are also capable of handling the FVS problem after a tiny modification. Specifically, for a candidate node i at time step t during a percolation on a given network $G(\mathcal{N}, \mathcal{M})$, one can easily verify that there is at least one cycle in the corresponding component $c(i)$ if its distinct nearest-neighbor $\hat{\Gamma}(i)$ (see also Eq. (3.25)) has a size less than $|\Gamma(i) \cap \mathcal{N}_o(t)|$ (i.e., the number of occupied neighbors of i). Thus, we only need to further constrain the candidate set. For example, considering ABonS1 (see Section 3.4.2), one can achieve that by replacing ii) with: repeat i) until there are no any unoccupied nodes having external degree less than α, satisfying $|\Gamma(i) \cap \mathcal{N}_o(t)| - |\hat{\Gamma}(i)| \equiv 0$ and $k_{c(i)}^{\mathrm{out}} > \alpha, \forall i \in \mathcal{N}_u(t)$. For ARRS and APRSrr, we only need to get

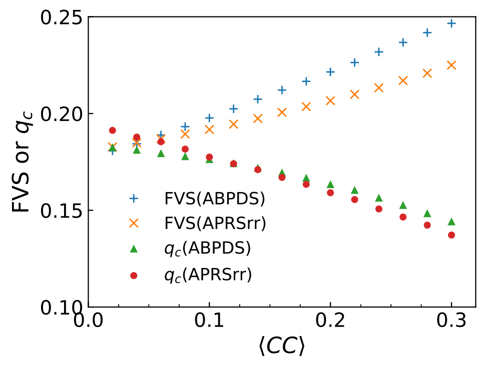
$$\mathcal{N}_c'(t) = \{i | i = \arg\min_j (|\Gamma(j) \cap \mathcal{N}_o(t)| - |\hat{\Gamma}(j)|), \forall j \in \mathcal{N}_c(t)\} \tag{3.34}$$

and choose the new occupied node through $\mathcal{N}_c'(t)$ (see also *line* 7 in Algorithm 3.2). Obviously, there is no any cycle in the occupied network if all occupied nodes satisfying $|\Gamma(i) \cap \mathcal{N}_o(t)| -$

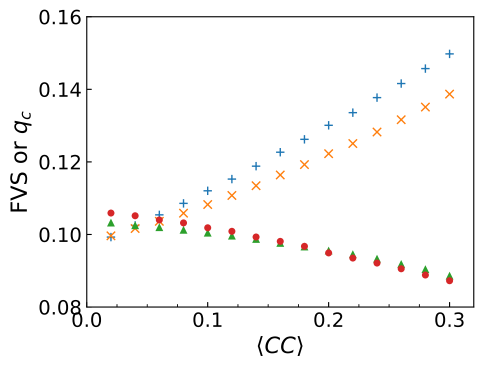
$|\hat{\Gamma}(i)| = 0$. Besides, another strategy adopted for ARRS and APRSrr regarding the FVS problem is: if the new occupied node i has $|\Gamma(i) \cap \mathcal{N}_o(t)| - |\hat{\Gamma}(i)| = 1$, which means that there would be one and only one cycle, then we swap places of i and one of its neighbors which are in the same component before the occupation of i. Obviously, the number of such neighbors is 2. Note that for bounded-size methods, we here only show ABonS1 as an example. One can employ the same strategy to extend ABonS1q, ABonS2, APRSs1, and APRSs1q for the FVS problem.

Networks	ABPDS	ABonS1fvs	APRSrrfvs	ARRSfvs
Power	516	**491**	***485.6***	**487.65**
CA-GrQc	1449	**1434**	***1426.2***	**1427**
p2p-Gnutella08	*1256*	1289	1276.85	1281
as-733	216	**209**	***208***	**208.6**
Scottish	444	445	***436.35***	**438**
CA-AstroPh	8626	**8549**	**8529.65**	***8525.8***
CA-CondMat	8323	**8245**	**8230.2**	***8228.4***
hep-th	12344	**12196**	***12097.45***	**12103.15**
Cit-HepPh	15405	**15235**	***15133.45***	**15139.8**
Email-Enron	7853	**7771**	**7748.7**	***7746.35***
p2p-Gnutella31	*9279*	9594	9459.35	9492.95
loc-Gowalla	38841	**37911**	***37690.2***	**37739**
Email-EuAll	1187	1226	***1182.8***	1193.65
com-Amazon	85274	**82839**	**82364.55**	***82263.8***
web-Google	208876	**205964**	***205231.85***	**205435.45**
PAroad	194443	**181268**	***176535***	**177536.8**
Txroad	239909	**222446**	***217066.25***	**217823.05**
as-Skitter	228775	**228191**	***224356.65***	**225329.9**

Table 3.4: Results of FVS on the 18 real-world networks. An item in bold represents that the corresponding method can find a smaller FVS set than ABPDS. One in italic means that the associated method has the best performance on the related network among all those mentioned methods.



(a)



(b)

Figure 3.18: The critical threshold q_c and the fraction FVS versus the average clustering coefficient $\langle CC \rangle$ for ABPDS and APRSrr in (a) ER networks with $\langle k \rangle = 3.5$ and $n = 10^4$, and (b) BA networks with $\langle k \rangle = 4.0$ and $n = 10^4$.

Following the mark used in ABonS1q and APRSs1q, we here refer the associated methods for the FVS problem as ABonS1fvs, APRSrrfvs, and ARRSfvs. Table 3.4 represents their performance compared to ABPDS. As we can see, all three methods surpass ABPDS to have smaller FVS sets in most networks, especially APRSrrfvs which is better in 16 out of 18

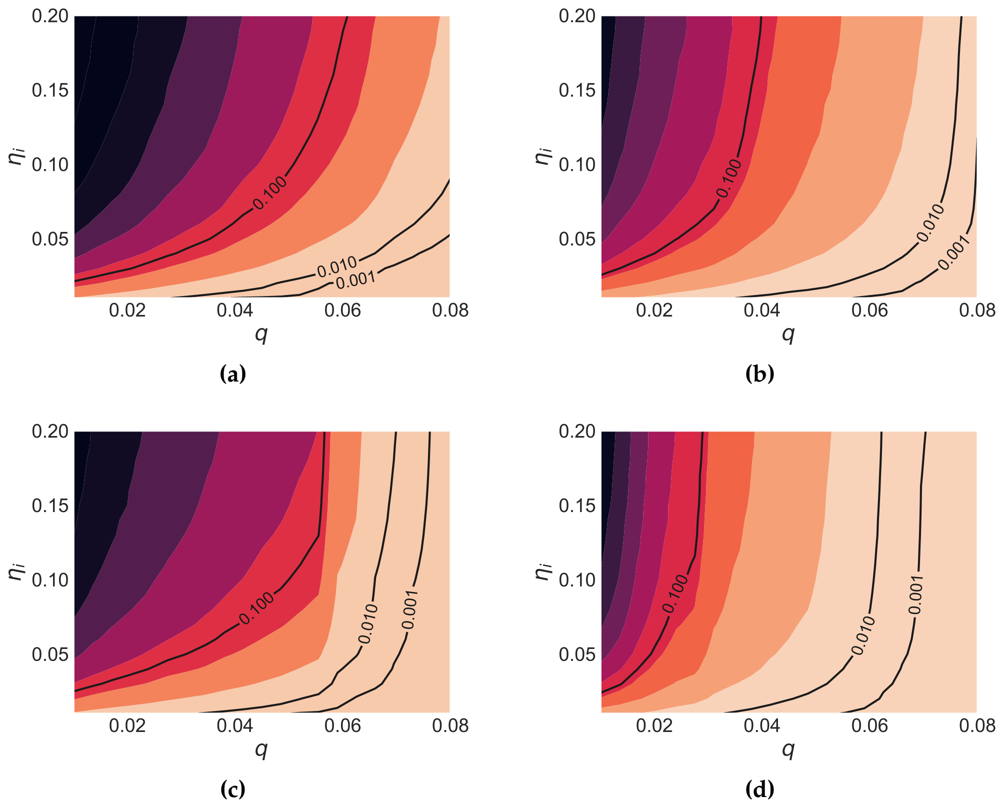
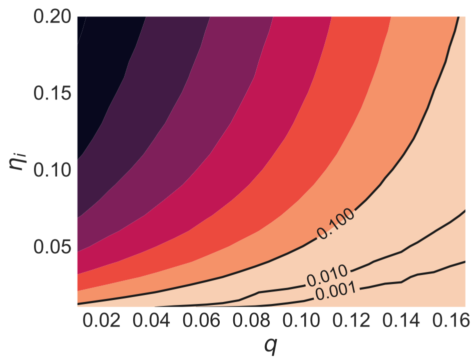


Figure 3.19: Contours of the average infected frequency $\langle \alpha_{\text{inf}} \rangle$ as a function of the immunized fraction q and the infected probability η_i on the Email-Enron network regarding (a) HubS, (b) ACIS, (c) AEIS, and (d) ARRS. In each figure, color from deep dark (i.e., top left) to light purple (bottom right) indicates that $\langle \alpha_{\text{inf}} \rangle$ changes from large to small. In addition, $\langle \alpha_{\text{inf}} \rangle = 0.001, 0.01$ and 0.1 are marked by the three solid curves, respectively.

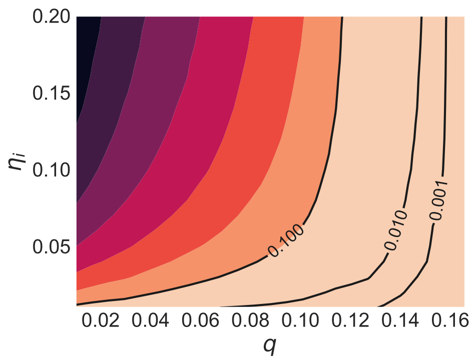
networks than ABPDS and holds the minimum in 12 among those methods. Besides, even ABonS1fvs also has better performance than ABPDS in 14 networks.

The different effectiveness of ABPDS on model networks and real-world networks (see Tables 3.3 and 3.4, and Figs. 3.16 and 3.17) motivates us to ask another question: how do local cycles influence the performance of ABPDS? To verify this, we consider the following way to enhance the clustering coefficients of both ER and BA networks: i) a network $G(\mathcal{N}, \mathcal{M})$ is generated based on the ER model or BA model; ii) randomly choose a node i and its two nearest neighbors u and v satisfying $e_{uv} = 0$, i.e., there is no edge between u and v; iii) further randomly select a node u' from $\Gamma(u) \setminus \{i\}$ and another one v' from $\Gamma(v) \setminus \{i\}$, where $u' \neq v'$ and $e_{u'v'} = 0$ must hold; iv) remove $e_{uu'}$ and $e_{vv'}$ from G, and add two new e_{uv} and $e_{u'v'}$ at the same time; v) repeat steps ii), iii), and iv) until the desired clustering coefficient is reached. In other words, we randomly select two nodes from $\partial\text{Ball}(i, 1)$ and further their corresponding neighbors from $\partial\text{Ball}(i, 2)$, and then replace old edges between those two layers with new edges in the same layer (refer to Fig. A.11). Apparently, during this enhanced process, the degree distribution would keep unchanged. Fig. 3.18 shows the related results, which again agree with our previous speculation, that is, comparing ABPDS with APRSrr, ABPDS is better in networks with rare local cycles while APRSrr has more advantage in those with a lot which most real-world networks own.

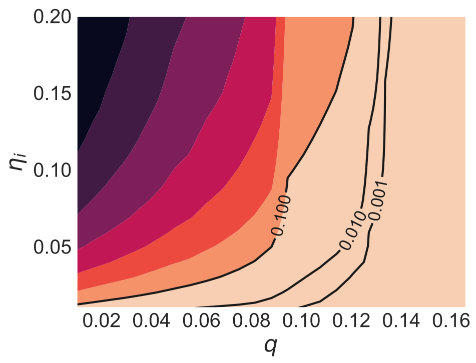
3.4.5.5 SIR results



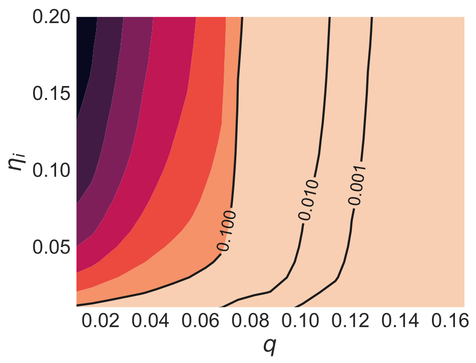
(a)



(b)



(c)



(d)

Figure 3.20: Contours of the average infected frequency $\langle \alpha_{\text{inf}} \rangle$ as a function of the immunized fraction q and the infected probability η_i on the loc-Gowalla network regarding (a) HubS, (b) ACIS, (c) AEIS, and (d) ARRS. In each figure, color from deep dark (i.e., top left) to light purple (bottom right) indicates that $\langle \alpha_{\text{inf}} \rangle$ changes from large to small. In addition, $\langle \alpha_{\text{inf}} \rangle = 0.001, 0.01$ and 0.1 are marked by the three solid curves, respectively.

We further validate the ability of ARRS to contain epidemics modeled by the SIR model (see also Section 3.1.2.1). In particular, we consider the following processes: i) for a given network $G(\mathcal{N}, \mathcal{M})$ and a sequence S (i.e., a specific method), we first remove (or immunize) q fraction of nodes from the network and assume that the epidemic could not occur on or pass through that group of nodes (see Section 3.1.2.3); ii) a node i is randomly chosen as the infectious source from those unimmunized nodes; iii) we then run SIR model on the remaining network until the epidemic dies out; iv) repeat ii) and iii) b times and count the number of times that a node gets infected, say b_j for node j; v) get the corresponding infected frequency $\alpha_{\text{inf}}(j) = b_j / b$ of j and the average one

$$\langle \alpha_{\text{inf}} \rangle = \frac{1}{n} \sum_{j \in \mathcal{N}} \alpha_{\text{inf}}(j)$$

over the whole network as well. Here $b = 10^5$ and $\eta_r = 0.05$ are conducted. Besides, we also verify the effective of ARRS by comparing with HubS, ACIS and AEIS. The associated results can be found in Figs. 3.19 and 3.20 where the Email-Enron network and the loc-Gowalla network are considered,

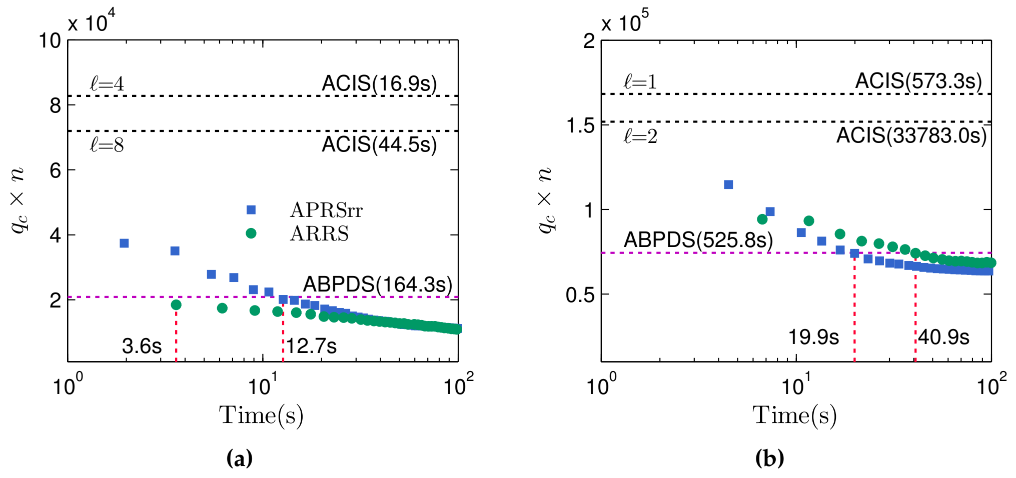


Figure 3.21: The running time (measured by second (s)) of ACIS, ABPDS, APRSrr (with $n_s(0) = 5$) and ARRS (with $n_s(0) = 10$) regarding critical threshold q_c on (a) the TXroad network and (b) the as-Skitter network. Thresholds of ACIS and ABPDS are characterized by the horizontal dashed lines, and the corresponding time is suspended around them. The values marked beside the vertical dashed lines are the point in time when APRSrr or ARRS begins to have smaller thresholds than both ACIS and ABPDS. All the results are obtained by averaging 20 implementations.

- Email-Enron: a network in which a node represents an email address, and an edge denotes that there is at least one mail transferred between the two addresses [93]. We choose it since some digital viruses may spread relying on it, which has bothered hundreds of thousands of people nowadays.
- loc-Gowalla: a location-based social network where misinformation might spread out.

As we can see from both Figs. 3.19 and 3.20, ARRS is much better than those compared methods, that is: i) for a specific q and η_i, nodes under the protection of ARRS have lower frequency to get infected, such as in the Email-Enron network where ARRS has improvements of 66.02%, 44.45%, and 63.74% accordingly against HubS, ACIS, and AEIS (obtained in the way similar to Eq. (3.33) by replacing F with $\langle\alpha_{\text{inf}}\rangle$) at $q = 0.042$ and $\eta_i = 0.100$; ii) for a specific q, networks under the protection of ARRS are more capable of resisting an epidemic, e.g., in the loc-Gowalla network where $\langle\alpha_{\text{inf}}\rangle = 0.050$ is reached at $0.020 < \eta_i < 0.030$ under HubS, $0.020 < \eta_i < 0.030$ under ACIS, $0.050 < \eta_i < 0.060$ under AEIS, and $0.190 < \eta_i < 0.200$ under ARRS when $q = 0.101$; iii) for a specific η_i, ARRS needs much less resource (i.e., the number of nodes) to immunize a network, for example, again in the Email-Enron network where $\langle\alpha_{\text{inf}}\rangle < 0.050$ under $\eta_i = 0.100$ is achieved by immunizing around $q = 0.066$, 0.052, 0.055, and 0.0380 fraction of nodes through HubS, ACIS, AEIS, and ARRS, respectively.

3.4.5.6 Running time

Since it is hard to explicitly analyze the computational complexity of those proposed methods, we here put APRSrr and ARRS [21] as well as CI and BPD (open-source codes written

[21]If viewing the time complexity of UF algorithm (see Algorithm A.2) as $O(1)$, then the time complexity of APRSrr and ARRS roughly follow $O(\hat{T}n_s n\langle k\rangle)$

by either C or C++ program) in the same simulated environment [22] and compare their time consumptions considering the two largest networks, i.e., the TXroad network (with maximal degree 12) and the as-Skitter network (with maximal degree 35455). As illustrated in Fig. 3.21, both APRSrr and ARRS acquire smaller thresholds than CI and BPD within quite a short time, in particular, for example, RR only takes 3.6s to obtain a better result than ACIS and ABPDS in the TXroad network. Note that the running time of CI and BPD reported here may be as a reference but not as a standard. With respect to those bounded-size methods, i.e., ABonS1, ABonS1q, ABonS2, APRSs1, and APRSs1q, they are of course very fast too. For instance, ABonS1q with $b = 3.0$ achieve $q_c \times n = 62665$ within 7s in the as-Skitter network and gets $q_c \times n = 15638$ for the TXroad network within 2s. Meanwhile, in the same condition, APRSs1q has 62302 and 15636 within 8s and 2s, respectively.

3.5 Evolutionary Framework for the Identification of Influential Nodes

The results from the previous section (Section 3.4) have shown us how effective those basic methods are. That is, on the one hand, ARRS has the capability to obtain comparable results to ABetS. Meanwhile, it needs much less time (see Fig. 3.12). On the other hand, compared to several state-of-the-art strategies which have their own advantages in different networks and over different metrics, almost all basic methods could surpass them in both performance and time consumption on almost all networks (see Tables 3.2 and 3.3, and Fig. 3.21), in particular APRSrr and ARRS. Yet, results in Fig. 3.12 also show that even ARRS still cannot outperform ABetS in some networks, which motivates us to further investigate why and how. That is, why are those basic methods still less effective than ABetS? And how can we improve those methods again, at least some of them, to truly approach or overtake ABetS in all kinds of networks? The first question might be ascribed to the fact that the related problem is a NP-hard problem[23]. But it is not such easy to answer how. Thus in what follows, we will mainly focus on how and give the answer of whether it is possible to develop better strategies [**L2**].

3.5.1 The advantage and disadvantage of ARRS

It is not very difficult to observe that all those basic methods in Section 3.4 rely on an initial sequence. This sequence could be random but also could be a one drawn from an existing method. Hence, we first need to figure out what function the initial sequence plays. Fig. 3.22 shows the effects of initial sequences on the proposed basic methods (see Section 3.4), and gives us the following observations. With regard to F (Fig. 3.22a), all those basic methods could benefit from a better initial sequence (e.g., comparing the results regarding AHubS and ABetS). But ABonS1, ABonS1q, ABonS2, APRSs1, and APRSs1q are easy to saturate (comparing the results among APagS, ACIS, and ABetS), that is, the improvement

[22]Lenovo NeXtScale nx360M5, Xeon E5-2667v3 8C 3.2GHz, Infiniband FDR14 with 8 threads.

[23]One can think about a special case regarding q_c that after the removal of part of nodes the remaining network only contains isolated nodes, which corresponds to the minimum vertex cover problem, a famous NP-hard problem.

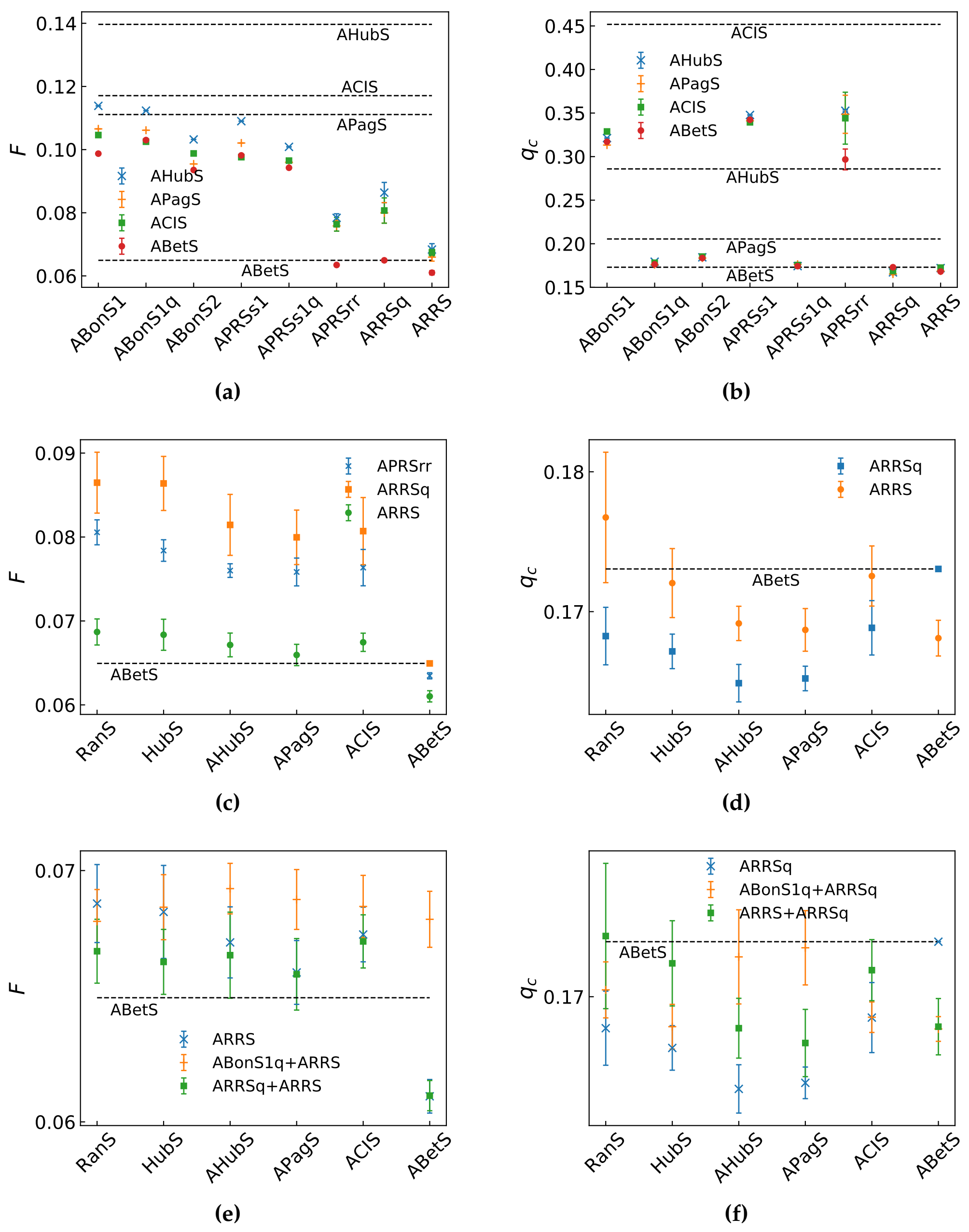


Figure 3.22: Effects of different initial sequences on F and q_c on the protein-protein interaction network. (a) F of the proposed basic methods over different initial sequences, i.e., HbuS, APagS, ACIS, and ABetS., where the dashed line marks F obtained based on the related initial sequence directly. (b) q_c. (c) F of different initial sequences regarding ARRSrr, ARRSq, and ARRS. (d) q_c of different initial sequences regarding ARRSq and ARRS. (e) and (f) Combinational results of the basic methods. Each error bar is obtained by calculation the standard deviation over 20 IIs.

of the initial sequence could not boost them again. On the contrary, APRSrr, ARRSq, and ARRS are only slightly influenced by the differences of AHubS, APagS, and ACIS and could be further improved by ABetS, where ARRSq is a method by simply replacing the global goal function of ARRS with $\xi_g(\cdot) = q_c$. Basically, for example, if ARRS can outperform a method with a random initial sequence, then the sequence based on that method would only have a very limited impact on ARRS. With respect to q_c (Fig. 3.22b), it seems to depend only on those basic methods themselves. But the results in Fig. 3.22d surprisingly show that the best q_c is obtained based on AHubS instead of ABetS. In fact, when initializing with

ABetS, ARRSq would be stuck at some value and has q_c even worse than the one based on RanS. Meanwhile, however, comparisons among other initial sequences illustrate that a start of smaller q_c is more likely to guide ARRSq to find a better q_c. Further regarding F of ARRS (Fig. 3.22c and 3.22e), obviously, an initial sequence with a smaller F conduces to a better result of F. Besides, the results (Fig. 3.22e) of the combination of ARRSq and ARRS shows that ARRSq could truly slightly improve the performance of ARRS, which indicates that ARRS might benefit from ARRSq. By contrast, as we observed in Fig. 3.22d, Fig. 3.22f again shows that an initial sequence with a small F might be not good to optimizing q_c. Moreover, Fig. A.21 (Appendix A.2.14) gives the comparisons of q_c obtained through ARRS and ARRSq on the 18 real-world networks (see Table 3.1). Obviously, in some networks, ARRS has smaller q_c than ARRSq, but ARRSq holds the best in most.

To sum up, we reach the following assumptions and problems:

(APa) Both ARRS and ARRSq can benefit from the initial sequence.

(APb) Usually, a better start would guide ARRS to find a smaller F.

(APc) But shallow strategies only have a limited impact on ARRS. Thus, problems arise as to why is the F based on ABetS much better than those based on others? And could we find a way to, at least, shorten the gap between them?

(APd) Meanwhile, a sequence with a smaller q_c might guide ARRS to find a better F.

(APe) By contrast, ARRSq would benefit more from a sequence with a larger F.

(APf) But still, a smaller F sometimes could result in a better q_c. Therefore, how should we balance F while optimizing q_c?

3.5.2 Pruning an existing method

We first consider (APc). From Section 3.4.4 we know that ARRS keeps or eliminates a new sequence S' based on the global goal function $\xi_g(\cdot)$ (see also Algorithm 3.2). This strategy, to some extent, is inefficient since it considers the whole sequence. That is, during an iteration, one part of S' maybe lead to a better result while another part might make the result worse. And if the worse one weighs more on F, then S' would be eliminated. Actually, such conflict becomes more and more frequent as T increases[24]. Indeed, the combination of n_s and r_u could overcome that and facilitate ARRS to have the ability to optimize a sequence locally (such as, a large n_s in tandem with a small r_u). But that ability is always limited.

Now assuming that there is a sequence S regarding a given network $G(\mathcal{N}, \mathcal{M})$ where each element of S corresponds to a unique node in $\mathcal{N}$, we define a slice of S as

$$S_p(t_1, t_1') = S[t_1 : t_1'], \tag{3.35}$$

[24]It is also the main reason that ARRS is stuck in some local optimization if it is initialized by some shallow strategies.

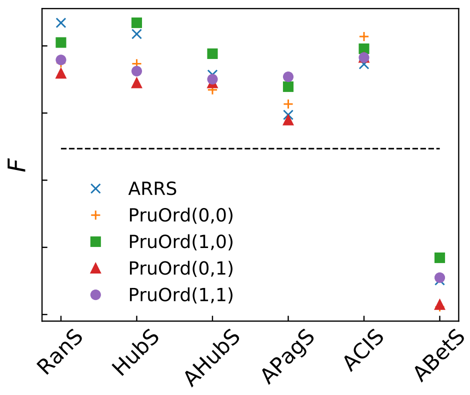
where $t_1 \leqslant t_1'$ are two given integers (refer to Section 2.1.2 for the definition of $S[t_1 : t_1']$). The corresponding local average of $\mathcal{G}_p(q)$ follows

$$F(S_p(t_1, t_1')) = \sum_{q=t_1/n}^{t_1'/n} \mathcal{G}_p(q). \tag{3.36}$$

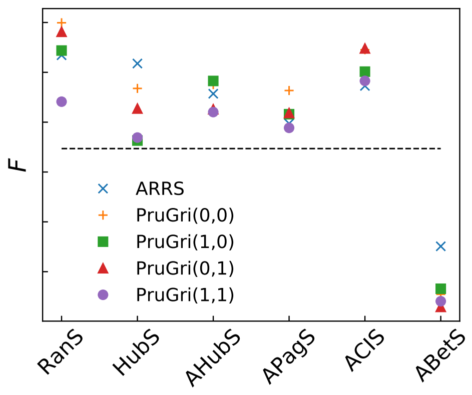
Then, one can easily observe that $S_p(t_1, t_1')$ would be independent of $S_p(t_2, t_2')$ if $t_1' < t_2$ or $t_2' < t_1$, that is, $F(S_p(t_1, t_1'))$ would keep unchanged, no matter what permutations $S_p(t_2, t_2')$ takes, vice versa. Note that here those elements in both $S_p(t_1, t_1;)$ and $S_p(t_2, t_2')$ are fixed, namely, they only change their orders internally.

Holding the above property, we can then try the following processes to prune a given sequence S, i.e., locally optimize an existing method. That is, i) randomly pick up two integers, and assign the small one to t_1 and the other one to t_1', respectively; ii) run ARRS on $S_p(t_1, t_1')$; and iii) repeat those two steps a number of times. Now another problem arises as to how should we choose those two integers?

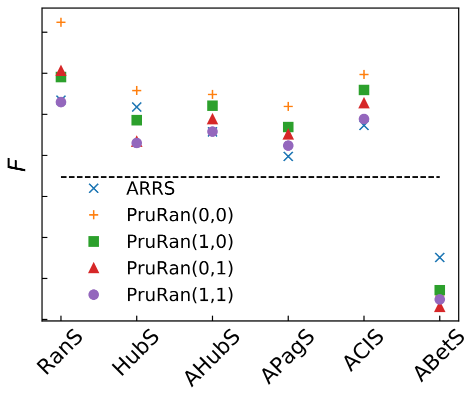
3.5.2.1 PruOrd



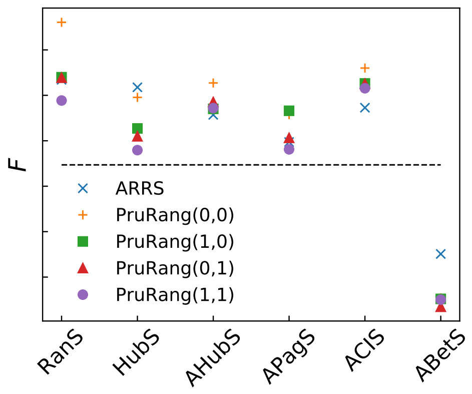
(a)



(b)



(c)



(d)

Figure 3.23: Performance of PruOrd, PruGri, PruRan and PruRang regarding F of different initial sequences compared to ARRS and ABetS (dashed lines). (a) PruOrd. (b) PruGri. (c) PruRan with uniformly random selection of t_1. (d) PruRang considering Eq. (3.38). The difference of F between every two ticks is 0.002. Each result is the mean of 20 IIs.

The first strategy still follows a routine similar to ARRS and we call it prune orderly (PruOrd). Specifically, letting $\hat{T}_p$ be the total pruning times and T_p be the current pruning iteration, PruOrd considers the following steps to achieve one round of pruning[25]:

1) set $a(T_p) = \lfloor a(0)nT_p \rfloor$, $b_1(T_p) = \lfloor b_1(0)(1-\delta_{b_1})^{T_p} \rfloor$ and $b_2(T_p) = \lfloor b_2(0)(1-\delta_{b_2})^{T_p} \rfloor$;

2) let $t_1 = a(T_p)$ and $t'_1 = t_1 + b_2(T_p)$;

3) run ARRS on $S_p(t_1, t'_1)$;

4) let $a(T_p) = a(T_p) + b_1(T_p)$;

5) repeat 2), 3), and 4) until the termination is reached.

Then, let us look into the reason that we have those control parameters. From Section 3.1 and 3.4 we know that the order parameter $\mathcal{G}_p(q)$ decrease as q increases and approaches 0 after the critical threshold q_c, i.e., $q > q_c$. Hence, when S becomes more and more orderly, the effect of a node from the subcritical regime on F would be less and less, especially those at the beginning of S[26]. Therefore, we use $a(T_p)$ to lock those nodes, and the number of such nodes increases as the rise of T_p. With respect to $b_1(T_p)$ and $b_2(T_p)$ which satisfies $b_1(T_p) < b_2(T_p)$, their combination ensures the interactions among different groups, whose significance have been demonstrated in Section 3.4. And similar to ARRS, we let both of them decrease along with the increase of T_p.

To validate the effectiveness of PruOrd, $\hat{T}_p = 10^4$, $a(0) = 0.9/\hat{T}_p$, $b_1(0) = 0.1n$, $\delta_{b_1} = 0.0001$, $b_2(0) = 0.3n$ and $\delta_{b_2} = 0.0005$ are conducted. Besides, since ARRS only runs on part of S, the following changes of its configuration are also considered: $\hat{T} = 20$, $r_u(0) = (t'_1 - t_1)/n$ and $\delta_{r_u} = \delta_{n_s} = 0.5$. Fig. 3.23a shows the corresponding results, where different combinations of $r_u(T)$ and $n_s(T)$ are studies as well. That is, for example, PruOrd(α_1, α_2) in which $\alpha_1 = 1$ corresponds to $r_u(T) = r_u(0)$ and $\alpha_2 = 1$ represents that $n_s(T)$ is randomly chosen from $[1, n_s(0)]$, otherwise, they follow the strategies same as ARRS. Moreover, if $\alpha_2 = 1$, we also associate $n_s(0)$ with T_p, e.g., here we let $n_s(0) = n_s(0) + 1$ if F does not have any improvement for 10 rounds.

As we can see from Fig. 3.23a, PruOrd truly works, in particular PruOrd(0,1). And its performance relies on both α_1 and α_2. Through comparing PruOrd(0,1) or PruOrd(1,0) with PruOrd(1,1), one can easily find that either $\alpha_1 = 0$ or $\alpha_2 = 1$ facilitates better results and PruOrd(0,1) accounts for the best. Besides, better performance could also be achieved by tuning those control parameters. But it is usually difficult to find the optimal configuration, and different networks might need a different one. Nevertheless, PruOrd converges very fast and could have a better result within 10 iterations than those compared methods that we mentioned in Section 3.4.

[25]Note that again $a(T_p)$, $b_1(T_p)$ and $b_2(T_p)$ are temporal parameters that we employ to help explain a method or strategy. Hence, one should refer to the specific place to check their associated meanings.

[26]Note that here a percolation process is considered instead of an attack process.

3.5.2.2 PruGri

The difficulty in managing those control parameters motivates us to simplify PruOrd. Hence, we have another strategy which prunes S based on a grid search (PruGri). In detail, PruGri conducts a round of pruning in the following ways:

1) given boundaries b'_l and b'_u satisfying $b'_u - b'_l > 0$, $b'_l > 0$ and $b'_u < n$;

2) let $a(T_p) = 0$ and get $b_2(T_p)$ by randomly picking up an integer from $[b'_l, b'_u]$;

3) set $t_1 = a(T_p)$ and $t'_1 = t_1 + b_2(T_p)$;

4) run ARRS on $S_p(t_1, t'_1)$;

5) let $a(T_p) = a(T_p) + b_2(T_p)$;

6) repeat 3), 4) and 5) until the termination is reached.

In this manner, we only need to tune b'_l and b'_u excluding those of ARRS. Note that the random selection of $b_2(T_p)$ achieves the similar goal that the combination of $b_1(T_p)$ and $b_2(T_p)$ has in PruOrd. And one can of course let $a(T_p)$ follow the way same as PruOrd. But here our purpose is to make PruGri as simple as possible. Besides, another advantage that PruGri has is that it could be paralleled since $b_2(T_p)$ is fixed for each round and $a(T_p)$ increases in steps of $b_2(T_p)$. As illustrated in Fig. 3.23b where $b'_l = 100$ and $b'_u = 0.2n$ are employed, PruGri(1,1) has the best performance on average. And compared to PruOrd (Fig. 3.23a), PruGri could benefit more from existing strategies except for a random sequence (i.e., RanS). Besides, considering ABetS, all four can find smaller F than ARRS. One can also compare PruOrd and PruGri directly, that is, count the number of ticks regarding F since the difference between every pair of ticks is same.

3.5.2.3 PruRan and PruRang

Though PruGri could find a really good result, it is sometimes inefficient in time consumption because it takes a grid search. Thus, we reach the third strategy which prunes a sequence by repeatedly conducting ARRS on a random slice of S (PruRan). Specifically,

1) give the boundary b'_u satisfying $0 < b'_u < n$;

2) assign a random integer $a(T_p)$ drawn from $[1, n]$ to t_1 and another one $b_2(T_p)$ from $[1, b'_u]$ for $t'_1 = t_1 + b_2(T_p)$, where $t'_1 < n$ must hold;

3) run ARRS on $S_p(t_1, t'_1)$;

4) repeat 2) and 3) until the termination is reached.

Indeed, PruRan is similar to PruGri but it gives us the freedom to control t_1. For instance, rather than randomly choose t_1 uniformly, we here consider a strategy similar to Eq. (3.27), that is, repeatedly sample a random integer $a(T_p) \in [1, n]$ until it is successfully assigned to t_1, which follows

$$t_1 = a(T_p) \;\text{ if } a(T_p) > b_1(T_p) \text{ or with a probability } A_p = e^{\frac{-[a(T_p)-b_1(T_p)]^2}{2[b_2(T_p)]^2}}. \tag{3.37}$$

As we mentioned during the introduction of PruOrd, the supercritical region makes the main contribution to F, which is also the reason that we have Eq. (3.37). For example, if letting $b_1(T_p) = q_c$, then Eq. (3.37) is less likely to consider nodes that is in the beginning of S. Further, if $b_2(T_p) \to 0$, then Eq. (3.37) degenerates to the original PruRan. Here we consider the settings of

$$b_1(T_p) = q_c n \text{ and } b_2(T_p) = \frac{q_c n}{b_2(0)\alpha^{T_p}} \tag{3.38}$$

with $\alpha > 1$ to ensure that $a(T_p)$ gradually converges to q_c from the subcritical regime as T_p increases. Note that q_c would also change with the rise of T_p.

To distinguish the one considering Eq. (3.37) from the original PruRan, we mark it with a 'g', namely, PruRang. Figs. 3.23c and 3.23d illustrate the comparisons of PruRan (with $b'_u = 0.2n$), PruRang (with $b'_u = 0.2n$ and using Eq. (3.38) with $b_2(0) = 0.01$ and $\alpha = 1.001$) and other methods. As we can see from them, both PruRan and PruRang hold the best performance on average by setting $\alpha_1 = \alpha_2 = 1$. And PruRang is more capable of approaching PruGri.

3.5.2.4 Summary

To sum up, considering the performance based on different initial sequences, those strategies could be truly possible to shorten the gap between ABetS and other shallow methods, such as PruGri(1,1) initialized with HubS. Besides, it is worth mentioning that all those 4 tested strategies have the ability to surpass ABetS, even though they are based on a shallow initial sequence. For instance, PruGri(1,1) has better performance in 10 out of 20 results than ABetS for the case of the initial sequence drawn based on HubS. In addition, in what follows, if there is no specific explanation, all those 4 strategies are considered with $\alpha_1 = \alpha_2 = 1$ where PruGri, PruRan, and PruRang have their best performance on average. Moreover, in the below sections, since we only aim to detect whether a new strategy works, $n_s(0) = n_s(0) + 1$ would be processed if 10 rounds of pruning result in no change of F, and a method terminates if either $n_s(0) > 50$ or $\hat{T}_p$ is reached. These two changes would speed up those strategies but reduce their effectiveness certainly.

3.5.3 Effects of the critical threshold on the average order parameter

We then consider (APd). From Fig. 3.22c we know that APRSrr is better than ARRSq, and ARRS has the best among them with respect to F. But the results in Fig. 3.24b show that ARRSq brings about comparable performance as ARRS does. And PruGri armed with ARRSq even has the ability to really approach ABetS (see Fig. 3.24a). Indeed, ARRSq ignores F when optimizing q_c, and thus always leads to a disordered sequence to F. In this case, those pruning strategies would work very well. Nevertheless, if q_c truly has a positive impact on F, then the pruning strategies would benefit a lot from some basic methods such as ABonS1q and APRSs1q, which are usually very fast to get a result. Another reason might be that the disorder of a sequence plays some role. We will discuss this later.

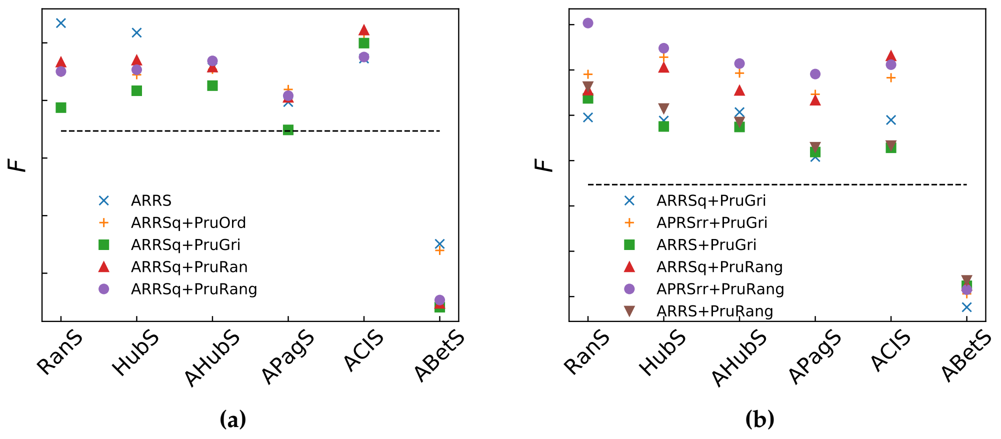


Figure 3.24: Influence of q_c on F considering that (a) has the configuration same as the one used in Fig. 3.23 and (b) conducts the new one, i.e., the pruning process would terminates when either $n_s(0) > 50$ or $\hat{T}_p$ is reached.

3.5.4 Optimization of the critical threshold

We further study (APe) and (APf). Similar to Fig. 3.23, the performance regarding those four pruning strategies are exhibited in Fig. 3.25. Compared to PruOrd, PruOrdq uses $\xi_g(\cdot) = q_c$ instead of F on the slice where critical threshold is, otherwise, still considers $\xi_g(\cdot) = F$. Others take the same change except for PruRangq. Since we are trying to optimize the threshold q_c, the condition $a(T_p) > b_1(T_p)$ of Eq. (3.37) would become useless. Hence, it is removed in PruRangq, that is, PruRangq chooses all $a(T_p)$ following some probability. In addition, we conduct Eq. (3.38) in the below way

$$b_1(T_p) = q_c n + b_1(0) - \alpha_3 T_p \text{ and } b_2(T_p) = \frac{b_1(T_p)}{b_2(0)\alpha^{T_p}} \tag{3.39}$$

to ensure that $a(T_p)$ has more chances choosing a value larger than q_c, where $\alpha_3 > 1$. As illustrated in Fig. 3.23 where $b_1(0) = 0.1n$ and $\alpha_3 = (b_1(0) - 100)/\hat{T}_p$ are conducted for PruRangq, all those four strategies share similar performance to ARRSq, which means that they do not really work, at least when they are initialized with those shallow methods. But surprisingly, ABetS guides them to have much better results than the one of ARRSq. Meanwhile, the optimal configurations regarding α_1 and α_2 are accordingly 0 and 1.

3.5.4.1 Effects of the average order parameter on the critical threshold

Now we further study the influence of F on q_c. Results from Table 3.3 tell us that the optimization of F truly conduces to the acquirement of really good q_c. Meanwhile, we also learn that ARRSq has a smaller q_c in most networks than ARRS from Fig. A.21. Moreover, ABetS would lock ARRSq (see Fig. 3.22d) and AHubS results in the best q_c. Besides, even RanS leads to a better result than ABetS. Therefore, we naturally ask what would happen regarding q_c if we change the way to control F.

A straightforward try is to ignore F, i.e., only the slice which q_c belongs to is dealt with. Since having to ensure that every node has the possibility to be checked, we here mainly consider an variant of PruRangq, say PruRangqv1, that is, choose t_1 and t_1' following

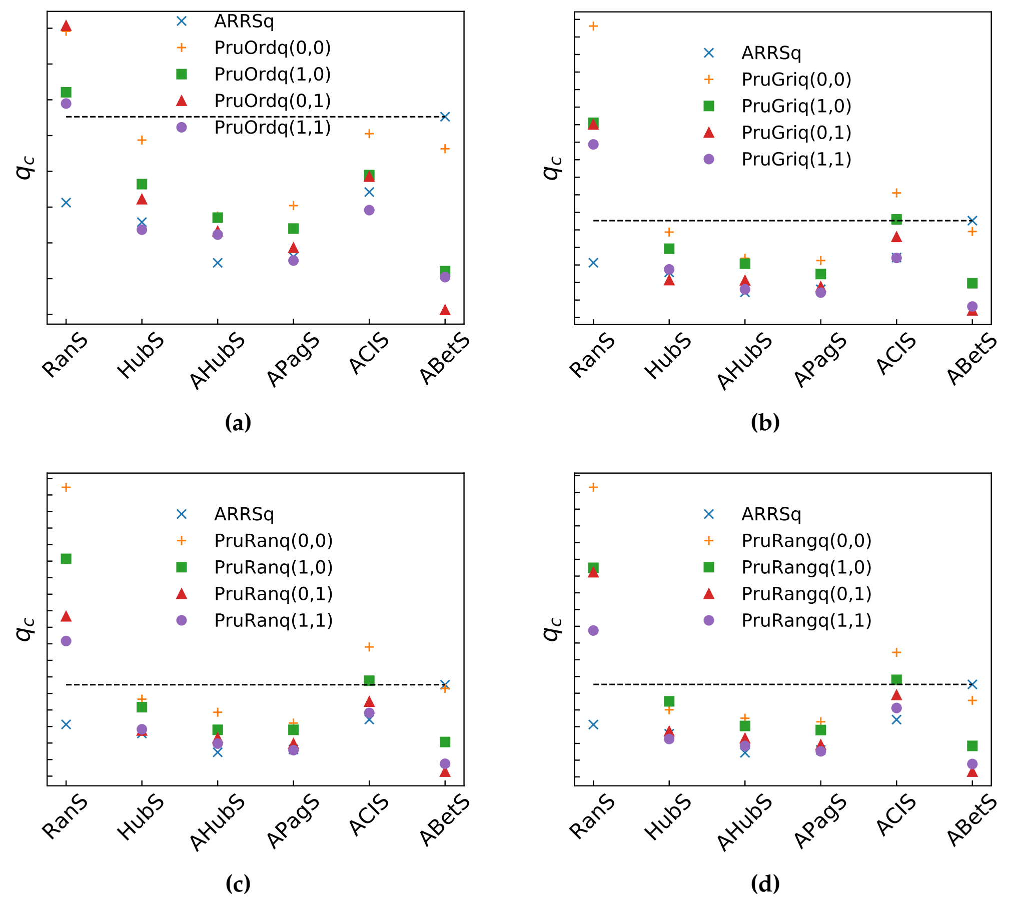


Figure 3.25: Performance of PruOrdq, PruGriq, PruRanq, and PruRangq regarding q_c of different initial sequences compared to ARRSq and ABetS (dashed lines). (a) PruOrdq. (b) PruGriq. (c) PruRanq with uniformly random selection of t_1. (d) PruRangq considering Eq. (3.38). The difference of q_c between every two ticks is 0.002. Each result is the mean of 20 IIs.

$t_1 < q_c < t_1'$, and $t_1 = a(T_p)$ (see Eqs. (3.37) and (3.38)) holding

$$b_1(T_p) = q_c n \text{ and } b_2(T_p) = \frac{n}{b_2(0)T_p}. \tag{3.40}$$

Apparently, $b_2(T_p)$ decreases as T_p increases (see Fig. A.22 in Appendix A.2.14), which indicates that $a(T_p)$ has higher probability to have a value closed to q_c with the rise of T_p (based on the assumption that the sequence becomes more and more orderly). Besides, one can tune $b_2(0) > 0$ to control the convergence rate of $a(T_p)$. Since the whole sequence is likely to be considered in the beginning, PruRangqv1 with $\alpha_1 = 0$ always has better performance than the one with $\alpha_1 = 1$. But still, this strategy with high probability has a local optimal solution because it could not effectively mix the whole sequence (i.e., the tail and head might not be visited often). Nevertheless, PruRangqv1 converges very fast and could obtain a better result than most state-of-the-art methods (those in Section 3.3) using much less time.

Another way of ignoring F is to disturb those groups which q_c does not belong to. Recall that the local goal function $\xi(\cdot)$ is designed to minimize $\mathcal{G}_p(q)$ (see Eq. (3.25)). Hence, the disorder of those groups is in a way equivalent to ignoring F. To achieve that, we only need to slightly modify PruRangq, that is, randomly permute a slice where q_c does not locate, which we call PruRangqv2.

The third way is loosing F on those groups which q_c does not belong to. From the design of ARRS we know that a small n_s cannot further optimize a sequence locally regarding F. Therefore, if n_s is fixed at a small value on all groups excluding the one that q_c belongs to, then somehow this strategy could benefit from both order and disorder of F, and the order of F is weaker than PruRangq, which we name PruRangqv3.

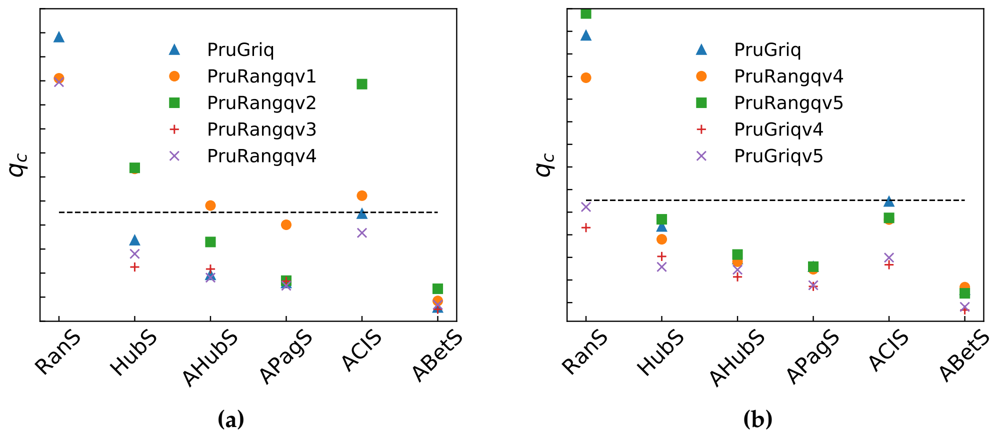


Figure 3.26: Performance of PruRangqv1, PruRangqv2, PruRangqv3, PruRangqv4, PruRangqv5, PruGriqv4 and PruGriqv5 regarding q_c of different initial sequences compared to PruGriq and ABetS (dashed lines). The difference of q_c between every two ticks is 0.002. Each result is the mean of 20 IIs.

The forth way still focuses on F of those groups that q_c does not belong to but chooses a group with some probability relying on F. Considering ARRS on a slice $S_p(t_1, t_1')$ that q_c does not locate, one can accept a new slice with a probability following either

$$\frac{S_p(T-1)}{S_p(T-1)+S_p(T)} \text{ or } \frac{S_p(T)}{S_p(T-1)+S_p(T)}, \tag{3.41}$$

and we accordingly call them PruRangqv4 and PruRangqv5. Obviously, PruRangqv4 would be more likely to choose a new slice if it has smaller F than the old one, while PruRangqv5 would reject it with a higher probability.

The corresponding results with respect to these strategies are shown in Fig. 3.26, in which the early termination is conducted, and PruGriqv4 and PruGriqv5 are two strategies based on PruGriq and Eq. (3.41), respectively. Besides, to verify them, PruGriq is chosen as a comparison, which accounts for the best on average in Fig. 3.25. Obviously, all those strategies based on PruRangq heavily rely on the initial sequence (comparing RanS and others). Specifically, in Fig. 3.26a, only PruRangqv4 has better performance than PruGriq, which indicates that F truly has influence on q_c. Meanwhile, from Fig. 3.26b, we learn that both PruRangq and PruGriq based on Eq. (3.41) are more effective than PruGriq, which means that the disorder of F also has an impact on q_c. It is worth noting that PruGriqv5 is better than PruGriqv4 if the initial sequence is based on HubS, which is very important because HubS is much easier to compute compared to APagS, ACIS, ABetS, and many others. This is also the reason that we choose PruGriqv5 in [**L2**].

3.5.4.2 Mutation operators

The influence of disorder of F on q_c motivates us to further introduce the mutation operators from the genetic algorithm. We conduct them on both S_p and S, i.e., local and global mutations. In detail, both of them, at each time, equally choose one with some probabilities from the following six mutation operators to produce the corresponding sequence:

i) the displacement mutation (DM) operator [103] usually randomly selects a fragment that would be moved from the sequence and eventually inserted in a random place;

ii) the exchange mutation (EM) operator [104] aims at choosing two nodes in the sequence at random and then exchanging them (a similar strategy could be found in ref. [5]);

iii) as for the insertion mutation (ISM) operator [103, 105], one random node is moved out the sequence and placed at a random position afterwards;

iv) the simple inversion mutation (SIM) operator [106] selects randomly two cut points in the sequence, and then reverses the fragment between these two cut points;

v) on the basis of SIM, we slightly change it by narrowing the cut points (S-SIM). Namely, the random selection happens in a narrow range;

vi) the inversion mutation (IVM) [107] operator works similarly to the DM. It also randomly selects a fragment, removes it from the sequence, and then inserts it in a randomly selected position, however, in the reversed order.

The corresponding performance can be found in Fig. A.23 in Appendix A.2.14.

3.5.5 Initialization based on graph partitioning

We now focus on (APa) and (APb). Those verifications in Figs. 3.22, 3.24, 3.25 and 3.26 have shown that the initial sequence truly plays an important role for both optimizations of F and q_c, especially ABetS. Therefore, here we further introduce another initial strategy based on graph partitioning regarding nodes [108], i.e., node (vertex) separator, which could achieve really good results particularly in networks of large size. Specifically, for a given network $G(\mathcal{N}, \mathcal{M})$, METIS[27] [109] obtains a node separator through the following steps.

1) Coarsen G to G_1 and further to $G_2, ..., G_i, G_j$ by merging nodes in $G, G_1, ..., G_i$, respectively. The coarsening strategy could be: i) randomly choosing an edge e_{uv}; ii) merging u and v if neither of them has been merged with other nodes; iii) updating the corresponding node set and edge set; and iv) repeating i)-iii) until there is no possibility for a new mergence. One can find more strategies from ref. [108], which basically follow the similar routine of i)-iv). Besides, it might also be possible to coarsen a graph based on those basic methods that we mentioned in Section 3.4, such as ABonS1[28].

[27]Particularly, we use metis-5.1.0 which one can find at `http://glaros.dtc.umn.edu/gkhome/views/metis`.
[28]Details regarding this are out of the scope of this thesis.

2) A graph partitioning algorithm is conducted on $G_j(\mathcal{N}_j, \mathcal{M}_j)$. The algorithm could be Kernighan-Lin algorithm[29] (KL) [110], which achieves a bisection of a network by: i) given the size of two groups, randomly assign nodes in $\mathcal{N}_j$ to the two groups; ii) considering a node u from one of the two groups and v from the other one, calculate the weight difference between before and after the interchanges of u and v, where the weight is the sum of all weights of edges connecting those two groups; iii) enumerate all such node pairs and interchange the one leading to the maximum difference; iv) repeat ii) and iii) until there is no difference larger than 0. Of course, KL is quite a simple strategy, but it is very effective [108]. One can also find some variants of it in ref. [108].

3) Uncoarsen $G_j(\mathcal{N}_j, \mathcal{M}_j)$ back to $G_i(\mathcal{N}_i, \mathcal{M}_i)$ by keeping the nodes in the two groups fixed (just simply expand merging nodes in the same group). After that, the KL strategy is conducted again on the two new groups. As ref. [108] mentioned, this round of KL would converge very fast.

4) Let $j = j - 1$ and $i = i - 1$, and repeat 2) and 3) until $G_i = G$. And then we have an edge separator, which is exactly the edge set of edges between the two groups.

5) The minimum vertex cover of the edge separator is considered as the node separator of G.

Since minimizing F is one of our main goals, we here acquire the new sequence S by repeatedly obtaining the node separator on the LCC. That is, i) obtain the LCC of the remaining network (which is usually the whole network in the beginning) as a subnetwork; ii) find the node separator of the subnetwork and remove them from the remaining network; and iii) repeat i) and ii) until the size of the LCC is less than 3.

We name the above strategy AMetisS since it is developed based on METIS. But AMetisS adds each node separator into S in random order, which obviously lacks effectiveness. Therefore, a straightforward way to overcome that drawback is to employ a greedy strategy similar to ABondS2 to reorder the node separator. In detail, one can achieve it through: i) remove the node separator from the given network; ii) a percolation process is conducted on the remaining network, e.g., greedily occupy those removed nodes following the sum rule, which we call AMetisSg. However, as we mentioned in Section 3.4, AMetisSg would easily result in local optimum and also locks the sequence, which means that the sequence could not jump out of the local optimal solution even using those pruning strategies. Therefore, AMetisS always guides such as PruGri to find a better final result than AMetisSg. In addition, from our testing results, both AMetisS and AMetisSg could sometimes obtain better results based on our basic method ABonS1q.

Fig. 3.27 shows the associated performance of AMetisS and AMetisSg as well as their combinations with PruGri, GruRan, and GruRang, etc., where GruRan and GruRang accordingly work in tandem with AMetisS and AMetisSg since AMetisS is locally disordered. As we can see from Figs. 3.27a and 3.27b, PruGri armed with ABetS still holds the smallest F among those methods. Besides, even though AMetisSg is more effective than AMetisS,

[29]This algorithm is also used for the community detection problem [1].

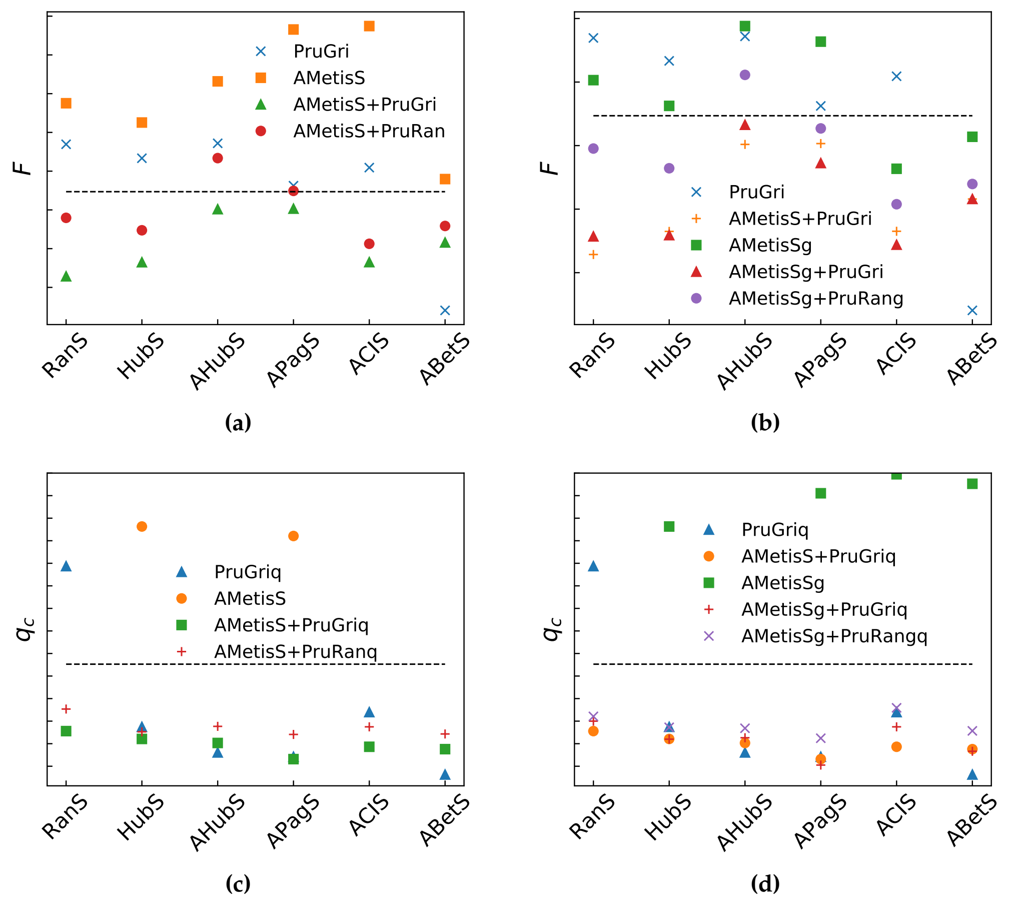


Figure 3.27: Performance of PruGri, GruRan, and GruRang regarding F and q_c of different initial sequences compared to PruGri, PruGriq, and ABetS (dashed lines). The difference of F and q_c between every two ticks is 0.002. Each result is the mean of 20 IIs.

their combinations with PruGri share similar results. Nevertheless, we finally achieve our goal: design strategies that could outperform ABetS. And more importantly, either RanS or HubS could guide these methods to achieve the goal. With respect q_c (see Figs. 3.27c and 3.27d), the minimum is also acquired by PruGriq with ABetS[30]. It is worth mentioning that PruRan and PruRang are much faster than PruGri if PruGri does not run in parallel, which is also the reason that we exhibit the results regarding PruRan and PruRang in Fig. 3.27. Nevertheless, both of them have comparable performance in most cases.

3.5.6 Evolutionary framework for the robustness and immunization problems

By now, we have verified and answered the assumptions and problems arisen in Section 3.5.1. The associated studies and strategies facilitate an evolutionary framework for the robustness and immunization problems, where we have finished

i) the initialization of populations, e.g., based on our basic methods or AMetisS;

ii) the introduction of mutation operators (see Section 3.5.4);

iii) the strategies for selections, e.g., Eq. (3.41);

[30]Note that one might get a good result of q_c through k-way partitioning, which is also out of the scope of this thesis.

iv) the management of diversity of populations, e.g., the sum and product rules.

Indeed, since there are several approaches to maintain the diversity of the sequence, we do not further introduce crossover operators. In what follows, we will first demonstrate the performance of the proposed framework in the way same as that we verify our basic methods. And more applications will be shown in the next chapter.

3.5.7 Applications

3.5.7.1 Data

Refer to Section 3.4.5.1.

3.5.7.2 Configurations of associated methods

- ARRS: the same parameters as those in Section 3.4.5.2 are conducted except that $\hat{T} = 20, r_u(0) = (t'_1 - t_1)/n$ and $\delta_{r_u} = \delta_{n_s} = 0.5$. Besides, $r_u(T) = r_u(0)$ if $\alpha_1 = 1$ and $n_s(T)$ is randomly chosen from $[1, n_s(0)]$ if $\alpha_2 = 1$. In addition, when $\alpha_2 = 1$, we also associate $n_s(0)$ with T_p, e.g., here we let $n_s(0) = n_s(0) + 1$ if F or q_c does not have any improvement for 10 rounds.
- PruOrd: $\hat{T}_p = 10^4$, $a(0) = 0.9/\hat{T}_p$, $b_1(0) = 0.1n$, $\delta_{b_1} = 0.0001$, $b_2(0) = 0.3n$ and $\delta_{b_2} = 0.0005$.
- PruGri: $b'_l = 100$ and $b'_u = 0.2n$.
- PruRan: $b'_u = 0.2n$.
- PruRang: $b'_u = 0.2n$, $b_2(0) = 0.01$ and $\alpha = 1.001$.
- PruRangq: same as PruRang and with $b_1(0) = 0.1n$ and $\alpha_3 = (b_1(0) - 100)/\hat{T}_p$.
- PruRangqv1, PruRangqv2, PruRangqv3, PruRangqv4 and PruRangqv5: same as PruRang.
- PruGriq, PruGriqv4 and PruGriqv5: same as PruGri.
- PruRangqv4m and PruRangqv5m: same as PruRang, and the local and global mutation probabilities are accordingly 0.1 and 0.3.
- PruGriqv4m and PruGriqv5m: same as PruGri, and the local and global mutation probabilities are accordingly 0.1 and 0.3.
- AMetisS and AMetisSg: NONE.
- EPruGri($\cdot$): PruGri with a specific initial sequence, e.g., EPruGri(HubS) means that we run PruGri starting from HubS.
- EPruGriqv5m($\cdot$): EPruGriqv5m with a specific initial sequence, e.g., EPruGriqv5m(HubS) means that we run EPruGriqv5m starting from HubS.
- Evol$_q$($\cdot$): EPruGriqv5m($\cdot$) with $b'_u = 0.1n$ followed by PruGriq.
- Evol$_F$($\cdot$): EPruGri($\cdot$) with $b'_u = 0.1n$.

The defaults of α_1 and α_2 are both taken 1, and all results regarding the above methods are the mean of 20 IIs if necessary.

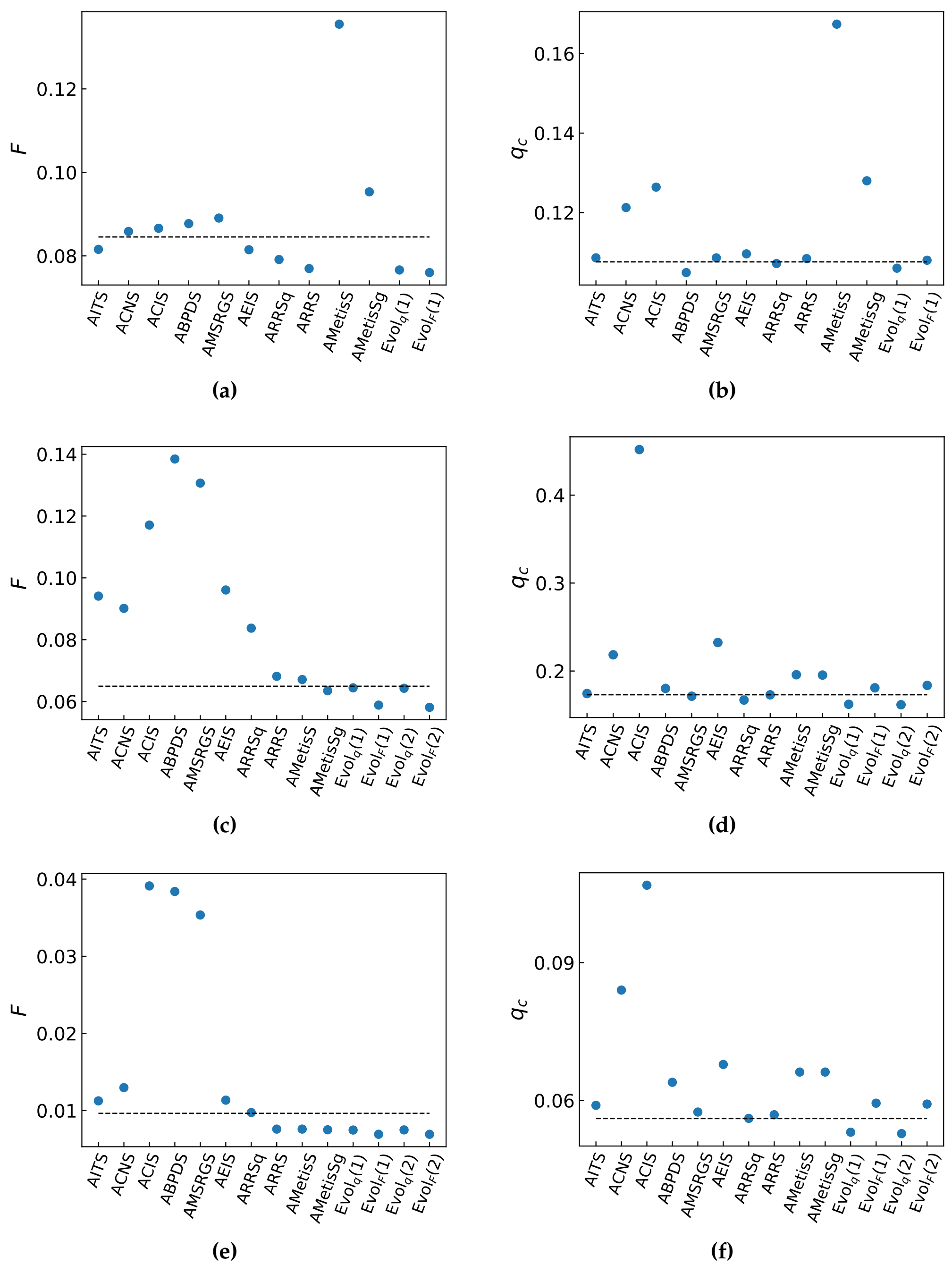


Figure 3.28: Comparisons among Evol$_q$, Evol$_F$ and many other methods on the three networks same as Fig. 3.12, i.e., (a) and (b) the BA network, (c) and (d) the power grid network, and (e) and (f) the yeast network, where dashed lines are associated with the results of ABetS. Evol$_q$(1) and Evol$_q$(2) correspond to Evol$_q$(AMetisSg) and Evol$_q$(AMetisS), respectively. Evol$_F$(1) and Evol$_F$(2) follow the same representations.

3.5.7.3 Percolation metrics

We still firstly consider the three small networks as we did in Section 3.4.5.3. Here rather than illustrate the order parameter $\mathcal{G}_a(q)$, we report more readable results regarding both F and q_c in Fig. 3.28. For the BA network (Figs. 3.28a and 3.28b), AMetisS and AMetisSg perform much worse than other methods with respect to both F and q_c. By contrast, Evol$_F$(1) acquires the best F, and Evol$_q$(1) has a smaller critical threshold than ABetS even though q_c

of AMetisSg in much larger than the one of ABetS. But ABPDS still accounts for the smallest. For the two real-world networks, however, both AMetisS and AMetisSg work very well, where AMetisSg even has smaller F than ABetS in both two networks. Meanwhile, Evol_q and Evol_F based on AMetisS are slightly better than those based on AMetisSg, respectively for q_c and F. Specifically, for F, $\text{Evol}_F(2)$ has improvements (see Eq. (3.33)) of 10.51% and 28.17% accordingly in the power grid network and the yeast network compared to ABetS, while $\text{Evol}_F(1)$ holds 9.39% and 28.15%, respectively. In addition, regarding q_c, $\text{Evol}_q(1)$ also surpasses ABetS with margins of 6.58% and 5.85% for the two networks and $\text{Evol}_q(2)$ gains 6.28% and 5.31%. Note that ABetS outperforms almost all existing methods that we considered in this thesis (see also Section 3.3).

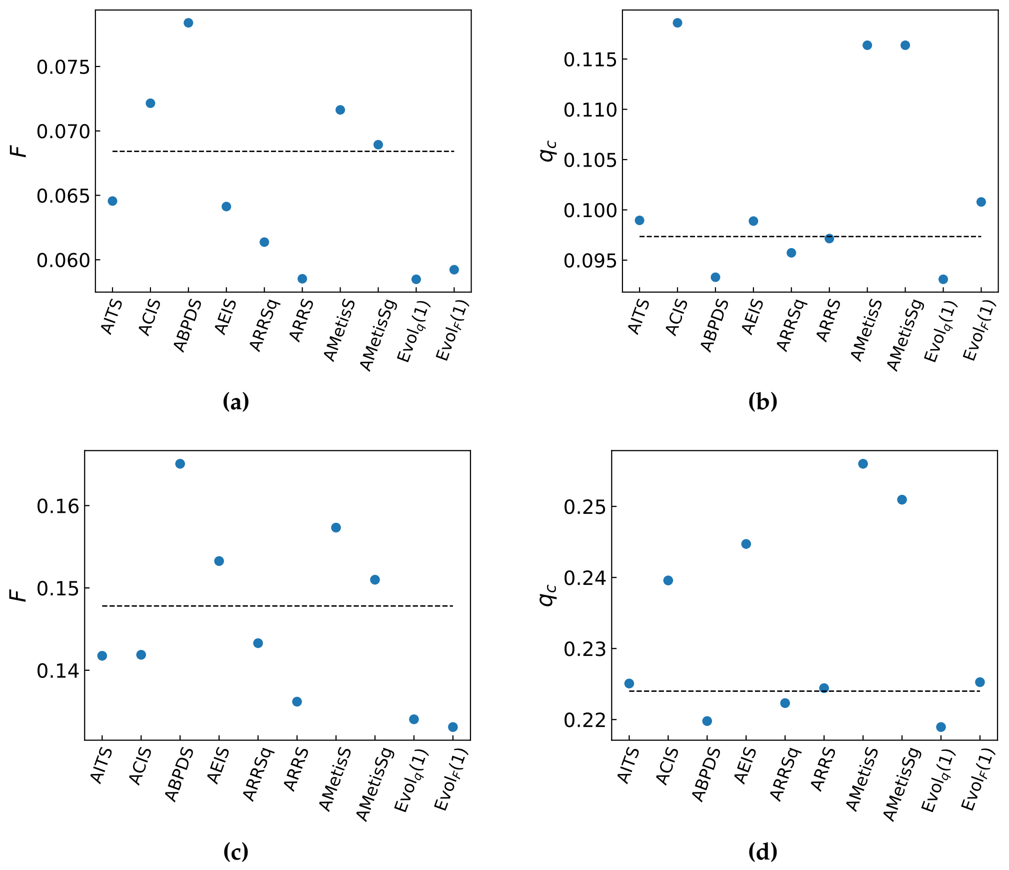


Figure 3.29: Performance of AMetisS and AMetisSg on networks generated through the configuration model based on degree sequences of (a) and (b) the power grid network, and (c) and (d) the yeast network, where dashed lines are associated with the results of ABetS. $\text{Evol}_q(1)$ and $\text{Evol}_F(1)$ correspond to $\text{Evol}_q(\text{AMetisSg})$ and $\text{Evol}_F(\text{AMetisSg})$, respectively.

The dramatic differences of results that AMetisS and AMetisSg have on the BA network and on the two real-world networks motivate us to further investigate their performance on networks constructed through the configuration model (see also Section 3.1.1.5). Specifically, we first draw a degree sequence based on either the power grid network or the yeast network, and then the configuration model is used to generate a network based that degree sequence. In this manner, we would have a new network which follows the similar degree distribution to either of the two networks but usually lacks community structure, degree correlation and local cycles as well (e.g., the network used in Fig. 3.29a has an assortativity of 0.0028 and a clustering coefficient of 0.0012). Fig. 3.29 shows the corresponding results. Apparently, AMetisS and AMetisSg fail in both networks for both F and q_c. And ABetS is also less effective than it does in the two real-world networks. In other words, for example, it is surpassed even by AITS and AEIS in Fig. 3.29a, and AITS and ACIS in Fig. 3.29c with regard to F. Since these networks are lack of local cycles, ABPDS is predictable to have really good performance regarding q_c (see Section 3.4.5.4). Nevertheless, $\text{Evol}_F(1)$ and $\text{Evol}_q(1)$ still accordingly acquire better F and q_c than almost all other methods. Based on those results, we could conclude that:

- if AMetisS or AMetisSg works well, Evol_q and Evol_F could benefit from them and further boost the performance of the corresponding sequence (see Fig. 3.28);
- if AMetisS or AMetisSg fails, Evol_q and Evol_F still have the ability to reorder the sequence and facilitate a really good result (see Figs. 3.28a, 3.28b and 3.29), which is actually usually better than almost all existing methods.

Note that the one constructed based on the power grid network only has the largest degree of 19, which is very small against its size ($n = 4941$)[31].

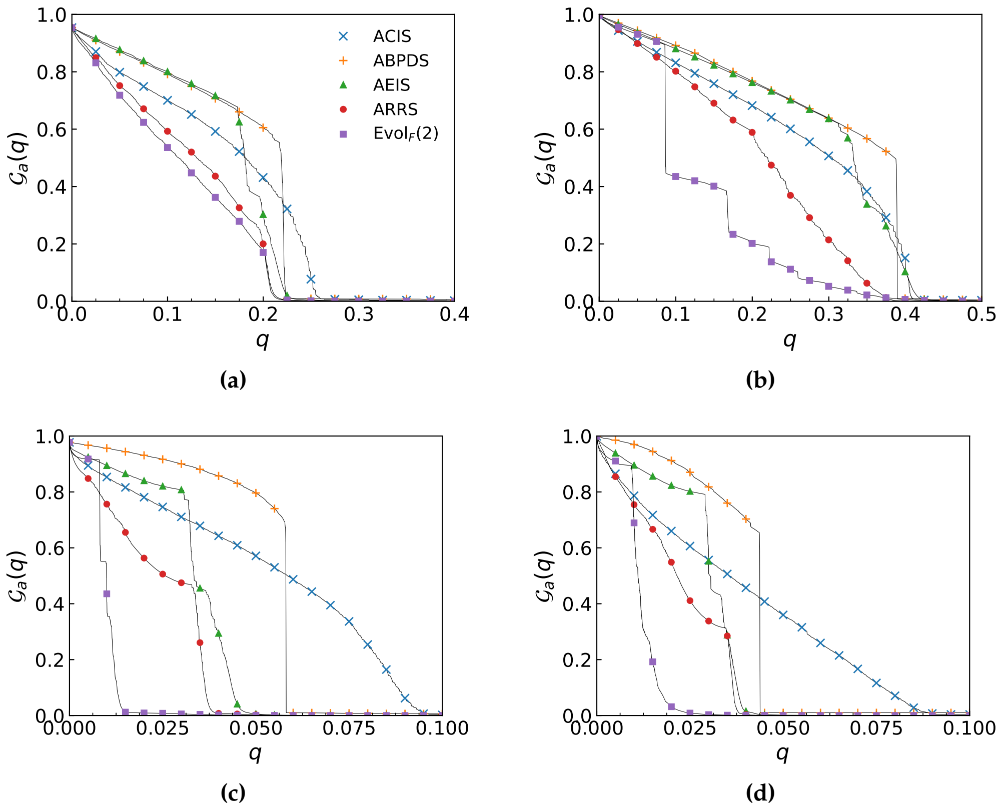


Figure 3.30: Performance of $\text{Evol}_F(2)$ validated by ACIS, ABPDS, AEIS, and ARRS on (a) the CA-AstroPh network, (b) the Cit-HepPh network, (c) the web-Google network, and (d) the as-Skitter network (ACIS with $\ell = 2$). $\text{Evol}_F(2)$ corresponds to $\text{Evol}_F(\text{AMetisS})$.

We further consider the order parameter $\mathcal{G}_a(q)$ of q regarding the four networks that we conducted in Fig. 3.14. Though our basic method ARRS already has better performance

[31]One can further study the influence of the network structure on AMetisS and AMetisSg. But it is out of the scope of this thesis.

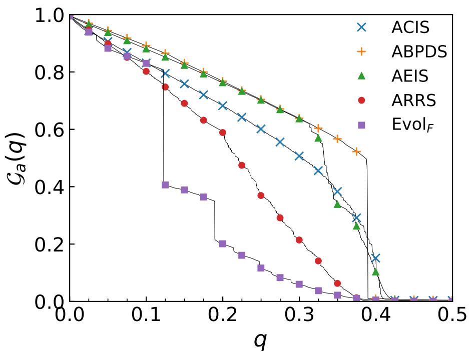


Figure 3.31: Tuning of F through the combination of ARRS and $\text{Evol}_F(2)$, which is represented by Evol_F, on the Cit-HepPh network. Here we only show this as an example. One can of course further tune them for a better result. $\text{Evol}_F(2)$ corresponds to $\text{Evol}_F(\text{AMetisS})$.

than ACIS, ABPDS, and AEIS in those networks, here Fig. 3.30 still gives the results of ACIS, ABPDS, and AEIS for the purpose of direct comparisons with Evol_F. As we can see from Fig. 3.30, indeed, only in the CA-AstroPh network $\text{Evol}_F(2)$ has smaller $\mathcal{G}_a(q)$ than all other methods for the whole rage of q, i.e., for $q \in (0, 1)$, which might be a problem for some cases. But it could be eased by combining with ARRS or other suitable methods, that is, we run Evol_F based on some strategy and fix part of the sequence at the same time (see Fig. 3.31). Nevertheless, for most cases of q, Evol_F is much better than other methods, including ARRS. For instance, in the CA-AstroPh network (Fig. 3.30a), $\text{Evol}_F(2)$ accordingly has advantages of 29.33%, 36.09%, 30.08%, and 7.97% compared to ACIS, ABPDS, AEIS, and ARRS. For the web-Google network (Fig. 3.30c), the margins are even much larger, 81.92%, 81.78%, 70.49%, and 58.14%, respectively.

More comparisons can be found in Table 3.5 and Table 3.6, where both results of F and q_c are reported accordingly. As we can see from Table 3.5, indeed, the greedy strategy (AMetisSg) works very well in some cases, like in the p2p-Gnutella08 network in which AMetisSg has much smaller F than AMetisS. But as we mentioned in Section 3.5.5, this greedy strategy is not so good for Evol_F, that is, $\text{Evol}_F(2)$ has better performance than $\text{Evol}_F(3)$ in 11 out of 18 networks. Nevertheless, the improvements in regard with $\text{Evol}_F(2)$ against $\text{Evol}_F(3)$ are very small ($-0.96\% \sim 1.03\%$), which indicates that $\text{Evol}_F(2)$ would be a better choice compared to $\text{Evol}_F(3)$. $\text{Evol}_F(2)$ also outperforms $\text{Evol}_F(1)$ in 15 networks, though $\text{Evol}_F(1)$ is actually better than AMetisSg in 10 networks. To sum up, $\text{Evol}_F(2)$ has average improvements of 60.62%, 54.83%, 54.49%, 43.23%, 21.69%, 14.09%, and 8.89% over HubS, AHubS, ACIS, AEIS, ARRS, AMetisS, and AMetisSg, respectively.

Now we move to the critical threshold q_c (Table 3.6). For this case, AMetisS and AMetisSg share almost the same value of q_c in each network. But $\text{Evol}_q(2)$ has smaller q_c than $\text{Evol}_q(3)$ in 14 out of 18 networks. Regarding $\text{Evol}_q(1)$, it has almost equal performance in small networks but is much worse in the four large networks compared to $\text{Evol}_q(2)$. Besides, it also has better results than AMetisSg in 14 networks. To summarize, $\text{Evol}_q(2)$ accordingly has mean advantages of 59.21%, 49.39%, 42.77%, 20.58%, 27.68%, 13.81%, 14.44%, 15.16%, and 13.26% over HubS, AHubS, ACIS, ABPDS, AEIS, ARRSq, ARRS, AMetisS, and AMetisSg.

Networks	HubS	AHubS	ACIS	AEIS	ARRS	AMetisS	AMetisSg
Power	$0.0636^{\ddagger\ddagger}$	$0.0524^{\ddagger\ddagger}$	$0.0449^{\ddagger\ddagger}$	$0.0112^{\ddagger\ddagger}$	$0.0076^{\dagger}$	$0.0076^{\dagger}$	$0.0075^{\dagger}$
CA-GrQc	$0.0825^{\ddagger\ddagger}$	$0.0685^{\ddagger\ddagger}$	$0.0527^{\ddagger\ddagger}$	$0.0347^{\ddagger}$	0.0289	$0.0346^{\ddagger}$	$0.0304^{\dagger}$
p2p-Gnutella08	$0.1993^{\ddagger\ddagger}$	$0.1574^{\ddagger}$	$0.1415^{\dagger}$	$0.1651^{\ddagger}$	0.1386	$0.2007^{\ddagger\ddagger}$	$0.1486^{\ddagger}$
as-733	$0.0128^{\ddagger\ddagger}$	$0.0125^{\ddagger\ddagger}$	$0.0150^{\ddagger\ddagger}$	$0.0097^{\ddagger}$	0.0087	$0.0109^{\ddagger\dagger}$	$0.0101^{\ddagger}$
Scottish	$0.0321^{\ddagger\dagger}$	$0.0272^{\ddagger}$	$0.0542^{\ddagger\ddagger}$	$0.0259^{\ddagger}$	0.0231	$0.0281^{\ddagger\dagger}$	$0.0270^{\ddagger}$
CA-AstroPh	$0.2508^{\ddagger\ddagger}$	$0.2084^{\ddagger\ddagger}$	$0.1562^{\ddagger\dagger}$	$0.1579^{\ddagger\ddagger}$	$0.1200^{\dagger}$	$0.1476^{\ddagger\dagger}$	$0.1224^{\dagger}$
CA-CondMat	$0.1238^{\ddagger\ddagger}$	$0.1103^{\ddagger\ddagger}$	$0.0832^{\ddagger\dagger}$	$0.0774^{\ddagger\dagger}$	0.0625	$0.0659^{\dagger}$	$0.0639^{\dagger}$
hep-th	$0.3751^{\ddagger\ddagger}$	$0.3048^{\ddagger\ddagger}$	$0.2541^{\ddagger\ddagger}$	$0.2742^{\ddagger\ddagger}$	$0.1915^{\ddagger\dagger}$	0.1539	0.1524
Cit-HepPh	$0.3665^{\ddagger\ddagger}$	$0.3062^{\ddagger\ddagger}$	$0.2645^{\ddagger\ddagger}$	$0.2860^{\ddagger\ddagger}$	$0.2056^{\ddagger\ddagger}$	0.1380	0.1372
Email-Enron	$0.0393^{\ddagger\ddagger}$	$0.0380^{\ddagger\ddagger}$	$0.0292^{\ddagger\ddagger}$	$0.0314^{\ddagger\ddagger}$	$0.0217^{\dagger}$	$0.0242^{\ddagger}$	$0.0226^{\ddagger}$
p2p-Gnutella31	$0.1287^{\ddagger\dagger}$	$0.1143^{\ddagger}$	$0.1015^{\dagger}$	$0.1172^{\ddagger}$	0.1003	$0.1169^{\ddagger}$	$0.1101^{\ddagger}$
loc-Gowalla	$0.1329^{\ddagger\ddagger}$	$0.1142^{\ddagger\ddagger}$	$0.0868^{\ddagger\ddagger}$	$0.0916^{\ddagger\ddagger}$	$0.0625^{\ddagger\ddagger}$	$0.0460^{\dagger}$	$0.0446^{\dagger}$
Email-EuAll	$0.0009^{\ddagger}$	$0.0009^{\dagger}$	$0.0056^{\ddagger\ddagger}$	$0.0019^{\ddagger\ddagger}$	0.0008	$0.0012^{\ddagger\ddagger}$	$0.0011^{\ddagger\dagger}$
com-Amazon	$0.1224^{\ddagger\ddagger}$	$0.1184^{\ddagger\ddagger}$	$0.0793^{\ddagger\ddagger}$	$0.0619^{\ddagger\ddagger}$	$0.0424^{\ddagger\ddagger}$	0.0250	0.0247
web-Google	$0.1141^{\ddagger\ddagger}$	$0.0886^{\ddagger\ddagger}$	$0.0526^{\ddagger\ddagger}$	$0.0322^{\ddagger\ddagger}$	$0.0227^{\ddagger\ddagger}$	$0.0103^{\dagger}$	$0.0101^{\dagger}$
PAroad	$0.1148^{\ddagger\ddagger}$	$0.0715^{\ddagger\ddagger}$	$0.0417^{\ddagger\ddagger}$	$0.0034^{\ddagger\ddagger}$	$0.0012^{\ddagger\ddagger}$	0.0006	0.0006
Txroad	$0.0859^{\ddagger\ddagger}$	$0.0652^{\ddagger\ddagger}$	$0.0342^{\ddagger\ddagger}$	$0.0019^{\ddagger\ddagger}$	$0.0007^{\ddagger\ddagger}$	$0.0003^{\dagger}$	0.0003
as-Skitter	$0.0706^{\ddagger\ddagger}$	$0.0487^{\ddagger\ddagger}$	$0.0394^{\ddagger\ddagger}$	$0.0287^{\ddagger\ddagger}$	$0.0215^{\ddagger\ddagger}$	$0.0138^{\ddagger}$	$0.0135^{\ddagger}$

Networks	AMetisSg(1)	Evol$_F$(1)	Evol$_F$(2)	Evol$_F$(3)	Evol$_q$(1)	Evol$_q$(2)	Evol$_q$(3)
Power	$0.0076^{\dagger}$	0.0070	**0.0069**	0.0069	$0.0076^{\dagger}$	$0.0075^{\dagger}$	$0.0075^{\dagger}$
CA-GrQc	$0.0317^{\ddagger}$	**0.0275**	0.0283	0.0284	0.0284	0.0282	0.0283
p2p-Gnutella08	$0.1495^{\ddagger}$	0.1327	0.1326	**0.1326**	0.1356	0.1356	0.1355
as-733	$0.0099^{\ddagger}$	0.0085	**0.0084**	0.0085	0.0088	0.0088	0.0087
Scottish	$0.0271^{\ddagger}$	0.0226	**0.0225**	0.0225	0.0227	0.0226	0.0226
CA-AstroPh	$0.1222^{\dagger}$	0.1130	**0.1104**	0.1109	$0.1244^{\ddagger}$	$0.1256^{\ddagger}$	$0.1247^{\ddagger}$
CA-CondMat	$0.0651^{\dagger}$	0.0599	**0.0597**	0.0600	$0.0645^{\dagger}$	$0.0655^{\dagger}$	$0.0661^{\dagger}$
hep-th	0.1523	$0.1836^{\ddagger}$	0.1516	**0.1510**	$0.2133^{\ddagger\dagger}$	$0.1992^{\ddagger\dagger}$	$0.1892^{\ddagger}$
Cit-HepPh	0.1372	$0.2011^{\ddagger\ddagger}$	0.1367	**0.1363**	$0.2316^{\ddagger\ddagger}$	$0.1822^{\ddagger\dagger}$	$0.1773^{\ddagger\dagger}$
Email-Enron	$0.0232^{\ddagger}$	0.0206	0.0198	**0.0198**	$0.0224^{\ddagger}$	$0.0225^{\ddagger}$	$0.0224^{\ddagger}$
p2p-Gnutella31	$0.1121^{\ddagger}$	**0.0959**	0.0961	0.0964	0.0992	0.0994	$0.1143^{\ddagger}$
loc-Gowalla	$0.0450^{\dagger}$	$0.0530^{\ddagger\dagger}$	**0.0416**	0.0417	$0.0575^{\ddagger\dagger}$	$0.0458^{\dagger}$	$0.0463^{\ddagger}$
Email-EuAll	$0.0010^{\ddagger}$	**0.0008**	0.0008	0.0008	0.0008	0.0008	0.0008
com-Amazon	0.0250	$0.0352^{\ddagger\dagger}$	0.0247	**0.0244**	$0.0423^{\ddagger\ddagger}$	$0.0313^{\ddagger\dagger}$	$0.0310^{\ddagger\dagger}$
web-Google	0.0099	$0.0178^{\ddagger\ddagger}$	**0.0095**	0.0096	$0.0190^{\ddagger\ddagger}$	$0.0101^{\dagger}$	$0.0102^{\dagger}$
PAroad	$0.0006^{\dagger}$	$0.0010^{\ddagger\ddagger}$	0.0006	**0.0006**	$0.0011^{\ddagger\ddagger}$	$0.0007^{\ddagger}$	$0.0007^{\ddagger}$
Txroad	0.0003	$0.0005^{\ddagger\ddagger}$	**0.0003**	0.0003	$0.0005^{\ddagger\ddagger}$	$0.0004^{\ddagger\dagger}$	$0.0004^{\ddagger}$
as-Skitter	$0.0137^{\ddagger}$	$0.0180^{\ddagger\ddagger}$	**0.0114**	0.0115	$0.0236^{\ddagger\ddagger}$	$0.0131^{\ddagger}$	$0.0133^{\ddagger}$

Table 3.5: Results of F on the 18 real-world networks, where AMetisSg(1) is the combination of ABondS1 and AMetisSg, Evol$_F$(1) is Evol$_F$(HubS), Evol$_F$(2) is Evol$_F$(AMetisS), Evol$_F$(3) is Evol$_F$(AMetisSg), Evol$_q$(1) is Evol$_q$(HubS), Evol$_q$(2) is Evol$_q$(AMetisS), and Evol$_q$(3) is Evol$_q$(AMetisSg). An item in bold represents that the corresponding method has the minimal F among all those mentioned methods in the associated network. Besides, $\dagger$, $\ddagger$, $\ddagger\dagger$ or $\ddagger\ddagger$ indicates that Evol$_F$(2) has an improvement of over 5%, 10%, 20%, or 30% compared to the corresponding strategies, respectively. Note that these improvements are calculated based on the related real values instead of the approximate results that are shown in this table (see also Eq. (3.33)).

3.5.7.4 SIR results

Similar to Section 3.4.5.5, we further verify the evolutionary framework by the SIR model, mainly considering Evol$_F$. But here, instead of the Email-Enron network and the loc-Gowalla network, we run the model on the global airline network which we collect from OpenFlights (`https://openflights.org`), where a node represents an airport and an edge between two nodes indicates that there is at least one airline between those two corresponding airports. We choose this network because it might be the main factor to facilitate a pandemic nowadays.

Networks	HubS	AHubS	ACIS	ABPDS	AEIS	ARRSq	ARRS	AMetisS
Power	$975^{\ddagger\ddagger}$	$762^{\ddagger\ddagger}$	$570^{\ddagger\ddagger}$	$316^{\ddagger}$	$337.1^{\ddagger\dagger}$	$275.85^{\dagger}$	$284.1^{\dagger}$	$327^{\ddagger\dagger}$
CA-GrQc	$912^{\ddagger\ddagger}$	$820^{\ddagger\ddagger}$	$1760^{\ddagger\ddagger}$	$398^{\ddagger}$	$428.25^{\ddagger}$	363.65	$377.55^{\dagger}$	$474^{\ddagger\dagger}$
p2p-Gnutella08	$2045^{\ddagger\ddagger}$	$1584^{\ddagger}$	$1444^{\dagger}$	**1300**	$1508.95^{\ddagger}$	1343.7	1375.35	$1925^{\ddagger\ddagger}$
as-733	$243^{\ddagger\ddagger}$	$248^{\ddagger\ddagger}$	$192^{\ddagger\dagger}$	$162^{\dagger}$	$169.35^{\ddagger}$	151.95	153.1	$186^{\ddagger}$
Scottish	$877^{\ddagger\ddagger}$	$603^{\ddagger\dagger}$	$2036^{\ddagger\ddagger}$	434	$471.05^{\dagger}$	437.05	443.6	$512^{\ddagger}$
CA-AstroPh	$8544^{\ddagger\ddagger}$	$6274^{\ddagger\ddagger}$	$4865^{\ddagger\dagger}$	$4198^{\dagger}$	$4320.6^{\ddagger}$	4031.9	4018.45	$4669^{\ddagger}$
CA-CondMat	$5726^{\ddagger\ddagger}$	$4500^{\ddagger\ddagger}$	$3217^{\ddagger\dagger}$	2569	$2700.8^{\dagger}$	2539.05	2538.8	$2971^{\ddagger}$
hep-th	$18097^{\ddagger\ddagger}$	$12092^{\ddagger}$	$11184^{\ddagger}$	$10294^{\dagger}$	$11002.85^{\ddagger}$	9817.85	9741.8	10031
Cit-HepPh	$22533^{\ddagger\ddagger}$	$15297^{\ddagger\dagger}$	$14164^{\ddagger}$	$13455^{\dagger}$	$14498.9^{\ddagger}$	$13011.95^{\dagger}$	$12999.2^{\dagger}$	$13429^{\dagger}$
Email-Enron	$4097^{\ddagger\ddagger}$	$4262^{\ddagger\ddagger}$	$3074^{\ddagger}$	$2621^{\dagger}$	$2764.35^{\dagger}$	2566.35	2578.25	$2876^{\ddagger}$
p2p-Gnutella31	$14111^{\ddagger\ddagger}$	$12424^{\ddagger\dagger}$	$10995^{\ddagger}$	**9287**	10127.2	9915.95	10196.3	$11790^{\ddagger}$
loc-Gowalla	$53828^{\ddagger\ddagger}$	$40168^{\ddagger\ddagger}$	$31386^{\ddagger\dagger}$	$26951^{\ddagger}$	$26916.7^{\ddagger}$	$25316^{\dagger}$	25023.7	$27002^{\ddagger}$
Email-EuAll	$1431^{\ddagger\dagger}$	$1282^{\ddagger}$	$1193^{\ddagger}$	1064	$6985.8^{\ddagger\ddagger}$	1067.45	1081.25	$1230^{\ddagger}$
com-Amazon	$78308^{\ddagger\ddagger}$	$68527^{\ddagger\ddagger}$	$42108^{\ddagger\ddagger}$	$29572^{\ddagger\dagger}$	$27471.15^{\ddagger}$	$27426.25^{\ddagger}$	$26358.95^{\ddagger}$	$25411^{\dagger}$
web-Google	$253099^{\ddagger\ddagger}$	$171550^{\ddagger\ddagger}$	$82525^{\ddagger\ddagger}$	$50861^{\ddagger\ddagger}$	$41948.85^{\ddagger\ddagger}$	$40326.2^{\ddagger\ddagger}$	$33679.6^{\ddagger\ddagger}$	$19256^{\ddagger\dagger}$
PAroad	$273899^{\ddagger\ddagger}$	$246270^{\ddagger\ddagger}$	$71134^{\ddagger\ddagger}$	$21172^{\ddagger\ddagger}$	$17204.05^{\ddagger\ddagger}$	$9576.25^{\ddagger\ddagger}$	$10124.85^{\ddagger\ddagger}$	5583
Txroad	$307413^{\ddagger\ddagger}$	$320991^{\ddagger\ddagger}$	$82744^{\ddagger\ddagger}$	$20873^{\ddagger\ddagger}$	$16800.1^{\ddagger\ddagger}$	$8505.6^{\ddagger\ddagger}$	$9373.65^{\ddagger\ddagger}$	4720
as-Skitter	$322128^{\ddagger\ddagger}$	$201670^{\ddagger\ddagger}$	$151846^{\ddagger\ddagger}$	$74286^{\ddagger\ddagger}$	$70901^{\ddagger\ddagger}$	$63547.4^{\ddagger\ddagger}$	$64164.8^{\ddagger\ddagger}$	$46291^{\ddagger}$

Networks	AMetisSg	AMetisSg(1)	$\text{Evol}_F(1)$	$\text{Evol}_F(2)$	$\text{Evol}_F(3)$	$\text{Evol}_q(1)$	$\text{Evol}_q(2)$	$\text{Evol}_q(3)$
Power	$327^{\ddagger\dagger}$	$325^{\ddagger}$	$284.65^{\dagger}$	$292.55^{\ddagger}$	$293.5^{\ddagger}$	263.4	**260.8**	262.3
CA-GrQc	$443^{\ddagger\dagger}$	$416^{\ddagger}$	$364.1^{\dagger}$	$374.9^{\dagger}$	$394.45^{\ddagger}$	348	**345.8**	347.95
p2p-Gnutella08	$1551^{\ddagger}$	$1559^{\ddagger}$	$1393.6^{\dagger}$	$1392.1^{\dagger}$	$1394.7^{\dagger}$	1318.85	1319.75	1318.05
as-733	$186^{\ddagger}$	$180^{\ddagger}$	153.55	154.85	156.8	151.55	**151.5**	151.6
Scottish	$508^{\ddagger}$	$520^{\ddagger}$	444.85	440.85	442.55	427.5	**426.35**	426.45
CA-AstroPh	$4420^{\ddagger}$	$4385^{\ddagger}$	4023.5	4072.95	$4171.3^{\dagger}$	3882.55	**3873**	3875.2
CA-CondMat	$2970^{\ddagger}$	$2936^{\ddagger}$	2548.45	$2744.05^{\ddagger}$	$2785.4^{\ddagger}$	**2436.05**	2445.3	2445.6
hep-th	10030	9921	9825.45	9880.2	9881.9	9518.05	9752.3	**9163**
Cit-HepPh	$13426^{\dagger}$	$13431^{\dagger}$	$13025.15^{\dagger}$	$13337.7^{\dagger}$	$13338.2^{\dagger}$	12690.25	**12217.3**	12237
Email-Enron	$2871^{\ddagger}$	$2848^{\ddagger}$	2590.65	$2653.95^{\dagger}$	$2663.05^{\dagger}$	**2478.55**	2487.95	2491.05
p2p-Gnutella31	$11304^{\ddagger}$	$11325^{\ddagger}$	$10300.95^{\dagger}$	$10331.45^{\dagger}$	$10312.25^{\dagger}$	9714.4	9701.1	$11894.25^{\ddagger}$
loc-Gowalla	$26999^{\ddagger}$	$27094^{\ddagger}$	24807.9	$25918^{\dagger}$	$25978.1^{\dagger}$	24283.05	**24014.5**	24059.75
Email-EuAll	$1183^{\ddagger}$	$1178^{\ddagger}$	1065.25	1064.4	1063.05	1041.15	1040.45	**1040.3**
com-Amazon	$25411^{\dagger}$	$25609^{\dagger}$	$25951^{\dagger}$	$24924.05^{\dagger}$	$24917.5^{\dagger}$	$25360.55^{\dagger}$	**23380.9**	23420.35
web-Google	$19255^{\ddagger\dagger}$	$20330^{\ddagger\ddagger}$	$27584.9^{\ddagger\ddagger}$	$15462.1^{\ddagger}$	$15473.2^{\ddagger}$	$21759^{\ddagger\ddagger}$	**13528**	13596.1
PAroad	5583	5752	$8717.75^{\ddagger\ddagger}$	5551.95	5547.55	$8785.5^{\ddagger\ddagger}$	**5502.75**	5506.7
Txroad	4718	4797	$7769.75^{\ddagger\ddagger}$	4686.8	4693.95	$7828.3^{\ddagger\ddagger}$	**4632.8**	4635.75
as-Skitter	$46287^{\ddagger}$	$46633^{\ddagger}$	$58889.35^{\ddagger\ddagger}$	41120.6	41335.05	$54455.5^{\ddagger\dagger}$	39986.6	**39693.05**

Table 3.6: Results of $q_c \times n$ on the 18 real-world networks, where AMetisSg(1) is the combination of ABondS1 and AMetisSg, $\text{Evol}_F(1)$ is Evol_F(HubS), $\text{Evol}_F(2)$ is Evol_F(AMetisS), $\text{Evol}_F(3)$ is Evol_F(AMetisSg), $\text{Evol}_q(1)$ is Evol_q(HubS), $\text{Evol}_q(2)$ is Evol_q(AMetisS), and $\text{Evol}_q(3)$ is Evol_q(AMetisSg). An item in bold represents that the corresponding method has the minimal q_c among all those mentioned methods in the associated network. Besides, †, ‡, ‡† or ‡‡ indicates that $\text{Evol}_q(2)$ has an improvement of over 5%, 10%, 20%, or 30% compared to the corresponding strategies, respectively. Note that these improvements are calculated based on the related real values instead of the approximate results that are shown in this table.

Fig. 3.32 shows the contours of the average infected frequency $\langle \alpha_{\text{inf}} \rangle$ (see also Section 3.4.5.5 for its definition) of the infected probability η_i and the immunized fraction q with respect to RanS, HubS, ACIS, AEIS, ARRS and Evol_F, where $\eta_r = 0.05$ and $b = 10^4$ independent simulations are conducted for every pair of η_i and q. As we can see from there, the network under the immunization of RanS is apparently much worse than it under targeted methods. For instance, even 20% nodes are immunized based on RanS. The network still cannot sustain an epidemic with $\eta_i = 0.01$. Amid those targeted methods, HubS is surprisingly better than ACIS when q is large, e.g., HubS has a larger margin of η_i to keep

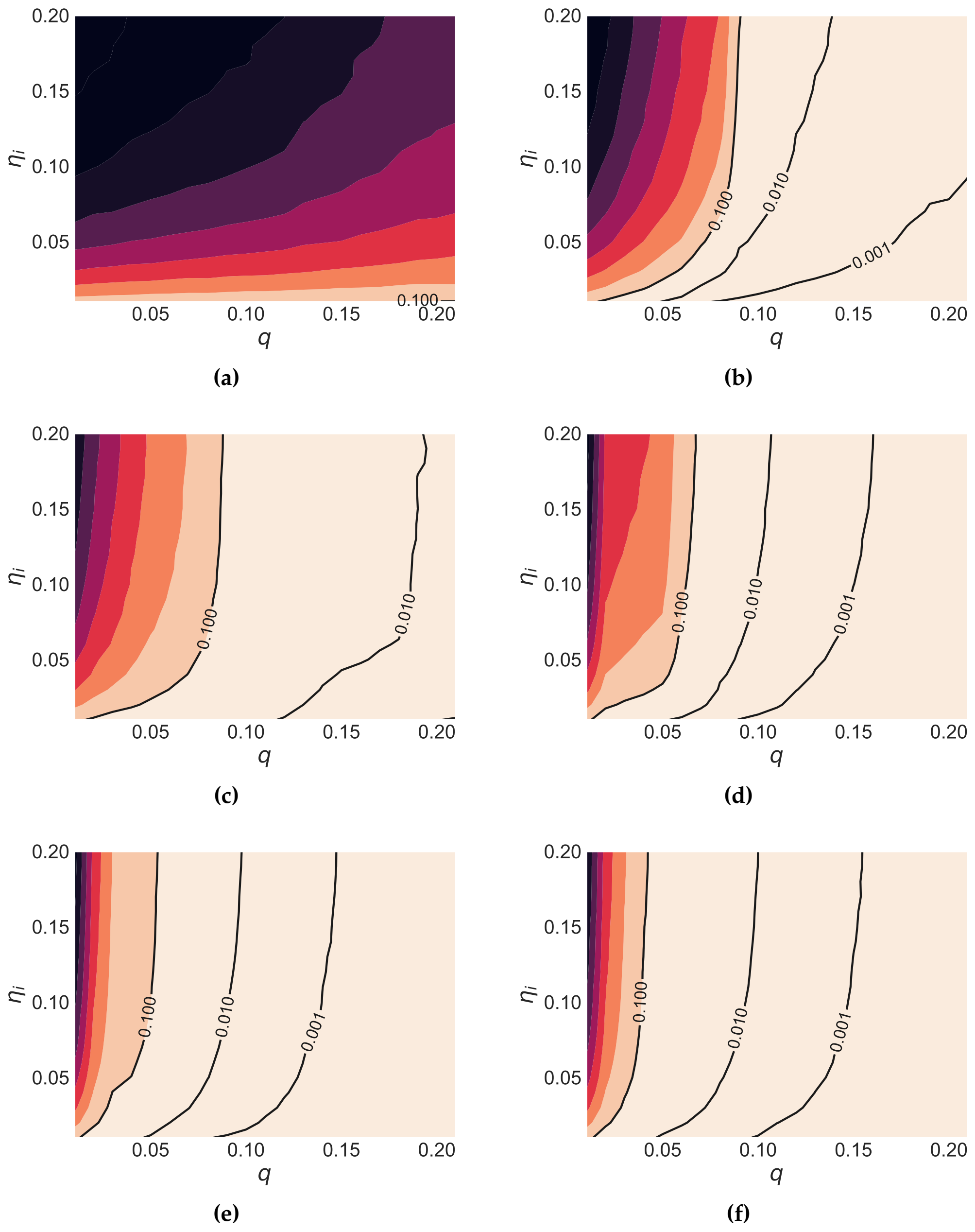


Figure 3.32: Contours of the average infected frequency $\langle \alpha_{\text{inf}} \rangle$ as a function of the immunized fraction q and the infected probability η_i on the global airline network regarding (a) RanS, (b) HubS, (c) ACIS, (d) AEIS, (e) ARRS, and (f) Evol$_F$. In each figure, color from deep dark (i.e., top left) to light purple (i.e., bottom right) indicates that $\langle \alpha_{\text{inf}} \rangle$ changes from large to small. In addition, $\langle \alpha_{\text{inf}} \rangle = 0.001, 0.01$, and 0.1 are marked by the three solid curves, respectively.

$\langle \alpha_{\text{inf}} \rangle$ less than 0.01 compared to ACIS. Evol$_F$ has similar behaviour. Namely, it only slightly outperforms AEIS and ARRS if q is large enough. Nevertheless, it is much more powerful than others to contain the spreading if the number of nodes that are allowed to be immunized is quite limited. For example, Fig. 3.33 shows the comparisons among those methods in regard to a specific case of $\eta_i = 0.2$ and $q = 0.05$. Obviously, Evol$_F$ is much better than other methods including ARRS, that is, $\langle \alpha_{\text{inf}} \rangle = 0.07$ against $\langle \alpha_{\text{inf}} \rangle = 0.13$. It is worth mentioning that one can always find a better result through Evol$_q$ if q is fixed.

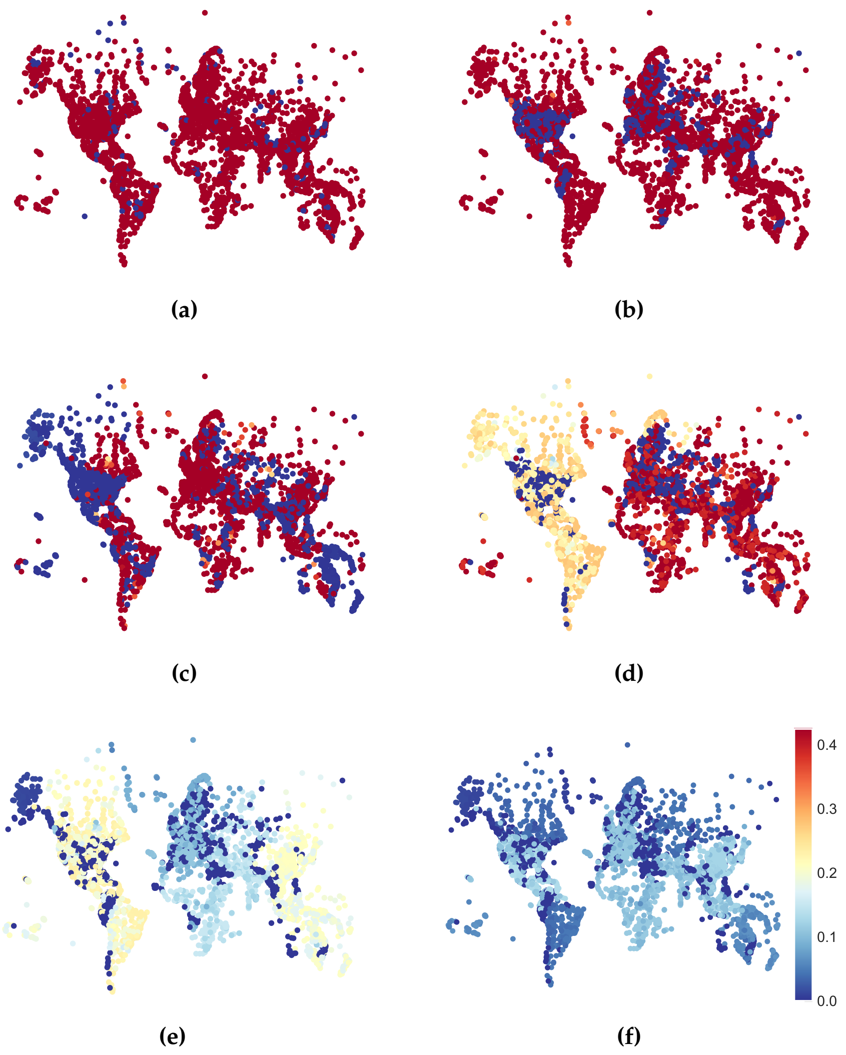


Figure 3.33: Specific patterns of SIR results on the global airline network regarding varied immunization methods, where η_i and q are fixed to 0.2 and 0.05, respectively. In particular, (a) RanS has $\langle\alpha_{\text{inf}}\rangle = 0.79$, (b) HubS has $\langle\alpha_{\text{inf}}\rangle = 0.50$, (c) ACIS has $\langle\alpha_{\text{inf}}\rangle = 0.29$, (d) AEIS has $\langle\alpha_{\text{inf}}\rangle = 0.28$, (e) ARRS has $\langle\alpha_{\text{inf}}\rangle = 0.13$, and (f) Evol$_F$ has $\langle\alpha_{\text{inf}}\rangle = 0.07$. The color bar shows the magnitude of $\langle\alpha_{\text{inf}}\rangle$ for each node.

3.6 Fast Scheme for the Suppression of F

Based on Sections 3.4 and 3.5, we could further have the following fast scheme for the suppression of F, which consists of three steps.

1) Repeatedly acquire the vertex separator, followed by a group reorder on the LCC. Since minimizing F is our main goal, the vertex separator can be obtained in the way same as AMetisS (Section 3.5.5). Meanwhile, on each separator, ARRS is conducted to achieve the local rank (Section 3.4.4), which should be much fater than the greedy strategy (see Fig. A.24).

2) Organize the tail. After a number of iterations of step 1), the related network transfers into a subnetwork containing a lot of small components. In this case, it would be very expensive to continue step 1) until the size of the LCC smaller than 3 (i.e., AMetisS and AMetisSg, see also Fig. A.24 in Appendix A.2.15). To overcome that problem, we

choose to stop step 1) when $\mathcal{G}(q)$ is less than a given threshold α (see also Eq. (3.8)). Now, if α is small, then the remaining network would be a network consisting of a number of small components. For this case, the probability that τ nodes randomly chosen from the remaining network belong to the same component would be bounded by $(\frac{\alpha}{1-q})^\tau$, which approaches 0 even τ is small. Therefore, we can also directly use the ARRS method to organize the tail, which is usually terminated within such as 5 rounds.

3) Prune the whole sequence. Neither steps 1) nor 2) considers nodes from other groups. This step therefore copes with nodes from different groups, which could be achieved by those strategies in Section 3.5. Besides, if letting $b(i)$ be the component size that node i leads to regarding a percolation process over a given sequence S (obtained based on steps 1) and 2)), then F could also benefit from the direct sort of S over $b(i)$.

Since the combination of steps 1) and 2), as well as the direct rank of phase 3), is enough to have a good result, we here particularly use GPEP to represent such combination and verify it as our main purpose. The corresponding performance can be found in Fig. A.24, Tables A.2, A.3, and A.4 in Appendix A.2.15.

3.7 Summary

Aiming at the proposal of advanced solutions for the network robustness and immunization problems on existing networks, we have profoundly studied and also developed a bunch of methods that are basically suitable for all kinds of networks. Specifically, we borrow ideas from the explosive percolation, and firstly demonstrated that rules delaying the percolation transition truly facilitate better solutions for the considered problems. In particular, based on the Bohman and Frieze's rule, bounded-size strategies including ABonS1, ABonS1q, ABonS2, APRSs1, and APRSs1q are designed, where ABonS1, ABonS2, and APRSs1 are for the minimization of F (Eq. (3.16)), and ABonS1q and APRSs1q are for the optimization of the critical threshold q_c. These approaches are very fast and usually have comparable results compared to the state of the art regarding both F and q_c (see Tables 3.2 and 3.3), e.g., ABonS1 and ABonS1q could accordingly obtain smaller F and q_c within 10 seconds in the as-Skitter network (with over 1.6 million nodes) than almost all existing methods[32] that we mentioned in Section 3.3. Besides, APRSs1 and APRSs1q give the option to score each node so that one can tune the score based on varied scenarios and further acquire a better result. But those bounded approaches could only outperform such as AEIS in a few networks especially in regard to F. Hence, we further considered and studied unbounded-size rules, particularly sum and product rules, based on which ARRS and APRSrr are proposed. Surprisingly, different from the explosive percolation, we found that ARRS with the product rule can always acquire better F comparing to the one with the sum rule. Besides, we also studied other unbounded rules but they are usually too time-consuming. In short, APRSrr is more capable of handling model or model-like networks, where there are usually a lack of community structures, short cycles, and correlations, such

[32]They are perhaps only less effective than ABetS. But it is almost impossible for ABetS to tackle a network of such size.

as the p2p-Gnutella08 network. ARRS instead has a good performance in most real-world networks regarding both F and q_c, and on average, surpasses all other basic methods. More importantly, ARRS also paves the way for the evolutionary framework. Note that both bounded and unbounded methods could be easily extended for the FVS problem. And either of them has a better performance than ABPDS in almost all networks that we considered in this thesis (Table 3.4).

For the evolutionary framework, we have introduced selection strategies, mutation operators, and the ways to initialize and maintain a population (sequence) as well. In particular, with respect to F, we firstly study PruOrd, which is yet effective enough to outperform ARRS. But it also faces a problem of the difficulty of the management of control parameters. To overcome that, PruGri is developed and takes a much easier way that only two parameter needs to be given beforehand. Usually, PruGri could obtain better results than PruOrd if we fail to give PruOrd the optimal configuration of the related parameters. However, the grid search strategy means that PruGri would be time-consuming if we do not have a parallel environment. Hence, we further have PruRan and PruRang, where PruRan chooses the slice simply following the uniform distribution, while PruRang does that by taking some probability from a variant of the Gaussian distribution. And therefore, PruRang is usually more capable of approximating PruGri. Note that one can also try other strategies of that probability. We then further study the effects of q_c on F and also the influences of F on q_c, and find that there is some conflict between the optimization of q_c and the optimization of F. And this conflict guides us to have PruRangqv4 and PruRangqv5, and further those mutation operators. In addition, through these investigations, we also find that the initial sequence plays an important role, especially one having advanced performance. Thus, we have another way based on graph partitioning to provide initial sequences for our framework, i.e., AMetisS and AMetisSg. And finally, we reach the evolutionary framework, that is, Evol$_F$ and Evol$_q$ accordingly for F and q_c.

All in all, for a given network, compared to a number of existing methods, including the state-of-the-art ones, the proposed approaches are much more capable of:

- identifying the fatal nodes, which is possibly related to, e.g., collapsing a criminal or corrupt organization, or offering an avenue to design new drugs to kill unwanted bacteria;
- revealing the true robustness of the network and then facilitating better ways to protect it, e.g., for a communication network against an intentional attack;
- preventing a possible outbreak of epidemics, e.g., taking more considerations on part of global airports;
- and preventing the spread of misinformation, which, as we mentioned before, has become one of the top threats to our society.

Besides, more applications will be found in the following Chapter.

4 Functions of Order Parameter as Measure

This chapter mainly discusses the following problems. Firstly, we are going to demonstrate whether F could be used as a measure to capture the network structure particularly considering the explosive synchronization. Then, viewing F as the goal, a few strategies are studies aiming to enhance or weaken the robustness of a given network (i.e., F), which is usually bounded by the most advanced attack strategy. Following that, we further investigate the role that F plays in the explosive synchronization dataset by the aid of machine learning methods.

The connections of this chapter and our previously published works are as follows. Section 4.1 is mainly based on the paper [**L3**]. Sections 4.2 and 4.3 are partly based on ideas and/or results from refs. [**L1**], [**L2**], and [**L3**]. Section 4.4 relies on the data generated through the method in ref. [**L3**].

4.1 Effects of Network Robustness on Explosive Synchronization

We particularly take the explosive synchronization (ES) as an example to study whether F (see also Eq. (3.16)) could be viewed as a measure to capture the network structure [**L3**]. The ES is another critical phenomenon that is observed when the coupled oscillators (e.g., the Kuramoto system [38]) are associated with a scale-free topology [13], i.e., the natural frequency of each oscillator proportionally corresponds to its number of connections, and they are coupled by the related adjacency matrix. The reasons that we choose ES are twofold. One the one hand, similar to the explosive percolation, the abrupt transition could also be found in the ES process. On the other hand, to some extent, the ES only relies on the network structure [111, 112, 113, 114], which is crucial for our verification in Section 4.4, since we could study and know more regarding the network structure compared to real dataset.

Specifically, we employ F to represent a network's robustness and view it as a global attribute of such network, and further study its influence on ES. Following the assumption of refs. [13, 113], we mainly find that ES depends not only on the network assortativity r (see also Eq. (2.26)) but also on the network robustness. In particular, there might exist a maximization of the hysteresis area, which can be achieved by adjusting the network assortativity while keeping F constant. However, this process cannot be inverted, i.e., a similar goal could not be achieved through the tuning of F over a fixed network assortativity. In addition, we further discuss the response of F and r to the change of each other, which results in a gap between the enhancement and weakness. We also find that this gap actually plays an important role regarding the ES.

4.1.1 Model

The reason that we choose F instead of q_c is as follows. As we studied in Section 3, for a given network $G(\mathcal{N}, \mathcal{M})$, there are usually two ways to measure its robustness, i.e., F and q_c, where F considers the whole order parameter $\mathcal{G}_a(q)$ while q_c only captures the moment of the disappearance of the giant component. From this point of view, F is obviously more capable of globally measuring the change of the order parameter. Thus, we here choose to mainly consider F instead of q_c. But there is still another problem. That is, from the results of Section 3 we know that G would have varied reactions of F if it is attacked by different strategies. Besides, so far we have studied over 20 attack strategies, which means that it is almost impossible to consider all of them. However, here we only need to know whether F could be viewed as a measure of the network structure. Namely, we could actually only study one of them, and others can follow the same way. Hence, we choose the simplest one, i.e., F under the attack of HubS.

Besides, following the assumption of ref. [13] viewing the dynamics of the nodes of G as phase oscillators coupled by the associated edges, we still employ the Kuramoto model [38] to govern the dynamics of the coupled oscillators:

$$\dot{\theta}_i = \omega_i + \lambda \sum_{j \in \mathcal{N}} A_{ij} \sin(\theta_j - \theta_i), \tag{4.1}$$

where $i \in \mathcal{N}$, θ_i is the phase of oscillator i, and ω_i denotes the corresponding natural frequency. A is the associated adjacency matrix (see Section 2.1.2). For a simple network, A_{ij} is a binary symbol, i.e., $A_{ij} = 1$ if $e_{ij} \in \mathcal{M}$, otherwise, $A_{ij} = 0$. Now if we control the coupling strength λ, the coherence (usually measured by the order parameter $\Re$) among the oscillators could be controlled, which follows, e.g.,

$$\Re(t) \mathrm{e}^{\mathrm{i}\psi(t)} = \frac{1}{n} \sum_{j \in \mathcal{N}} \mathrm{e}^{\mathrm{i}\theta_j(t)}, \tag{4.2}$$

where $\mathrm{i} = \sqrt{-1}$, e is Euler's number, and $\psi(t) = \langle \theta(t) \rangle$ denotes the phase of the mean field of the system under the evolution of time t.

4.1.2 Rewiring strategy

To tune the values of F and r of G, we first define the following procedures:

$$\alpha_-(e_{ij}) := \text{cut the edge between nodes } i \text{ and } j,$$
$$\alpha_+(\sigma_{uv}) := \text{add an edge between nodes } u \text{ and } v,$$

where σ_{uv} means that there is no direct connection between nodes $u \in \mathcal{N}$ and $v \in \mathcal{N}$, i.e., $e_{ij} \in \mathcal{M}$ and $\sigma_{uv} \notin \mathcal{M}$. Further, letting

$$\alpha_-(e_{ij}, e_{uv}) := \alpha_-(e_{ij}) \text{ and } \alpha_-(e_{uv}),$$
$$\alpha_+(\sigma_{iu}, \sigma_{jv}) := \alpha_+(\sigma_{iu}) \text{ and } \alpha_+(\sigma_{jv}),$$

we have the following observations:

i) the average degree $\langle k \rangle$ keeps constant if $\alpha_-(e_{ij})$ and $\alpha_+(\sigma_{uv})$ appear in a pair;

ii) the node degree distribution keeps unchanged if $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ appear in a pair;

iii) r keeps constant if $k_i = k_v$ or/and $k_j = k_u$ under the condition of observation ii).

Note that both e_{ij} and e_{uv} are randomly selected from G in this thesis. Besides, for convenience of description, we define the goal function $\xi^\delta(g)$ based on the cut-add strategy, i.e., $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ appear in a pair, where g is the metric associated with a certain property of G, like r or F, and δ is a symbol corresponding to either the enhancement $(+)$ or the reduction $(-)$ of g. Moreover, we employ $\xi^\delta(g|a_g)$ to represent the evolution of $\xi^\delta(g)$ under the condition of a_g. For example, $\xi^+(F_0|r \equiv r_0)$ means that the robustness F of G is enhanced to F_0, while its assortativity r is fixed to r_0, where both F_0 and r_0 are given values. In addition, when we specify a certain value, e.g., $r = 0.000$, it means that the difference between the given value and the real value is within 1×10^{-5}. In this case, we use $\xi^\delta(g)$ instead of $\xi^+(g)$ or $\xi^-(g)$ because both of them are employed to adjust the network in a small region around the given value. In other words, for instance, we first use either $\xi^+(r_0)$ or $\xi^-(r_0)$ to make the network assortativity close to r_0, and then employ both $\xi^+(r_0)$ and $\xi^-(r_0)$ to further adjust the network which finally has $r = r_0 \pm 10^{-5}$.

In particular, for a given network and g_0 (assuming $g_0 > g$), one procedure $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ is accepted if it increases g, otherwise, is ignored. Similarly, when $g_0 < g$, we adopt the exchange $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ if it decreases g. Then, repeat those two procedures until desired g_0 is reached, which corresponds to $\xi^\delta(g)$. During the process to g_0, we may capture some points (say g_0'), where g only undergoes an enhancement or a reduction. This case is associated with $\xi^+(g_0')$ or $\xi^-(g_0')$, respectively. Moreover, if there is no special explanation, we also relate $g_0 > g$ to the enhancement of the attribute g and $g_0 < g$ to the reduction.

4.1.3 Results

In this section, we will numerically demonstrate the effects of F as well as the combinations of F and r on the system of Eq. (4.1), in which the natural frequency ω_i

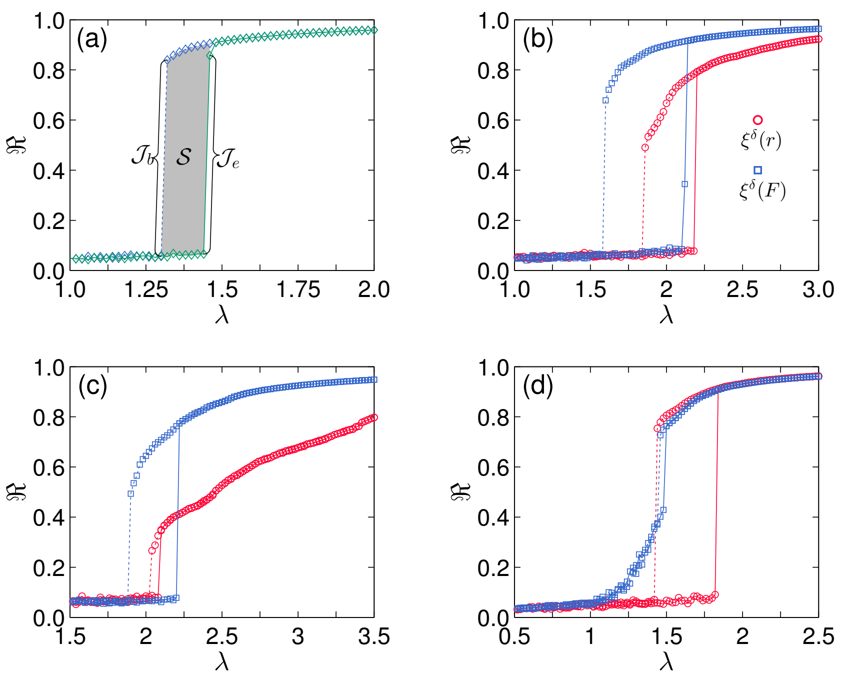


Figure 4.1: The magnitude of synchronization $\Re$ versus the coupling strength λ for the forward and backward transitions in networks with $n = 10^3$ and $\langle k \rangle = 6.0$. (a) The initial SF network constructed using the BA model with $F = 0.197$ and $r = -0.067$. The solid and dashed lines respectively correspond to the forward and backward transitions, between which is the hysteresis area represented by $\mathcal{S}$. Besides, the maximal jump sizes are accordingly denoted by $\mathcal{J}_e$ and $\mathcal{J}_b$. (b) $r = 0.000$ with $F = 0.278$ adjusted by $\xi^\delta(r)$ (red circle) and $F = 0.250$ with $r = -0.033$ through $\xi^\delta(F)$ (blue square) on the initial SF network. (c) $r = 0.050$ with $F = 0.354$ and $F = 0.300$ with $r = -0.006$. (d) $r = -0.050$ with $F = 0.213$, and $F = 0.150$ with $r = -0.077$.

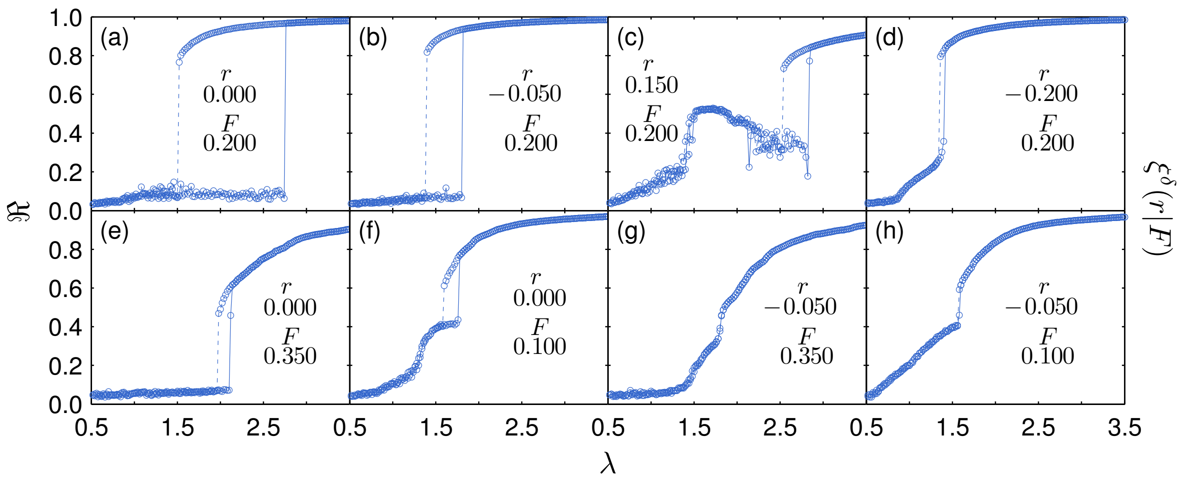


Figure 4.2: The magnitude of synchronization $\Re$ versus the coupling strength λ for forward and backward transition on networks with $n = 10^3$ and $\langle k \rangle = 6.0$ under different assortativity r and robustness F. (a-d) $\xi^\delta(r|\xi^\delta(F) \equiv 0.20)$. (e) and (g) $\xi^\delta(r|\xi^\delta(F) \equiv 0.35)$. (f) and (h) $\xi^\delta(r|\xi^\delta(F) \equiv 0.10)$.

of each oscillator is fixed to the associated node degree k_i. In addition, the order parameter $\Re$ (Eq. (4.2)) is calculated by simulating the system long enough until it is stable using the adaptive Runge-Kutta-Fehlberg method (Fehlberg's 4(5) method) [115] with error tolerance 1×10^{-4}, respectively for the forward and backward evolutions of the coupling strength $\lambda(\tau) := \lambda_0 + \tau\Delta\lambda, \forall \tau \in [0, L], \tau, L \in \mathbb{N}$, where λ_0, $\Delta\lambda$ and L are given values. In other

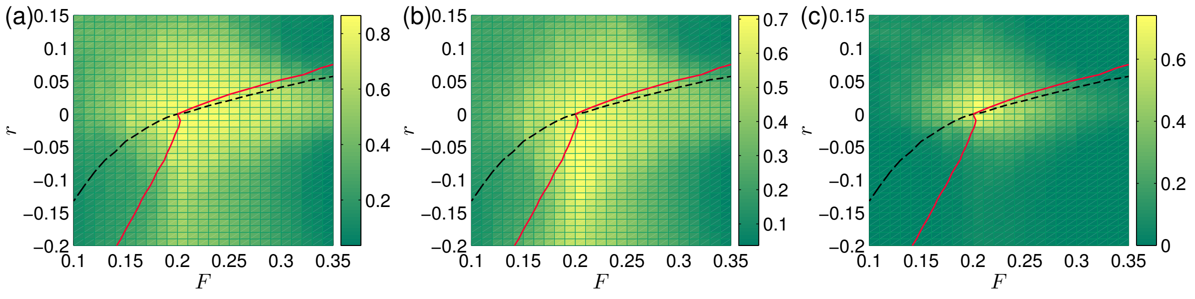


Figure 4.3: The jump size J and hysteresis area $\mathcal{S}$ of the network robustness F and assortativity r. (a) The forward synchronization $\mathcal{J}_e$. (b) The backward synchronization $\mathcal{J}_b$. (c) $\mathcal{S}$. The solid and dashed curves correspond to F of r with $\xi^\delta(r)$ and r of F with $\xi^\delta(F)$, respectively. Each result is the average of 20 network realizations.

words, each $\lambda(\tau)$ corresponds to a steady state of $\Re(\lambda(\tau))$, and the forward transition of $\Re$ evolves with an ascending order of τ and the backward transition with a descending order of τ. Note that $\Delta\lambda = 0.02$ is considered here and all of the initial SF networks are constructed using the BA model [18]. Besides, For a convenient description, the symbols 'e' and 'b' are accordingly used to mark the forward and backward evolutions of λ.

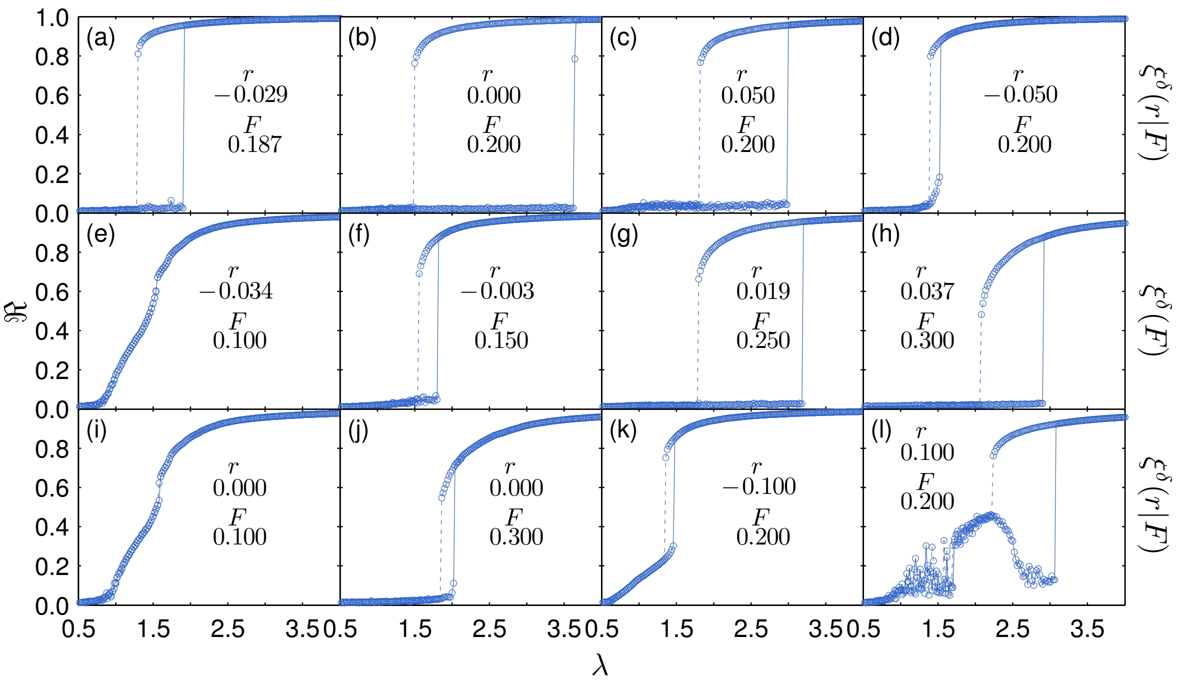


Figure 4.4: The magnitude of synchronization $\Re$ versus the coupling strength λ for forward and backward transition on networks with $n = 10^4$ and $\langle k \rangle = 6.0$ for different r and F. The result of the paradigm network from the BA model is reported in (a).

We first investigate the effects of the network robustness F and assortativity r on ES through $\xi^\delta(F)$ and $\xi^\delta(r)$ on networks with $\langle k \rangle = 6.0$, respectively. Fig. 4.1a shows the order parameter $\Re$ dependent on the coupling strength λ for the forward and backward transitions on the initial BA network, where $\mathcal{J}_e$ and $\mathcal{J}_b$ accordingly represent the maximal jump size of $\Re$ of the forward and backward transitions. Mathematically, they are obtained through the following ways,

$$\begin{aligned}\mathcal{J}_e &= \Re_e(\lambda(\tau_e+1)) - \Re_e(\lambda(\tau_e)), \tau_e := \arg\max_{\tau}\left(\Re_e(\lambda(\tau+1)) - \Re_e(\lambda(\tau))\right),\\ \mathcal{J}_b &= \Re_b(\lambda(\tau_b)) - \Re_b(\lambda(\tau_b-1))), \tau_b := \arg\max_{\tau}(\Re_b(\lambda(\tau)) - \Re_b(\lambda(\tau-1))).\end{aligned} \tag{4.3}$$

And

$$\mathcal{S} = \Delta\lambda \sum_{\tau'=\tau_b}^{\tau_e} [\Re_b(\lambda(\tau')) - \Re_e(\lambda(\tau'))] \tag{4.4}$$

denotes the hysteresis area, in which $\tau' \in \mathbb{N}$. Indeed (see the red circle in Fig. 4.1(b-d)), on the one hand, both $\mathcal{S}$ and $\mathcal{J}$ vary over the network assortativity and are suppressed by either a large or a small r, which is in line with the results in refs. [112, 113]. On the other hand, a similar change of $\mathcal{S}$ and $\mathcal{J}$ could be achieved through the adjustment of the network robustness (see the blue square in Fig. 4.1(b-d)). That is, there is also a specific range of F where both $\mathcal{S}$ and $\mathcal{J}$ could reach larger values than those that do not belong to that region. But in general, we also have: no matter which one of r and F is adjusted, the other one would positively increase or decrease.

Thus, we next fix either F or r, and then vary the other one to separately verify the dependence of ES on the network robustness and assortativity. Fig. 4.2 illustrates the forward and backward synchronization on networks with different r and F of 0.100, 0.200 and 0.350. Note that $\xi^\delta(r|\xi^\delta(F))$ means that we first enhance or weaken the network robustness F and then adjust the network assortativity r to a certain value by keeping F constant. As manifested in Fig. 4.2(a-d), though r can narrow the hysteresis area $\mathcal{S}$ and decrease the jump size $\mathcal{J}$, both $\mathcal{S}$ and $\mathcal{J}$ still exist even r taking a much large or small values (0.150 and -0.200) if F is fixed to 0.200. In contrast (Fig. 4.2(e-h)), the change of F sharply decreases the size of $\mathcal{S}$ and $\mathcal{J}$, and in some cases, they even disappear, which indicates that we could certainly control $\mathcal{S}$ and $\mathcal{J}$ through the interaction of r and F.

Fig. 4.3 shows the results of the impacts of the interaction of the network assortativity and robustness on the jump size and hysteresis area of ES, as well as F of r with $\xi^\delta(r)$ and r of F with $\xi^\delta(F)$. Without loss of generality, we also consider the cases of BA networks with $n = 10^3$ and $\langle k\rangle = 6.0$. Due to the limitation of n, networks constructed by the BA model are a little disassortative and their F is slightly less than 0.200. Therefore, we employ $\xi^\delta(r|\xi^\delta(F) \equiv 0.200)$ to reconstruct the paradigmatic BA networks and then generate our experimental networks. As a result, all networks used for further study in this simulation are with $\langle k\rangle = 6.0$, $F = 0.200$ and $r = 0.000$ (the differences of both F and r among those networks are within 1×10^{-5}). Finally, based on those networks, we derive Fig. 4.3 through $\xi^\delta(r|\xi^\delta(F))$, which means that $\xi^\delta(F)$ and $\xi^\delta(r|F)$ are successively used to adjust the network structure.

From Fig. 4.3, the following conclusions could be drawn:

1) the explosive synchronization is more likely in assortative networks with an enhancement of robustness compared to those with disassortativity and weak robustness;

2) extreme values of r and/or F would suppress the jump size $\mathcal{J}$ and hysteresis area $\mathcal{S}$;

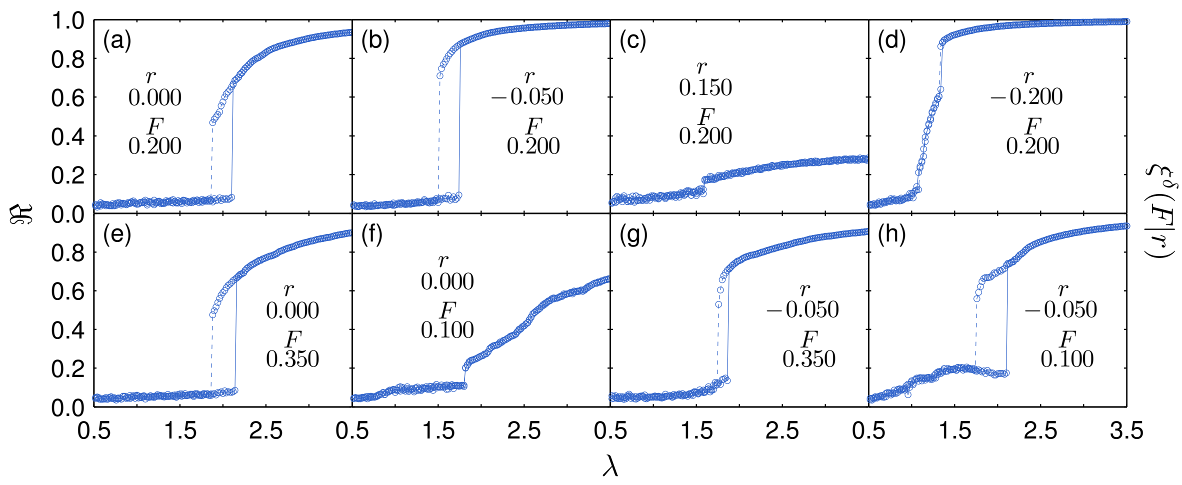


Figure 4.5: The magnitude of synchronization $\Re$ versus the coupling strength λ for forward and backward transitions on networks with $n = 10^3$ and $\langle k \rangle = 6.0$ under different assortativity r and robustness F. (a, e, f) $\xi^\delta(F|\xi^\delta(r) \equiv 0.000)$. (b, g, h) $\xi^\delta(F|\xi^\delta(r) \equiv -0.050)$. (c) $\xi^\delta(F|\xi^\delta(r) \equiv +0.150)$. (d) $\xi^\delta(F|\xi^\delta(r) \equiv -0.200)$.

3) the existence range of $\mathcal{J}$ is much larger than that of $\mathcal{S}$;

4) $\mathcal{J}_e$ is larger than $\mathcal{J}_b$ in assortativity networks but smaller in disassortativity networks;

5) there is an area within which both $\mathcal{J}$ and $\mathcal{S}$ reach peaks under the interaction of r and F.

In detail, the solid curve in Fig. 4.3 represents the influence of r on $\mathcal{J}$ and $\mathcal{S}$ without the control of F, which is related to refs. [112, 113, 114]. In addition, F shoots up with the increase of r for $r > 0$, but slowly falls when r decreases if $r < 0$. This means that $\mathcal{J}$ actually remains in a similar range of assortativity and disassortativity if F is unfixed. With respect to $\mathcal{S}$, we have results similar to ref. [113], namely, $\mathcal{S}$ reaches its maximum in weak assortativity networks and vanishes quickly as networks become disassortative. A contrary trend of r is observed in the process of $\xi^\delta(F)$ (see the dashed lines in Figs. 4.3), under which $\mathcal{J}$ and $\mathcal{S}$ disappear dramatically with the decrease of F and stay in a large range of increasing F.

From Fig. 4.3, the following conclusions could be drawn: 1) the explosive synchronization is more likely in assortative networks with an enhancement of robustness compared to those with disassortativity and weak robustness; 2) extreme values of r and/or F will refrain the jump size $\mathcal{J}$ and hysteresis area $\mathcal{S}$; 3) the existence range of $\mathcal{J}$ is much larger than that of $\mathcal{S}$; 4) $\mathcal{J}_e$ is larger than $\mathcal{J}_b$ in assortativity networks but smaller in disassortativity networks; 5) there is an area within which both $\mathcal{J}$ and $\mathcal{S}$ reach peaks under the interaction of r and F. In detail, the solid curve in Fig. 4.3 represents the influence of r on $\mathcal{J}$ and $\mathcal{S}$ without the control of F, which is related to refs. [112, 113, 114]. In addition, F shoots up with the increase of r for $r > 0$, but slowly falls when r decreases if $r < 0$. This means that $\mathcal{J}$ actually remains in a similar range of assortativity and disassortativity if F is unfixed. With respect to $\mathcal{S}$, we have results similar to ref. [113], namely, $\mathcal{S}$ reaches its maximum in weak assortativity networks and vanishes quickly as networks become disassortative. A contrary trend of r is observed in the process of $\xi^\delta(F)$ (see the dashed lines in Figs. 4.3), under which $\mathcal{J}$ and $\mathcal{S}$ disappear dramatically with the decrease of F and stay in a large range of increasing F.

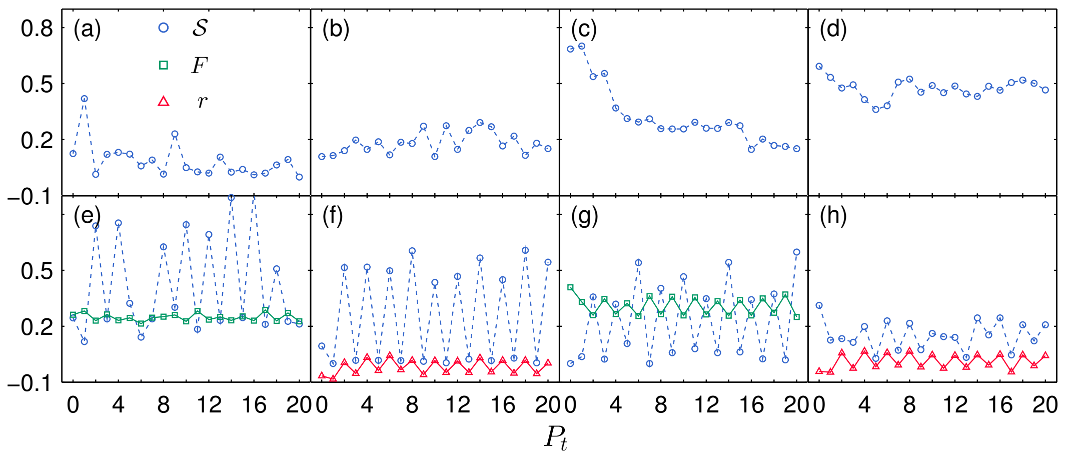


Figure 4.6: The hysteresis area $\mathcal{S}$ versus the perturbations of network assortativity r and robustness F on networks with $\langle k \rangle = 6.0$, $n = 10^3$ (the same initial network as Fig. 4.2) for (a,b,e-h) and $n = 10^4$ (the same initial network as in Fig. 4.4) for (c,d). (a,b) Respectively for $\xi^\delta(r|F)$ and $\xi^\delta(F|r)$ with $r(0) = -0.067$ and $F(0) = 0.197$. (c,d) Respectively for $\xi^\delta(r|F)$ and $\xi^\delta(F|r)$ with $r(0) = -0.029$ and $F(0) = 0.187$. (e, g) $\xi^\delta(r)$ with $r(0) = 0.000$ and $r(0) = 0.050$, accordingly. (f, h) $\xi^\delta(F)$ with $F(0) = 0.197$ and $F(0) = 0.250$, accordingly.

We further validate the above conclusions in a much larger network with $n = 10^4$ and $\langle k \rangle = 6.0$. Figs. 4.4(b, i, j) and Figs. 4.4(b-d, k, l) depict that how ES is influenced by varying F or r, while the other one is fixed: making network fragile is more likely to collapse ES than enhancing the network's robustness, and a larger $\mathcal{S}$ can be observed in assortative networks compared to disassortative networks. When r is not fixed (Figs. 4.4(e-f)), $\mathcal{J}$ and $\mathcal{S}$ disappear in a high speed with decreasing of F and stay in a large range for increasing F. Moreover, in these cases, $\mathcal{S} = 1.979$ reaches its maximum when $r = 0.000$ and $F = 0.200$, and it is also much larger than that in the network with $n = 10^3$.

4.1.4 Discussions

By now, we have presented our results of $\xi^\delta(r|\xi^\delta(F) \equiv F_0)$, namely, the network robustness F is initially given for a certain value F_0, and then the network assortativity is further adjusted with the constraint $F \equiv F_0$. But what would happen if we employ $\xi^\delta(F|\xi^\delta(r) \equiv r_0)$ to modify the network? Fig. 4.5 shows the corresponding results of $\xi^\delta(F|\xi^\delta(r))$ on the same network in Fig. 4.2. The strongest difference between them is in Fig. 4.2 and Fig. 4.5(c,d,g,h), where the hysteresis area vanishes in one network, while it still exists in the corresponding other one, even though they both have the same r and F. One reason may be ascribed to the fact that both r and F are disturbed heavily in Figs. 4.2(g,h) and Figs. 4.5(c,d), e.g., for Fig. 4.5c, F will considerably increase with the rise of r and then be dragged back to 0.200 (also see the solid line in Fig. 4.3). This can be verified in a similar way as in Fig. 4.3.

Another interesting difference is that the hysteresis area in Fig. 4.2a is much larger than that in Fig. 4.5a (also see Fig. 4.8). The main distinct process between them is that the change trend of F, i.e., for $\xi^\delta(r|\xi^\delta(F))$ (Fig. 4.2a), F only undergoes an increasing process (ignore the tiny adjustment), while it firstly increases with the rise of r and then decreases to 0.200 through $\xi^\delta(\xi^\delta(F)|r)$ (Fig. 4.5a). Thus, how do the increasing and decreasing behaviours

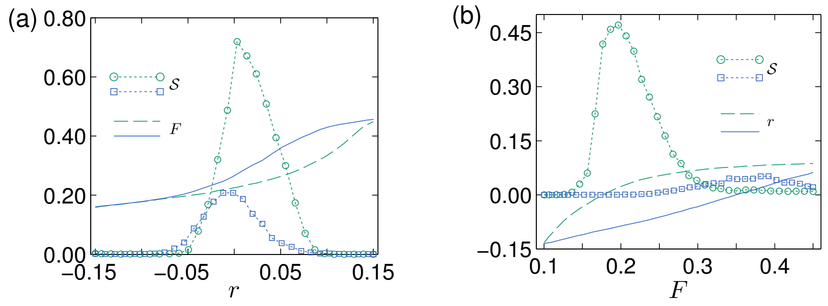


Figure 4.7: The hysteresis area $\mathcal{S}$ and the network robustness F or assortativity r versus r or F for forward (blue) and backward (green) evolutions on networks with $\langle k \rangle = 6.0$ and $n = 10^3$.

influence the ES? Fig. 4.6 shows $\mathcal{S}$ of the perturbations of r and F. For a convenient description, we employ $r(P_t)$ and $F(P_t)$ to be associated with the perturbation step P_t. In detail, at each step (P_t), the associated property (r or/and F) is either increased ($\xi^+(g)$) or decreased ($\xi^-(g)$) (assuming 10^4 times of the cut-add strategy), and then adjusted back to the given value, i.e., networks for each data point have similar properties. Besides, we here set odd steps ($P_t = 1, 3, \ldots$) as the decreasing process and even steps ($P_t = 2, 4, \ldots$) as the increasing process. Considering Fig. 4.6(e) as an example, we firstly employ $\xi^-(r)$ (10^4 times of the cut-add strategy) to reduce the network assortativity and then adjust it to $r(1) \approx r(0)$ through $\xi^\delta(r)$ to obtain the network for $P_t = 1$. Based on this network, we further use $\xi^+(r)$ to enhance the network assortativity and then tune it back to $r(2) \approx r(1) \approx r(0)$ to get the network for $P_t = 2$, etc. In this manner, we continually disturb r but at each data point $r(P_t) \approx r(0), P_t = 1, 2, 3 \ldots$.

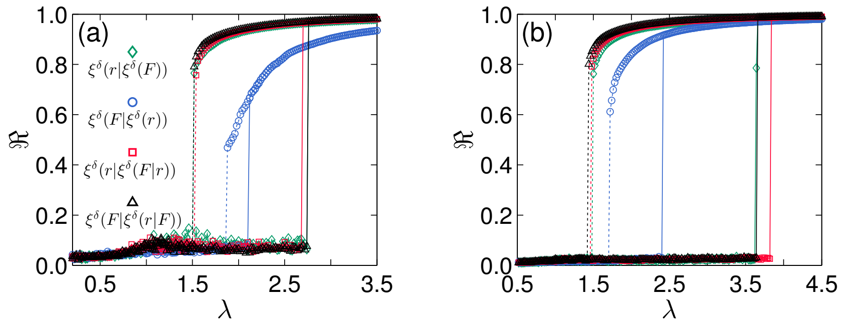


Figure 4.8: The magnitude of synchronization $\Re$ versus the coupling strength λ for forward and backward transition on networks with $\langle k \rangle = 6.0$, $r = 0.000$ and $F = 0.200$ for (a) $n = 10^3$ and (b) $n = 10^4$. $\xi^\delta(r|\xi^\delta(F|r))$ here means that F is firstly adjusted to 0.200 by keeping r constant and then change r to 0.000 with fixed $F \equiv 0.200$.

As shown in Fig. 4.6 (a-d), the hysteresis area $\mathcal{S}$ decreases with the increase of P_t if we disturb the network assortativity when keeping F constant, and it undergoes a slight fluctuation by $\xi^\delta(F|r)$. Note that the perturbation process is different from that in Fig. 4.3. Besides, the utmost interest may locate in Fig. 4.6 (e-h), where we only consider the perturbation from either r or F. For both cases, $\mathcal{S}$ periodically fluctuates in the evolution of P_t. Specifically, when disturbing r, $\mathcal{S}$ negatively changes as the network robustness evolves, while it is positively correlated to r through $\xi^\delta(F)$. In addition, the fluctuation between two adjacent P_t is very large. This suggests that quite different $\mathcal{S}$ will be obtained even though two networks have almost similar r or F.

To further demonstrate the results of Fig. 4.6 (e-h), we show the hysteresis area $\mathcal{S}$ and the network robustness F (or assortativity r) against the forward and backward evolutions of r (or F) in Fig. 4.7. For the backward evolution of r (Fig. 4.7a), we firstly disturb r through $\xi^{\delta}(r)$ until the network reaches a stable state (assuming $P_t \in [1, 100]$), where $\lfloor 10000/P_t \rfloor$ is conducted for the cut-add strategy at each perturbation step. After this, r is increased to 0.15 and then gradually decreased to -0.15, during which we capture the temporary networks with an interval of around 0.01 of r. We independently repeat this process 20 times and obtain the backward transition of r, which is shown as the green-dotted-circle curve in Fig. 4.7a. The forward evolution of r is gathered in a similar way, but with the process that r is decreased to -0.15 and then gradually increased to 0.15. The similar strategy is also conducted to evolve F.

As illustrated in Fig. 4.7, there is a gap of F or r between $\xi^{-}(\cdot)$ and $\xi^{+}(\cdot)$. This gap changes the place of the peak of $\mathcal{S}$, which indicates that δ of $\xi^{\delta}(\cdot)$ truly plays an important role in $\mathcal{S}$. Moreover, the magnitude of synchronization $\Re$ versus the coupling strength λ for the forward and backward transitions on networks with the same $\langle k \rangle$, r and F is demonstrated in Fig. 4.8. These results are also twofold. On the one hand, there might be a maximum of $\mathcal{S}$ achieved by appropriate adjustments of r and F. On the other hand, the order of the adjustment plays a crucial role, which means that r and F are not the only two properties influencing ES. By and large, this problem is still open and needs further study.

4.1.5 Summary

The robustness of networks as a fundamental problem in network science has been widely studied during the past decade [2]. Those studies mainly consider how the structure of networks influences their robustness. Here we view the network robustness F as a measure of networks and employ it to capture the change of networks' structure. More specifically, we have numerically studied effects of the network robustness as well as its interaction with network assortativity r on explosive synchronization (ES), and have found that both extreme values of F and r would suppress ES, especially the hysteresis area between the forward and backward transitions. In particular, for a network constructed by the BA model, there is a maximum of hysteresis area achieved by appropriate adjustments of F and r. In addition, our discussion reveals that the change trends of both the network robustness and assortativity play important roles in ES. In other words, different behaviours of ES are found in networks with similar F or/and r if the networks undergo different change processes, which remains the problem of effects of the network structure on ES still partly open. More discussions will be found in Section 4.4.

4.2 Ways to Enhance the Robustness of a Network

From Section 3 we know that the robustness of a network is associated with a number of factors, such as the dense of edges (average degree), the degree distribution, the attack strategy, the correlations among nodes, the existence of community structures, etc. Usually, the denser a network is, the more robust it would be. But for most networks, like constructing a new road in the road network, a new edge means a huge amount of cost. Besides, the degree distribution represents the capabilities of nodes, which is also usually difficult to manipulate, e.g., the number of airlines that an airport could handle always keeps unchanged. Regarding the correlations and community structures, one can easily observe their influences from the comparisons of Figs. 3.28 and 3.29. Therefore, in this section, following the assumption that the degree of each node is fixed [5], we are going to mainly study how to overcome the 'bucket effect' and increase a network's robustness regarding varied intentional attacks.

4.2.1 The power of selection

As studied in Section 4.1.2, for a given network $G(\mathcal{N}, \mathcal{M})$, its degree sequence would keep unchanged if we tune the network through the cut-add strategy, i.e., $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ appear in a pair. Hence, a straightforward way to improve G's robustness regarding a specific attack strategy S is as follows[1]:

1) randomly pick up two edges e_{ij} and e_{uv} satisfying $i \neq j \neq u \neq v$ and $\sigma_{iu}, \sigma_{jv} \notin \mathcal{M}$;

2) get a new network G' through $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$;

3) obtain S' of G';

4) let $G \leftarrow G'$ and $S \leftarrow S'$ if $F'(S') > F(S)$, where $F'(S')$ and $F(S)$ are associated with G' and G, respectively;

5) repeat those steps t times.

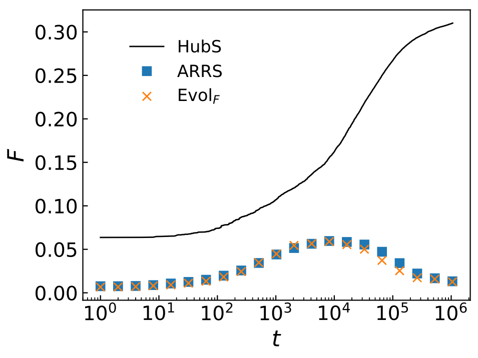
In general, step 3) of the above procedure accounts for the main part of the time consumption, and step 4) follows (calculation of $F'(S')$). Here, for a convenient description, we still use the notation from 4.1.2, i.e., $\xi^+(g)$, in which g corresponds to $F(S)$.

Apparently, most of the advanced strategies that we studied in Section 3 would face the problem from step 3). For instance, compared to HubS, ARRS would take much more time to get a result, even though it could have better results than ACIS, AEIS, and ABPDS using much less time. Note that t is usually much larger than n to enhance a network, no matter whether there are other constraints[2]. One can also imagine a scenario that we try to enhance the robustness of the as-Skitter network, on which even ABonS1 would take a few seconds to get a result. All those facts indicate that we cannot employ $g = F(S)$ directly for an advanced strategy, particularly on networks of large size.

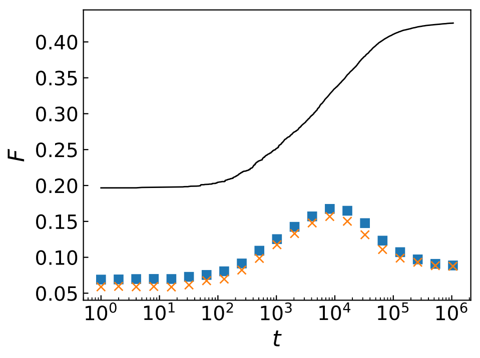
Instead, a trick that we could use is to let $g = F(\text{HubS})$ in $\xi^+(g)$ since $S \equiv S'$ holds for HubS, which means step 3) could be effectively eliminated. But we still face another problem: the 'bucket effect'. Fig. 4.9 illustrates the evolution of F as a function of t considering $g = F(\text{HubS})$. Indeed, regarding HubS, both networks become more and more robust as t increases. But if they are attacked by ARRS and Evol$_F$, one can easily observe that there

[1]Here we mainly consider F and one can study q_c in the similar way.

[2]We mainly consider a constraint of keeping the node degree fixed since others could follow a similar way to study, e.g., accept a switch if it does not result in an increment of the edge weight (which could be the length of airlines or the cost of relationships).



(a)

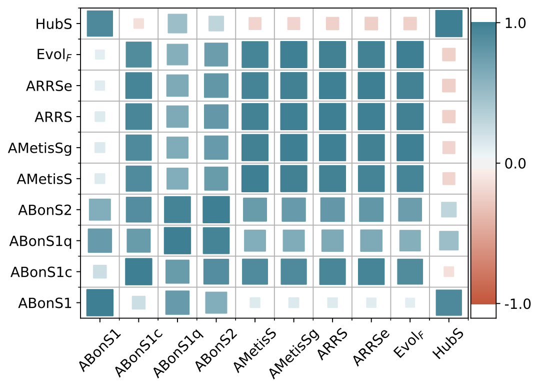


(b)

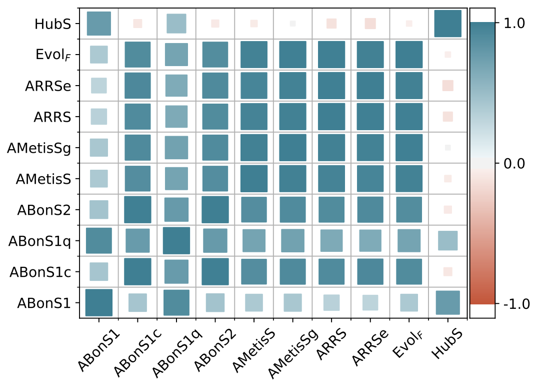
Figure 4.9: Evolution of F under the cut-add strategy regarding $g = F(\text{HubS})$ on (a) the power grid network and (b) the yeast network. Marks with respect to ARRS and Evol$_F$ are obtained on networks at $t = 2^0, 2^1, 2^2, ..., 2^{20}$.

are some critical t_c, after which both $F(\text{ARRS})$ and $F(\text{Evol}_F)$ would decrease as $F(\text{HubS})$ increases. In other words, there is some peak of F associated with an advanced strategy if a network is enhanced by $g = F(\text{HubS})$.

A straightforward way to acquire that peak is using a selection strategy similar to the one conducted by ARRS. That is, starting from G' at t, we run the cut-add method a number of times based on $g = F(\text{HubS})$ and then get a new network G''. After that, the corresponding advanced strategy, e.g., ARRS, is used to decide whether G'' should be kept. But we might still need a number of such selections to find the optimum. Hence, as an alternative, we may rely on some basic method which could of course get a result using much less time than ARRS. Meanwhile, it also has a similar trend of F as ARRS. A such fast strategy might be ABonS1, ABonS2, or ARRS with an early stop.



(a)



(b)

Figure 4.10: Illustration of the correlation matrix based on the Pearson correlation coefficient of F at $t = 2^0, 2^1, 2^2, ..., 2^{20}$ on (a) the power grid network and (b) the yeast network under the condition same as Fig. 4.9. One can read the magnitude of corresponding coefficient through both the color and the size of each square (i.e., the larger the magnitude is, the bigger the related square is).

Fig. 4.10 shows the correlation pattern in regard to the Pearson correlation coefficient Coef of F. Specifically, same as Fig. 4.9, we first obtain a bunch of networks at $t = 2^0, 2^1, 2^2, ..., 2^{20}$ using $g = F(\text{HubS})$. Then, all the considered methods run on those networks, and a number of F regarding different t and methods could be obtained. For each method, we have a series of F as a function of t, and the Pearson correlation coefficient is drawn between every pair methods. As we can see from Fig. 4.10, though ABonS1 fails, both ABonS1c and ARRSe could effectively capture the trend of Evol_F, where ABonS1c[3] is a variant of ABonS1 by directly considering the component size, and ARRSe is the case of ARRS with $T = 10$. Recall that our finial goal is to tackle the 'bucket effect' problem, namely, to enhance a network to be against the most advanced attack strategy, i.e., Evol_F. More precisely, ABonS1c has Coef $= 0.90$ in both network while ARRSe holds 0.98.

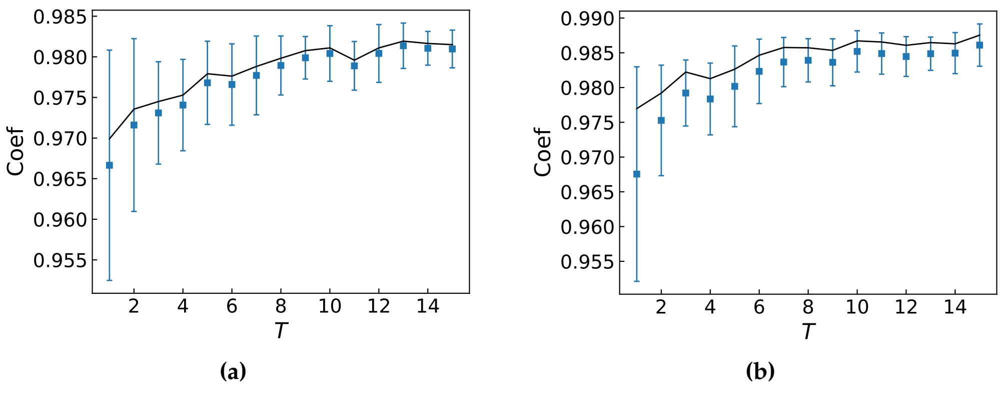


Figure 4.11: The Pearson correlation coefficient Coef of F between Evol_F and ARRSe with different T under the condition same as Fig. 4.9. The solid line is drawn based on the mean of F over 20 IIs. Meanwhile, the square mark corresponds to the case of obtaining F by running ARRSe once. Each error bar is obtained by calculation the standard deviation over 20 IIs.

Therefore, we also depict the Pearson correlation coefficient Coef of F between Evol_F and ARRSe with different T under the condition same as Fig. 4.9 in Fig. 4.11. Here we consider two cases: i) run ARRSe 20 times on a specific network and use the mean to measure F (solid lines in Fig. 4.11) and ii) run ARRSe only once (square marks). As we can see from there, for the same T, Coef of case i) is always larger than the mean of case ii). But case i) also suffers a problem that it has to run ARRSe a number of times for the detection[4]. Indeed, one could overcome that problem by parallelizing it. But in what follows, we would mainly consider case ii) because it only needs to run ARRSe once each time, which makes it usually more effective compared to case i), especially when a network has a large size. Besides, as T increases, it becomes more and more stable. Nevertheless, one should find a balance between the time consumption and Coef. To sum up, among those methods in Fig. 4.10, ARRSe would be more preferable. And if a network has a very large size, then ABonS1c could be an alternative.

So far we have studied which method could be used to detect the critical t_c of Evol_F during a process of enhancement of robustness regarding $g = F(\text{HubS})$. For a convenient description, we employ $\hat{\alpha}$ to quantify $F(\text{Evol}_F)$ at t_c. Then, another problem arises as to how could we effectively further boost $\hat{\alpha}$? A straightforward approach might be: 1) run the cut-add strategy a number of times; 2) use ARRSe to decide whether those exchanges of edges could be accepted; 3) repeat the previous steps until this process is stable. Following

[3]ABonS1c is usually more effective than ABonS1 in small networks but less in large networks with respect to the robustness and immunization problems.

[4]One might also further study the influence of the number of repeated times on ARRSe.

that, now HubS could also be replaced by ARRSe, that is, step 1) is conducted on $g = F(S')$, where S' is fixed to ARRSe from step 2). In this manner, $\hat{\alpha}$ of Fig. 4.9a is increased to 0.1728 from 0.1568, and Fig. 4.9b to 0.0626 from 0.0598. Besides, if we initialize ARRSe based on different attack strategies (see Section 3) instead of HubS, then $\hat{\alpha}$ could be improved again. The reason might be ascribed to the fact that different initial attack strategies indirectly help keep the diversity of the network structure, which one can study more following the routine and also strategies similar to Section 3.5. All in all, we reach the following processes to enhance the robustness of a network $G(\mathcal{N}, \mathcal{M})$, that is,

e1) initialize ARRSe based on a specific attack strategy;

e2) run ARRSe on G and get the corresponding attack strategy S;

e3) run $\xi^+(g)$ a number of times (say t_l) on G and then get a new network G', where $g = F(S)$;

e4) run ARRSe on G' and get a new attack sequence S';

e5) $G \leftarrow G'$ and $S \leftarrow S'$ if $F(S')$ of G' is larger than $F(S)$ of G;

e6) get the enhanced network G by repeating steps e3), e4), and e5) a number of rounds (say t_g).

We call the above process WayEnhS. Note that t_g corresponds to the number of times that we run ARRSe. And in general, for the two networks in Fig. 4.11, G would become stable within 10 of t_g when $t_l = 10^3$. Besides, as we mentioned, one might of course conduct e1)-e6) a few times to acquire a more robust network (i.e., get a larger $\hat{\alpha}$) through initializing ARRSe based on different attack strategies per round.

But instead, we are here more interested in how to further speed up WayEnhS since our aim is to tackle networks of very large size, e.g., a network drawn over the whole population. From Section 4.1, particularly from Fig. 4.7, we learn that there is a positive feedback between the assortativity and $F(\text{HubS})$, i.e., a large r is always associated with a large $F(\text{HubS})$, which means that we can actually improve $F(\text{HubS})$ by increasing r. Besides, from the conclusion of iii) in Section 4.1.2, we know that r could be controlled locally (see also Eq. (2.26)). That is, we could also indirectly control $g = F(S)$ in step e3) locally, at least for the case of $g = F(\text{HubS})$.

Therefore, we reach a problem similar to the one associated with the explosive percolation (see Section 2.5). Specifically, considering the rewiring strategy in Section 4.1.2, if a pair of $\alpha_-(e_{ij}, e_{uv})$ and $\alpha_+(\sigma_{iu}, \sigma_{jv})$ is only accepted when

$$b_r = (k_i k_u + k_j k_v) - (k_i k_j + k_u k_v) \tag{4.5}$$

is larger than 0, then the assortativity r would be a monotonically increasing function of t. For a convenient of description, we name this process WayEnhSrv1. Note that WayEnhSrv1 only needs to consider the degree of the related four nodes rather than calculate $F(\text{HubS})$ for per t.

As exhibited in Fig. 4.12, where we verify the corresponding strategies through ARRS with $T = 10^3$ since the associated trend of F is actually our utmost interest, WayEnhSrv1

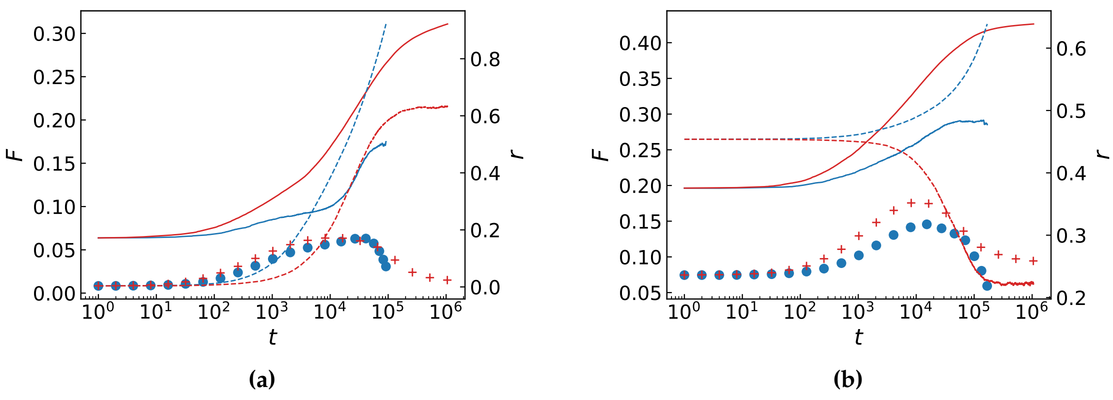


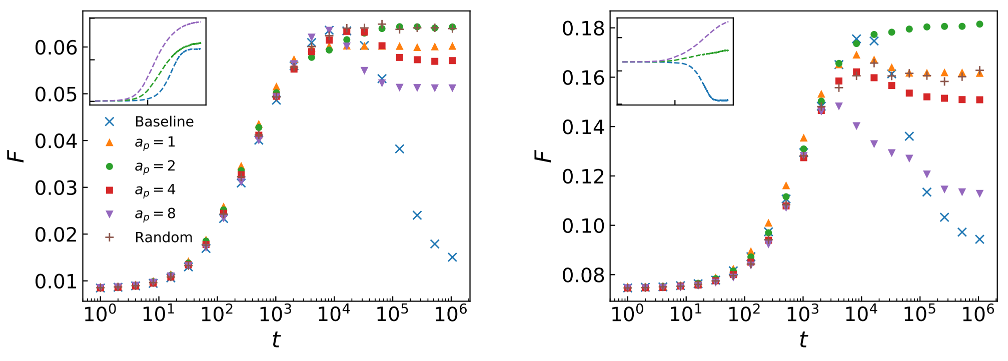
Figure 4.12: Evolution of F (regarding ARRS with $T = 10^3$ and HubS) and r versus t on (a) the power grid network and (b) the yeast network, where dashed lines correspond to r and others are associated with F. Amid them, $F(\text{HubS})$ is represented by solid lines and marks are $F(\text{ARRS})$. Meanwhile, the strategy based on HubS (see also Fig. 4.9) is colored red and the one based on assortativity is blue. All results are drawn from 20IIs.

acquires similar peak of $F(\text{ARRS})$ compared to the one based on HubS in the power grid network, which indicates that it truly works in this case. But for the yeast network, the performance of WayEnhSrv1 is worse than the one based on HubS. When looking into these two results, one can easily observe that r of the strategy based on HubS in Fig. 4.12a has a positive feedback of $F(\text{HubS})$ while a negative in Fig. 4.12b, which might account for the different performances that WayEnhSrv1 has. Nevertheless, if one needs a really fast way to enhance the robustness of a network, WayEnhSrv1 in tandem with the check of ARRSe would be a good choice.

Considering the different behaviors of r in Figs. 4.12a and 4.12b, we naturally ask what role does r play in WayEnhS? Specifically, for per t, if a few pairs of edges, say a_p, are selected, and only the one leading to the largest increase[5] of b_r is chosen to conduct the cut-add strategy, then could WayEnhS reach its peak sooner? Or could that strategy help WayEnhS acquire better results?

Firstly, we verify the effect of the above strategy on WayEnhSrv1, say WayEnhSrv1(a_p). As illustrated in Fig. 4.13a, both WayEnhSrv1(2) and WayEnhSrv1(Random) would become stable as t increases, which is a really good property since we do not need to check the critical point anymore as we do for the one based on HubS (i.e., the Baseline). And apparently, both of them could acquire networks which share the largest $F(\text{ARRS})$ with the Baseline, WayEnhSrv1(4) and WayEnhSrv1(8). But for the yeast network, only WayEnhSrv1(2) have the ability to surpass the Baseline. And the peak of WayEnhSrv1(8) is even much worse than the one that the Baseline has, which might also be ascribed to the differences of the assortativity r. Therefore, in the insert panels of Fig. 4.13, the assortativity r against t is also given with respect to the cases of the Baseline, WayEnhSrv1(2), and WayEnhSrv1(8). For the power grid network, WayEnhSrv1(2) undergoes the same trend of r as the Baseline, which indicates that r plays an important role in this type of network. But for the yeast network, a

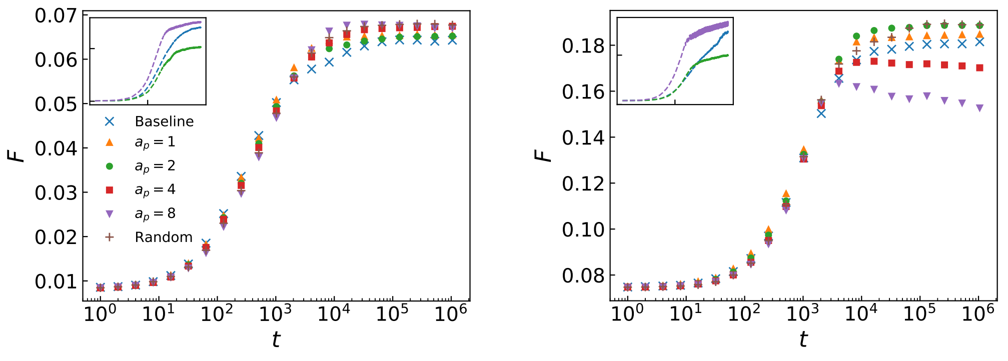
[5]Note that here we only study this special case. Other strategies over the choices of edges could be investigated following the same routine.



(a) **(b)**

Figure 4.13: F of ARRS (with $T = 10^3$) on networks enhanced by WayEnhSrv1(a_p) considering (a) the power grid network and (b) the yeast network, where the Baseline is the one based on HubS in Fig. 4.12, and the Random corresponds to the case of a_p randomly drawn from $[1, 10]$ for per $t_l = 10^3$. The dashed lines in the insert panel are associated with the assortativity r as a function of t in regard to the Baseline (Blue), $a_p = 2$ (green), and $a_p = 8$ (purple).

slight increase of r has a similar effect on $F(\text{ARRS})$ compared to a sharp decrease of r, which tells us that the robustness might also be controlled by some other attributes.



(a) **(b)**

Figure 4.14: F of ARRS (with $T = 10^3$) on networks enhanced by WayEnhS(a_p) considering (a) the power grid network and (b) the yeast network, where the Baseline is WayEnhSrv1(2) in Fig. 4.13, and the Random corresponds to the case of a_p randomly drawn from $[1, 10]$ for per $t_l = 10^3$. The dashed lines in the insert panel are associated with the assortativity r as a function of t in regard to the Baseline (Blue), $a_p = 2$ (green), and $a_p = 8$ (purple).

We then conduct the same strategy on WayEnhS, say WayEnhS(a_p), in which $t_l = 10^3$. The corresponding results are shown in Fig. 4.14, where the Baseline is WayEnhSrv1(2) in Fig. 4.13 instead of the one based on HubS. Different from Fig. 4.13a, WayEnhS(8) is better than others in the power grid network, where the performance of WayEnhS(a_p) actually increases as the rise of a_p. However, for the yeast network, $a_p = 8$ is again much worse than others. In addition, different from Fig. 4.14a, $F(\text{ARRS})$ has an inverse trend against a_p in Fig. 4.14b, that is, it sharply decreases as a_p increases. In other words, WayEnhS(a_p) could not find such a a_p like that $a_p = 2$ works on both networks for WayEnhSrv1(a_p). Nevertheless, WayEnhS(Random) works and could overcome that problem, which also has better results than WayEnhSrv1(2) (the Baseline).

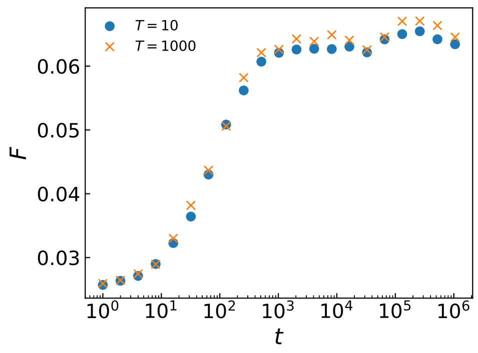


Figure 4.15: F of ARRS (with $T = 10^3$) on networks enhanced by WayEnhS(a_p) regarding different T for ARRSe on the German power grid network.

We finally consider the influence of ARRSe on WayEnhS. As we mentioned, the time consumption would increase as the rise of T for ARRSe. Hence, we here employ a quite small network, the German power grid network (with $n = 511$ and $m = 679$), to verify the effects of T on WayEnhS. As shown in Fig. 4.15, ARRSe with $T = 10^3$ could truly obtain better results than the one with $T = 10$. But the differences of F between them are very small, which, in tandem with the results from Figs. 4.10 and 4.11, indicates that $T = 10$ is enough for ARRSe for WayEnhS.

In conclusion, the significant performances of both WayEnhS(Random) and WayEnhSrv1(a_p) show us 'the power of selection'. And such selection mainly includes two parts. On the one hand, the local selection based on the assortativity could help us quickly enhance the robustness of a given network. One the other hand, the selection based on ARRSe accounts for the global one and could improve the associated performance again. In particular, for a network of small size, one could use WayEnhS(Random) to acquire a strongly robust network. Meanwhile, WayEnhSrv1(2) would be an effective alternative to tackle a network of very large size. In what follows, if there is no special explanation, we would mainly consider WayEnhS(Random) and WayEnhSrv1(2), and accordingly refer to them as WayEnhS and WayEnhSr.

4.2.2 Applications

We first verify WayEnhS by considering varied attack strategies (see also Section 3) on two small networks, the power grid network and the yeast network. Specifically, $t_l = 10^3$ and $t_g = 10^3$ are conducted for WayEnhS, based on which the enhanced networks are acquired. Following that, attacks are considered on both the original and enhanced networks. As we can see from Fig. 4.16, WayEnhS not only improve the 'bottle neck' of the robustness of a network against but also makes the network more capable of impeding attacks from all kinds of strategies. In particular, for the power grid network, if we let $F(\text{Evol}_F)$ of the enhanced network be F' and the one of the original network be F'', then F'/F'' approaches 9.15, which indicates that the enhanced network might be around 9 times more robust than the original one. For the yeast network, F'/F'' is even much larger, reaching 26.80. With respect to other strategies, the improvement is also significant. Besides, for both enhanced networks, ARRS and Evol$_F$ are much better than other methods, in particular, even ARRS accordingly has improvements (see also Eq. (3.33)) of 8.15% and 9.01% on the power grid network and the

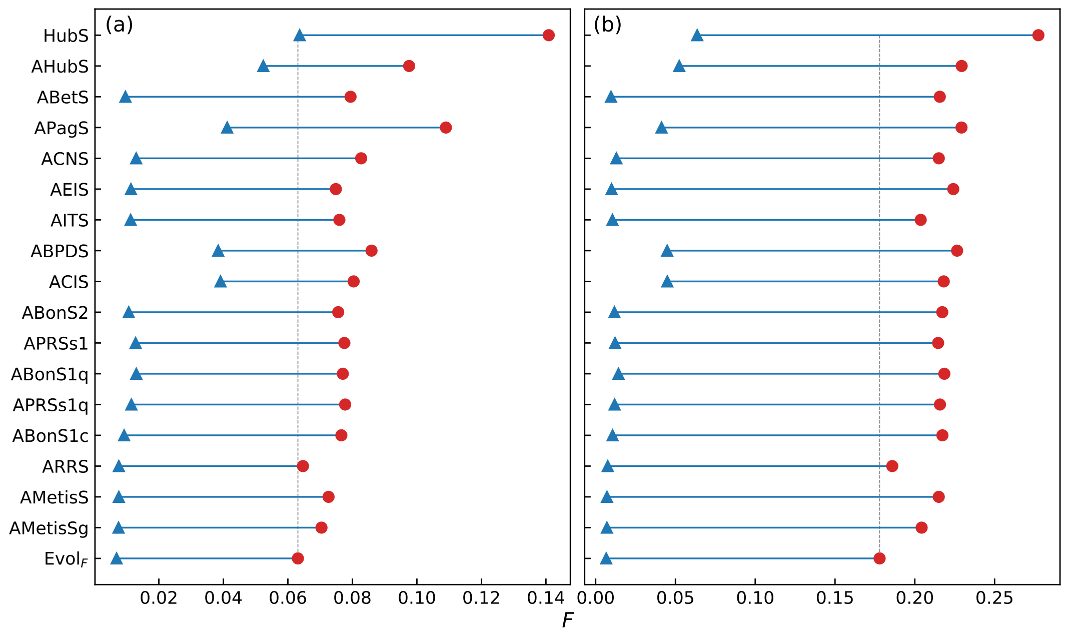


Figure 4.16: Performances of WayEnhS on (a) the power grid network and (b) the yeast network. The solid triangle is related to the original networks and the filled circle corresponds to the networks enhanced by WayEnhS. The dashed line marks F of Evol$_F$.

yeast network compared to AMetisSg, which means that both AMetisS and AMetisSg to some extend failed on both enhanced networks[6].

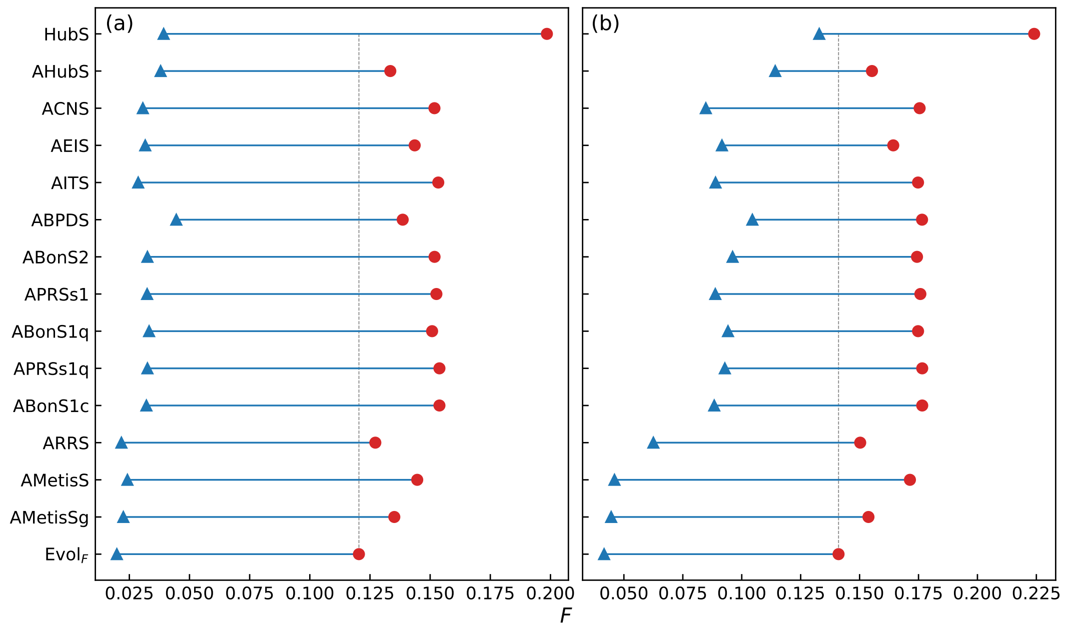


Figure 4.17: Performances of WayEnhSr on (a) the Email-Enron network and (b) the loc-Gowalla network. The solid triangle is related to the original network and the filled circle corresponds to the network enhanced by WayEnhSr. The dashed line marks F of Evol$_F$.

We then demonstrate the effectiveness of WayEnhSr on two large networks, the Email-Enron network and the loc-Gowalla network, where 10^8 exchanges are conducted. As exhibited in Fig. 4.17, WayEnhSr could also effectively increase the ‘bottle neck’ of the

[6]One can study more regarding the condition that AMetisS and AMetisSg would fail, in tandem with the results in Fig. 3.29.

network robustness. More precisely, it has $F'/F'' = 6.08$ and 3.39 for the Email-Enron network and the loc-Gowalla network. Though the improvements are not significant as the one that WayEnhS has, WayEnhSr is much faster than WayEnhS, e.g., it could acquire an enhanced network within 1 minutes in our simulation environment regarding the loc-Gowalla network.

4.2.3 Summary

As we have discussed, the true robustness of a network is detected by the most advanced attack strategy which, however, is usually too time-consuming to be a criterion for the cut-add strategy. To overcome this problem, we firstly study which naive method could be an alternative of the advanced one and reach ARRSe (our basic method ARRS with early stop), which could effectively check the critical point during the enhancement process based on HubS. We then further investigate that process and acquire our first method, WayEnhS, which has a better performance and similar time consumption against the one based on HubS. But WayEnhS is still hardly employed to tackle large networks, since it has to calculate $F(S)$ per t. Hence, we further investigate the effects of assortativity on WayEnhS, an attribute that highly correlates with the network robustness and could be controlled locally. Such trick not only helps us improve the performance of WayEnhS again but also gives us the second method, WayEnhSr, which is totally local. To demonstrate the effectiveness of both WayEnhS and WayEnhSr, four real-world networks are considered. The results regarding attacks from over ten state-of-the-art strategies illustrate that both of them have the ability to greatly improve the robustness of a network. Nevertheless, both WayEnhS and WayEnhSr are indirect methods to increase the bound of the robustness against the most advanced attack strategy, e.g., ARRSe with $T = 10^3$ is more effective than the one with $T = 10$, since the first has larger correlation with Evol_F. Thus, this problem is still open and more studies are needed, including itself of the most advanced attack strategy and its more reliable representative.

4.3 Influences of Acquaintances on the Containment of Epidemics

In this part, rather than consider the case in Section 3.1.2, i.e., the influence of the degree distribution on the epidemic spreading, we mainly focus on existing networks and study effects of the change of the corresponding structure on the spreading patterns under immunization. In particular, we will see how we tune the network structure to make the immunization strategy work more effectively.

4.3.1 Ways to weaken the robustness of a network

From Section 3, we know that the immunization problem and the robustness problem in a degree are equivalent to each other. Therefore[7], for the case of keeping the degree sequence fixed, we could actually directly use the strategies that we introduced in Section

[7]One potential application is that, e.g., one can temporarily reconstruct the contact network of a company or institute, so that the immunization of such network could be achieved by a small amount of staffs instead of all working from home during a crisis such as a pandemic.

4.2 but in a reverse process, i.e., $\xi^-(g)$ instead of $\xi^+(g)$. In other words, we replace step 4) in Section 4.2 with: let $G \leftarrow G'$ and $S \leftarrow S'$ if $F'(S') < F(S)$, where $F'(S')$ and $F(S)$ are associated with G' and G, respectively. To distinguish from WayEnhS and WayEnhSr, we mark their corresponding inverse processes with 'i', namely, WayEnhSi and WayEnhSri. Table 4.1 shows the associated results. As we can see from there, ARRSe only works in the power grid network (S2, S3, and S5), which indicates that WayEnhSi is not suitable for this problem. By contrast, $\xi^-(g)$ based on HubS (S1) works well. But again, it would suffer a problem of the time consumption. As an alternative, $\xi^-(r)$ (S8) has comparable results against S1 in the yeast network and the global airline network, but it is much worse than S1 in the power grid network and the as-733 network. Indeed, one could study more regarding this problem following a routine similar to Section 4.2. But here we are more interested in another problem.

Conf.	r	Random	HubS	ARRSe	Power	Yeast	as-733	Airline
S1			✓		0.0123	0.0315	0.0020	0.0206
S2				✓	0.0084	0.0754	0.0088	0.0363
S3		✓		✓	0.0084	0.0772	0.0089	0.0366
S4	✓		✓		0.0259	0.0325	0.0072	0.0216
S5	✓			✓	0.0082	0.0762	0.0089	0.0359
S6					0.0600	0.1617	0.0107	0.1174
S7		✓			0.0751	0.0940	0.0148	0.0618
S8	✓				0.0638	0.0328	0.0098	0.0218

Table 4.1: Performances of different configurations (Conf.) regarding WayEnhSi, WayEnhSri, and other strategies on the power grid network (Power), the yeast network (Yeast), the as-733 network (as-733), and the global airline network (Airline). S1 is $\xi^-(g)$ with $g = F(\text{HubS})$. S2 is WayEnhSi(1). S3 is WayEnhSi(Random). S4 is $\xi^-(g)$ with $g = F(\text{HubS})$ and only pairs leading to decreases of r are considered. S5 follows the same constrain from r as S4 but with ARRSe criteria. S6 is randomly rewiring the network a number of times. S7 is WayEnhSri (Random). And S8 is $\xi^-(g)$ with $g = r$. The numbers are the related results, i.e., F of ARRS (with $T = 10^3$) on networks drawn based on each configuration.

Supposing that a number of edges, say $m_r = |\mathcal{M}_u|$, are allowed to be removed from a network, we ask how this removal would influence the effectiveness of an immunization strategy. Specifically, considering a network $G(\mathcal{N}, \mathcal{M})$, we are interested in the following problems:

i) if m_r is given, then tackle
$$\arg\min_{\mathcal{M}_u} F'(\cdot), \tag{4.6}$$
where $F'(\cdot)$ corresponds to the remaining network $G' = G(\mathcal{N}, \mathcal{M} \setminus \mathcal{M}_u)$;

ii) if a certain $F'(\cdot)$ needs to be achieved, then tackle
$$\min m_r; \tag{4.7}$$

iii) a bi-objective problem regarding both Eqs. (4.6) and (4.7).

In what follows, we particularly focus on HubS as a case study, and others could be investigated in the similar way[8].

From the results in Section 4.2 and Table 4.1, we know that the assortativity r truly has effects on $F(\text{HubS})$. In tandem with the fact that a more practicable strategy should usually be local, an edge e_{ij} in our framework is removed if it has

Prodi1: maximum $k_i \times k_j$,

Prodi2: minimum $k_i \times k_j$,

Subti1: minimum $|k_i - k_j|$, or

Subti2: maximum $|k_i - k_j|$

among a_r candidates. Indeed, it again looks similar to the explosive percolation problem. Fig. A.25 (Appendix A.3.1) illustrates the associated results. As we can see from there, Prodi1 only works on the power grid network, where it is also better than Prodi2. But on other networks, Prodi2 surpasses Prodi1 and holds the best on two of them. Subti2 is slightly worse than Prodi2 on the yeast network and the global airline network, but it works on all four networks. Note that Prodi2 considers both the degree itself and the difference of corresponding nodes while Subti2 only captures the difference.

4.3.2 The role of less connected acquaintance

Based on those results, now we could further study the following scenario[9]: assuming that you are asked to temporarily stop contacting one or a few of your acquaintances (including all people whom you have connections with), who of them would contribute more to the protection of the whole network? Since aiming to design a local strategy, the below part would consider the degree of each node from the original network rather than keep updating as the removal of edges (e.g., the case in Section 4.3.1). Specifically, for a given network $G(\mathcal{N}, \mathcal{M})$, the following processes are conducted: i) randomly pick up a node i from G; ii) a_r random selections (might account for the irrational decision) are conducted on its corresponding nearest neighbors; iii) choose the node j from the a_r candidates and then remove e_{ij} based on Prodi1, Prodi2, Subti1, or Subti2; iv) repeat i)-iii) a number of times. As illustrated in Fig. A.26 (Appendix A.3.1), still only Prodi2 and Subti2 work. It is worth mentioning that Prodi2 is simply equivalent to choosing the node with the smallest degree[10] among the a_r candidates, which greatly facilitates the possibility of its implementation. Hence, in what follows, we would mainly consider and verify Prodi2.

[8] All three problems are actually NP-hard.

[9] In general, an epidemic would die out if we effectively control the basic reproductive number η_0 (see also Eq. (3.10)), such as $\eta_i a$ could be curbed by social distancing, face masks, or / and hands hygiene etc. Indeed, when the spread is mild, the Test-Track-Treat (test communities for diagnosis, track contacts of infection, and treat by the quarantine of those cases) strategy might be the most efficient approach. But when the situation becomes severe, that strategy would only have limited effect and a national restriction might be needed. Hence, here we discuss a local approach could play potential role in such as the suppression of a severe spread.

[10] This could be achieved by the aid of machine learning methods, such as train a model correlated with node degrees based on information like age, career, etc. Here we consider the case that the degrees of neighbors are inferred by the focused node.

4.3.3 Applications

Similar to Sections 3.4.5.5 and 3.5.7.4, we still employ the SIR model and follow the same settings (i.e., $\eta_r = 0.05$ and $b = 10^5$) to demonstrate the effectiveness of Prodi2. Specifically, we conduct the verification through the below processes: i) based on Prodi2, acquire a new network G' by removing m_r edges from the given network G; ii) immunize G' using RanS, AcqI, or HubS, and then independently run the SIR model on the associated G'. The sequence with respect to AcqI is obtained through: i) for a node i in G', randomly target one of its nearest neighbors; ii) check every node of G' and then score each node based on the number of targeted times; iii) independently repeat ii) 20 times and sum up their corresponding scores; iv) acquire the sequence based on the related score.

a_r	$0.5n$	$1.0n$	$1.5n$	$2.0n$	$2.5n$	$3.0n$
1	0.3546	0.2382	0.1563	0.0903	0.0461	0.0231
2	0.3570	0.2445	0.1555	0.0888	0.0453	0.0208
8	0.3533	0.2357	0.1484	0.0795	0.0380	0.0161

Table 4.2: The mean of the average infected frequency $\langle\alpha_{\text{inf}}\rangle$, $\langle\langle\alpha_{\text{inf}}\rangle\rangle$, over the infected probability $\eta_i \in [0.01, 0.20]$ (with interval 0.01) and the immunized fraction $q \in [0.01, 0.16]$ (with interval 0.0075) of RanS, in regard to networks drawn at $m_r = 0.5n, 1.0n, ..., 3.0n$ through Prodi2 with $a_r = 1, 2$ and 8 on the Email-Enron network.

a_r	$0.5n$	$1.0n$	$1.5n$	$2.0n$	$2.5n$	$3.0n$
1	0.0459	0.0342	0.0225	0.0124	0.0057	0.0025
2	0.0428	0.0304	0.0178	0.0087	0.0038	0.0014
8	0.0394	0.0241	0.0121	0.0051	0.0018	0.0005

Table 4.3: $\langle\langle\alpha_{\text{inf}}\rangle\rangle$ of Prodi2 regarding AcqI on the Email-Enron network.

a_r	$0.5n$	$1.0n$	$1.5n$	$2.0n$	$2.5n$	$3.0n$
1	0.0457	0.0341	0.0225	0.0125	0.0060	0.0026
2	0.0434	0.0304	0.0185	0.0094	0.0041	0.0015
8	0.0398	0.0248	0.0131	0.0057	0.0021	0.0005

Table 4.4: $\langle\langle\alpha_{\text{inf}}\rangle\rangle$ of Prodi2 regarding HubS on the Email-Enron network.

Tables 4.2, 4.3, and 4.4 exhibit the associated results on the Email-Enron network, where the mean of the average infected frequency $\langle\alpha_{\text{inf}}\rangle$ (see also Section 3.4.5.5 for its definition) are given, say $\langle\langle\alpha_{\text{inf}}\rangle\rangle$. As we can see from there, for all three cases (i.e., networks accordingly under the immunization of RanS, AcqI, and HubS), Prodi2 with $a_r = 8$ is much better than the one that randomly removes edges (namely, Prodi2 with $a_r = 1$), especially when m_r becomes large, e.g., the fraction of $\langle\langle\alpha_{\text{inf}}\rangle\rangle$ of $a_r = 8$ and the one of $a_r = 1$ is less 0.5 for $m_r = 2.0n$ regarding AcqI and HubS (Tables 4.3 and 4.4). Besides, Prodi2 with $a_r = 2$ is also much better than the random one for most cases. Meanwhile, for a specific a_r, $\langle\langle\alpha_{\text{inf}}\rangle\rangle$ would hugely decrease as m_r increases, even though each node is only asked to collapse one edge with its neighbors on average.

We then consider the contours of $\langle\alpha_{\text{inf}}\rangle$ as a function of q and η_i at $m_r = 2.0n$ on the Email-Enron network and the loc-Gowalla network. As depicted in Figs. A.27, A.28, and A.29 (Appendix A.3.1), on both networks, Prodi2 with larger a_r could achieve better results, namely, for a particular q, it is more resistance against larger η_i, and for a particular η_i, it needs less resources to reach the same $\langle \alpha_{\text{inf}} \rangle$. More precisely, to achieve $\langle \alpha_{\text{inf}} \rangle < 0.01$ at $\eta_i = 0.05$ on the loc-Gowalla network, AcqI accompanied by Prodi2 with $a_r = 8$ only needs to immunize $q = 0.06$ fraction of nodes compared to $q = 0.07$ and $q = 0.09$ needed for $a_r = 2$ and $a_r = 1$, respectively. In other words, Prodi2 with $a_r = 8$ could help save 0.03 fraction of nodes against the one with $a_r = 1$.

4.3.4 Summary

In this section, we have studied the immunization problem from another perspective, that is, the effects of the removal of edges on immunization strategies. Particularly, we want to know whether a simple removal of edges would boost the effectiveness of immunization strategies, and if it does, which strategy would be more effective. To answer these problems, we first investigated strategies extended from Section 4.2 and find that the assortativity also has impacts on immunization. In tandem with the fact that a practicable method should need information as little as possible, we proposed a very simple strategy called Prodi2, which searches for and targets edges connecting to nodes with a small degree. The results on the SIR model demonstrate that our strategy could help save a huge amount of resources (less nodes are needed to be immunized or removed to achieve the same goal). It is worth mentioning that the developed strategy is totally local. Hence, the combination of it and AcqI would be really useful in some cases, e.g., help distribute vaccines if the amount is very limited.

4.4 Prediction of the Hysteresis in Explosive Synchronization

4.4.1 Problems and motivations

From Section 4.1, we learned that both the robustness and assortativity have effects on the ES[11]. Meanwhile, from the comparisons of Figs. 4.2 and 4.5, and the results in Fig. 4.8, we also knew that they are not the only attributes to influence the ES. Particularly, taking the hysteresis area $\mathcal{S}$ between the forward and backward transitions as an example, one can easily observe that it has far different behaviours in Figs. 4.2(a) and 4.5(a), even though the degree sequence, $F(\text{HubS})$, and r of the two networks are same, which indicates that $\mathcal{S}$ is also influenced by other unknown attributes. Hence, is there any way that could help us to figure out what those unknown attributes are?

Further, though another conclusion which could be drawn from Section 4.1 is that the size of a network also plays an important role in the ES, we still do not know the exact relationship between $\mathcal{S}$ and n. To answer this, we first have to overcome two subproblems. That is, on the one hand, even a network of size 10^3 needs more than 15 hours[12] to acquire a result. And Fig. 4.3 is obtained based on $18,720$ networks, which indicates that it would be very expensive to get similar datasets for $n = 2 \times 10^3, 3 \times 10^3, \ldots$. On the other hand, if

[11] Note that all studied networks share the same degree distribution.

[12] We implement the corresponding numerical algorithm ourselves for this particular problem. And our testing results show that it is faster than ‘DifferentialEquations.jl’ and much faster than MATLAB.

a network is too large, then Eq. (4.1) would become stiff. Thus, is there any other way to predict the behaviour of a system on large networks rather than run the system directly?

In addition, regarding the ways to enhance the robustness of a network (Section 4.2), a goal function based on a more advanced strategy usually leads to a better result. But, as we mentioned, a more advanced strategy is always associated with more time consumption. Therefore, is it possible to find a faster and more suitable way to measure a network, by aid of which we could optimize the network directly rather than calculate, e.g., $F(\text{Evol}_F)$ per time? Furthermore, why did AMetisSg fail in cases of Figs. 3.28, 3.29, 4.16, and 4.17 etc.? And is there any better way than Prodi2 to boost the effectiveness of existing immunization strategies?

Indeed, we again cannot answer all those problems. In what follows, we are going to particularly consider the first problem and try to tackle it by the aid of machine learning tools. And others are partly or fully equivalent to that problem. More specifically, we will focus on the prediction of $\mathcal{S}$ and try to figure out whether it could be manipulated as that we did in Sections 4.2 and 4.3.

4.4.2 Basic idea

The reason that we choose to consider $\mathcal{S}$ is two-fold. On the one hand, compared to the critical threshold of the order parameter $\Re$, the maximum jump size $\mathcal{J}_e$, or $\mathcal{J}_b$, $\mathcal{S}$ is more robust against the numerical error and the coupling strength $\Delta\lambda$. On the other hand, since the natural frequency is correlated with the node degree, and the associated degree sequence keeps fixed, $\mathcal{S}$ to some extent only relies on the adjacency matrix, i.e., the network structure. In this manner, we reach a special case of the following problem: assuming that a given system[13] mainly or only depends on the network structure, i) study the effects of the network structure on such a system; and ii) control and optimize the network structure to optimize the behaviour of the corresponding system.

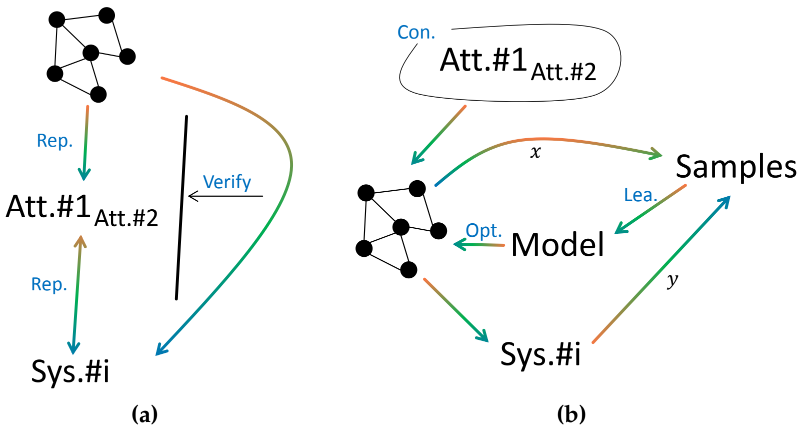


Figure 4.18: Framework of the investigation of effects of the network structure on dynamic systems.

Over the past decades, most works including Sections 4.1 and 4.2 and many others [9, 49, 113, 37] basically follow a routine from Fig. 4.18a. That is,

[13]We particularly consider systems without analytical solutions, which most complex systems follow.

a) represent a given network G by its corresponding attributes, say Att.#1 and Att.#2 etc.;

b) study the correlation between the behaviour $f(\cdot)$ of a considered system Sys.#i and those attributes by the aid of the verification from the direct investigation of Sys.#i on G;

c) achieve the prediction through the leading one, say Att.#1, or the combination of a few;

d) achieve the optimization by controlling Att.#1 or others.

However, the above process would suffer several problems. The first one is from b), i.e., the direct investigation of Sys.#i on G is usually very expensive: it is either hard to solve the problem analytically or would take a long time to get a numerical solution. Besides, as we studied in Section 4.1, the attribute is always correlated with each other and it would be really difficult to manipulate them simultaneously. Meanwhile, there are also some attributes that we might never know.

Due to those problems, in what follows, we are going to consider and study a routine from Fig. 4.18b. Specifically, from existing works, such as [116, 37], we could know what attributes have effects on the considered system. Through them, the network structure could be controlled and tuned, and further we can acquire a number of new networks (x in Fig. 4.18b), on which the corresponding $f(\cdot)$ (y in Fig. 4.18b) could be obtained too. Viewing x and y as samples, a model is learned by aid of machine learning tools. And the prediction and optimization might be achieved based on the learned model.

To sum up, we will particularly focus on $\mathcal{S}$ and first construct several datasets based on the studies in Section 4.1. Then, we are going to verify the effectiveness of a few machine learning methods on those datasets and see whether $\mathcal{S}$ is controllable and optimizable.

4.4.3 Essentials

4.4.3.1 Problems

As we mentioned, for a given network G and a system f (here Eq. (4.1)), we aim to learn a function $f' : x^{\alpha} \to \mathcal{S}'$ which could achieve the goal similar to $f : G \to \mathcal{S}$ (i.e., $|\mathcal{S}' - \mathcal{S}|$ should be as small as possible), where x^{α} is a feature vector with α dimensions. Meanwhile, $f'' : G \to x^{\alpha}$ should be much easier and also take much less time to be acquired compared to f. For instance, regarding the assortativity r, Eq. (2.26) could obtain $x = r$ with a time complexity of $O(m)$.

Then, we are going to gradually study the following problems:

P1) classification: predict whether $\mathcal{S}$ exists;

P2) regression: predict $\mathcal{S}$.

And verify

P3) optimization: $\max \mathcal{S}$.

All these problems would also be investigated with respect to varied attributes, degree sequences, and network sizes etc.

4.4.3.2 Baseline

The baseline is set to be the case that $x^{\alpha} \leftarrow [r, F]$[14] and f' is learned through random forests (RF)[15] [118, 117].

4.4.3.3 Data

The corresponding data are mainly from Section 4.1. Particularly, for the classification problem, networks with $0.01 \leq \mathcal{S} \leq 0.03$ are removed to avoid numerical errors. As a result, we have $16,655$ valid samples out of $18,720$ networks, which includes $7,115$ cases of nonexistence (i.e., $\mathcal{S} < 0.01$) and $9,540$ cases of existence ($\mathcal{S} > 0.03$) (see also Fig. A.30 in Appendix A.3.2 for visual examples). The regression problem considers all $18,720$ networks.

4.4.3.4 Experimental configurations

We particularly consider four cases summarized in Table 4.5. Specifically, based on different initial networks, we divide the data used by Fig. 4.3 into 20 groups. As a result, each group contains 936 networks which share the same degree sequence. But different groups correspond to different sequences[16]. Note that all networks in the 20 groups also have the same average degree $\langle k \rangle$ and network size n. Then, EXP1 and EXP2 are associated with the verification on a particular group, where the only difference between them is that EXP1 would shuffle the whole group while EXP2 does not. That is, EXP2 would usually not cover the bounds of r and F, e.g., learn a model based on networks with $F < 0.3$ and then predict $\mathcal{S}$ of those with $F > 0.3$. EXP3 is the case that a model is learned based on some groups and verified by the others. EXP4 is similar to EXP2 but the verification would be conducted based on networks in other groups.

Conf.	Same size	Same degree sequence	Mixed
EXP1	✓	✓	✓
EXP2	✓	✓	
EXP3	✓		✓
EXP4	✓		

Table 4.5: Experimental configurations.

4.4.4 Results

4.4.4.1 Method based on the eigenvalue

A straightforward way is to construct x^{α} based on the spectrum of a graph, whose effectiveness has been demonstrated in a large range of fields [119, 120, 121]. We here mainly consider the eigenvalues of the related normalized Laplacian matrix $\hat{\mathcal{L}}$ [119][17], which is

[14]One could of course add more attributes.

[15]If there is no special explanation, RF is conducted under default parameters given by ref. [117] (version 0.23.1).

[16]Different sequences follow a similar degree distribution.

[17]Others could follow the same manner to investigate.

defined as $\hat{\mathcal{L}} := D^{-1/2}\mathcal{L}D^{-1/2} = I - D^{-1/2}AD^{-1/2}$, where D is a diagonal matrix of the degree sequence k, and $\mathcal{L} = D - A$ is the Laplacian matrix. In addition, as the baseline does, RF is conducted on features generated based on the largest α eigenvalues of $\hat{\mathcal{L}}$, namely, $x^\alpha \leftarrow [\lambda_1, \lambda_2, ..., \lambda_\alpha]$. For a convenient description, we refer to this process as EigRF(α).

Conf.	Baseline	2	8	32	128	512
EXP1	97.55 ± 0.53	72.36 ± 3.64	80.28 ± 2.92	84.87 ± 3.13	87.64 ± 2.77	93.54 ± 1.47
EXP2	85.84 ± 3.06	60.23 ± 4.25	64.00 ± 4.72	67.13 ± 5.21	67.96 ± 5.89	74.67 ± 3.70
EXP3	93.25 ± 1.09	69.59 ± 2.34	77.64 ± 2.44	82.74 ± 1.65	86.40 ± 1.78	92.85 ± 1.23
EXP4	90.04 ± 1.09	58.93 ± 1.80	63.27 ± 3.55	67.06 ± 2.92	72.07 ± 2.65	85.65 ± 2.22

Table 4.6: Accuracy (in percentage for both the mean and standard deviation) of EigRF(α) regarding varied feature dimensions α based on the eigenvalue of $\hat{\mathcal{L}}$, namely, $\alpha = 2, 8, ..., 512$. The baseline is the case of $x^\alpha \leftarrow [r, F]$. Results of EXP1 and EXP2 are drawn on 20 groups, on each of which 10-fold cross-validation is adopted. Instead, EXP3 and EXP4 directly conduct 10-fold cross-validation on all groups, i.e., learn a model based on networks from 18 groups and verify it on the others.

Table 4.6 reports the corresponding results with respect to the classification problem, where $\alpha = 2, 8, ..., 512$ of EigRF(α) are investigated[18]. As we can see from there, the effectiveness of EigRF(α) increases as the rise of α, i.e., the mean increases followed by a decrease of the standard deviation. But it could not surpass the baseline[19] for all cases, even when $\alpha = 512$. For EXP2, one can also easily observe a larger gap between the baseline and EigRF(512) compared to EXP1. Nevertheless, comparing the results of EXP1 and EXP3, i.e., from networks with the same degree sequence to networks with massive sequences, the baseline has a much larger drop of accuracies than the one of EigRF(α), which indicates that EigRF(α) might be more capable of handling unknown networks. Besides, different from a drop between EXP1 and EXP3, the results of EXP2 and EXP4 show us that the increase of the number of samples could alleviate the boundary problem.

Conf.	Baseline	2	8	32	128	512
EXP1	2.89 ± 0.46	10.44 ± 1.37	8.74 ± 1.13	6.89 ± 1.01	6.34 ± 0.90	5.34 ± 0.71
EXP2	5.45 ± 0.73	12.28 ± 1.53	10.88 ± 1.36	9.11 ± 1.25	9.19 ± 1.24	8.82 ± 1.26
EXP3	5.33 ± 0.51	11.11 ± 0.99	9.36 ± 0.90	7.50 ± 0.86	6.94 ± 0.73	5.83 ± 0.73
EXP4	6.32 ± 0.55	12.42 ± 0.98	10.80 ± 0.87	9.13 ± 0.88	9.00 ± 0.76	8.48 ± 0.87

Table 4.7: MAE (in percentage for both the mean and standard deviation) of EigRF(α) regarding varied feature dimensions α. The baseline is the case of $x^\alpha \leftarrow [r, F]$.

We further consider EigRF(α) on the regression problem, where the mean absolute error (MAE) is conducted during the training phase. As reported in Table 4.7, for networks generated from the same degree sequence (EXP1), the baseline has a much smaller MAE than EigRF(α). And similar to the one in the classification task, the increment of MAE from EXP1 to EXP3 with respect to the baseline is much larger than the one of EigRF(α). But comparisons between EXP2 and EXP4 show that the increase of the number of samples only slightly lowers the MAE of EigRF(α) while it actually increases the one regarding the baseline.

[18]One could tune parameters of RF to boost those accuracies but the improvement is very limited.

[19]Again, we view the baseline as a touchstone to verify which machine learning strategy would work.

To sum up, for both problems, EigRF(α) could not facilitate better results than the baseline in all cases, no matter whether networks are generated from the same degree sequence or the attributes are mixed. But the increases of α and the number of samples could help it approach the baseline (see EXP3 in Tables 4.6 and 4.7). More validations regarding EigRF(α) would be conducted later, in particular using networks from Figs. 4.4, 4.5, and 4.6 in Section 4.1.

4.4.4.2 Method based on the graph kernel

Since $\mathcal{S}$ relies on both the degree sequence and the network structure, the graph kernel might also be an alternative tool. Here, we particularly verify the Weisfeiler-Lehman subtree kernel (WLsubK(t)) [122], where t is a control variable to indicate how many iterations the kernel would run. Roughly, WLsubK(t) is a method to measure the similarity of graphs by repeatedly calculating the inner product of the color-frequency sequences[20]. We choose WLsubK(t) because other kernels are usually too time-consuming to tackle networks that we considered here [123]. Besides, to achieve classification or regression, WLsubK(t) is usually followed by a support-vector machine strategy which the default one from ref. [117] is considered here.

Conf.	Classification			Regression
	5	10	SP	5
EXP1	58.87 ± 3.36	58.80 ± 3.35	72.22 ± 6.38	12.13 ± 1.13
EXP2	50.74 ± 14.29	50.72 ± 14.31	58.95 ± 7.76	13.05 ± 1.23
EXP3	75.58 ± 1.79	76.05 ± 1.61	80.16 ± 3.46	13.29 ± 0.68

Table 4.8: Accuracy and MAE (in percentage for both the mean and standard deviation) of WLsubK(t) regarding $t = 5$ and 10. SP corresponds to the shortest path kernel [124].

The associated results are illustrated in Table 4.8, where each node is initially labeled based on their related degrees. Obviously, compared to EigRF(α) and the baseline (see Tables 4.6 and 4.7), both accuracy and MAE of WLsubK(t) are much worse, which indicates that the graph-kernel-based method might not be a good tool to tackle this type of problem.

4.4.4.3 Method based on the graph neural network

The graph neural network (GNN) is a special group of neural networks aiming to tackle irregular graph-structure data [125], which, from the perspective of network science, could be mainly classified into two categories. The first category (GNN1), such as the one in ref. [126], is trying to learn the network spectrum and handle a graph in a global view. On the contrary, the second category (GNN2) measures a network in a graph-kernel-based way, i.e., it also repeatedly integrates information from nodes' neighbors and identifies a network based on its all nodes (capture local patterns), e.g., GIN [127]. In what follows, we are going to mainly verify GNN2, particularly considering GIN as an example, since GNN1 is mainly designed to deal with node-related problems.

[20]During each iteration, nodes in a network relabel (color) themselves based on labels from their nearest neighbors. After that, one could count the number of nodes that have the same color, which gives us some color-frequency sequences. Then, the similarity could be drawn based on those sequences.

For a given network G, GIN measures it through h_G,

$$h_G = \sum_{\ell} \mathrm{MLP}_1^{(\ell)}\Big(\sum_{v\in\mathcal{N}} h_v^{(\ell)}\Big), \tag{4.8}$$

where MLP represents multilayer perceptrons, ℓ is a control variable, and h_v follows

$$h_v^{(\ell)} = \mathrm{ReLU}\Big(\mathrm{MLP}_2^{(\ell)}\Big(\sum_{u\in\Gamma(v)\cup v} h_u^{(\ell-1)}\Big)\Big), \tag{4.9}$$

in which ReLU is the activation function [128]. Besides, the blow configurations are taken for GIN: the number of hidden units for both MLP_1 and MLP_2 is fixed to 64; MLP_1 and MLP_2 are accordingly with 1 and 2 layers, and they are followed by dropouts with rates of 0 and 0.5 [129], respectively; $\ell = 5$ and h_u^0 is initialized by the degree[21], that is, $h_u^0 = k_u$; the batch size for EXP1 and EXP2 is 32, otherwise, 128 is considered; the batch normalization strategy [130] is also used for each hidden layer; and to learn the associated variables, the Adam optimizer [131] is employed with an initial learning rate of 0.01 which is meanwhile decayed with a fixed ratio of 0.5 every 50 epochs.

Conf.	Classification			Regression		
	Baseline	GIN	GIN-RK4	Baseline	GIN	GIN-RK4
EXP1	97.55 ± 0.53	96.30 ± 1.10	96.47 ± 0.89	2.89 ± 0.46	3.09 ± 0.40	3.04 ± 0.40
EXP2	85.84 ± 3.06	90.09 ± 2.43	90.47 ± 2.60	5.45 ± 0.73	11.30 ± 1.70	11.19 ± 1.42
EXP3	93.25 ± 1.09	97.71 ± 0.94	97.44 ± 0.73	5.33 ± 0.51	5.55 ± 0.86	5.57 ± 0.44

Table 4.9: Accuracy and MAE (in percentage for both the mean and standard deviation) of GIN and GIN-RK4.

Table 4.9 shows the corresponding results, where GIN-RK4 is the case that we rewrite Eq. (4.8) in a 'fourth-order Runge–Kutta method' manner[22]. Besides, for both GIN and GIN-RK4, accuracy or MAE is calculated at the epoch where the validation set performs best, and the ratio of the training set, validation set, and testing set follows $8 : 1 : 1$. As we can see from there, for the classification problem, GIN could surpass the baseline in both EXP2 and EXP3, and it is only slightly worse than the baseline in EXP1. But for the regression task, the baseline still holds the best, especially in EXP2, where GIN is much worse than the baseline. Nevertheless, GIN could obtain much better results than EigRF(α) and WLsubK(t) for almost all cases that we considered here. Besides, GIN-RK4 is usually more stable than GIN (see the associated standard deviation), and also slightly better for most cases[23]. It is worth mentioning that the mean of the test set of EXP3 is 0.1259 regarding the regression.

4.4.4.4 More validations

Now we move to networks from Figs. 4.4 (with mean of $\mathcal{S}$ 0.5346), 4.5 (0.0849) and 4.6 (0.2611), in which networks of Figs. 4.4 and 4.6(c,d) are with size $n = 10^4$. The corresponding

[21]The one-hot encoding strategy is not conducted here since we need a model which could tackle unknown networks.

[22]Roughly, the architecture of Eq. (4.8) could be viewed as 'Euler's method'.

[23]For a system like Eq. (4.1), we could construct a neural network in RK4 way to track its trajectory. In this manner, only the sin function is needed to be precisely learned. But it is always very difficult for a neural network to achieve. One could verify this by truncating RK4's results during each iteration.

Networks	Baseline	EigRF(α)					GIN	GIN-RK4
		2	8	32	128	512		
Fig. 4.4	31.45	44.09	56.02	56.60	58.10	49.36	30.67	34.94
Fig. 4.5	14.56	7.51	4.40	6.22	7.83	8.80	3.17	2.09
Fig. 4.6	18.56	16.46	12.68	12.22	13.16	15.78	12.19	13.05

Table 4.10: Mean MAE (in percentage) of networks from Figs. 4.4, 4.5, and 4.6 regarding varied methods.

Methods	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
GIN	4.27	6.83	19.18	25.67	16.82	9.09	8.01	7.69
GIN-RK4	6.09	4.76	25.19	31.65	16.16	10.56	5.10	4.86

Table 4.11: Mean MAE (in percentage) of networks from Fig. 4.6 regarding GIN and GIN-RK4.

results are shown in Tables 4.10 and 4.11, where the prediction model is acquired based on EXP3. As reported there, the baseline fails as expected. And others do not work on networks from Fig. 4.4 either, namely, all of them could not achieve the prediction of networks with a much larger size than those used to train the model[24]. For networks from Fig. 4.5, GIN-RK4 works extremely well compared to others, which indicates that it has the ability to capture the change of the network structure that both r and F fail to do. Besides, since Fig. 4.6 is based on networks of size both $n = 10^3$ and 10^4, we also show the MAE of GIN and GIN-RK4 in regard to each subfigure (Table 4.11). Again, both GIN and GIN-RK4 fail to predict $\mathcal{S}$ of a large network. Meanwhile, even for small networks, either GIN or GIN-RK4 has a large MAE, which means that we cannot cope with the optimization problem (i.e., P3) in Section 4.4.3.1) relying on them. Hence, more studies might be needed, including the improvement of existing models and the development of new models.

4.4.4.5 Effects of network robustness on $\mathcal{S}$

We further divide the training set of EXP3 into two equal parts, say D_1 and D_2, and constrain them in the same way of EXP3. That is, D_1 consists of 7,488 networks reconstructed based on 8 initial BA networks. Then, the following process is considered: 1) for D_1, each network is relabeled by the mean of 20 $F(\text{GPEP})$ (see also Section 3.6); 2) learn a model based on D_1 and the corresponding new labels; 3) fix MLP$_2$ of the learned model (here we arbitrarily choose the one at 500 epochs) and retrain MLP$_1$ based on D_2 (see Eqs. (4.8) and (4.9)), i.e., transfer learning from D_1 to D_2 [132]. We name this process as GIN-$F(b)$, where b represents the number of groups of D_2 used to retrain (recall that each group contains 936 networks). Besides, GIN-3(b) and GIN-null(b) are conducted as baselines, where GIN-3(b) is GIN but with different MLP$_1$, and GIN-null(b) is similar to GIN-$F(b)$ but with random initialization of MLP$_2$. Meanwhile, we also refer to GIN-F-3(b) as the case that both MLP$_1$ and MLP$_2$ of GIN-$F(b)$ are retained. All of them employ the same architecture of GIN except for MLP$_1$ with 3 layers.

The corresponding results are reported in Table 4.12. Compared to GIN-3(b), GIN-F-3(b) has smaller MAE for all b, and GIN-$F(b)$ holds better in two and similar in the rest, which

[24]Perhaps one reason is that the predicted networks are too large. But this is one of the final goals with respect to this problem.

Methods	1	2	4	8
GIN-3(b)	9.52 ± 1.64	7.87 ± 0.91	6.73 ± 1.48	5.51 ± 1.03
GIN-null(b)	8.97 ± 1.36	8.14 ± 1.28	8.67 ± 1.61	7.97 ± 1.34
GIN-F(b)	8.92 ± 1.87	6.85 ± 1.13	6.87 ± 1.14	5.83 ± 1.16
GIN-F-3(b)	7.76 ± 1.57	6.30 ± 0.80	6.54 ± 1.08	5.31 ± 0.77

Table 4.12: Mean MAE (in percentage) regarding varied training sets of sizes $b \in \{1, 2, 4, 8\}$.

indicates that both GIN-F(b) and GIN-F-3(b) could benefit from the pre-trained model[25]. That is, we could rely on the methodology of Chapter 3 to train a model, and then use such model to capture the transition of the network structure, which might have potential applications in such as climate networks where usually only a few networks have labels or / and the number of networks is quite limited. Note that both GIN-null(b) and GIN-F(b) only retrain MLP$_1$.

4.4.5 Summary

To sum up, we have studied a few strategies (which, to our knowledge, should be the most suitable tools here) considering the problems raised in Section 4.1, in particular the predictability of $\mathcal{S}$. Our results show that some of them could truly help us to acquire either a high accuracy or a low MAE regarding the prediction of $\mathcal{S}$. But none of them could actually further be used to investigate the corresponding optimization problem, since the precision of those tools are so far not enough to tackle it, such as only GIN and GIN-RK4 could just approach the baseline in some cases. Indeed, one might follow a similar strategy of Section 4.2 to indirectly achieve that. But that strategy would drive us away from the real optimum. Thus, this problem is still open and needs more studies.

[25]We didn't tune hyperparameters for this problem. But we truly did that when verifying GIN-F-3(8) on the empirical networks studied in ref. [127]. And GIN-F-3(8) could acquire slightly better results than GIN in most of those networks.

5

Conclusion and Outlook

5.1 Conclusion

This thesis particularly studied a number of attack and immunization strategies regarding the robustness and immunization problems, including bounded and unbounded strategies, and the evolutionary framework. Based on them, a fast scheme for the suppression of F – a parameter characterizing the network robustness – was further developed. Those three sections correspond to the identification of nodes (Chapter 3). Following that, Chapter 4 discussed whether F could be viewed as a measure to capture the network structure considering effects of network robustness on explosive synchronization. Meanwhile, ways to enhance the robustness of a network, influences of acquaintances on the containment of epidemics, and prediction of the hysteresis in explosive synchronization were also studied, which are related to the identification of networks. Specifically, the main contributions of this thesis are summarized as follows.

Identification of nodes: bounded and unbounded strategies

The order parameter of an explosive percolation usually undergoes an irreversible process. But for most regular percolation, like the one on a two-dimensional square lattice (Section 2.4.1), the order parameters of the forward transition (percolation) and backward transition (attack) are in principle equivalent to each other. That is, they are reversible, which motivates us to ask whether the rule leading to an explosive percolation could also be used to attack an existing (usually finite) network. If it does, then the rule could usually heavily delay the critical threshold, which corresponds to the solution of the robustness and immunization problems, that is, finding the minimum node set to collapse or immunize the given network. Focusing on that, the strategies – ABonS1, ABonS1q, ABonS2, ARRS, APRSs1, APRSs1q, and APRSrr – are developed, which are basically suitable for all kinds of networks. Among them, ABonS1, ABonS1q, ABonS2, APRSs1, and APRSs1q are bounded-size strategies (Section 3.4.2) but the others are unbounded (Section 3.4.4). Indeed, an unbounded-size method usually has a better performance than a bounded one. But the bounded methods have the advantage in the time consumption, e.g., ABonS1 and ABonS1q could accordingly obtain smaller F and q_c (a parameter capturing the percolation transition) within 10 seconds than almost all existing methods that we mentioned in Section 3.4 in a network with over 1.6 million nodes (they are perhaps only less effective than ABetS, which, however, is almost impossible to tackle a network of such size.). Besides, APRSs1 and APRSs1q give the option to score each node so that one can tune the score based on varied scenarios and further acquire a better result. With respect to the two unbounded methods, APRSrr is more capable of handling the model or model-like network, where there is usually a lack of community structures, short cycles, and correlations, such as the p2p-Gnutella08 network. ARRS instead has good performances in most real-world networks regarding both F and q_c, and on average, surpasses all other basic methods. More importantly, ARRS also paves the way for the evolutionary framework. Note that both bounded and unbounded methods could be easily extended for the FVS problem. And they both have better performances than ABPDS in almost all networks that we considered in this thesis (Table 3.4).

Identification of nodes: evolutionary framework for the identification of influential nodes

But the above strategies are still easy to fall into a local optimum. To overcome that, we further studied the effects of the initial sequence and also the corresponding control variables on the basic method ARRS, which guided us to our second main contribution, i.e., the evolutionary framework for investigation of influential nodes, where we have introduced selection strategies, mutation operators, and the ways to initialize and maintain a population (sequence) as well (Section 3.5). In particular, in regard to F, we firstly investigated PruOrd, which is yet effective enough to outperform ARRS. But it also faces a problem in the difficulty of the management of control parameters. To tackle that, PruGri was developed and it takes a much easier way that only two parameters are needed to be given beforehand. Usually, PruGri could obtain better results than PruOrd if we fail to give PruOrd the optimal configuration of the related parameters. However, the grid search strategy means that PruGri would be time-consuming if we do not have a parallel environment. Hence, we further had PruRan and PruRang, where PruRan chooses the slice simply following the uniform distribution, while PruRang does that by taking some probability from a variant of the Gaussian distribution. Therefore, PruRang is usually more capable of approximating PruGri. Note that one can also try other strategies of that probability. We then further studied the effects of q_c on F and also the influences of F on q_c, and found that there is some conflict between the paralleled optimization of q_c and F. This conflict guided us to have PruRangqv4 and PruRangqv5, and further those mutation operators. In addition, through these investigations, we found that the initial sequence plays an important role, especially one having advanced performance. Thus, we had another approach based on graph partitioning to provide initial sequences for our framework, i.e., AMetisS and AMetisSg. Finally, we reached the evolutionary framework, that is, Evol_F and Evol_q accordingly for F and q_c.

We then verified the effectiveness of the proposed framework considering a number of real-world networks. Regarding F, our framework accordingly has average improvements (Eq. (3.33)) of 8.89%-60.62% over HubS, AHubS, ACIS, AEIS, ARRS, AMetisS, and AMetisSg, while such improvements become 13.26%-59.21% over HubS, AHubS, ACIS, ABPDS, AEIS, ARRSq, ARRS, AMetisS, and AMetisSg for q_c, respectively.

Identification of networks: effects of network robustness on explosive synchronization

A lot of literatures have shown that the robustness of a network is influenced by a number of associated attributes, such as the density of edges (average degree), the assortativity, the clustering coefficient, and the degree distribution, etc. This inspired us to ask what effects the robustness has on a more complex phenomenon, since, on the one hand, the robustness (particularly $F(S)$) could globally capture the transition of the network structure under the corresponding attack, and on the other hand, it could also be acquired very easily, especially for, e.g., HubS. To investigate that, we took the explosive synchronization regarding the Kuramoto model as an example, and numerically studied how $F(\text{HubS})$ (view it as a global attribute) as well as its interaction with the network assortativity r influence the behavior of the explosive synchronization (Section 4.1). We found that both extreme values of F and r would suppress the explosive synchronization, especially the hysteresis area between the forward and backward transitions. In particular, for a network constructed by the BA model, there is a maximum of hysteresis area achieved by appropriate adjustments of F and r. In addition, our discussion reveals that the change trends of both the network robustness and assortativity play important roles in the explosive synchronization. In other words, different behaviours of the explosive synchronization are found in networks with similar F or/and r if the networks undergo different change processes. Thus, the problem of effects of the network structure on ES still remains partly open.

Identification of networks: ways to enhance the robustness of a network

Our fourth main contribution is the investigation of possible approaches to enhance the robustness of a given network (Section 4.2). From our first and second studies, we knew that a network would have varied reactions of robustness over different attack strategies, which indicates that the true robustness of a network is detected by the most advanced attack strategy. But such a strategy is usually too time-consuming to be a practicable criterion for the cut-add approach (an approach could modify the network structure by keeping the degree sequence fixed). To overcome that problem, we firstly studied which naive method could be an alternative of the advanced one and reached ARRSe, our basic method ARRS with an early stop trick, which could effectively check the critical point during the enhancement process based on HubS. We then further investigated such process and acquired our first method, WayEnhS, which has a better performance and similar time consumption against the one based on HubS. But WayEnhS is still hardly employed to tackle large networks, since it has to calculate $F(S)$ per t. Hence, we further studied the effects of assortativity on WayEnhS, an attribute that highly correlates with the network robustness and could be controlled locally. More importantly, such strategy not only helped us to improve the performance of WayEnhS again but also led us to the second method, WayEnhSr, which is totally local. To demonstrate the effectiveness of both WayEnhS and WayEnhSr, four real-world networks are analyzed. The results regarding attacks from over ten state-of-the-art strategies illustrate that both of them have the ability to greatly improve the robustness of a network. Nevertheless, both WayEnhS and WayEnhSr are indirect methods to increase the bound of the robustness against the most advanced attack strategy, e.g., ARRSe with $T = 10^3$ is more effective than the one with $T = 10$, since the first has a larger correlation with Evol_F. Thus, this problem is still open and more studies are needed, including itself of the most advanced attack strategy and its more reliable representative.

Identification of networks: influences of acquaintances on the containment of epidemics

In Section 4.3, rather than consider the case in Section 3.1.2, i.e., the influence of the degree distribution on the epidemic spreading, we chose to focus on existing networks and study effects of the change of the corresponding structure on the spreading patterns under immunization. In particular, we wanted to know whether a simple removal of edges would boost the effectiveness of immunization strategies, and if it does, which strategy would be more effective. To cope with these problems, we firstly investigated strategies extended from the previous part and found that the assortativity also has impacts on immunization. In tandem with the fact that a practicable method should need information as little as possible, we proposed a very simple strategy, called Prodi2, which searches for and targets edges connecting to nodes with a small degree. The results on the SIR model demonstrate that our strategy could help to save a huge amount of resources (less nodes are needed to be immunized or removed). It is worth mentioning that the developed strategy is totally local. Hence, the combination of it and AcqI (a well-known local immunization strategy) would be really useful in some cases, e.g., help to distribute vaccines if the amount is very limited.

Identification of networks: from a machine learning perspective

Last but not least, our sixth main contribution could be viewed as a proof of concept. To further tackle the problems raised in the previous sections, especially the one from Section 4.1, we turned our attention to machine learning methods. In Section 4.4, we first employed the data from Section 4.1 to set up four experimental configurations, which aim to verify the predictability of $\mathcal{S}$ (the hysteresis area of the explosive synchronization) under varied conditions. Then, we studied methods based on the spectrum of the related normalized Laplacian matrix, the graph kernel, and the graph neural network. Our results show that some of them could truly help us to acquire either a high accuracy or a low MAE regarding the prediction of $\mathcal{S}$. But none of them could actually further be used to investigate the corresponding optimization problem, since the precision of those tools are so far not enough to tackle it, such as only GIN and GIN-RK4 could just approach the baseline in some cases. Indeed, one might follow a similar strategy as in Section 4.2 to indirectly achieve that. But that strategy would drive us away from the real optimum. Note that, the considered strategies, to our knowledge, should be the most suitable tools here. Thus, as a bunch of problems that we touched in the previous sections, more studies are needed, including the improvement of existing models and the development of new models.

5.2 Outlook

Several problems have arisen during our journey to this thesis. For part of them, one could predictably have solutions, or at least some approximate solutions, following the routines that we have already done. But for others, we still do not know how to handle them so far. A few instances are as below.

Temporal networks. By now we have only studied static networks. When it comes to temporal networks, the corresponding problems would become much more difficult. For example, the human being interactions could be modeled by a network, which evolves day by day. In this case, how could we achieve the immunization of such a network with minimum resources during a pandemic? Besides, in a short term, that interaction network could be viewed as a special case where only edges evolve. For more complex situations including the change of nodes (e.g., the online social network), what strategy should be designed?

Incomplete networks. For many empirical networks, we only have partial information regarding them, e.g., the human-being-contact network. In this case, how should we design the corresponding strategy? In Section 4.3, we have shown a possible alternative. But that is only a very special example. More significant approaches should be further developed.

Predictability and controllability of F. As we showed in Section 4.1, $F(\text{HubS})$ truly plays an important role on, at least the explosive synchronization, which indicates that F could be viewed as a global attribute to capture the structure of a network. Hence, a natural question is: what about the true robustness regarding such measure? To answer this question, we however firstly need to solve a problem from Section 4.2, that is, finding a more reliable representative of the most advanced attack strategy.

Machine learning methods. As we mentioned and verified, machine learning methods might play functions on the problem from 4.1 and the methods in Sections 4.2 and 4.3. But the question is how could we achieve that? In Section 4.4, we have shown that both the precision and the learning models themselves are challenged by the considered problems. Hence, are there more suitable models? Or do we need to develop new models to cope with those problems?

References

[1] Mark Newman. *Networks*. Oxford university press, 2018.

[2] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.

[3] Peter J Menck et al. “How dead ends undermine power grid stability”. In: *Nature Communications* 5 (2014), p. 3969.

[4] Mark EJ Newman. “Spread of epidemic disease on networks”. In: *Physical review E* 66.1 (2002), p. 016128.

[5] Christian M Schneider et al. “Mitigation of malicious attacks on networks”. In: *Proceedings of the National Academy of Sciences* 108.10 (2011), pp. 3838–3841.

[6] Goylette F Chami et al. “Social network fragmentation and community health”. In: *Proceedings of the National Academy of Sciences* 114.36 (2017), E7425–E7431.

[7] Yong Zou et al. “Complex network approaches to nonlinear time series analysis”. In: *Physics Reports* 787 (2019), pp. 1–97.

[8] Dirk Brockmann and Dirk Helbing. “The hidden geometry of complex, network-driven contagion phenomena”. In: *Science* 342.6164 (2013), pp. 1337–1342.

[9] Réka Albert, Hawoong Jeong, and Albert-László Barabási. “Error and attack tolerance of complex networks”. In: *Nature* 406.6794 (2000), pp. 378–382.

[10] Duncan S Callaway et al. “Network robustness and fragility: Percolation on random graphs”. In: *Physical Review Letters* 85.25 (2000), p. 5468.

[11] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. “Efficient immunization strategies for computer networks and populations”. In: *Physical Review Letters* 91.24 (2003), p. 247901.

[12] Zhen Wang et al. “Statistical physics of vaccination”. In: *Physics Reports* 664 (2016), pp. 1–113.

[13] Jesús Gómez-Gardenes et al. “Explosive synchronization transitions in scale-free networks”. In: *Physical Review Letters* 106.12 (2011), p. 128701.

[14] Daqing Li et al. “Percolation transition in dynamical traffic network with evolving critical bottlenecks”. In: *Proceedings of the National Academy of Sciences* 112.3 (2015), pp. 669–672.

[15] Abhijeet Ghadge, Hendrik Wurtmann, and Stefan Seuring. “Managing climate change risks in global supply chains: a review and research agenda”. In: *International Journal of Production Research* 58.1 (2020), pp. 44–64.

[16] OECD. *Climate-resilient Infrastructure*. 2018. URL: `http://www.oecd.org/environment/cc/policy-perspectives-climate-resilient-infrastructure.pdf` (visited on 04/23/2020).

[17] Paul Baran. "On distributed communications networks". In: *IEEE transactions on Communications Systems* 12.1 (1964), pp. 1–9.

[18] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *Science* 286.5439 (1999), pp. 509–512.

[19] Reuven Cohen et al. "Resilience of the internet to random breakdowns". In: *Physical Review Letters* 85.21 (2000), p. 4626.

[20] Reuven Cohen et al. "Breakdown of the internet under intentional attack". In: *Physical Review Letters* 86.16 (2001), p. 3682.

[21] Christian M Schneider, Tamara Mihaljev, and Hans J Herrmann. "Inverse targeting—An effective immunization strategy". In: *EPL (Europhysics Letters)* 98.4 (2012), p. 46002.

[22] Yang Liu et al. "Immunization strategy based on the critical node in percolation transition". In: *Physics Letters A* 379.43-44 (2015), pp. 2795–2801.

[23] Flaviano Morone and Hernán A Makse. "Influence maximization in complex networks through optimal percolation". In: *Nature* 524.7563 (2015), pp. 65–68.

[24] Pau Clusella et al. "Immunization and targeted destruction of networks using explosive percolation". In: *Physical Review Letters* 117.20 (2016), p. 208301.

[25] Salomon Mugisha and Hai-Jun Zhou. "Identifying optimal targets of network attack by belief propagation". In: *Physical Review E* 94.1 (2016), p. 012305.

[26] Alfredo Braunstein et al. "Network dismantling". In: *Proceedings of the National Academy of Sciences* 113.44 (2016), pp. 12368–12373.

[27] C Drew Harvell et al. "Climate warming and disease risks for terrestrial and marine biota". In: *Science* 296.5576 (2002), pp. 2158–2162.

[28] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*. Vol. 2. Springer, 2012.

[29] In Wikipedia, the free encyclopedia. *Economic impact of the COVID-19 pandemic*. 2020. URL: `https://en.wikipedia.org/wiki/Economic_impact_of_the_COVID-19_pandemic#cite_note-2` (visited on 08/07/2020).

[30] Anthony J McMichael et al. *Climate change and human health: risks and responses*. World Health Organization, 2003.

[31] Dean T Jamison et al. "Global health 2035: a world converging within a generation". In: *The Lancet* 382.9908 (2013), pp. 1898–1955.

[32] Renee Cho. *How Climate Change Is Exacerbating the Spread of Disease*. 2014. URL: `https://blogs.ei.columbia.edu/2014/09/04/how-climate-change-is-exacerbating-the-spread-of-disease/` (visited on 05/08/2020).

[33] Matt J Keeling and Ken TD Eames. "Networks and epidemic models". In: *Journal of the Royal Society Interface* 2.4 (2005), pp. 295–307.

[34] Romualdo Pastor-Satorras et al. "Epidemic processes in complex networks". In: *Reviews of modern physics* 87.3 (2015), p. 925.

[35] Peng Ji et al. "Cluster explosive synchronization in complex networks". In: *Physical Review Letters* 110.21 (2013), p. 218701.

[36] Yong Zou et al. "Basin of attraction determines hysteresis in explosive synchronization". In: *Physical Review Letters* 112.11 (2014), p. 114102.

[37] Francisco A Rodrigues et al. "The Kuramoto model in complex networks". In: *Physics Reports* 610 (2016), pp. 1–98.

[38] Yoshiki Kuramoto. "Self-entrainment of a population of coupled non-linear oscillators". In: *International Symposium on Mathematical Problems in Theoretical Physics*. Springer. 1975, pp. 420–422.

[39] Gino Del Ferraro et al. "Finding influential nodes for integration in brain networks using optimal percolation theory". In: *Nature Communications* 9.1 (2018), p. 2274.

[40] Jun Meng et al. "Percolation framework to describe El Niño conditions". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.3 (2017), p. 035807.

[41] Béla Bollobás. *Modern graph theory*. Vol. 184. Springer Science & Business Media, 2013.

[42] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), p. 440.

[43] Phillip Bonacich. "Power and centrality: A family of measures". In: *American journal of sociology* 92.5 (1987), pp. 1170–1182.

[44] Leo Katz. "A new status index derived from sociometric analysis". In: *Psychometrika* 18.1 (1953), pp. 39–43.

[45] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine". In: (1998).

[46] Alex Bavelas. "Communication patterns in task-oriented groups". In: *The journal of the acoustical society of America* 22.6 (1950), pp. 725–730.

[47] Linton C Freeman. "A set of measures of centrality based on betweenness". In: *Sociometry* (1977), pp. 35–41.

[48] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. "Dynamical and correlation properties of the Internet". In: *Physical Review Letters* 87.25 (2001), p. 258701.

[49] Mark EJ Newman. "Assortative mixing in networks". In: *Physical Review Letters* 89.20 (2002), p. 208701.

[50] Dietrich Stauffer and Ammon Aharony. *Introduction to percolation theory*. CRC press, 2018.

[51] Ray Solomonoff and Anatol Rapoport. "Connectivity of random nets". In: *The bulletin of mathematical biophysics* 13.2 (1951), pp. 107–117. ISSN: 0007-4985.

[52] Edgar N Gilbert. "Random graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144. ISSN: 0003-4851.

[53] Paul Erdős and Alfréd Rényi. "On random graphs I." In: *Publicationes mathematicae* 6.26 (1959), pp. 290–297.

[54] Paul Erdős and Alfréd Rényi. "On the evolution of random graphs". In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1 (1960), pp. 17–60.

[55] Béla Bollobás and Bollobás Béla. *Random graphs*. 73. Cambridge university press, 2001.

[56] Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

[57] Oliver Riordan, Lutz Warnke, et al. "Achlioptas process phase transitions are continuous". In: *The Annals of Applied Probability* 22.4 (2012), pp. 1450–1464.

[58] Dimitris Achlioptas, Raissa M D'Souza, and Joel Spencer. "Explosive percolation in random networks". In: *Science* 323.5920 (2009), pp. 1453–1455.

[59] Nuno AM Araujo and Hans J Herrmann. "Explosive percolation via control of the largest cluster". In: *Physical Review Letters* 105.3 (2010), p. 035701.

[60] Raissa M D'Souza et al. "Explosive phenomena in complex networks". In: *Advances in Physics* 68.3 (2019), pp. 123–223.

[61] Jingfang Fan et al. "Universal gap scaling in percolation". In: *Nature Physics* 16.4 (2020), pp. 455–461.

[62] Tom Bohman and Alan Frieze. "Avoiding a giant component". In: *Random Structures & Algorithms* 19.1 (2001), pp. 75–85.

[63] Joel Spencer and Nicholas Wormald. "Birth control for giants". In: *Combinatorica* 27.5 (2007), pp. 587–628.

[64] Eric J Friedman and Adam S Landsberg. "Construction and analysis of random networks with explosive percolation". In: *Physical Review Letters* 103.25 (2009), p. 255701.

[65] Robert M Ziff. "Explosive growth in biased dynamic percolation on two-dimensional regular lattice networks". In: *Physical Review Letters* 103.4 (2009), p. 045701.

[66] Young Sul Cho et al. "Percolation transitions in scale-free networks under the Achlioptas process". In: *Physical Review Letters* 103.13 (2009), p. 135702.

[67] SS Manna and Arnab Chatterjee. "A new route to explosive percolation". In: *Physica A: Statistical Mechanics and its Applications* 390.2 (2011), pp. 177–182.

[68] Jan Nagler, Anna Levina, and Marc Timme. "Impact of single links in competitive percolation". In: *Nature Physics* 7.3 (2011), pp. 265–270.

[69] Rui A da Costa et al. "Explosive percolation transition is actually continuous". In: *Physical Review Letters* 105.25 (2010), p. 255701.

[70] Oliver Riordan and Lutz Warnke. "Explosive percolation is continuous". In: *Science* 333.6040 (2011), pp. 322–324.

[71] Tom Bohman, Alan Frieze, and Nicholas C Wormald. "Avoidance of a giant component in half the edge set of a random graph". In: *Random Structures & Algorithms* 25.4 (2004), pp. 432–449.

[72] Wei Chen and Raissa M D' Souza. "Explosive percolation with multiple giant components". In: *Physical Review Letters* 106.11 (2011), p. 115701.

[73] Young Sul Cho, Byungnam Kahng, and Doochul Kim. "Cluster aggregation model for discontinuous percolation transitions". In: *Physical Review E* 81.3 (2010), p. 030103.

[74] AA Moreira et al. "Hamiltonian approach for explosive percolation". In: *Physical Review E* 81.4 (2010), p. 040101.

[75] Christian Von Mering et al. "Comparative assessment of large-scale data sets of protein–protein interactions". In: *Nature* 417.6887 (2002), pp. 399–403.

[76] *Protein-protein interaction network*. URL: `http://www.linkprediction.org/index.php/link/resource/data/1` (visited on 04/23/2020).

[77] Xiao-Long Ren et al. "Generalized network dismantling". In: *Proceedings of the national academy of sciences* 116.14 (2019), pp. 6554–6559.

[78] Michael Molloy and Bruce Reed. "A critical point for random graphs with a given degree sequence". In: *Random structures & algorithms* 6.2-3 (1995), pp. 161–180.

[79] Reuven Cohen and Shlomo Havlin. *Complex networks: structure, robustness and function*. Cambridge university press, 2010.

[80] William Ogilvy Kermack and Anderson G McKendrick. "A contribution to the mathematical theory of epidemics". In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.

[81] Romualdo Pastor-Satorras and Alessandro Vespignani. "Epidemic dynamics and endemic states in complex networks". In: *Physical Review E* 63.6 (2001), p. 066117.

[82] Yamir Moreno, Romualdo Pastor-Satorras, and Alessandro Vespignani. "Epidemic outbreaks in complex heterogeneous networks". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 26.4 (2002), pp. 521–529.

[83] Wikipedia. *List of epidemics*. 2020. URL: `https://en.wikipedia.org/wiki/List_of_epidemics` (visited on 05/08/2020).

[84] Nita Madhav et al. "Pandemics: risks, impacts, and mitigation". In: *Disease Control Priorities: Improving Health and Reducing Poverty. 3rd edition*. The International Bank for Reconstruction and Development/The World Bank, 2017.

[85] European Parliament. *Economic impact of epidemics and pandemics*. 2020. URL: `https://www.europarl.europa.eu/RegData/etudes/BRIE/2020/646195/EPRS_BRI(2020)646195_EN.pdf` (visited on 05/08/2020).

[86] WHO. *Global epidemics and impact of cholera*. 2020. URL: `https://www.who.int/topics/cholera/impact/en/` (visited on 05/08/2020).

[87] Linyuan Lü et al. "Vital nodes identification in complex networks". In: *Physics Reports* 650 (2016), pp. 1–63.

[88] Ron Milo et al. "Network motifs: simple building blocks of complex networks". In: *Science* 298.5594 (2002), pp. 824–827.

[89] Asim K Dey, Yulia R Gel, and H Vincent Poor. “What network motifs tell us about resilience and reliability of complex networks”. In: *Proceedings of the National Academy of Sciences* 116.39 (2019), pp. 19368–19373.

[90] Maksim Kitsak et al. “Identification of influential spreaders in complex networks”. In: *Nature physics* 6.11 (2010), pp. 888–893.

[91] Richard M Karp. “Reducibility among combinatorial problems”. In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[92] Zvi Galil and Giuseppe F Italiano. “Data structures and algorithms for disjoint set union problems”. In: *ACM Computing Surveys (CSUR)* 23.3 (1991), pp. 319–344.

[93] Jure Leskovec et al. “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters”. In: *Internet Mathematics* 6.1 (2009), pp. 29–123.

[94] Pu Wang et al. “Understanding the spreading patterns of mobile phone viruses”. In: *Science* 324.5930 (2009), pp. 1071–1076.

[95] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. “Experimental evidence of massive-scale emotional contagion through social networks”. In: *Proceedings of the National Academy of Sciences* 111.24 (2014), pp. 8788–8790.

[96] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graphs over time: densification laws, shrinking diameters and possible explanations”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 177–187.

[97] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graph evolution: Densification and shrinking diameters”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 2.

[98] Ripeanu Matei, Adriana Iamnitchi, and P Foster. “Mapping the Gnutella network”. In: *IEEE Internet Computing* 6.1 (2002), pp. 50–57.

[99] Johannes Gehrke, Paul Ginsparg, and Jon Kleinberg. “Overview of the 2003 KDD Cup”. In: *ACM SIGKDD Explorations Newsletter* 5.2 (2003), pp. 149–151.

[100] Bryan Klimt and Yiming Yang. “Introducing the Enron Corpus.” In: *CEAS*. 2004.

[101] Eunjoon Cho, Seth A Myers, and Jure Leskovec. “Friendship and mobility: user movement in location-based social networks”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 1082–1090.

[102] Jaewon Yang and Jure Leskovec. “Defining and evaluating network communities based on ground-truth”. In: *Knowledge and Information Systems* 42.1 (2015), pp. 181–213.

[103] Zbigniew Michalewicz. *Genetic Algorithms + Data structures = Evolution Programs*. Springer, 1996.

[104] Wolfgang Banzhaf. “The “molecular” traveling salesman”. In: *Biological Cybernetics* 64.1 (1990), pp. 7–14.

[105] David B Fogel. “An evolutionary approach to the traveling salesman problem”. In: *Biological Cybernetics* 60.2 (1988), pp. 139–144.

[106] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[107] David B Fogel. “Applying evolutionary programming to selected traveling salesman problems”. In: *Cybernetics and Systems* 24.1 (1993), pp. 27–36.

[108] George Karypis and Vipin Kumar. “A fast and high quality multilevel scheme for partitioning irregular graphs”. In: *SIAM Journal on scientific Computing* 20.1 (1998), pp. 359–392.

[109] George Karypis. “METIS: Unstructured graph partitioning and sparse matrix ordering system”. In: *Technical report* (1997).

[110] Brian W Kernighan and Shen Lin. “An efficient heuristic procedure for partitioning graphs”. In: *The Bell system technical journal* 49.2 (1970), pp. 291–307.

[111] Liuhua Zhu, Liang Tian, and Daning Shi. “Criterion for the emergence of explosive synchronization transitions in networks of phase oscillators”. In: *Physical Review E* 88.4 (2013), p. 042921.

[112] Ping Li et al. “Reexamination of explosive synchronization in scale-free networks: The effect of disassortativity”. In: *Physical Review E* 87.4 (2013), p. 042803.

[113] I Sendiña-Nadal et al. “Effects of degree correlations on the explosive synchronization of scale-free networks”. In: *Physical Review E* 91.3 (2015), p. 032811.

[114] Thomas K DM Peron et al. “Effects of assortative mixing in the second-order Kuramoto model”. In: *Physical Review E* 91.5 (2015), p. 052805.

[115] Erwin Fehlberg. “Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems”. In: (1969).

[116] Alex Arenas et al. “Synchronization in complex networks”. In: *Physics Reports* 469.3 (2008), pp. 93–153.

[117] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[118] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[119] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. 92. American Mathematical Soc., 1997.

[120] Andrew Y Ng, Michael I Jordan, and Yair Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[121] Santo Fortunato. “Community detection in graphs”. In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[122] Nino Shervashidze et al. “Weisfeiler-lehman graph kernels.” In: *Journal of Machine Learning Research* 12.9 (2011).

[123] Giannis Siglidis et al. “GraKeL: A Graph Kernel Library in Python.” In: *Journal of Machine Learning Research* 21.54 (2020), pp. 1–5.

[124] Karsten M Borgwardt and Hans-Peter Kriegel. "Shortest-path kernels on graphs". In: *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE. 2005, 8–pp.

[125] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[126] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems*. 2016, pp. 3844–3852.

[127] Keyulu Xu et al. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[128] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *International Conference on Machine Learning*. 2010.

[129] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[130] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[131] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[132] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[133] Stéphane Coulomb et al. "Gene essentiality and the topology of protein interaction networks". In: *Proceedings of the Royal Society B: Biological Sciences* 272.1573 (2005), pp. 1721–1725.

[134] Sebastian Wandelt et al. "A comparative analysis of approaches to network-dismantling". In: *Scientific reports* 8.1 (2018), pp. 1–15.

[135] Filippo Radicchi and Claudio Castellano. "Fundamental difference between superblockers and superspreaders in networks". In: *Physical Review E* 95.1 (2017), p. 012318.

[136] Raj Kumar Pan et al. "Using explosive percolation in analysis of real-world networks". In: *Physical Review E* 83.4 (2011), p. 046112.

[137] Peter Sanders and Christian Schulz. "Think Locally, Act Globally: Highly Balanced Graph Partitioning". In: *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*. Vol. 7933. LNCS. Springer, 2013, pp. 164–175.

A

A.1 Complex Network Theory and Network Percolation

A.1.1 Networks

Network	Node	Edge	Examples
Social	Person	Friendship	Facebook
Co-purchasing	Product	Co-purchasing frequency	Online shop
Web	Web page	Hyperlink	World Wide Web
Airline	Airport	Airline	Global airline network
Metabolic	Protein	Metabolic interaction	Yeast protein-interaction network
Collaboration	Person	Collaboration	• Publication • Movie
Climate	Local measurement	• Location-adjoin • Correlation	ERA5
Road	• City • Interaction • Endpoint	Road	EU road network
Power grid	• Generator • Transformator • Substation	Power supply line	US power grid
Citation	• Article • Patent	Citation	Publication
Internet	• Host • Router	• Cable • Wireless data connection	Internet topology graph
Communication	• Person • Recipient	• Face-to-face talk • Contact	• Conference • Email system

Table A.1: Examples of networks from the real world.

A.1.2 Eigenvector centrality

With some rearrangement, Eq. (2.13) can be rewritten as

$$\lambda\mathcal{H} = A\mathcal{H}, \lambda \neq 0 \tag{A.1}$$

in matrix notation. Obviously, the solution should be that $\mathcal{H}$ is an eigenvector of the adjacency matrix A, and λ is the related eigenvalue. Besides, since A is a non-negative matrix and all entries in $\mathcal{H}$ should be non-negative, the largest eigenvalue λ_1 and the associated eigenvector are considered as the preferable solution (according to the *Perron-Frobenius theorem*). Thus, the main goal of the eigenvector centrality is actually to find out the eigenvector that the largest eigenvalue corresponds to. One way to approximate[1] this eigenvector is the *power method*. Let's randomly initialize $\mathcal{H}(0)$ (should be a non-zero vector) and based on it obtain the centrality at step t,

$$\mathcal{H}(1) = A\mathcal{H}(0), \mathcal{H}(2) = A\mathcal{H}(1) = A^2\mathcal{H}(0), ..., \mathcal{H}(t) = A^t\mathcal{H}(0).$$

One can also write $\mathcal{H}(0)$ as a linear combination of all eigenvectors $\mathbf{v}_i$ of A,

$$\mathcal{H}(0) = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{A.2}$$

where α_i is some constant. Thus, we have

$$\begin{aligned} \mathcal{H}(t) &= A^t \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \\ &= \sum_{i=1}^{n} \alpha_i \lambda_i^t \mathbf{v}_i \\ &= \lambda_1^t \sum_{i=1}^{n} \alpha_i (\frac{\lambda_i}{\lambda_1})^t \mathbf{v}_i, \end{aligned} \tag{A.3}$$

in which λ_i is the eigenvalue that $\mathbf{v}_i$ is associated with. Apparently, when $t \to \infty$, $\mathcal{H}(t) \approx \alpha_1 \lambda_1^t \mathbf{v}_1$, which means that $\mathcal{H}(t)$ is proportional to[2] $\mathbf{v}_1$.

To sum up, starting with a random $\mathcal{H}(0)$, the eigenvector centrality can be iteratively acquired through $\mathcal{H}(t)$ with $A\mathcal{H}(t-1)$.

A.1.3 Katz centrality

Consider Eq. (2.14) and recall that the i^{th}-row-j^{th}-column value of A^t is the number of walks with length t from node i to node j. If letting $\alpha < \frac{1}{\lambda_1}$, then one can have $\alpha^t A^t \to 0$ when $t \to \infty$. Thus, we can get

$$Z = (I - \alpha A)^{-1} - I, \tag{A.4}$$

[1]Approximation solution is enough for the eigenvector centrality because we only need to find out which nodes are relatively important than others.

[2]Again, this is enough for the eigenvector centrality.

where I represents the identity matrix and $(I - \alpha A)^{-1}$ is the inverse of matrix $I - \alpha A$. In this way, the Katz centrality can be obtained through calculating $(I - \alpha A)^{-1}$ directly.

Since we only need to identify each node, we can remove the term $-I$ in Eq.(A.4) and get

$$\mathcal{H}^{\mathrm{T}} = \mathbf{1}^{\mathrm{T}} Z = \mathbf{1}^{\mathrm{T}} (I - \alpha A)^{-1}, \tag{A.5}$$

where $\mathcal{H}^{\mathrm{T}}$ is the transpose of $\mathcal{H}$ and $\mathbf{1}$ represents a vector $(1, 1, 1, ...)$. Multiplying $(I - \alpha A)$ on both sides of Eq.(A.5) and doing some transposition and rearrangement, we have

$$\mathcal{H} = \alpha A^{\mathrm{T}} \mathcal{H} + \mathbf{1}. \tag{A.6}$$

Now we can also obtain $\mathcal{H}$ in the way like the eigenvector centrality does.

A.1.4 Percolation on random graph

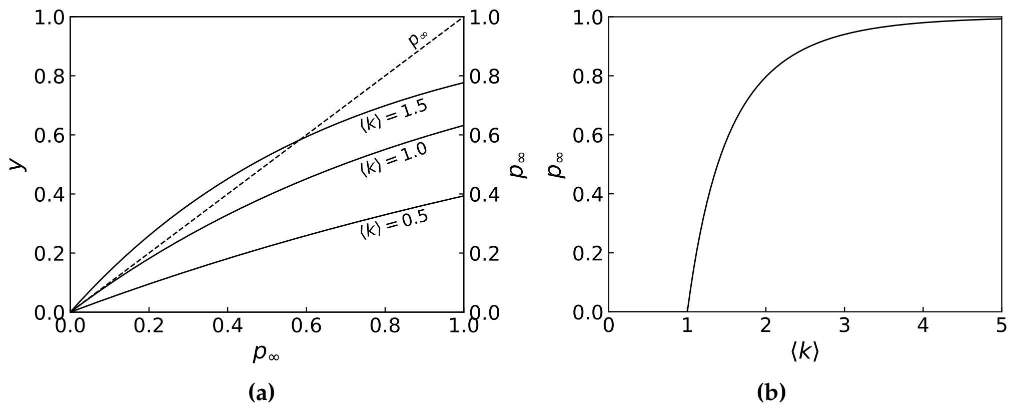


Figure A.1: Illustration regarding the giant component p_∞ of the average degree $\langle k \rangle$. (a) The solid line and the dashed line are accordingly associated with $y = 1 - \mathrm{e}^{-p_\infty \langle k \rangle}$ and p_∞. Considering three different $\langle k \rangle$, only $\langle k \rangle = 1.5$ has a non-trivial solution (they three share a same trivial solution $p_\infty = 0$). (b) The numerical solution of p_∞ versus $\langle k \rangle$.

Following ref. [1] and assuming that p_∞ is the probability that a randomly chosen node i belongs to the giant component, which also indicates that i connects to at least one node from the giant component, thus we have

$$p_\infty = 1 - (1 - p)^{(n-1)p_\infty}. \tag{A.7}$$

It is worth noting that $(1 - p)^{(n-1)p_\infty}$ is the probability that i does not connect to any nodes in the giant component. Using Eq. (2.30) to eliminate p from Eq. (A.7) we get

$$p_\infty = 1 - (1 - \frac{\langle k \rangle}{n-1})^{(n-1)p_\infty}. \tag{A.8}$$

After rearrangement, we have

$$1 - p_\infty = (1 - \frac{\langle k \rangle}{n-1})^{(n-1)p_\infty}. \tag{A.9}$$

Then,

$$
\begin{aligned}
\ln(1-p_\infty) &= (n-1)p_\infty \ln(1-\frac{\langle k\rangle}{n-1}) \\
&\approx (n-1)p_\infty(-\frac{\langle k\rangle}{n-1}) \\
&= -p_\infty\langle k\rangle.
\end{aligned} \tag{A.10}
$$

Further, we get

$$p_\infty = 1-\mathrm{e}^{-p_\infty\langle k\rangle}, \tag{A.11}$$

which does not have a simple solution. Therefore, in Fig. A.1a[3], one can find the change of $y = 1-\mathrm{e}^{-p_\infty\langle k\rangle}$ and p_∞ as a function of p_∞ considering different average degrees $\langle k\rangle$, respectively. Since Eq. (A.11) has a trivial solution $p_\infty = 0$, we can identify the phase transition through

$$
\begin{aligned}
1 &= \frac{\mathrm{d}}{\mathrm{d}p_\infty}(1-\mathrm{e}^{-p_\infty\langle k\rangle}) \\
&= \langle k\rangle\mathrm{e}^{-p_\infty\langle k\rangle}.
\end{aligned} \tag{A.12}
$$

Rearranging it, we have

$$p_\infty = \frac{\ln\langle k\rangle}{\langle k\rangle}, \tag{A.13}$$

which indicates that a non-trivial solution only exists for $\langle k\rangle > 1$ in a view of the fact $1-\mathrm{e}^{-p_\infty\langle k\rangle} < 1$ and $\frac{\mathrm{d}}{\mathrm{d}p_\infty}(\frac{\mathrm{d}y}{\mathrm{d}p_\infty}) < 0$. Besides, we also show the numerical results of p_∞ against $\langle k\rangle$ in Fig. A.1b.

[3] Note that these behaviors correspond to $n \to \infty$.

A.1.5 BFW Algorithm

Algorithm A.1: BFW

```
Input: $n, m$
Output: $G(n, m)$
1  Initialize $G(n, \mathcal{M} = \emptyset)$, $\alpha = 2$, $a = 0$, $b = 0$, $\mathcal{M}_u(t)$ and $\mathcal{M}_o(t)$
2  $t \leftarrow 0$
3  while $t < m$ do
4      $e_{ij} \leftarrow RS(\mathcal{M}_u(t), 1)$
5      $a \leftarrow$ The size of the LCC of $G(n, \mathcal{M}_o(t) \cup \{e_{ij}\})$
6      if $a \leqslant \alpha$ then
7          $t \leftarrow t + 1$
8          $\mathcal{M}_u(t) \leftarrow \mathcal{M}_u(t) \setminus \{e_{ij}\}$
9          $\mathcal{M}_o(t) \leftarrow \mathcal{M}_o(t) \cup \{e_{ij}\}$
10         $b \leftarrow b + 1$
11     else if $t/b < 1/2 + \sqrt{1/(2\alpha)}$ then
12         $\alpha \leftarrow \alpha + 1$
13     else
14         $b \leftarrow b + 1$
15 $\mathcal{M} \leftarrow \mathcal{M}_o(t)$
```

A.2 Ways to Fragment Networks

A.2.1 Molloy-Reed criterion

We here show an intuitive explanation of the Molloy-Reed criterion [78, 79]. Given a network $G(n, m)$ constructed through the configuration model, we assume that there is a giant component, and node j belongs to it. Further, supposing that a randomly chosen node i has degree k_i, then the average degree of nodes in the giant component can be expressed as a conditional expectation $\langle k_i | e_{ij} \rangle$ which means that node i has nearest neighbors in the giant component if it belongs to the giant component. Apparently, $\langle k_i | e_{ij} \rangle$ should be at least $2 - 2/n$ (imagining that the giant component is a tree). $\langle k_i | e_{ij} \rangle$ can also be obtained through

$$\langle k_i | e_{ij} \rangle = \sum_{k_i} k_i P(k_i | e_{ij}), \tag{A.14}$$

where $P(k_i | e_{ij})$ is the corresponding conditional probability that the randomly chosen node i with degree k_i connects to a node j from the giant component. Further, since

$$P(k_i | e_{ij}) = \frac{P(e_{ij} | k_i) P(k_i)}{P(e_{ij})} \tag{A.15}$$

where $P(k_i) = p_{k_i}$ (the probability that a random chosen node has degree k_i), $P(e_{ij}) = 2m/n^2$ (recall that G is constructed through the configuration model) and $P(e_{ij} | k_i) = k_i / n$ (which is the probability that a node i with degree k_i has an edge to node j), we have

$$\langle k_i | e_{ij} \rangle = \frac{\langle k^2 \rangle}{\langle k \rangle} \geqslant 2 - \frac{2}{n}, \tag{A.16}$$

in which the equal holds if the giant component is a tree. But usually, there should be a few self-loops and multi-edges (see Section 3.1.1.5) in the network G [1]. Thus, we get the Molloy-Reed criterion as Eq. (3.5).

A.2.2 Attacks on the configuration model network

Given a network G generated by the configuration model, we randomly select q fraction of its nodes and then remove them from the network, including the incident edges. Apparently, this move will change the degree of the remaining nodes and also lead to a subnetwork G', which has a different degree distribution $p'_{k'}$. Assuming that node i is held by both G and G', the probability that its degree k decreases to a specific degree k' should be

$$\tbinom{k}{k'}(1-q)^{k'} q^{k-k'},$$

i.e., each of its nearest neighbors has q probability of being removed. Since $k \geqslant k'$ and the probability that a randomly chosen node has degree k obeys p_k in G, the new degree distribution $p'_{k'}$ follows

$$p'_{k'} = \sum_{k=k'}^{\infty} p_k \tbinom{k}{k'}(1-q)^{k'} q^{k-k'}. \tag{A.17}$$

To determine whether there is a giant component in G', we need to obtain both $\langle k' \rangle$ and $\langle k'^2 \rangle$ (see Eq. (3.5)). For $\langle k' \rangle$, we can get it through

$$\langle k' \rangle = \sum_{k'=0}^{\infty} k' p'_{k'} = \sum_{k'=0}^{\infty} k' \sum_{k=k'}^{\infty} p_k \tbinom{k}{k'}(1-q)^{k'} q^{k-k'}. \tag{A.18}$$

After some algebraic calculations, we get

$$\langle k' \rangle = (1-q)\langle k \rangle. \tag{A.19}$$

In the similar way, one can obtain $\langle k'^2 \rangle$,

$$\langle k'^2 \rangle = (1-q)^2 \langle k^2 \rangle + q(1-q)\langle k \rangle. \tag{A.20}$$

G', on the other hand, represents a network constructed with part of edges from G through the configuration model. Thus, we can use the Molloy-Reed criterion (Eqs. (3.5)) to determine whether there is a giant component in network G', i.e., $\langle k'^2 \rangle / \langle k' \rangle > 2$ which leads us to get the critical threshold q_c, i.e., Eq. (3.6).

A.2.3 Attacks on ER network

Intentional attack on hubs regarding ER networks. Refer to Fig. A.2.

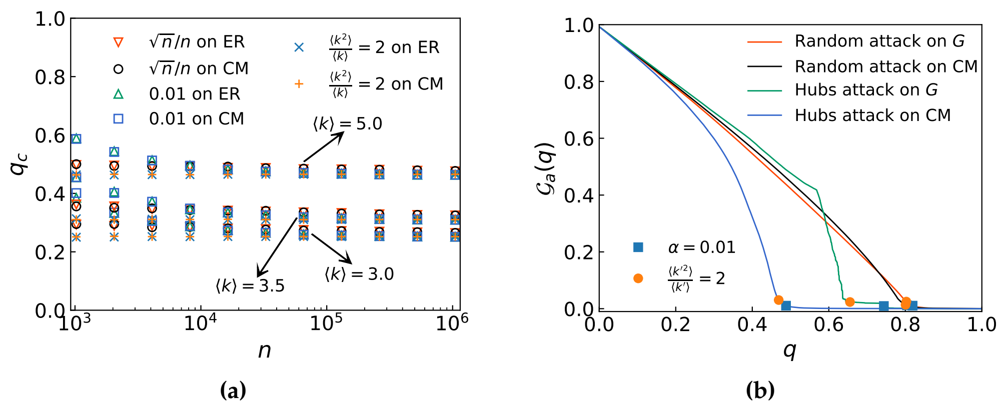


Figure A.2: (a) The critical threshold q_c as a function of network size n under the intentional attack on hubs on ER networks and the associated configuration model networks (CM). Each symbol from top to bottom, e.g., the square, corresponds to a different average degree $\langle k\rangle$. (b) The fraction of the LCC $\mathcal{G}_a(q)$ versus removed fraction q for the random attack and the intentional attack on hubs. The network G is an enhanced network considering the intentional attack.

A.2.4 Attacks on scale-free network

Letting k be continuous and $p(k) = \alpha k^{-\gamma}$, then we have $\int_{k_{\min}}^{\infty} p(k)\mathrm{d}k = 1$ and thus $\alpha = (\gamma-1)k_{\min}^{\gamma-1}$, where $k_{\min}$ is the minimum degree. Therefore,

$$p(k) = (\gamma-1)k_{\min}^{\gamma-1}k^{-\gamma}. \tag{A.21}$$

Besides, assuming that there is only one node[4] [2] which bounds the node degree in a scale-free network $G(n,m)$, say $k_{\max}$, one can get

$$\int_{k_{\max}}^{\infty} p(k)\mathrm{d}k = \frac{1}{n}, \tag{A.22}$$

and further (with Eq. (A.21))

$$k_{\max} = k_{\min} n^{1/(\gamma-1)}. \tag{A.23}$$

Hence, one can approximate $\langle k\rangle$ and $\langle k^2\rangle$ of p_k through $p(k)$ [19] (in particular $k_{\min} \ll k_{\max}$), i.e., $p_k \approx \int_k^{k+1} p(k)\mathrm{d}k$ and

$$\langle k\rangle = \sum_{k_{\min}}^{k_{\max}} kp_k \approx \int_{k_{\min}}^{k_{\max}} kp(k)\mathrm{d}k = (\gamma-1)k_{\min}^{\gamma-1}\frac{k_{\max}^{2-\gamma} - k_{\min}^{2-\gamma}}{2-\gamma}, \tag{A.24}$$

Similarly, we get

$$\langle k^2\rangle \approx (\gamma-1)k_{\min}^{\gamma-1}\frac{k_{\max}^{3-\gamma} - k_{\min}^{3-\gamma}}{3-\gamma}. \tag{A.25}$$

[4]This can be found in many real-world networks.

Thus, according to Eqs. (3.6), (A.23), (A.24) and (A.25), the threshold of scale-free networks is approximated as

$$q_c \approx \begin{cases} 1 - \dfrac{1}{ak_{\max} - 1}, & \text{if } 1 < \gamma < 2, \\ 1 - \dfrac{1}{ak_{\max}^{3-\gamma} k_{\min}^{\gamma-2} - 1}, & \text{if } 2 < \gamma < 3, \\ 1 - \dfrac{1}{ak_{\min} - 1}, & \text{if } \gamma > 3, \end{cases} \tag{A.26}$$

where $a = |(2-\gamma)/(3-\gamma)|$. Eq. (A.26) indicates that: for $\gamma > 3$, q_c is independent from the network size n; for $2 < \gamma < 3$, $k_{\max}^{3-\gamma} k_{\min}^{\gamma-2} \sim n^{(3-\gamma)/(\gamma-1)}$ means that $q_c \to 1$ if $n \to \infty$ and its convergence speed increases as γ decreases; for $1 < \gamma < 2$, q_c converges even faster as the increase of n.

For the intentional attack on hubs [20], we first assume that the new degree bound is $k'_{\max}$ after the removal of q fraction of hubs. Then, we have

$$q = \sum_{k=k'_{\max}}^{k_{\max}} p_k \approx \int_{k'_{\max}}^{\infty} p(k) \mathrm{d}k - \frac{1}{n}. \tag{A.27}$$

If $q \gg 1/n$ which always holds, we can get (with Eq. (A.21))

$$k'_{\max} = k_{\min} q^{1/(1-\gamma)}. \tag{A.28}$$

Since networks are constructed through the configuration model, edges are independent of each other. That is, each edge has the same probability of connecting to those hubs. In other words, the removal of hubs would remove each edge with the same probability which follows the fraction of removed edges

$$q_e = \frac{\int_{k'_{\max}}^{k_{\max}} k p_k \mathrm{d}k}{\langle k \rangle} \approx (\frac{k'_{\max}}{k_{\min}})^{2-\gamma}, \gamma > 2 \tag{A.29}$$

by the aid of Eq. (A.24) and ignoring $k_{\max}$ (for $\gamma > 2$) which is usually much larger than both $k_{\min}$ and $k'_{\max}$. Inserting Eq. (A.28) into Eq. (A.29), one can get

$$q_e = q^{(2-\gamma)/(1-\gamma)} \tag{A.30}$$

which also indicates that $q_e \to 1$ as $\gamma \to 2$. In other words, for networks with $\gamma \to 2$, hubs dominate almost all edges (also see Eq. (A.21)). Now considering the remaining nodes, each of their incidental edges has the same probability q_e of being removed. Therefore, the intentional attack on hubs also follows Eq. (A.17) but with $k'_{\max}$ instead of $k_{\max}$, i.e.,

$$p'_{k'} = \sum_{k=k'}^{k'_{\max}} p_k \tbinom{k}{k'} (1-q_e)^{k'} q_e^{k-k'}, \tag{A.31}$$

which indicates that the critical threshold under the intentional attack on hubs can also be obtained through Eq. (3.6) if we replace q with q_e and $k_{\max}$ with $k'_{\max}$ in Eqs. (A.24) and (A.25). Specifically, according to Eqs. (A.24), (A.25) and (A.28), one can have a new $\langle k^2 \rangle / \langle k \rangle$

as

$$\frac{\langle k^2 \rangle}{\langle k \rangle} = \frac{2-\gamma}{3-\gamma} k_{\min} \frac{q^{(3-\gamma)/(1-\gamma)} - 1}{q^{(2-\gamma)/(1-\gamma)} - 1}. \tag{A.32}$$

Inserting it and Eq. (A.30) into Eq. (3.6), i.e.,

$$q_c^{(2-\gamma)/(1-\gamma)} = 1 - \frac{1}{\frac{2-\gamma}{3-\gamma} k_{\min} \frac{q_c^{(3-\gamma)/(1-\gamma)} - 1}{q_c^{(2-\gamma)/(1-\gamma)} - 1} - 1}, \tag{A.33}$$

and after some rearrangements, Eq. (3.9) is reached.

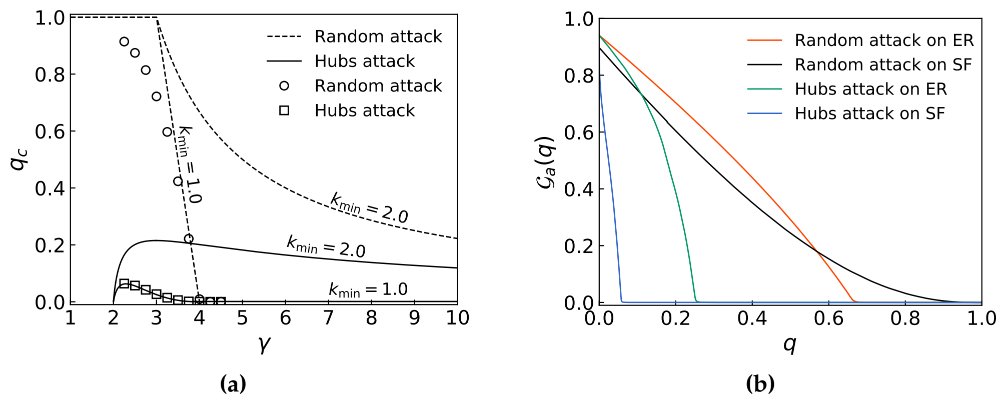


Figure A.3: (a) The critical threshold q_c as a function of γ for scale-free networks under random and intentional attacks. The dashed and solid lines are obtained through Eqs. (A.26) and (3.9), respectively. The circle and square symbols correspond to the results on networks with $n = 2^{20}$ and $\alpha = 0.01$ (see Eq. (3.8)). (b) Comparison of ER networks (with $\langle k \rangle = 3.0$ and $n = 2^{20}$) and SF networks (with $k_{\min}$, $\langle k \rangle = 3.0$, $n = 2^{20}$ and $\gamma = 2.5$).

Fig. A.3a shows the results of q_c versus γ predicted by Eqs. (A.26) and (3.9) at $n \to \infty$ regarding $k_{\min} = 1.0$ and $k_{\min} = 2.0$. To verify those results, we also give q_c of γ on networks with size $n = 2^{20}$ considering $k_{\min} = 1.0$. As we can see from there, q_c under the random and intentional attacks are getting closer as the increase of γ, which follows the conclusion from Eq. (A.23), that is, a SF network would not have hubs whose degrees are much larger than others if γ is large. Besides, it is worth mentioning that we construct a SF network in the following way: i) generate a random number a drawn from power-law distribution regarding node i; ii) let $k_i = \lfloor a + 0.5 \rfloor$ where $\lfloor \cdot \rfloor$ is the floor function; iii) get the degree sequence based on i) and ii), and then generate the SF network through the configuration model. In Fig. A.3b, we compare the response of q_c of ER networks and SF networks to the random attack and intentional attack on hubs. Apparently, for the case of Fig. A.3b, SF networks are more robust against random failure but also much more fragile to intentional attacks compared to ER networks. In addition, $\langle k \rangle = 3.0$ also follows the conclusion given by Eq. (A.24).

A.2.5 Susceptible-Infected-Recovered model

Since the population is fully mixed, the effective number that per infectious person leads to on susceptible individuals is $\eta_i a \mathrm{S}(t)$, namely, on average, $a\mathrm{S}(t)$ of a individuals are susceptible and $\eta_i a \mathrm{S}(t)$ of them would be infected by an infectious individual. Thus, to the next step $t+1$, the fraction of susceptible individuals would decrease, i.e.,

$$\frac{\mathrm{dS}(t)}{\mathrm{d}t} = -\mathrm{I}(t)\eta_i a \mathrm{S}(t) \tag{A.34}$$

which based on the fact that $\mathrm{I}(t)$ fraction of infectious individuals exist at t[5]. Eq. (A.34) accounts for the infection phase. Meanwhile, the recovery phase assumes that η_r fraction of current infected individuals would recover at step t. Therefore, the fraction of recovered individuals would increase,

$$\frac{\mathrm{dR}(t)}{\mathrm{d}t} = \eta_r \mathrm{I}(t). \tag{A.35}$$

Regarding the number of infected individuals to the next time step $t+1$, it depends on both the infection phase and the recovery phase, namely,

$$\frac{\mathrm{dI}(t)}{\mathrm{d}t} = \mathrm{I}(t)\eta_i a \mathrm{S}(t) - \eta_r \mathrm{I}(t). \tag{A.36}$$

Apparently, an epidemic can outbreak only if $\frac{\mathrm{dI}(t)}{\mathrm{d}t} > 0$, that is,

$$\mathrm{I}(t)\eta_i a \mathrm{S}(t) > \eta_r \mathrm{I}(t), \tag{A.37}$$

which gives us Eq. (3.11).

Fig. A.4 shows an example regarding the SIR model.

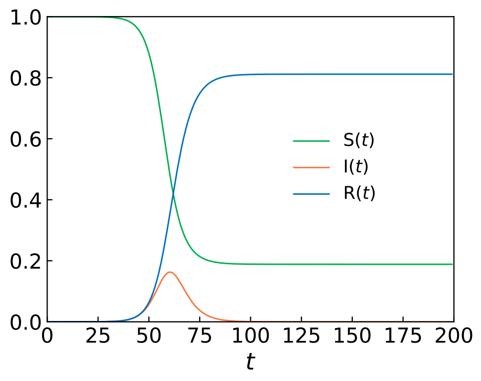


Figure A.4: An example of the SIR model with $\eta_i a = 0.5$, $\eta_r = 0.25$, $\mathrm{S}(0) = (n-1)/n$ and $\mathrm{I}(0) = 1/n$ on a population of size $n = 10^6$. That is, the basic reproductive number here is $\eta_0 = \eta_i a/\eta_r = 2$.

A.2.6 From SIR to percolation

Following ref. [2], we here show some results on networks generated through the configuration model. Specifically, for a given network $G(n, m)$, we firstly divide its nodes into different groups based on their degrees. Consider the group consisting of nodes with

[5]One can understand this in the following ways. The number of infected individuals is $n\mathrm{I}(t)$. Each of them would infect $\eta_i a\mathrm{S}(t)$ people. Then, the number of new infected individuals is $n\mathrm{I}(t)\eta_i a\mathrm{S}(t)$. Thus, the lost fraction of susceptible individuals is $\mathrm{I}(t)\eta_i a\mathrm{S}(t)$, i.e., Eq. (A.34).

degree k and let $\mathrm{I}_k(t)$ be the fraction of infected nodes in this group, i.e.,

$$\mathrm{I}_k(t) = \frac{\#\text{infected nodes have degree } k \text{ at } t}{np_k}, \tag{A.38}$$

Similarly, we have $\mathrm{S}_k(t)$ and $\mathrm{R}_k(t)$. Now focusing on the change of $\mathrm{I}_k(t)$, obviously, the second term on the right side of Eq. (A.36) has similar formation, namely, $\eta_r \mathrm{I}_k(t)$. For the first term, a can be replaced by k, η_i and η_r are same, and $\mathrm{S}(t)$ turns into $\mathrm{S}_k(t)$; however, the infected neighbors that the group has are different since they might come from other groups. Therefore, we can rewrite Eq. (A.36) as

$$\frac{\mathrm{dI}_k(t)}{\mathrm{d}t} = \Theta(t)\eta_i k \mathrm{S}_k(t) - \eta_r \mathrm{I}_k(t), \tag{A.39}$$

where $\Theta(t)$ is called density function [2, 81, 82], which characterizes the average fraction of infected nodes that a node with degree k contacts. Further, because the network generated through configuration model, the probability that a node with degree k connects to a node with degree k' is $k'p_{k'}/\langle k \rangle$ according to Eq. (2.22). For a particular infected node with k', it could at most further infect $k'-1$ other nodes because it gets the virus from at leat one of its neighbors. Thus, we have $\Theta(t)$ as

$$\Theta(t) = \sum_{k'} \frac{(k'-1)p_{k'}\mathrm{I}_{k'}(t)}{\langle k \rangle}, \tag{A.40}$$

which is independent of k. Now considering the early stage of an epidemic where $\mathrm{S}_k(t) \approx 1$, we have

$$\sum_k \frac{(k-1)p_k}{\langle k \rangle}\frac{\mathrm{dI}_k(t)}{\mathrm{d}t} = \sum_k \frac{(k-1)p_k}{\langle k \rangle}\Theta(t)\eta_i k - \sum_k \frac{(k-1)p_k}{\langle k \rangle}\eta_r \mathrm{I}_k(t) \tag{A.41}$$

by multiplying $\sum_k \frac{(k-1)p_k}{\langle k \rangle}$ on Eq. (A.39). After some rearrangements, one can get

$$\frac{\mathrm{d}\Theta(t)}{\mathrm{d}t} = (\frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle}\eta_i - \eta_r)\Theta(t), \tag{A.42}$$

where the derivative of Eq. (A.40) is used. Through solving Eq. (A.42), we have

$$\Theta(t) = ae^{bt}, \tag{A.43}$$

where a is a constant and $b = \frac{\langle k^2 \rangle - \langle k \rangle}{\langle k \rangle}\eta_i - \eta_r$. Apparently, if $b < 0$, $\Theta(t)$ will exponentially decrease to zero as t increases. Thus, an epidemic that can outbreak should satisfy $b > 0$, which gives us the epidemic threshold on networks, i.e., Eq. (3.12).

Assuming that we occupy each edge (bond) with probability $1 - e^{-\eta_i t_r}$, then we can get a subnetwork consisting of a number of connected components, which possibly has a giant component. For this case, the spread size of an epidemic is equivalent to the size of the connected component where the infected source locates at. Apparently, if there is a giant component and the infected source is randomly chosen (i.e., with high probability that the infected source is in the giant component), then with high probability, the final size of an outbreak is equal to the size of the giant component. And thus, the epidemic threshold is associated with the critical threshold of the related bond percolation. Mathematically, considering a particular node i and an associated edge e_{ij}, the probability α that i does not connect to the giant component through j with degree k_j comprises the following two parts: i) the probability that the edge e_{ij} is unoccupied, i.e., $e^{-\eta_i t_r}$; ii) the probability that node j does not belong to the giant component if e_{ij} is occupied, that is, $(1-e^{-\eta_i t_r})\alpha^{k_j-1}$ (recall that we should subtract the one connecting to i), where α^{k_j-1} is the probability that j does not connect to the giant component through its other neighbors. Thus, the probability that node i does not connect to the giant component through a particular node j with k_j is $e^{-\eta_i t_r} + (1-e^{-\eta_i t_r})\alpha^{k_j-1}$. Further, according to Eq. (2.22) and recalling that G is constructed through the configuration model, the probability P_{k_j} that j has degree k_j is independent of k_i and it is given by $P_{k_j} = \frac{k_j p_{k_j}}{\langle k \rangle}$. Now averaging over k_j, we have α which follows

$$\begin{aligned}\alpha &= \sum_k \frac{k p_k}{\langle k \rangle}(e^{-\eta_i t_r} + (1-e^{-\eta_i t_r})\alpha^{k-1}) \\ &= e^{-\eta_i t_r} + (1-e^{-\eta_i t_r})\sum_k \frac{k p_k \alpha^{k-1}}{\langle k \rangle},\end{aligned} \tag{A.44}$$

where we simply replace k_j with k. According to refs. [4, 1], the critical threshold (regarding percolation) can be obtained through the derivative of Eq. (A.44) at $\alpha = 1$. Thus, we have

$$1 - e^{-\eta_i t_r} = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}, \tag{A.45}$$

which indicates that a giant component exists or an epidemic outbreaks if the occupied probability $1 - e^{-\eta_i t_r}$ or the epidemic threshold is larger than $\frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$. After some rearrangements, one can also get [6]

$$\eta_i t_r = -\ln(1 - \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}). \tag{A.46}$$

A.2.7 Metrics to Methods

$\mathcal{G}_a(q)$ and $\langle k'^2 \rangle / \langle k' \rangle$ of q on ER networks regarding RanS and HubS. Refer to Fig. A.5.

A.2.8 General methods

Fig. A.6 gives some comparisons among those methods on one artificial network and two empirical networks. The BA network is constructed through the BA model [18] (check details from Section 3.1.1.4). The power grid network (Power) is a network containing 4941 nodes and 6594 edges, where a node represents either a generator, a transformator or a substation, and an edge is a power supply line [42]. Here we are interested in how the LCC changes if an attack occurs on some of those nodes. Although this imagined attack might never happen, climate change indeed increases the frequency and intensity of extreme

[6]If replacing t_r with the mean recovered time $1/\eta_r$ in Eq. (A.46), one can get η_i/η_j which is actually different from the one in Eq. (3.12), even though many literatures view Eqs. (3.12) and (A.45) as the same thing, such as [2, 12]. The reason, as mentioned in ref. [1], is that Eqs. (3.12) and (A.45) are derived based on different assumption. In this thesis, we mainly consider the epidemic threshold given by Eq. (3.12) but sometimes we use Eq. (A.45) when referring to percolation.

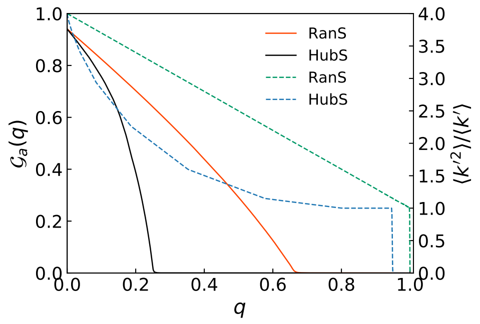


Figure A.5: $\mathcal{G}_a(q)$ and $\langle k'^2 \rangle / \langle k' \rangle$ versus q on ER networks with $\langle k' \rangle = 3.0$ and $n = 2^{20}$. The solid and dashed lines are associated with $\mathcal{G}_a(q)$ and $\langle k'^2 \rangle / \langle k' \rangle$, respectively. RanS is the random attack strategy, and HubS is the intentional attack on hubs. On the one hand, the removal of nodes would isolate a network, and an epidemic could only outbreak at most as the size of the LCC if the infectious source is unique. On the other hand, the removal also increases the epidemic threshold, i.e., decreases $\langle k'^2 \rangle / \langle k' \rangle$. Apparently, for both cases, HubS is much more efficient than RanS.

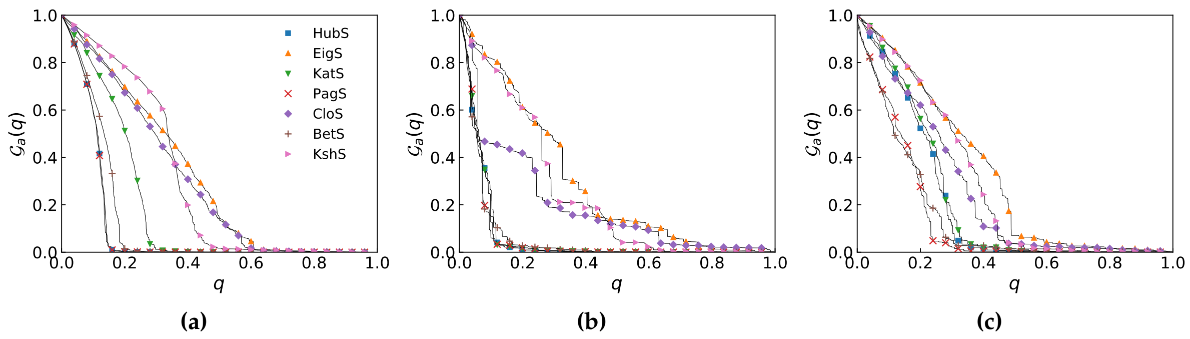


Figure A.6: Performance of HubS, EigS, KatS, PagS, CloS, BetS, and KshS regarding $\mathcal{G}_a(q)$ of q on (a) a BA network with $n = 10^4$ and $\langle k \rangle = 4.0$, (b) a power grid network of the western states of US with $n = 4941$, $\langle k \rangle = 2.7$, and (c) a protein-protein interaction network in yeast with $n = 2375$ and $\langle k \rangle = 9.8$. EigS is with $\alpha = 0.1$. KatS is with $\alpha = 0.04$ for the BA and the power grid networks, and $\alpha = 0.01$ for the yeast network (recall that α of EigS should be smaller than the reciprocal of the largest eigenvalue of the adjacent matrix to ensure convergence). PagS is with $\alpha = 0.85$. All symbols are sampled over the same interval of q. Thus one can also compare $\mathcal{G}_a(q)$ for a specific q through vertically considering those symbols.

events that possibly disable a few of the lines or generators simultaneously. The other real-world network is the protein-protein interaction network in yeast, in which nodes represent proteins, and the metabolic interaction among them is captured by edges [75, 76]. Research has shown that proteins have more interactions with others are more important for the yeast to survive [133], which is directly associated with one of our potential goals to design drugs to kill unwanted bacteria [2].

Apparently, for all those three networks, PagS performs better than others, that is, for almost all q, PagS has smaller $\mathcal{G}_a(q)$ compared to other methods. In particular, KatS is more effective than EigS but less than PagS, which exactly follows those explanations in Section 2.2. For HubS, it is comparable to PagS in the BA network and the power grid network but worse than both PagS and BetS in the yeast network. In real-world networks, there are always some critical nodes that connect two or several communities. Though those nodes sometimes have a small degree, they are very important to keep or block the transition of information among communities. For this case, BetS might be a good choice. BetS usually has better performance in real-world networks than that in model networks where a community structure rarely exists. Among those strategies, CloS and KshS account for the worst performance. That may be because both of them try to find those influential nodes which could maximize the spread. Therefore, one should notice that there are some differences between finding a group of nodes to maximize a spread and finding a group of nodes to prevent an outbreak effectively. In addition, comparing PagS on those three networks, the density of edges, i.e., $\langle k \rangle$, plays a very important role regarding the network robustness. Moreover, regarding the critical threshold q_c, PagS only needs to remove 1583 nodes to disintegrate the BA network. However, KatS needs 6604 to achieve the same goal. Further, in the same network, PagS also has much smaller F than KatS, 0.099 compared to 0.318.

A.2.9 Heuristic methods

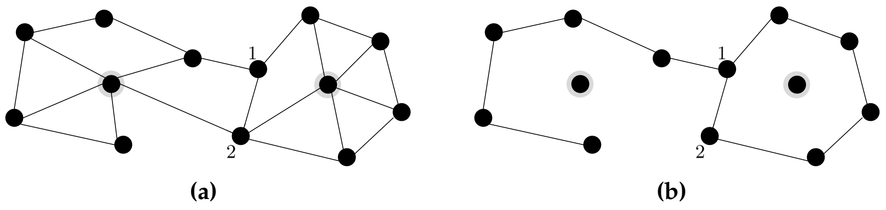


Figure A.7: An example of AHubS. (a) Apparently, node 2 has larger degree than node 1, i.e., $k_2 = 4$ compared to $k_1 = 3$. (b) After the removal of two hubs under shadow, node 1 emerges as a new hub in the remaining network.

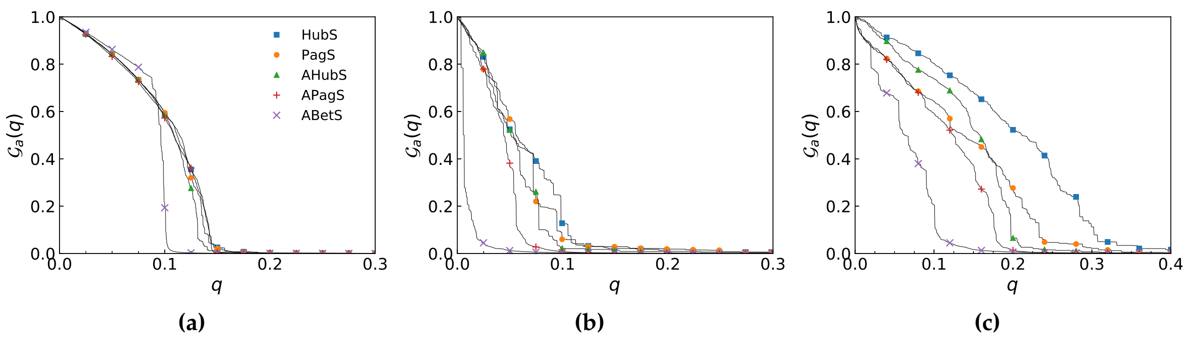


Figure A.8: Performance of HubS, PagS, AHubS, APagS, and ABetS regarding $\mathcal{G}_a(q)$ of q on the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network. AHubS, APagS and ABetS repeatedly identify and remove the node with highest score from the corresponding remaining network.

Fig. A.8 shows the performance of AHubS, APagS, and ABetS compared to HubS and PagS. As we can see from Fig. A.8a, even though the adaptive process is conducted, AHubS is only slightly better than HubS. The reason is that a network generated through the BA model theoretically has an absence of degree correlation (see Section 2.3.1), which means that the removal of hubs would not influence other hubs. In real-world networks, however, AHubS always has much better results than HubS. PagS and APagS share a similar pattern. To some extent, PagS indirectly relies on the degree from both a node itself and others[7].

[7]PagS (see also Section 2.2.4) converges to the leading eigenvector of the related matrix which is indirectly associated with the node degree.

Again, in the BA network where there is the absence of degree correlation, PagS would be conquered by the degree of node itself (see also Fig. A.6a).

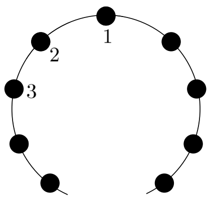


Figure A.9: An example with respect to BetS and ABetS. Both BetS and ABetS would firstly remove node 1. But after that, BetS might choose node 2 while ABetS choose node 3 instead. Apparently, the combinational configuration of 1 and 3 is better than the one of 1 and 2 to disintegrate this network.

The most surprising results are from ABetS. From Fig. A.6, we know that PagS actually has better performance than BetS. But, for the adaptive case, on the contrary, ABetS is much more effective than APagS. More specifically, for example, ABetS collapses the BA network with only 1076 nodes compared to APagS with 1459, i.e., $(1459-1076)/1459 = 26.25\%$ saving of nodes. From Section 2.2.6, we know that BetS aims at those nodes that a lot of shortest paths rely on. Though BetS usually works well on varied networks (see Fig. A.6), it faces a problem that a large-score node would have neighbors with a large score when coming to collapsing a network (see Fig. A.9). One can also imagine a special case on an unbalanced tree network where ABetS would be much more capable than BetS. Indeed, ABetS could obtain smaller q_c and F than many other methods [134]. But it suffers a problem from the time consumption. Therefore usually, ABetS is not suitable to tackle large networks. Thus, in what follows, we employ ABetS as a criterion to verify recently developed methods on small networks but would not consider it on large networks.

A.2.10 Decycling-based methods

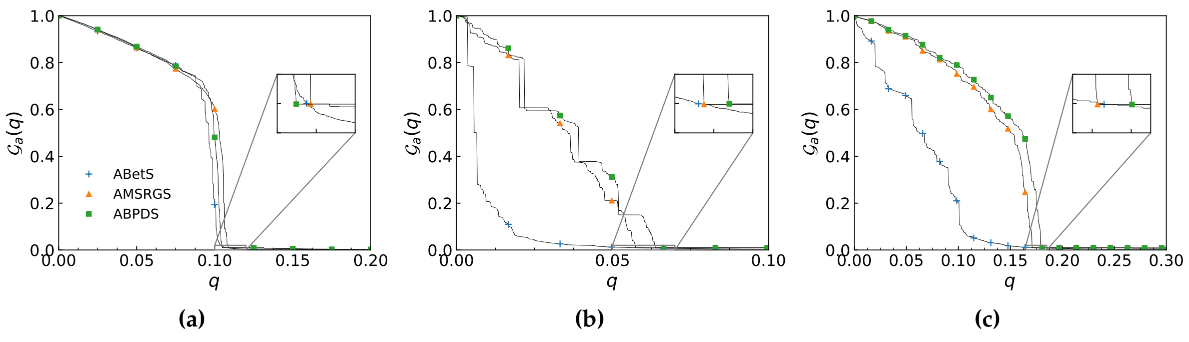


Figure A.10: Performance of ABetS, AMSRGS, and ABPDS regarding $\mathcal{G}_a(q)$ of q on the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network. The inserted panel shows the critical threshold q_c marked by correspond symbols. AMSRGS is conducted with $\alpha_1 = 3/n$ and $\alpha = 0.01$ while ABPDS is with $\alpha = 0.01$ and $x = 12$ of Eq. (4) in ref. [25].

Fig. A.10 shows the results of both ABPDS and AMSRGS compared to ABetS on the BA network, the power grid network, and the yeast network, respectively. Since ABPDS and AMSRGS are only designed for the critical threshold q_c, we here mainly consider the insert in each figure. Obviously, ABetS and AMSRGS have similar performance on those three networks, and ABetS is better on two of them. ABPDS, instead, surpasses both ABetS and AMSRGS on the BA network but is worse than them on the two real-world networks. Besides, regarding F, both ABPDS and AMSRGS are much less effective than ABetS, especially in real-world networks. To sum up, on average, ABetS still outperforms both AMSRGS and ABPDS that are comparable with each other. But again, ABetS is too time-consuming for large networks. In addition, since ABPDS is usually much faster than AMSRGS to get a result (check details from refs. [25, 26]), we will then mainly verify the proposed methods through the comparison with ABPDS instead of AMSRGS.

A.2.11 Collective influence approach

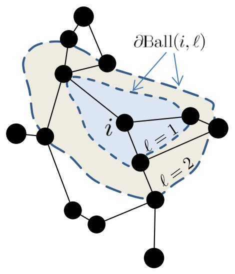


Figure A.11: An example of the collective influence strength. Considering node i and the case of $\ell = 1$, $\partial\text{Ball}(i, 1)$ is a node set comprising of the nearest neighbors of i, i.e., the three nodes on the inner dashed curve. Similarly, $\partial\text{Ball}(i, 2)$ contains 5 nodes located on the outer dashed curve.

Fig. A.11 shows an instance regarding how to calculate the collective influence strength.

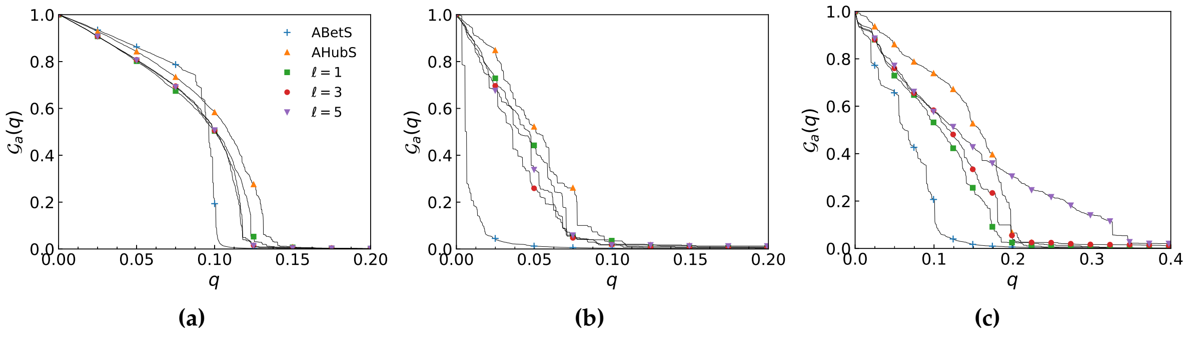


Figure A.12: Performance of ACIS with $\ell = 1$, $\ell = 2$ and $\ell = 3$ compared to ABetS regarding $\mathcal{G}_a(q)$ of q on the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network.

Indeed, for a tree, one can easily see the connection between the CI strength and the root of a balanced tree, i.e., a node i with large $\text{CI}_\ell(i)$ is more likely to be the root of a balanced tree. And from ABetS, AMSRGS, and ABPDS, we know that the removal of the root is fatal to a tree. However, this also indicates that ACIS could not find real optimization, or at least does not have the ability to outperform ABetS. To verify this, Fig. A.12 gives results of $\mathcal{G}_a(q)$ as a function of q respecting ACIS with different ℓ compared to ABetS. As it illustrates, even though ACIS outperforms AHubS that is mainly considered by ref. [23], ABetS is much more effective than ACIS, particularly in real-world networks. Besides, as shown in Fig. A.12c, the performance of ACIS might become worse as ℓ increases. Moreover, it is also worth mentioning that ref. [23] views the immunization problem and the influence maximization problem as the same, but they are an actually different problem. One can find more comparisons in ref. [135].

A.2.12 Percolation-based methods

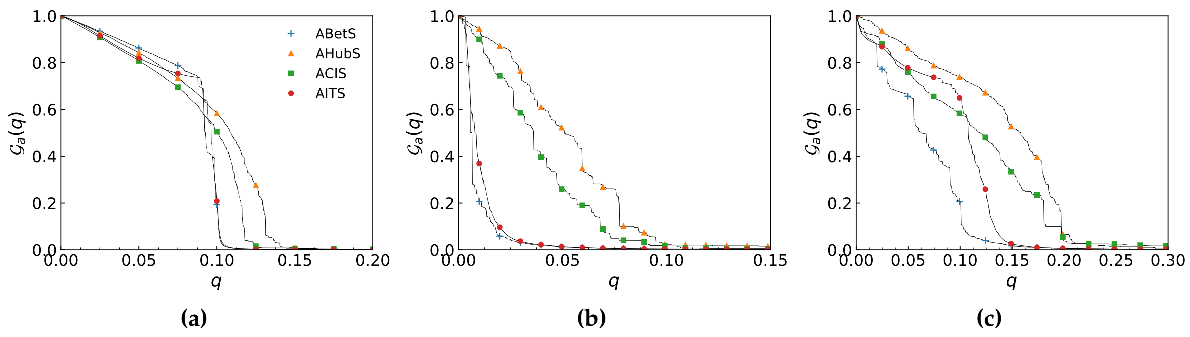


Figure A.13: Performance of AITS compared to ABetS, AHubS, and ACIS (with $\ell = 3$) regarding $\mathcal{G}_a(q)$ of q on the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network. Each AITS is an average of 20 independent implementations.

Fig. A.13 shows compared results among AITS, ABetS, AHubS, and ACIS under $\ell = 3$. For all three networks, AITS and ABetS can obtain similar critical thresholds that are much smaller than those through AHubS and ACIS. In the BA network, AITS even has a smaller F than ABetS. However, for real-world networks, this advantage disappears, especially in the yeast network, where the ratio of F of ABetS and AITS is 0.68.

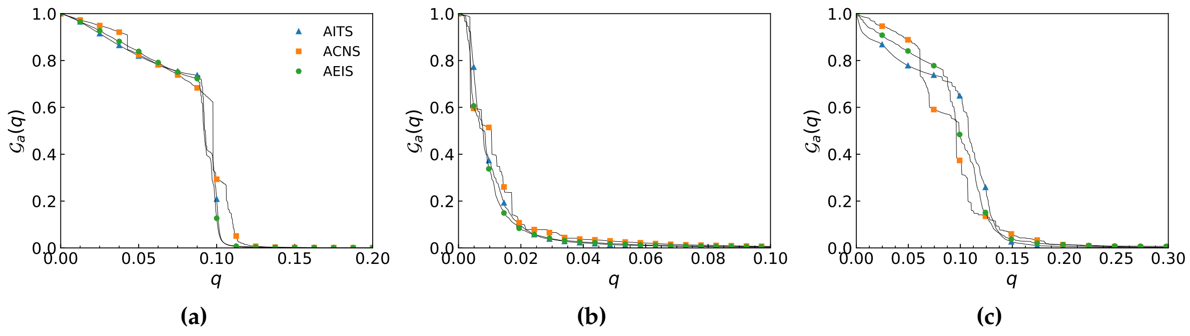


Figure A.14: Comparisons among AITS, ACNS, and AEIS regarding $\mathcal{G}_a(q)$ of q on the same networks as Fig. A.6, i.e., (a) the BA network, (b) the power grid network, and (c) the yeast network. Following the suggestions from ref. [24], here n_s and K of AEIS are fixed to 2000 and 6, respectively. All of them are obtained by averaging the results of 20 independent implementations.

Fig. A.14 gives the performance of ACNS and AEIS compared to AITS. Obviously, they have similar performance in the three considered networks, which means that both ACNS and AEIS would be less effective than ABetS in real-world networks. In addition, ACNS is slightly worse than AITS and AEIS in the BA network and the power grid network but better in the yeast network. Besides, ACNS could always obtain results faster than AITS since it only needs to remove nodes until the remaining network only contains isolated nodes.

A.2.13 Bounded and Unbounded Strategies

Union-Find Algorithm. Refer to Algorithm A.2. Besides, Fig. A.15 shows an example, where the starting node of an arrow can be viewed as node i, and the blue node is the corresponding root $a[i]$.

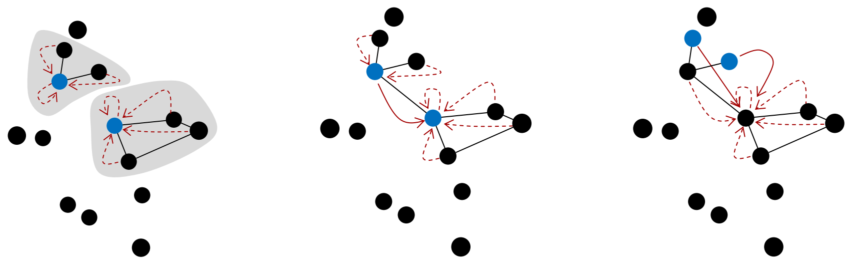


Figure A.15: An example of the Union-Find algorithm. (a) Considering the nodes in the two components (marked by shadows) at time t, they point to the two roots (in blue color), respectively. (b) Union: assuming that the two components in (a) merge together by the aid of an edge between the two roots, then the root in the small component points to the one in the large component (see the solid arrow). Note that the remaining nodes in the smaller component still point to the old root. (c) Find: those remaining nodes (colored blue) follow their old root to find and change to the new root.

Algorithm A.2: Union-Find

```
Function Root($i, a$):
    while $i \neq a[i]$ do
        $a[i] \leftarrow a[a[i]]$
        $i \leftarrow a[i]$
    return $i$
Function Find($i, j, a$):
    return Root($i, a$) $=$ Root($j, a$)
Function Union($i, j, a, c$):
    $u \leftarrow$ Root($i, a$)
    $v \leftarrow$ Root($j, a$)
    if $|c(u)| < |c(v)|$ then
        $a[u] \leftarrow v$
        $c(v) \leftarrow c(v) \cup c(u)$
        return $|c(v)|$
    else
        $a[v] \leftarrow u$
        $c(u) \leftarrow c(u) \cup c(v)$
        return $|c(u)|$
```

Bounded-size strategies. Performance of the bounded-size approach regarding Eq. (3.23) is shown in Fig. A.16.

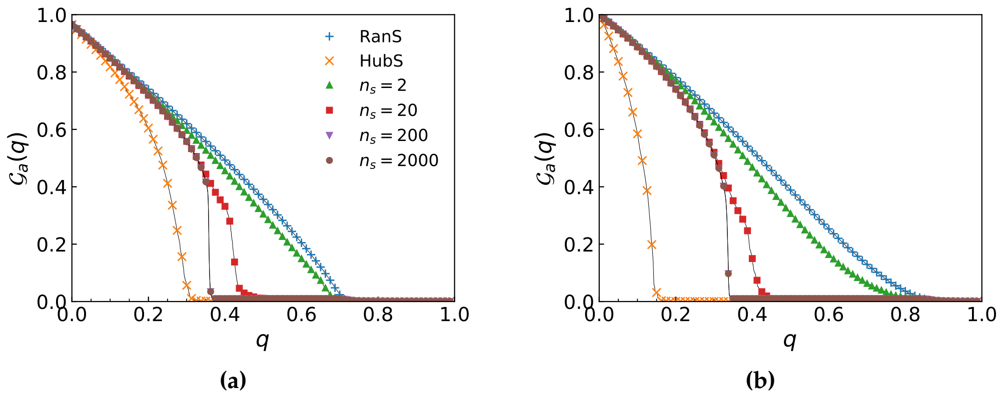


Figure A.16: Performance of basic bounded-size rule with $\alpha = 0.01 \times n$ on (a) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$ and (b) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$.

Performance of ABonS1. As we can see from Fig. A.17, apparently, ABonS1 can get much smaller critical threshold q_c than AHubS, and it is even better than ACIS. Here it is also worth mentioning that, for example on the BA network regarding ACIS with $\ell = 4$ and ABonS1 with $b = 1.2$, ABonS1 can obtain the corresponding result within 1 second while ACIS needs over 10 minutes in the same computational environment.

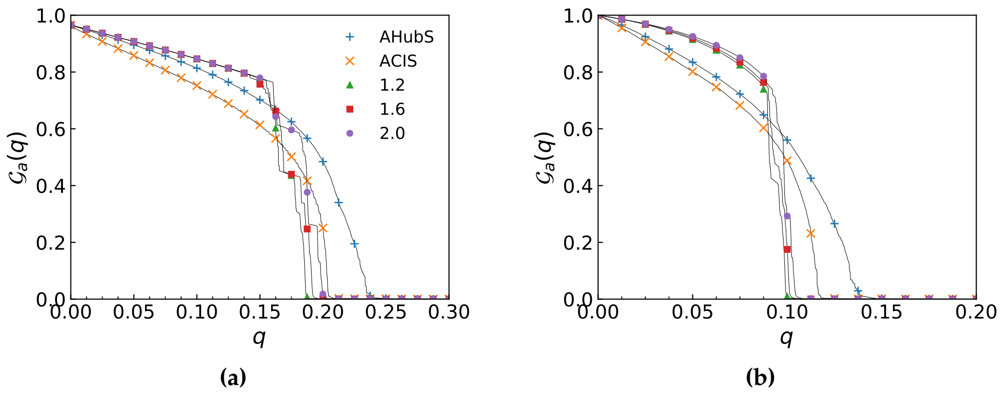


Figure A.17: Performance of ABonS1 (with $b = 1.2, 1.6$ and 2.0) compared to AHubS and ACIS (with $\ell = 4$) on (a) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$ and (b) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$.

Performance of ABonS2. Though the results in Fig. A.18 show ABonS2 is comparable to ABonS1, one can truly tune b_c and b_o to get better results, especially in real-world networks. Note that ABonS1 and ABonS2 surpass most existing methods and have comparable results to the others except for ABetS.

Performance of ASumRS and AProRS. Fig. A.19 shows their performance validated by ABonS1. Apparently, both ASumRS and AProRS are less effective than ABonS1, and ASumRS is better than AProRS, which is different from the one in Fig. 2.5. Specifically, for example, ASumRS converges very fast as the increase of n_s at the early stage and becomes stable around $n_s = 2000$. Actually, the results with $n_s = 20000$ are slightly worse than those with $n_s = 2000$ in some cases, which indicates the optimum is not obtained at $n_s = n$, in contrast to the conclusions[8] given in refs. [136, 24].

[8]Different strategies share the similar properties.

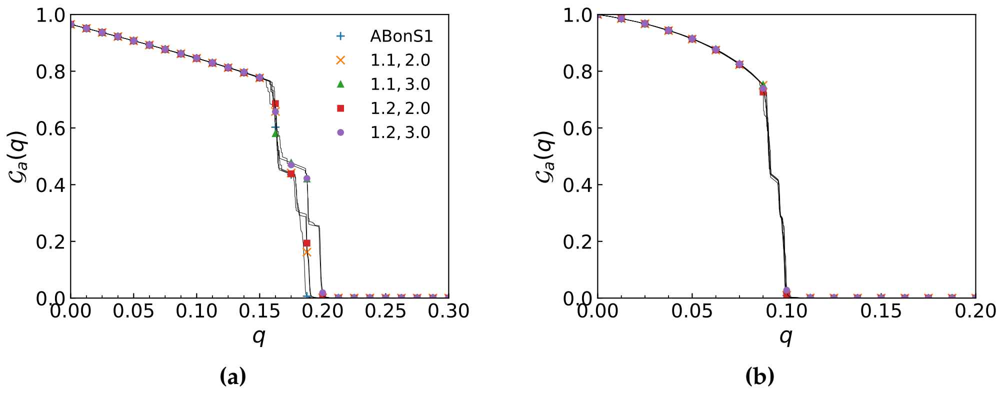


Figure A.18: Performance of ABonS2 (with $b_c = 1.1$ and $b_o = 2.0$, and so forth) compared to ABonS1 (with $b = 1.2$) on (a) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$ and (b) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$.

Performance of APRSs1 and APRSrr. Results in Fig. A.20 demonstrate that APRSs1 is slightly more effective than ABonS1 under same d with respect to the critical threshold. Note that one can tune $n_s(0)$ and $r_u(0)$ to arm APRSrr to get better results than APRSs1, e.g., set $n_s(0)$ and $r_u(0)$ large enough.

A.2.14 Evolutionary Framework for the Identification of Influential Nodes

Comparisons of q_c between ARRS and ARRSq. Refer to Fig. A.21.

An example of the accepted probability A_p against $a(T_p)$. Refer to Fig. A.22.

Performance of mutation operators regarding q_c. Fig. A.23 shows the corresponding comparisons among ARRSq, PruGriq, PruGriqv4, PruGriqv4m (PruGriqv4 with mutation operators) and PruGriqv5m (PruGriqv5 with mutation operators), where the probabilities of the local mutation and the global mutation are 0.1 and 0.3, respectively. Note that all those results are based on initial sequences drawn from HubS. Compared to ARRSq, PruGriq can find smaller q_c in 10 out of 18 networks. PruGriqv4 further improves PruGriq and it surpasses PruGriq in all networks (see also Eq. (3.41)). We then add mutation operators into PruGriqv4 (PruGriqv4m). As we can see from PruGriqv4 vs. PruGriqv4m, PruGriqv4m only has negative $D(q_c)$ in 5 networks. For PruGriqv5m, it has better performance in 13 networks than PruGriqv4m, in 15 than PruGriqv4, in 18 than PruGriq and in 16 than ARRSq. It is worth mentioning that PruGriqv5m truly outperforms PruGriqv4m on average, but one should know that it is possible of PruGriqv4m to have smaller critical threshold than PruGriqv5m if they are initialized based on other methods instead of HubS (see Fig. 3.26).

A.2.15 Fast Scheme for the Suppression of F

The computing time of GPEP is illustrated in Fig. A.24 and the corresponding results of F are reported in Table A.2. Meanwhile, performance of GPEP regarding varied imbalances and $\hat{\tau}$ is presented in Tables A.3 and A.4, respectively.

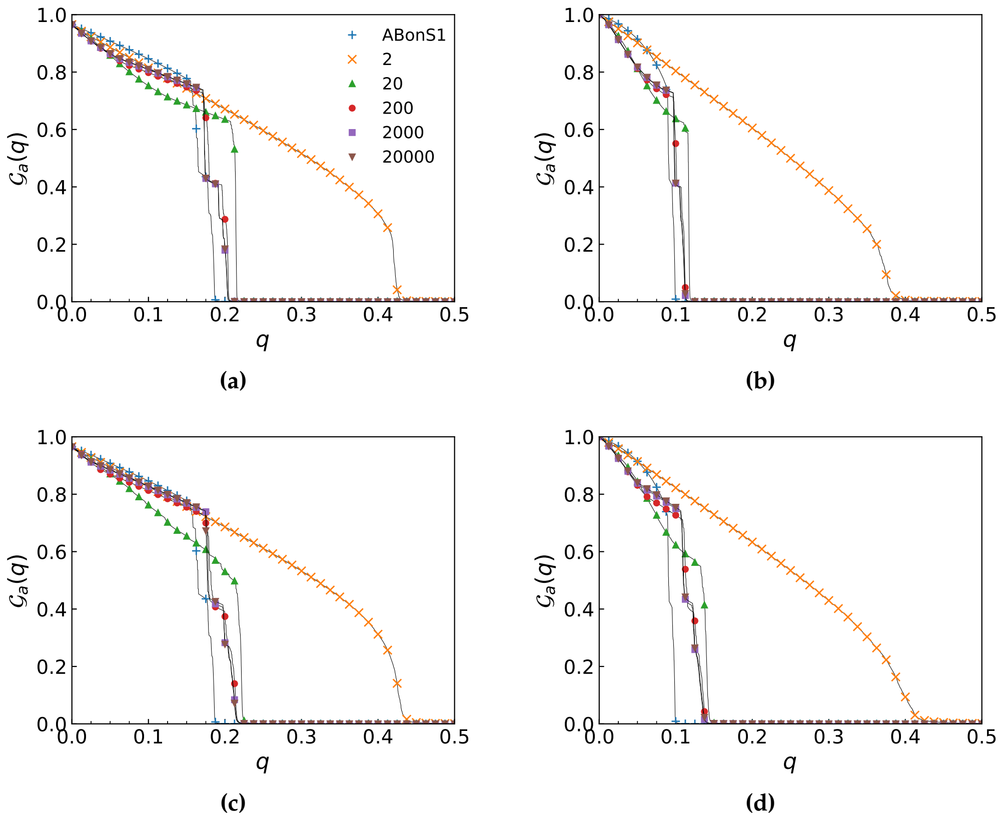


Figure A.19: Performance of ASumRS ((a) and (b)) and AProRS ((a) and (b)) compared to ABonS1 (with $b = 1.2$) on (a) and (c) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$, and (b) and (d) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$. Here $n_s = 2, 20, 200, 2000$ and 20000 are considered.

Networks	AMetisS	AMetisSg	AMetisSe	AMetisSge	GPEPK	GPEP	Evol$_F$(1)	Evol$_F$(2)	Evol$_F$(3)	Evol$_F$(4)
Power	0.0076	0.0075	0.0099‡†	0.0098‡†	0.0071	0.0074	0.0070	**0.0069**	0.0069	0.0069
CA-GrQc	0.0346‡	0.0304	0.0356‡	0.0312	0.0293	0.0300	**0.0275**	0.0283	0.0284	0.0288
p2p-Gnutella08	0.2007‡†	0.1486	0.2011‡†	0.1464	0.1454	0.1454	0.1327	0.1326	**0.1326**	0.1333
as-733	0.0109‡	0.0101†	0.0111‡	0.0105‡	0.0094	0.0094	0.0085	**0.0084**	0.0085	0.0087
Scottish	0.0281†	0.0270	0.0288†	0.0278	0.0250	0.0266	0.0226	**0.0225**	0.0225	0.0226
CA-AstroPh	0.1476‡	0.1224	0.1490‡	0.1196	0.1412‡	0.1201	0.1130	**0.1104**	0.1109	0.1166
CA-CondMat	0.0659	0.0639	0.0669†	0.0639	0.0658	0.0627	0.0599	**0.0597**	0.0600	0.0610
hep-th	0.1539	0.1524	0.1556	0.1540	0.1790‡	0.1515	0.1836‡	0.1516	**0.1510**	0.1511
Cit-HepPh	0.1380	0.1372	0.1395	0.1387	0.1370	0.1360	0.2011‡‡	0.1367	0.1363	**0.1359**
Email-Enron	0.0242†	0.0226	0.0248‡	0.0228	0.0225	0.0219	0.0206	0.0198	0.0198	**0.0198**
p2p-Gnutella31	0.1169‡	0.1101†	0.1172‡	0.1048	0.1048	0.1044	**0.0959**	0.0961	0.0964	0.1006
loc-Gowalla	0.0460	0.0446	0.0470†	0.0452	0.0629‡†	0.0444	0.0530‡	**0.0416**	0.0417	0.0417
Email-EuAll	0.0012‡	0.0011	0.0013‡†	0.0012‡	0.0011	0.0011	**0.0008**	0.0008	0.0008	0.0008
com-Amazon	0.0250	0.0247	0.0267†	0.0263†	0.0352‡†	0.0248	0.0352‡†	0.0247	**0.0244**	0.0247
web-Google	0.0103	0.0101	0.0125‡†	0.0122‡	0.0239‡‡	0.0100	0.0178‡‡	**0.0095**	0.0095	0.0096
PAroad	0.0006	0.0006	0.0031‡‡	0.0031‡‡	0.0006	0.0006	0.0010‡‡	0.0006	**0.0006**	0.0006
Txroad	0.0003	0.0003	0.0028‡‡	0.0028‡‡	0.0003	0.0003	0.0005‡†	**0.0003**	0.0003	0.0003
as-Skitter	0.0138†	0.0135†	0.0162‡†	0.0151‡	0.0250‡‡	0.0126	0.0180‡†	**0.0113**	0.0114	0.0117
Ave. imp.	6.65%	0.87%	21.50%	15.27%	10.53%		6.68%	−9.25%	−9.16%	−7.01%

Table A.2: Results of F on the 18 real-world networks regarding GPEP. Evol$_F$(1) is Evol$_F$(HubS), Evol$_F$(2) is Evol$_F$(AMetisS), Evol$_F$(3) is Evol$_F$(AMetisSg), Evol$_F$(4) is Evol$_F$(GPEP).

A.3 Functions of Order Parameter as Measure

A.3.1 Influences of Acquaintances on the Containment of Epidemics

***F* of m_r/m regarding varied removal criteria.** Refer to A.25.

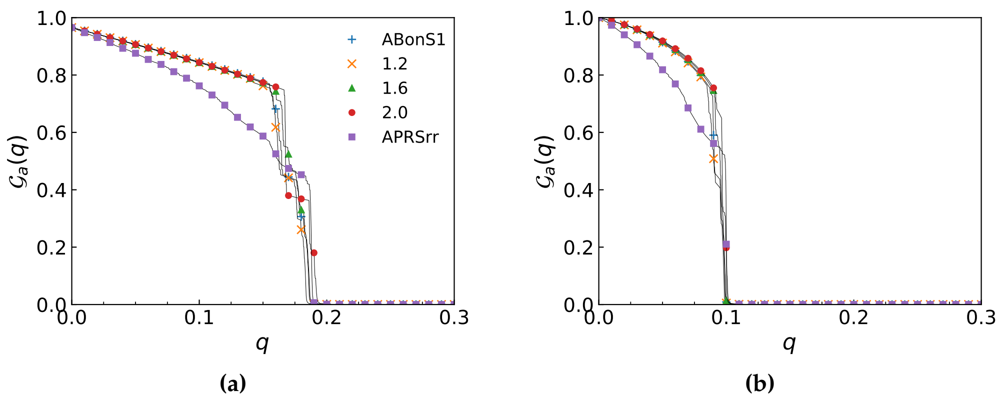


Figure A.20: Performance of APRSs1 (with $b = 1.2$, 1.6 and 2.0) and APRSrr (with same configuration as ARRS in Fig. 3.10) compared to ABonS1 (with $b = 1.2$) on (a) an ER network with $n = 10^5$ and $\langle k \rangle = 3.5$ and (b) a BA network with $n = 10^5$ and $\langle k \rangle = 4.0$.

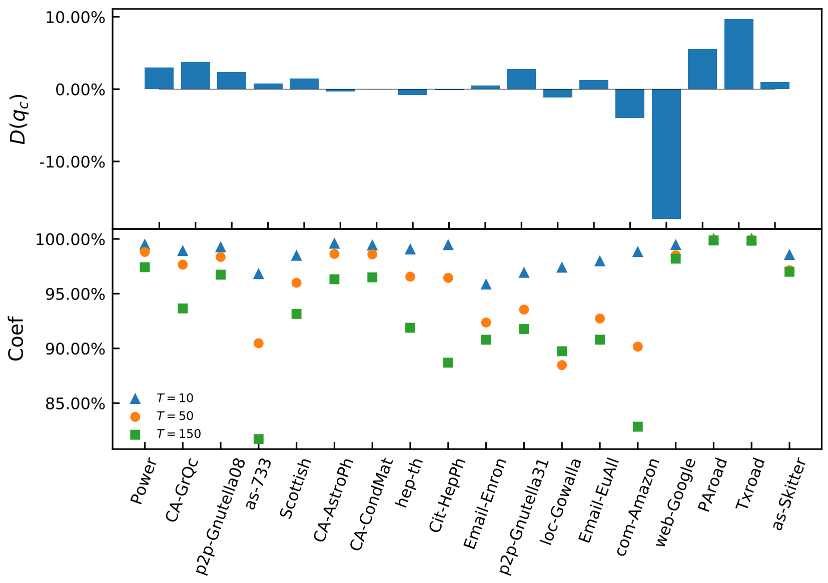


Figure A.21: Comparisons of q_c between ARRS and ARRSq on the 18 real-world networks, where the normalized q_c difference $D(q_c)$ is calculated through $D(q_c) = (q_c(\text{ARRS}) - q_c(\text{ARRSq}))/((q_c(\text{ARRS}) + q_c(\text{ARRSq}))/2)$. Coef is the Pearson correlation coefficient of F and q_c as a function of T considering ARRS.

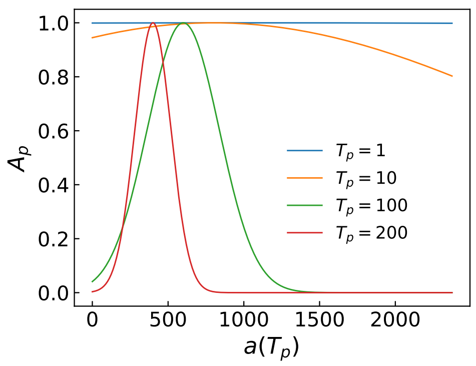


Figure A.22: An example of A_p against $a(T_p)$ regarding Eq. (3.40), where $b_2(0) = 0.05$ and $n = 2375$ are considered based on an assumed scenario of $b_1(1) = 1000$, $b_1(10) = 800$, $b_1(100) = 600$, and $b_1(200) = 400$.

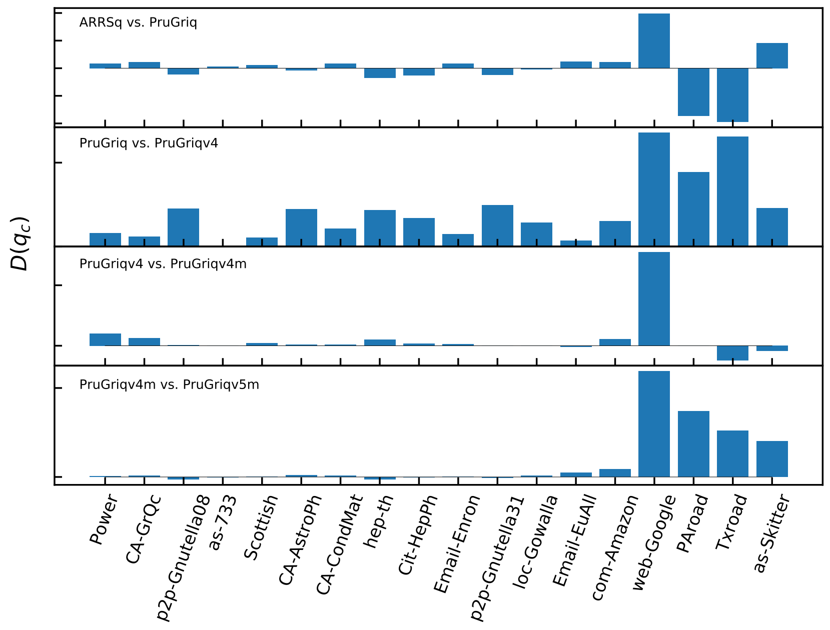


Figure A.23: Comparisons of q_c among ARRSq, PruGriq, PruGriqv4, PruGriqv4m (PruGriqv4 with mutation operators) and PruGriqv5m (PruGriqv5 with mutation operators) on the 18 real-world networks, where $D(q_c) = (q_c(\text{ARRSq}) - q_c(\text{PruGriq}))/((q_c(\text{ARRSq}) + q_c(\text{PruGriq}))/2)$ is calculated for ARRSq vs. PruGriq, and so forth. The value between every two ticks is uniquely 0.1 and each dark solid line corresponds to $D(q_c) = 0$.

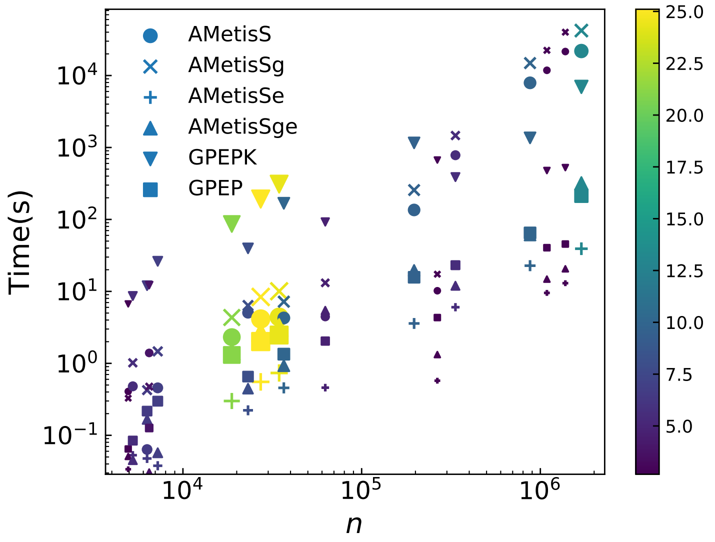


Figure A.24: Computing time (in second) of GPEP as a function of the network size n compared to AMetisS, AMetisSg, AMetisSe, AMetisSge, and GPEPK, where the average degree could be read by both the size and corresponding color. AMetisSe is AMetisS with an early stop (till $\mathcal{G}_a(q) < \alpha$) and AMetisSge is AMetisSg with an early stop. GPEPK is the case that the node separator is acquired based on KaHIP [137].

***F* of m_r/m regarding varied removal criteria from local perspective.** Refer to A.26.

Contours of $\langle \alpha_{\text{inf}} \rangle$ regarding Prodi2 on the Email-Enron network and loc-Gowalla network. Accordingly refer to Figs. A.27 and A.28.

Contours of $\langle \alpha_{\text{inf}} \rangle$ regarding Prodi2 with different m_r on the Email-Enron network. Refer to Fig. A.29.

Networks	10	20	40	80	160	320	640	GPEP
Power	0.0080†	0.0077	0.0082‡	0.0077	**0.0073**	0.0076	0.0075	0.0074
CA-GrQc	0.0314	0.0300	**0.0297**	0.0305	0.0301	0.0307	0.0323†	0.0300
p2p-Gnutella08	0.1468	0.1463	0.1456	0.1482	**0.1448**	0.1455	0.1526	0.1454
as-733	**0.0091**	0.0096	0.0092	0.0094	0.0096	0.0094	0.0097	0.0094
Scottish	0.0258	0.0262	0.0259	**0.0254**	0.0259	0.0265	0.0265	0.0266
CA-AstroPh	0.1254	0.1242	0.1234	0.1242	0.1235	0.1172	**0.1161**	0.1201
CA-CondMat	0.0626	0.0622	0.0635	0.0619	**0.0606**	0.0661†	0.0643	0.0627
hep-th	0.1510	0.1632†	0.1564	0.1492	**0.1485**	0.1540	0.1534	0.1515
Cit-HepPh	0.1357	**0.1357**	0.1362	0.1368	0.1374	0.1386	0.1464†	0.1360
Email-Enron	0.0225	0.0248‡	0.0223	0.0227	0.0223	0.0220	0.0222	**0.0219**
p2p-Gnutella31	0.1088	0.1100†	0.1060	0.1040	**0.1034**	0.1054	0.1058	0.1044
loc-Gowalla	0.0460	0.0470†	0.0471†	0.0456	0.0459	0.0451	**0.0424**	0.0444
Email-EuAll	0.0009	**0.0009**	0.0009	0.0010	0.0010	0.0010	0.0010	0.0011
com-Amazon	0.0272†	0.0262†	0.0261†	0.0253	0.0252	**0.0224**	0.0235	0.0248
web-Google	0.0102	0.0090	0.0104	0.0104	0.0089	0.0096	**0.0083**	0.0100
PAroad	0.0007‡	0.0007‡	0.0007†	0.0007	0.0007	0.0006	0.0007‡	**0.0006**
Txroad	0.0004‡	0.0004‡	0.0004†	0.0004	0.0003	**0.0003**	0.0003	0.0003
as-Skitter	0.0148‡	0.0139†	0.0138†	0.0135†	0.0130	0.0118	**0.0115**	0.0126
Ave. imp.	2.96%	2.41%	2.25%	1.63%	−0.63%	−0.97%	−0.56%	

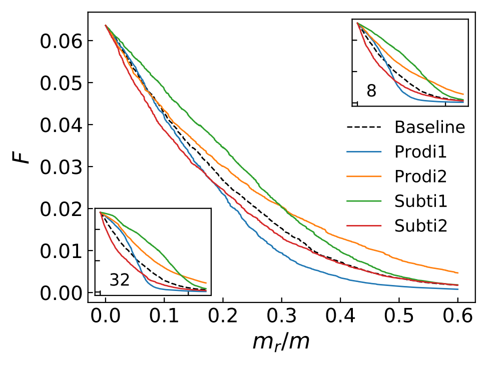
Table A.3: Performance of GPEP regarding varied imbalance. Note that GPEP is with 100.

Networks	10	20	40	80	160	320	640	1280	GPEP
Power	0.0074	0.0074	0.0074	0.0074	0.0074	0.0074	0.0074	0.0074	**0.0074**
CA-GrQc	0.0304	0.0303	0.0302	0.0301	0.0300	**0.0300**	0.0300	0.0300	0.0300
p2p-Gnutella08	0.1485	0.1462	0.1457	0.1455	0.1455	0.1454	0.1455	**0.1454**	0.1454
as-733	0.0096	0.0095	0.0095	0.0094	**0.0094**	0.0094	0.0094	0.0095	0.0094
Scottish	0.0269	0.0268	0.0267	0.0266	0.0266	0.0266	0.0266	0.0266	**0.0266**
CA-AstroPh	0.1339‡	0.1292†	0.1235	0.1201	0.1201	**0.1197**	0.1198	0.1198	0.1201
CA-CondMat	0.0633	0.0631	0.0630	0.0627	0.0625	**0.0625**	0.0625	0.0625	0.0627
hep-th	0.1515	0.1515	0.1515	0.1515	0.1515	0.1514	0.1513	**0.1513**	0.1515
Cit-HepPh	0.1361	0.1361	0.1361	0.1361	0.1360	0.1360	0.1360	**0.1360**	0.1360
Email-Enron	0.0224	0.0222	0.0221	0.0220	**0.0219**	0.0219	0.0219	0.0220	0.0219
p2p-Gnutella31	0.1074	0.1067	0.1057	0.1045	0.1042	0.1042	**0.1042**	0.1042	0.1044
loc-Gowalla	0.0448	0.0448	0.0447	0.0445	0.0441	0.0440	0.0439	**0.0439**	0.0444
Email-EuAll	0.0011	0.0011	0.0011	0.0011	**0.0011**	0.0011	0.0011	0.0011	0.0011
com-Amazon	0.0248	0.0248	0.0248	0.0248	0.0247	0.0246	0.0246	**0.0245**	0.0248
web-Google	0.0101	0.0101	0.0101	0.0100	0.0099	0.0099	0.0098	**0.0098**	0.0100
PAroad	0.0006	0.0006	0.0006	0.0006	0.0006	**0.0006**	0.0006	0.0006	0.0006
Txroad	0.0003	0.0003	0.0003	0.0003	**0.0003**	0.0003	0.0003	0.0003	0.0003
as-Skitter	0.0132	0.0132	0.0130	0.0127	0.0126	0.0125	0.0125	**0.0125**	0.0126
Ave. imp.	1.80%	1.21%	0.69%	0.16%	−0.13%	−0.25%	−0.27%	−0.24%	

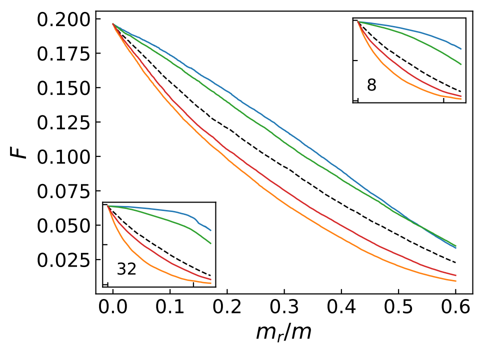
Table A.4: Performance of GPEP regarding varied $\hat{\tau}$. Note that GPEP is with 100.

A.3.2 Prediction of the Hysteresis in Explosive Synchronization

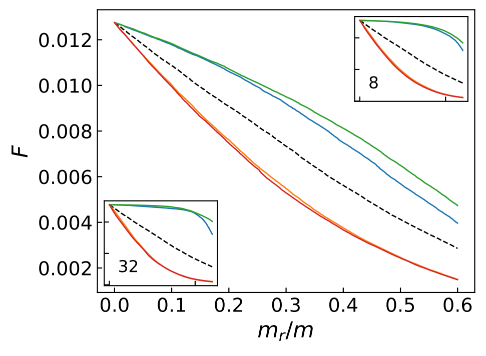
Examples regarding $\mathcal{S}$. Refer to Fig. A.30.



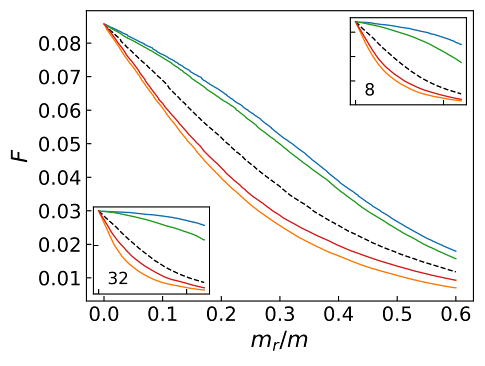
(a)



(b)

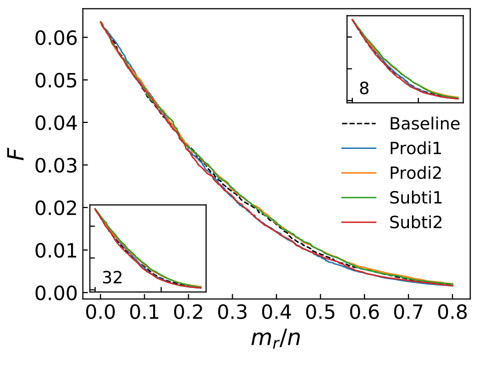


(c)

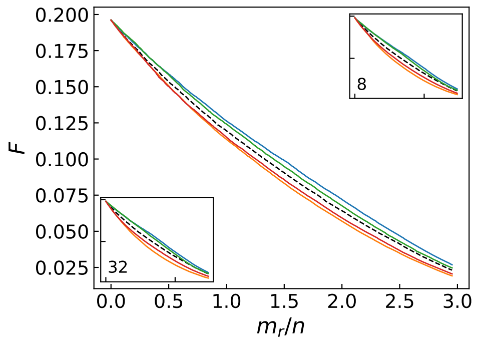


(d)

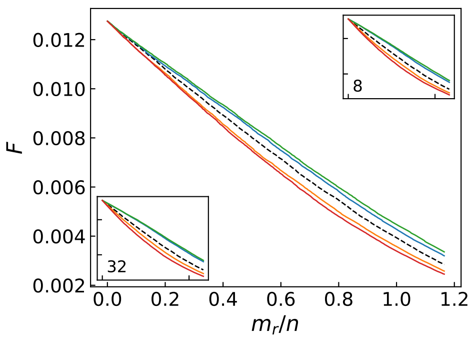
Figure A.25: F against the fraction of removed edges m_r/m regarding varied criteria on (a) the power grid network, (b) the yeast network, (c) the as-733 network, and (d) the global airline network. The Baseline (the dashed line) corresponds to the case that m_r/m fraction of edges are randomly removed from the network. The main figure is related to $a_r = 2$ and the two inserts are associated with $a_r = 8$ and $a_r = 32$, respectively.



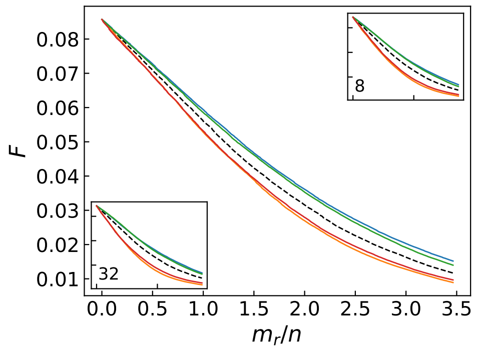
(a)



(b)

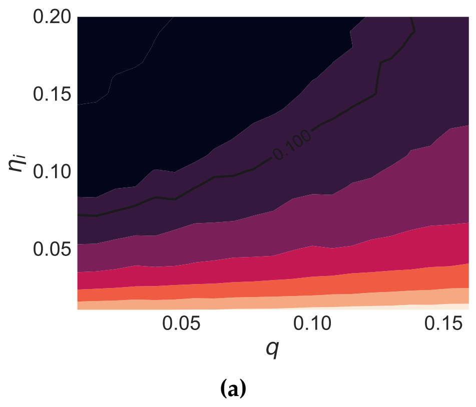


(c)

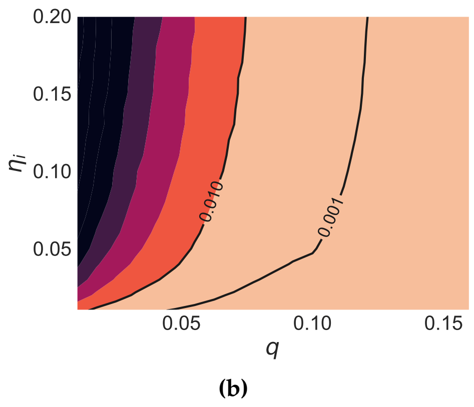


(d)

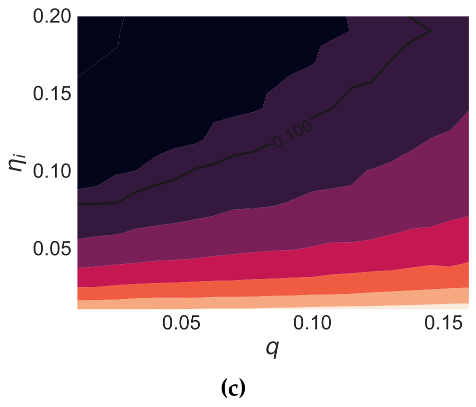
Figure A.26: F against the fraction of removed edges m_r and the network size n regarding varied criteria on (a) the power grid network, (b) the yeast network, (c) the as-733 network, and (d) the global airline network. The Baseline (the dashed line) corresponds to the case that m_r edges are randomly removed from the network. The main figure is related to $a_r = 2$ and the two inserts are associated with $a_r = 8$ and $a_r = 32$, respectively.



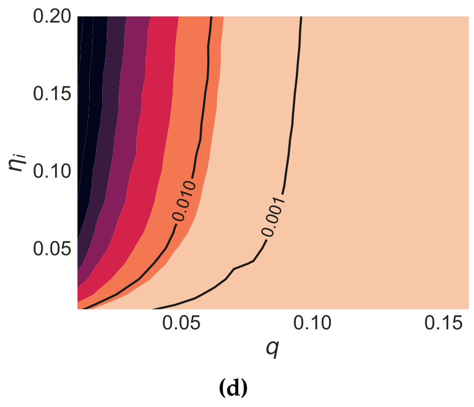
(a)



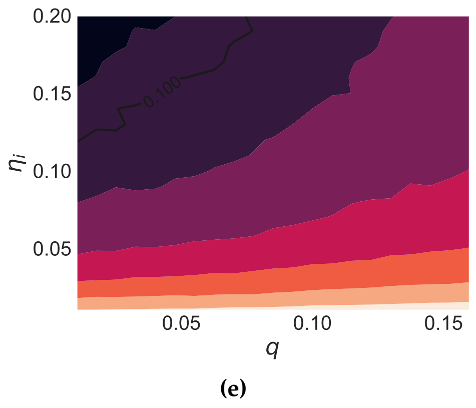
(b)



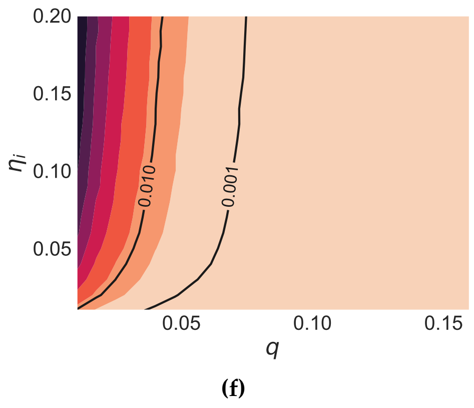
(c)



(d)

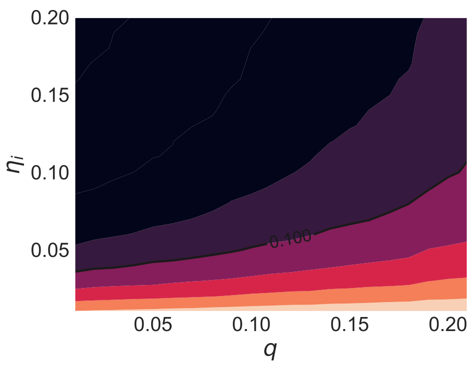


(e)

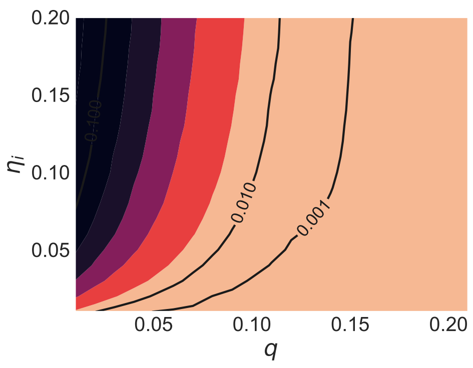


(f)

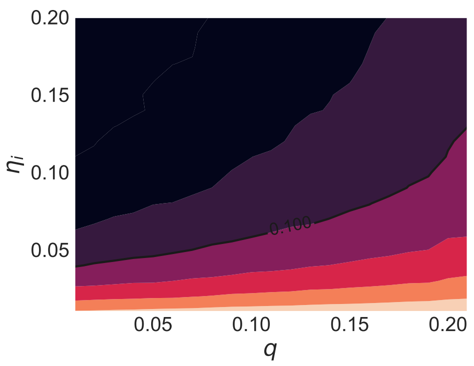
Figure A.27: Contours of $\langle \alpha_{\mathrm{inf}} \rangle$ as a function of the immunized fraction q and the infected probability η_i on the Email-Enron network regarding Prodi2 with $m_r = 2.0n$ for RanS in (a), (c), and (e), and AcqI in (b), (d), and (f). Note that RanS and AcqI are in different scales of the color, e.g., the same colors represent the same magnitudes of $\langle \alpha_{\mathrm{inf}} \rangle$ for (a) and (c) but they are different for (a) and (b). In each figure, color from deep dark (i.e., top left) to light purple (bottom right) indicates that $\langle \alpha_{\mathrm{inf}} \rangle$ changes from large to small. In addition, $\langle \alpha_{\mathrm{inf}} \rangle = 0.001, 0.01$ and 0.1 are marked by the three solid curves, respectively.



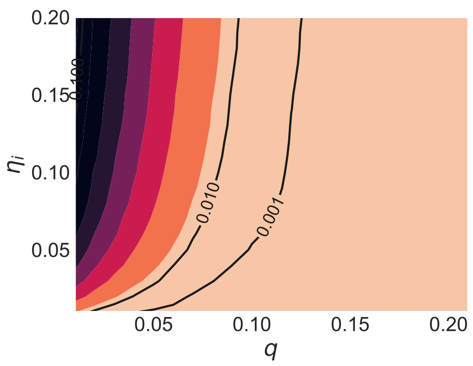
(a)



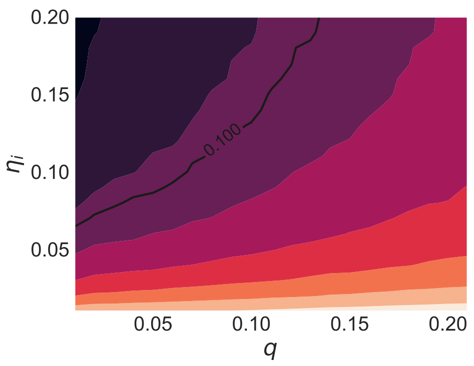
(b)



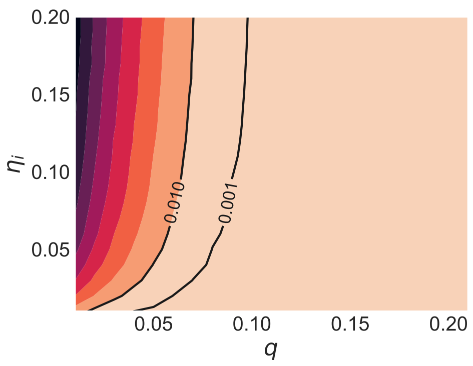
(c)



(d)

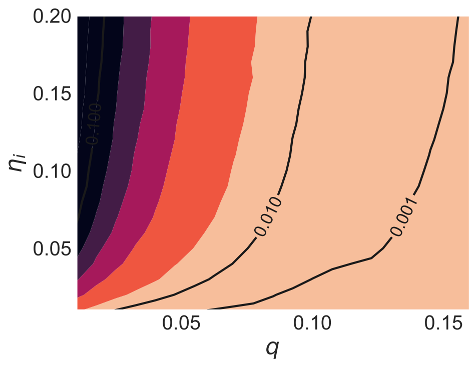


(e)

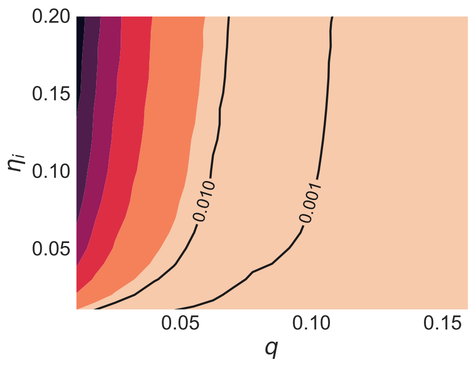


(f)

Figure A.28: Contours of $\langle \alpha_{\text{inf}} \rangle$ as a function of q and η_i on the loc-Gowalla network regarding Prodi2 with $m_r = 2.0n$ for RanS in (a), (c), and (e), and AcqI in (b), (d), and (f).

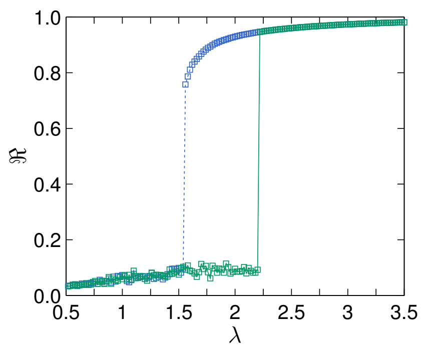


(a)

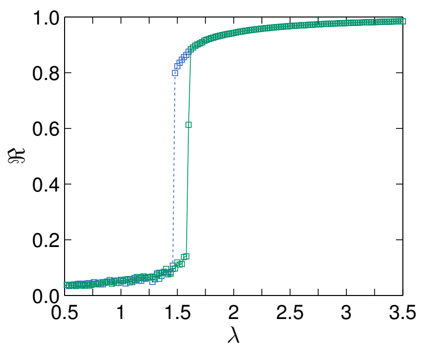


(b)

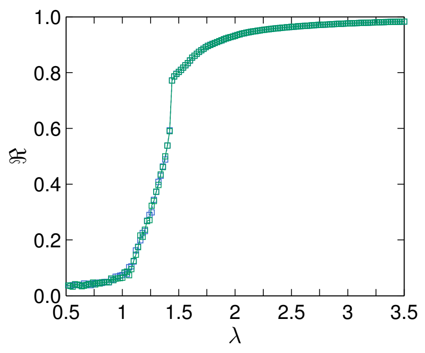
Figure A.29: Contours of $\langle \alpha_{\text{inf}} \rangle$ as a function of q and η_i on the loc-Gowalla network regarding Prodi2 with (a) $m_r = 1.0n$ and (b) $m_r = 1.5n$ considering AcqI.



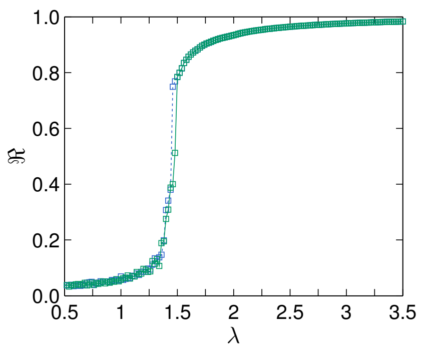
(a)



(b)



(c)



(d)

Figure A.30: Examples regarding $\mathcal{S}$. (a) $\mathcal{S} = 0.5323$. (b) $\mathcal{S} = 0.0913$. (c) $\mathcal{S} = 0.0000$. (d) $\mathcal{S} = 0.0121$.