ABSTRACT

Title of dissertation:	Diffusion, Infection and Social (Information) Network Database
	Chanhyun Kang, Doctor of Philosophy, 2015
Dissertation directed by:	Professor V.S. Subrahmanian Department of Computer Science

Research to analyze diffusive phenomena over large rich datasets has received considerable attention in recent years. Moreover, with the appearance and proliferation of online social network services, social (information) network analysis and mining techniques have become closely intertwined with the analysis of diffusive and infection phenomena. In this dissertation, we suggest various analysis and mining techniques to solve problems related to diffusive and infection phenomena over social (information) networks built from various datasets in diverse areas. This research makes five contributions. The first contribution is about influence analysis in social networks for which we suggest two new centrality measures, Diffusion Centrality and Covertness Centrality. Diffusion Centrality quantifies the influence of vertices in social networks with respect to a given diffusion model which explains how a diffusive property is spreading. Covertness Centrality quantifies how well a vertex can communicate (diffuse information) with (to) others and hide in networks as a common vertex w.r.t. a set of centrality measures. The second contribution is about network simplification problems to scale up analysis techniques for very large networks. For this topic, two techniques, CoarseNet and Coarsened Back and Forth (CBAF), are suggested in order to find a succinct representation of networks while preserving key characteristics for diffusion processes on that network. The third contribution is about social network databases. We propose a new network model, STUN (Spatio-Temporal Uncertain Networks), whose edges are

characterized with uncertainty, space, and time, and develop a graph index structure to retrieve graph patterns over the network efficiently. The fourth contribution develops epidemic models and ensembles to predict the number of malware infections in countries using past detection history. In our fifth contribution, we also develop methods to predict financial crises of countries using financial connectedness among countries.

Diffusion, Infection and Social (information) Network Database

by

Chanhyun Kang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Contents

Chapter 1: Introduction

With the proliferation of rich data-sets from various scientific fields such as economics, social science, finance, ecology, biology and computer science, there is now much ongoing research to analyze such data sets. Moreover, because it is possible to get fine and abundant historical and (or) interaction data from various online social network services, the analysis of diffusive and infection phenomenon has been extensively studied and social (information) network analysis and mining techniques have become closely intertwined with the analysis of diffusive and infection phenomena. Moreover, as graph data models can naturally express the relationships and interactions among data elements, graph models have been widely used for solving various problems in diverse areas.

In this thesis, we suggest various analysis and mining techniques to solve problems related to diffusive and infection phenomena over social (information) networks built from various data-sets in diverse areas. In short, we focus on developing data mining techniques for *social influence analysis* to quantify the relative importance of vertices in networks, *network simplification* to scale up social analysis techniques for very large networks, *social network database* for networks whose edges have space, time and uncertainty information, *malware infection prediction* to predict the volume of malware infections in each country and *systematic banking crises prediction* over countries. The short descriptions of the contributions are below.

Social Influence Analysis. Influence analysis in social networks has received considerable attention in recent years due to its wide range of applications, e.g viral marketing, personalized recommendation, link prediction and ranking of feeds. Centrality of a vertex as a measure of its relative importance within a graph is a fundamental concept in network analysis. It has been extensively studied and widely used for social network influence analysis. In Chapter 2, we develop two new centrality measures, *Diffusion Centrality* and *Covertness Centrality* for different analysis purposes.

In different areas including economics, epidemiology, physics, sociology and computer science, there has been much research on analyzing how various diffusive properties of vertices spread through the networks and various diffusion models have been developed. However, existing centrality measures mainly depend on the structural properties of networks and do not explicitly consider diffusion processes, how properties diffuse through social networks. They also do no consider the semantical properties of nodes, and the nature of the links between them. Therefore, in Chapter 2.1, we suggest Diffusion Centrality to quantify the influence of vertices in social networks with respect to a given diffusion model. We then develop a general framework which consists of syntax/semantics to express diffusion models and suggest a hypergraph-based algorithm, *HyperDC*, to compute diffusion centrality efficiently with optimizations in the social information networks for a given diffusion model.

It has been known for some time that in terror networks, money laundering networks, and criminal networks, important players want to stay off the radar and to blend into the crowd. But they hold sufficient ability to communicate with or to diffuse information to the rest of their network. In order to capture these, Chapter 2.2 proposes Covertness Centrality. Covertness Centrality measures how common a vertex is w.r.t. a set of centrality measures (common-ness) and how well the vertex can communicate with a user-specified set of vertices. We first propose probabilistic models to measure common-ness of a vertex and then, define Covertness Centrality of a vertex as a linear combination of common-ness and the communication ability of the vertex.

Network Simplification. In very large networks, it would be not feasible to apply network analysis techniques directly and exactly.Because of this, several network simplification techniques have been developed in order to analyze huge social networks. But there are no network simplification techniques to capture smaller equivalent representation of the network that preserves its diffusion and infection characteristics with considering not only structural aspects but also semantic aspects of social networks. So In Chapter 3 we suggest two network simplification techniques, COARSENET and Coarsened Back and Forth (*CBAF*).

In Chapter 3.1, we first formulate a novel Graph Coarsening Problem to find a succinct representation of any graph while preserving key characteristics for diffusion processes on that graph. We then provide a fast and effective near-linear-time (in nodes and edges) algorithm, COARSENET and show how our method can help in diverse applications like influence maximization and detecting patterns of propagation at the level of automatically created groups on real cascade data. In Chapter 3.2, we develop another network simplification technique which is extended from COARSENET to support various diffusion models explicitly and semantic aspects of social networks. Then, we suggest *CBAF* algorithm to find top-k vertices having the highest diffusion centrality in very large social information networks approximately and efficiently.

Spatio-Temporal Uncertain Network. In prior social network analysis research, an edge expresses the relationship or the interaction between two vertices with labels. However, due to the widespread adoption of location-sensing mobile devices, people can share information about their geographic position with others easily via online social network services. As a consequence, service providers have access to the data on online relations and interactions with spatial and temporal information among users. Many companies already provide services taking into account spatial and temporal information with social networks. There are also numerous ongoing efforts to infer spatial, temporal and other properties of social networks. The spatial, temporal and uncertainty information allow us to build reliable social network models and the models can help us enrich network analysis techniques. Moreover, it is obvious that diffusive and infection phenomena could be (very) different according to different spatio-temporal situations. But the framework to express social networks whose edges have space, time and uncertainty information has not been studied yet. So, in Chapter 4, we propose a new network model, STUN (Spatio-Temporal Uncertain Networks), to formally define social networks whose edges can be characterized with uncertainty, space, and time information. We then develop an index structure to store the networks and to retrieve graph patterns efficiently with space, time, and uncertainty constraints over the networks. We also introduce STUN ranking queries which attempt to identify important nodes in a STUN network, extending the well-known PageRank algorithm.

In the previous techniques suggested in this thesis, we assume that diffusion models are given. In this thesis, we also suggest how diffusion models can be built for capturing the diffusion and infection process of specific diffusive and infection phenomenon from infection history data. Our research does not suggest a general framework to build diffusion models for various datasets, because the infection properties (features) of diffusive and infection phenomenon would be (very) different for different application domains. For example, the diffusion and infection features of epidemic diseases would be very different from the diffusion process of information and rumors. In our research, we show how social network analysis techniques can be used for the analysis of diffusive and infection phenomenon and how the analyzed features can be used for building diffusion and infection models to predict the infections of diffusive objects. We specifically study the diffusion process and infection (diffusion) phenomenon of two different data sets.

Malware Infection and Diffusion prediction model. Given a history of detected malware attacks, can we predict the number of malware infections in a country? Can we do this for different malware and countries? This is an important question which has numerous implications for cyber security, right from designing better anti-virus software, to designing and deloying targeted patches to more accurately measuring the economic impact of breaches. This problem is compounded by the fact that, as externals, we can only detect a fraction of actual malware infections. In Chapter 5, we address this problem using data from Symantec covering more than *1.4 million hosts* and 50 malware spread across 2 years and multiple countries. We first carefully design domain-based features from both malware and machine-hosts perspectives. Secondly, inspired by epidemiological and information diffusion models, we design a novel temporal non-linear model for malware spread and detection. Finally we present ESM, an ensemble-based approach which combines both these methods to construct a more accurate algorithm. Using extensive experiments spanning multiple malware and countries, we show that ESM can effectively predict malware infection ratios over time (both the actual number and trend) and the performance is stable and robust even when the number of detected infections is low.

Systematic Banking Crises (infection) prediction model. The global financial crisis has reignited interest in crisis prediction models. It has also raised the question whether financial interconnectedness can serve as an early warning indicator of crises. In Chapter 6, first we suggest network indicators that measure the connectedness among countries in the global network of financial linkages. The indicators can be obtained by applying social network analysis techniques and can capture the infection (diffusion) properties of crises over countries implicitly and relatively. Second, we suggest a infection prediction model to predict which countries will experience systemic banking crises with the indicators. We then examine the ability of connectedness in the global network of financial linkages to predict systemic banking crises during 1978-2010 year period.

Chapter 2: Social Influence Analysis

Influence analysis in social networks has received considerable attention in recent years due to its wide range of applications, e.g viral marketing, personalized recommendation, link prediction and ranking of feeds. Various centrality measures have been used for social influence analysis. However, there is no centrality measure to consider the infection (propagation) flow of diffusive properties directly and explicitly in various diffusive and infection phenomenon over networks. No centrality measure also exists to capture the features of important players who want to hide in a crowd and to have sufficient communication ability with others in crime social networks. In Chapter 2, we suggest two new centrality measures, *Diffusion Centrality* and *Covertness Centrality* for them. *Diffusion Centrality* is to measure the relative influence of a vertex in a social information network with considering explicitly the diffusion process of a given diffusion model and proposed in Section 2.1. *Covertness Centrality* quantifies the abilities of a vertex, how well the vertex can be hidden in crowds and communicate with others. We develop it in Section 2.2.

2.1 Diffusion Centrality

2.1.1 Introduction

An increasingly important problem in social networks (SNs) is that of assigning a 'centrality" value to vertices reflecting their importance within the SN. Well-known measures such as *degree centrality* [83, 163], *betweenness centrality* [37, 82], *PageRank* [**?**], *closeness centrality* [177, 21], and *eigenvector centrality* [36] only take the structure of the network into account—they do not differentiate between vertices that are central w.r.t. spreading one topic or meme or sentiment vs. spreading another. A vertex that is important in spreading awareness of a mobile phone program may be very unimportant in spreading information about a restaurant. Likewise, most past work assumes that there is no information about properties of the vertices/edges or edge weights—but in modern social networks, at least some self-declared properties exist and, in many cases, analysis of tweets and posts can provide further information. These omissions can cause serious problems as shown in the following toy example.

Example 2.1.1 (HIV). *Figure 2.1 shows four people a, b, c, d, where b has HIV. Solid edges denote sexual relationships, while dashed edges denote friend relationships (these edges are undirected in this example, i.e. sexual partners and friends are considered symmetric relationships). Edge weights denote the intensity of these relationships.*

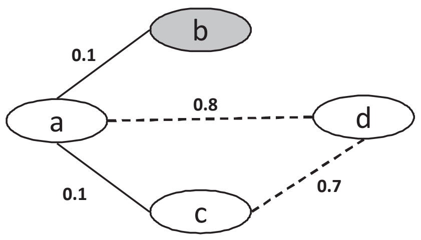


Fig. 2.1: [

]A small HIV social network. Shaded vertices denote people with HIV. Solid edges denote sexual relationships, dashed edges denote friend relationships.

The table below shows the centrality of a, b, c, d w.r.t. various centrality measures[1].

Intuitively, the only person in this network capable of spreading HIV is b. However, b has the lowest centrality according to all five centrality measures mentioned above.

Past centrality measures *ignore how properties (e.g., HIV in the above example) diffuse*

[1] The classical definitions of the centrality measures are used as opposed to their weighted versions.

Centrality Measure	**a**	**b**	**c**	**d**
Degree	1	0.33	0.66	0.66
Betweenness	2	0	0	0
PageRank	0.367	0.141	0.246	0.246
Closeness	0.33	0.2	0.25	0.25
Eigenvector	0.375	0.125	0.25	0.25

through the SN, solely focusing on the structure of the network. We can readily think of networks (e.g., Twitter) where person A has the highest centrality in terms of spread of support for Republicans, while person B is the central player in terms of spread of support for conserving gorillas. The network in both cases is the same (Twitter), but the centrality of vertices should be measured both by structural graph properties and by a vertex's ability to diffuse a given property. In Section 2.1, *we propose the novel notion of* diffusion centrality *that takes an SN, a diffusive property p, and a previously learned diffusion model Π for p, and defines centrality of vertices based on these inputs.* We do not provide algorithms to automatically learn diffusion models—interested readers may find one such algorithm in [41].

In Section 2.1, we also show how diffusion centrality can be used to achieve significantly higher spread of a diffusive property p by using diffusion models for p rather than classical centrality measures. We further show that this can be done for most diffusion models we have seen in the literature. Moreover, this can often be done in time comparable to that of Pagerank. Last but not least, our methods are shown to scale to social networks with over 2M vertices and 20M edges.

Section 2.1 is organized as follows.

- We define *Diffusion Centrality (DC)* in Section 2.1.3 to quantify the influence of vertices in a social network with respect to *(i)* a given diffusion model which explains how a diffusive property is spreading, *(ii)* the structural properties and *(iii)* the semantic properties of the network. Diffusion models are expressed via Generalized Annotated (logic) Programs or GAPs [131] that were previously shown [184] to express most diffusion models studied in the literature, including a variety of cascade models, tipping models, disease spread models, and homophilic models.

- Section 2.1.4 proposes a general ‘hypergraph fixed point algorithm” (HyperLFP) to efficiently compute the likelihood that an arbitrary vertex has certain properties according to the GAP diffusion model. We also define novel classes of GAPs (such as p-monotone and p-dwindling), together with a novel suite of network filtering methods to develop the HyperDC algorithm. These optimization significantly reduce the time required to compute the expected spread of p through the network w.r.t. a diffusion model Π.

- Section 2.1.5 describes extensive experiments comparing DC with classical centrality measures in terms of both runtime and the “spread” generated by central vertices. Experimental results on multiple real-world social networks and widely used diffusion models in the literature show that when the top-k vertices selected by diffusion centrality are given property p (e.g., an incentive to spread good news about a mobile phone plan or a political candidate), they usually achieve much bigger spreads (i.e., expected number of vertices in the network having property p) than when these top-k vertices are selected by classical centrality measures. This maximization of spread w.r.t. a wide variety of diffusion models is the main goal of this paper. Moreover, the experiments show that HyperDC is much faster than betweenness and closeness centrality—its runtime is often similar to PageRank, eigenvector, and degree centrality.

In Section Ap.1.4, we summarize all the notations used in Section 2.1.

2.1.2 Preliminaries

In this section, we formally define social networks (SNs), recall generalized annotated programs (GAPs) from [131], and show how diffusion models can be expressed with GAPs.

2.1.2.1 Social Networks

We assume the existence of a set VP of unary predicate symbols (to capture properties of vertices in a social network), called *vertex predicate symbols* (or *properties*), and a set EP of binary predicate symbols (intended to capture relationships between vertices in a social network), called *edge predicate symbols*.

Definition 1 (Social Network)**.** *A* social network *(SN) is a tuple* $(V, E, \mathsf{VL}, \omega)$ *where:*

1. V *is a finite set of* vertices*;*
2. $E \subseteq V \times V \times \mathsf{EP}$ *is a finite set of* (labeled) edges*;*
3. $\mathsf{VL} : V \rightarrow 2^{\mathsf{VP}}$ *assigns a set of properties to each vertex;*
4. $\omega : E \rightarrow (0, 1]$ *assigns a weight to each edge.*

Intuitively, an SN is a directed graph where VL assigns a set of properties to each vertex and there can be multiple labeled edges between a given pair of vertices, each of which is associated with a weight and a unique edge predicate symbol.

Example 2.1.2. *Consider the SN of Example 2.1.1 (cf. Figure 2.1). Here,* $\mathsf{VP} = \{\mathsf{hiv}\}$ *and* $\mathsf{EP} = \{\mathsf{sp}, \mathsf{fr}\}$*, where* sp *and* fr *stand for sexual and friend relationships, respectively. The SN is* *defined as:*

1. $V = \{a, b, c, d\}$.

2. $E = \{\langle a, b, \mathsf{sp}\rangle, \langle b, a, \mathsf{sp}\rangle, \langle a, c, \mathsf{sp}\rangle, \langle c, a, \mathsf{sp}\rangle, \langle a, d, \mathsf{fr}\rangle, \langle d, a, \mathsf{fr}\rangle, \langle c, d, \mathsf{fr}\rangle, \langle d, c, \mathsf{fr}\rangle\}$.

3. $\mathsf{VL}(b) = \{\mathsf{hiv}\}$; $\mathsf{VL}(a) = \mathsf{VL}(c) = \mathsf{VL}(d) = \emptyset$.

4. $\omega(\langle a, b, \mathsf{sp}\rangle) = \omega(\langle b, a, \mathsf{sp}\rangle) = \omega(\langle a, c, \mathsf{sp}\rangle) = \omega(\langle c, a, \mathsf{sp}\rangle) = 0.1$;

   $\omega(\langle a, d, \mathsf{fr}\rangle) = \omega(\langle d, a, \mathsf{fr}\rangle) = 0.8$; $\omega(\langle c, d, \mathsf{fr}\rangle) = \omega(\langle d, c, \mathsf{fr}\rangle) = 0.7$.

Our definition of social networks is much broader than that used by several researchers [**?**, 63, 115, 127] who often do not consider either vertex properties or edge labels or edge weights—it is well known in marketing that intrinsic properties of vertices (customers, patients) and the nature and strength of the relationships (edges) are critical for decision-making.

2.1.2.2 Generalized Annotated Programs (GAPs)

Syntax. We assume the existence of a set $\mathcal{T}$ of variable symbols ranging over the unit real interval $[0, 1]$ and a set $\mathcal{F}$ of function symbols (corresponding to continuous functions), each of which has an associated arity.

Definition 2 (Annotation Term)**.** *Annotation terms are defined as follows:*

1. *Any member of* $[0, 1] \cup \mathcal{T}$ *is an* annotation term.

2. *If $f \in \mathcal{F}$ is an n-ary function symbol and $t_1, \ldots, t_n$ are annotation terms, then $f(t_1, \ldots, t_n)$ is an* annotation term.

For instance, 0.5 and X are annotation terms (X is assumed to be a variable in $\mathcal{T}$). If $+$ is a binary function symbol in $\mathcal{F}$, then $X + 1$ is an annotation term.

We now define a logical language whose predicate symbols are $\mathsf{VP} \cup \mathsf{EP}$. We assume the existence of a finite set V of constants (the universe of vertices) and a set $\mathcal{V}$ of variable symbols ranging over the constants (vertices). No function symbols are present. Any member of $\mathsf{V} \cup \mathcal{V}$ is a *term*. If $p \in \mathsf{VP}$ and t is a term, then $p(t)$ is a *vertex atom*. If $p \in \mathsf{EP}$ and t_1, t_2 are terms, then $p(t_1, t_2)$ is an *edge atom*.

Definition 3 (Annotated Atom/GAP-Rule/GAP)**.** *If A is an atom and μ is an annotation term, then $A : \mu$ is an* annotated atom*; if A is a vertex (resp. edge) atom, then $A : \mu$ is also called a* vertex *(resp.* edge*) annotated atom. A* GAP-rule *(or simply* rule*) is of the form $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$ where $n \geq 0$ and every $A_i : \mu_i$ is an annotated atom. $A_0 : \mu_0$ is the* head *of the rule, while $A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$ is the* body *of the rule. A* generalized annotated program *(GAP) is a finite set of rules.*

For notational simplicity, we will write a ground rule $A_0 : \mu_0 \leftarrow$ simply as $A_0 : \mu_0$, that is, we drop $\leftarrow$. Such a rule is called a *fact*. An annotated atom (resp. rule, GAP) is *ground* iff there are no occurrences of variables from either $\mathcal{T}$ or $\mathcal{V}$ in it. We use $\mathcal{A}$ to denote the set of all ground atoms. Throughout this paper we consider a restricted class of GAPs: every rule with a non-empty body has a vertex annotated atom in the head ([131] allows any annotated atom in the head of a rule). Thus, edge atoms can appear only in rule bodies or rules with an empty body. This restriction results from the set of diffusion models we consider in this paper: neither edge weights nor edge labels change as the result of the diffusion.

We use $grd(r)$ to denote the *ground instances* of a rule r, i.e. the set of all rules obtained from r by replacing every occurrence of a variable in $\mathcal{T}$ (i.e., appearing in annotation terms) with a real number in $[0, 1]$, and every occurrence of a variable in $\mathcal{V}$ (i.e., appearing in vertex or edge atoms) with a vertex in V—multiple occurrences of the same variable are replaced in the same way. Given a GAP Π, we use $grd(\Pi)$ to denote the set of all ground instances of rules in Π, i.e. $grd(\Pi) = \bigcup_{r \in \Pi} grd(r)$.

We assume that VP contains a distinguished vertex predicate symbol vertex that represents the presence of a vertex in an SN. Every SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ can be represented by a GAP, denoted $\Pi_{\mathcal{S}}$, as follows:

$$\Pi_{\mathcal{S}} = \{\mathsf{vertex}(v) : 1 \mid v \in V\} \cup \{p(v) : 1 \mid v \in V \wedge p \in \mathsf{VL}(v)\} \cup \\ \{ep(v_1, v_2) : \omega(\langle v_1, v_2, ep\rangle) \mid \langle v_1, v_2, ep\rangle \in E\}$$

Example 2.1.3. *The GAP associated with the SN of Example 2.1.1 is the following:*

$$\begin{array}{llll} \mathsf{vertex}(a) : 1 & \mathsf{sp}(a,b) : 0.1 & \mathsf{fr}(a,d) : 0.8 & \mathsf{hiv}(b) : 1 \\ \mathsf{vertex}(b) : 1 & \mathsf{sp}(b,a) : 0.1 & \mathsf{fr}(d,a) : 0.8 & \\ \mathsf{vertex}(c) : 1 & \mathsf{sp}(a,c) : 0.1 & \mathsf{fr}(c,d) : 0.7 & \\ \mathsf{vertex}(d) : 1 & \mathsf{sp}(c,a) : 0.1 & \mathsf{fr}(d,c) : 0.7 & \end{array}$$

When we augment $\Pi_{\mathcal{S}}$ with other rules, such as rules describing how certain properties diffuse through the social network, we get a GAP $\Pi \supseteq \Pi_{\mathcal{S}}$ that captures both the structure of the SN and the diffusion principles. Below is a small example.

A GAP Π_{hiv} for the SN of Example 2.1.1 might be:

$$r_1: \ \mathsf{hiv}(V) : 0.9 \times X \times Y \leftarrow \mathsf{sp}(V, V') : Y \wedge \mathsf{hiv}(V') : X$$

$$r_2: \ \mathsf{hiv}(V) : 0.4 \times X \times Y \times Y' \leftarrow \mathsf{fr}(V, V') : Y \wedge \mathsf{sp}(V', V'') : Y' \wedge \mathsf{hiv}(V'') : X$$

$$r_3: \ \mathsf{hiv}(V) : 0.6 \times X \times Y \times Y' \leftarrow \mathsf{sp}(V, V') : Y \wedge \mathsf{sp}(V', V'') : Y' \wedge \mathsf{hiv}(V'') : X$$

The first rule says that the confidence that a vertex V has HIV, given that a partner V' has HIV with confidence X, is $0.9 \times X \times Y$, where Y is the weight of the sexual relationship between the two vertices. The other rules can be similarly read.

Semantics. An interpretation I is a mapping from the set of all ground atoms $\mathcal{A}$ to $[0,1]$. The set of all interpretations can be partially ordered as follows: given two interpretations I_1 and I_2, then $I_1 \preceq I_2$ iff for all ground atoms $A \in \mathcal{A}$, $I_1(A) \leq I_2(A)$. An interpretation I *satisfies* a ground annotated atom $A : \mu$, denoted $I \models A : \mu$, iff $I(A) \geq \mu$. I *satisfies* a ground rule r of the form $AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$, denoted $I \models r$, iff *(i)* I satisfies AA_0 or *(ii)* there exists an $1 \leq i \leq n$ such that I does not satisfy AA_i. I *satisfies* a (possibly non-ground) rule iff I satisfies all ground instances of it. I *satisfies* a GAP Π iff I satisfies every rule in Π. A GAP Π *entails* a ground annotated atom AA, denoted $\Pi \models AA$, iff every interpretation I that satisfies Π also satisfies AA.

[131] associates an operator $\mathbf{T}_\Pi$ that maps interpretations to interpretations with any GAP Π. Suppose I is an interpretation. Then,

$$\mathbf{T}_\Pi(I)(A) = \max(\{I(A)\} \cup \{\mu \mid A : \mu \leftarrow AA_1 \wedge \ldots \wedge AA_n \text{ is in } grd(\Pi) \text{ and}$$
$$\text{for all } 1 \leq i \leq n,\ I \models AA_i\})$$

The *iteration* of $\mathbf{T}_\Pi$ is defined as follows:

$\mathbf{T}_\Pi \uparrow 0$ is the interpretation that assigns 0 to all ground atoms.

$\mathbf{T}_\Pi \uparrow (i+1) = \mathbf{T}_\Pi(\mathbf{T}_\Pi \uparrow i)$.

[131] shows that $\mathbf{T}_\Pi$ is monotonic (w.r.t. $\preceq$) and has a least fixed point $\mathsf{lfp}(\mathbf{T}_\Pi)$. Moreover, they show that a GAP Π entails a ground annotated atom $A : \mu$ iff $\mu \leq \mathsf{lfp}(\mathbf{T}_\Pi)(A)$ and therefore $\mathsf{lfp}(\mathbf{T}_\Pi)$ precisely captures the ground atomic logical consequences of Π. We will denote the least fixed point $\mathsf{lfp}(\mathbf{T}_\Pi)$ also as $\mathsf{lfp}(\Pi)$.

Example 2.1.4. *Consider again the HIV SN $\mathcal{S}$ of Example 2.1.1 and the GAP Π_{hiv} of Example 2.1.2.2. Let $\Pi = \Pi_{\mathcal{S}} \cup \Pi_{hiv}$. $\mathbf{T}_\Pi$ has a least fixed point that coincides with $\mathbf{T}_\Pi \uparrow 3$ as shown in Table 2.1 (edge atoms are not reported as the values assigned to them at iteration 1 are those reported in Example 2.1.3 and do not change after iteration 1). Thus, the least fixed point assigns* 0.09 *to* $\mathsf{hiv}(a)$*, 1 to* $\mathsf{hiv}(b)$*, 0.0081 to* $\mathsf{hiv}(c)$*, and 0.032 to* $\mathsf{hiv}(d)$.

2.1.2.3 Expressing Diffusion Models with GAPs

Diffusion models specify how vertex properties "propagate" and fall into three categories: *tipping models*, in which a vertex adopts a behavior when a sufficiently large percentage of its neighbors adopt the behavior [179, **?**, 115]; *cascade models*, in which diffusion cascades across the network (cascade models have been developed for the SIR model of disease spread [10], marking photos

$\mathbf{T}_\Pi \uparrow i$	$\mathsf{hiv}(a)$	$\mathsf{hiv}(b)$	$\mathsf{hiv}(c)$	$\mathsf{hiv}(d)$
0	0	0	0	0
1	0	1	0	0
2	0.09	1	0.006	0.032
3	0.09	1	0.0081	0.032
4	0.09	1	0.0081	0.032

Tab. 2.1: Iterations of $\mathbf{T}_\Pi$.

as favorites in Flickr [**?**]); and *homophilic models*, in which vertices adopt behaviors on the basis of their intrinsic properties but not on the basis of the network structure [12]. Many existing diffusion models for a variety of phenomena can be expressed as GAPs. [183] shows that GAPs can represent many different diffusion models, including: the *Susceptible-Infectious-Removed* (SIR) model of disease spread [10], which is a classic disease model; the *Susceptible-Infectious-Susceptible* (SIS) model [110], which is a variant of the SIR model; the *Big Seed* marketing approach [200], which is a strategy of advertising to a large group of individuals who are likely to spread the advertisement further through network effects; models of diffusion of "favorited" pictures in Flickr [49]; tipping models like the Jackson-Yariv product diffusion model [115].

Example 2.1.5. *Suppose the SN in Figure 2.1 represents* cell phone users *and vertices have properties like* male, female, young, old, *and* adopter *telling us if the user adopted a cell phone plan. The phone company wants to identify important users. Suppose* d *is male and everyone else is female; initially nobody is an* adopter. *A cell phone provider may have a diffusion rule learned from past promotions:*

$$\mathsf{adopter}(V') : 0.6 \times X \times Y \leftarrow \mathsf{adopter}(V) : X \wedge \mathsf{male}(V) : Y \wedge \mathsf{fr}(V, V') : W$$

The vertex which has the greatest influence, if given a free mobile phone plan and if the above diffusion model is used, is clearly d *(because this is the only vertex that can "influence"* *others to adopt the plan). However, we see from the table in Example 2.1.1 that* d *is not the most relevant vertex w.r.t. to all centrality measures. It is also interesting to note that* c *and* d *have the same centrality w.r.t. all standard centrality measures (because their properties and the diffusion model are ignored).*

Other examples of diffusion models expressed with GAPs, such as a conditional probability model, the Jackson-Yariv and the SIR model, are reported in Section Ap.1.1, as they have been used in our experiments.

2.1.3 Diffusion Centrality

Diffusion centrality tries to measure how well a vertex v can diffuse a property p (e.g., the hiv property). Given an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a vertex predicate symbol p, and a vertex $v \in V$, the *insertion* of $p(v)$ into $\mathcal{S}$, denoted $\mathcal{S} \oplus p(v)$, is the SN $(V, E, \mathsf{VL}', \omega)$ where VL' is exactly like VL except that $\mathsf{VL}'(v) = \mathsf{VL}(v) \cup \{p\}$. In other words, inserting $p(v)$ into a social network merely says that vertex v has property p and that everything else about the network stays the same. Likewise, the *removal* of $p(v)$ from $\mathcal{S}$, denoted $\mathcal{S} \ominus p(v)$, is the social network $(V, E, \mathsf{VL}'', \omega)$ which is just like $\mathcal{S}$ except that $\mathsf{VL}''(v) = \mathsf{VL}(v) - \{p\}$.

Definition 4 (Diffusion Centrality)**.** *Let* $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ *be an SN,* Π *a GAP, and* p *a property. The* diffusion centrality *(DC for short) of a vertex* $v \in V$ *w.r.t.* Π*,* p*, and* $\mathcal{S}$*, denoted* $\mathsf{dc}_{\Pi,p,\mathcal{S}}(v)$*, is defined as follows:*

$$\sum_{v' \in V - \{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \oplus p(v)})(p(v')) \; - \sum_{v'' \in V - \{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \ominus p(v)})(p(v''))$$

Whenever Π, p, and $\mathcal{S}$ are clear from the context, we denote the diffusion centrality of a vertex v simply as $\mathsf{dc}(v)$.

This definition says that computing the diffusion centrality of vertex v involves two steps. First, we find the least fixed point of the diffusion model and the SN $\mathcal{S} \oplus p(v)$, i.e. we assume that vertex v has property p and see how much diffusion occurs. Then, we find the least fixed point of the diffusion model and the SN $\mathcal{S} \ominus p(v)$, i.e. we assume that vertex v does not have property p and see how much diffusion occurs. The difference between these two numbers captures the "impact" that would occur in terms of diffusion of property p if vertex v had property p.[2]

Example 2.1.6. *Consider again the HIV SN of Example 2.1.1 and the GAP of Example 2.1.2.2. Recall that the only vertex with property* hiv *is b. It can be easily verified that the values for the positive and negative summands of Definition 4 for all vertices are as reported in the following table.*

	a	b	c	d
Positive Summand	1.122	0.13	1.122	1.0981
Negative Summand	1.0401	0	1.122	1.0981
Diffusion Centrality	0.0819	0.13	0	0

Thus, b has the highest centrality w.r.t. hiv *and Π_{hiv}— classical centrality measures (Example 2.1.1) do not capture this because b is not a "central" vertex from a purely topological perspective. However, b should have the highest centrality because it is the only one with HIV. Vertices c and d do not increase the confidence of any vertex to have HIV. So their diffusion centrality is zero.*

Example 2.1.7. *If we return to the cell phone case (Example 2.1.5), we see that the DC of d*

[2] Considering just the first summation of Definition 4 is wrong. Suppose we have an SN and a vertex v s.t. the first summation of $\mathsf{dc}(v)$ is a high number N (i.e., the expected number of vertices that would have property p assuming that v has property p is N). Suppose that when we assume that v does not have property p, the same value N is determined (i.e., this is the value of the second summation). Then, intuitively, v should not have a high diffusion centrality since the expected number of vertices with property p is the same regardless of whether v has property p or not (hence v does not seem to play a central role in the diffusion of p). In contrast, considering just the first summation would give v a high centrality.

is 1.2, while all other vertices have 0 as their DC. Thus, d *has the highest diffusion centrality. Furthermore, as opposed to classical centrality metrics,* c *and* d *do not have the same centrality, because their properties and the diffusion of interest make them important to a different extent.*

Diffusion Centrality Problem (DCP). Given an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP Π, and a property p, the diffusion centrality problem consists of finding the DC (w.r.t. Π, p, and $\mathcal{S}$) of every vertex of $\mathcal{S}$.

2.1.4 The HyperDC Algorithm for Exact Diffusion Centrality Computation

In this section, we present the *HyperDC* algorithm (Section 2.1.4.3), which solves DCP exactly using the *HyperLFP* algorithm (Section 2.1.4.2) to compute the least fixed point of a GAP. We start with a set of optimization that can speed up HyperDC (Section 2.1.4.1).

2.1.4.1 Preprocessing Optimization Steps

When computing $\mathsf{dc}(v)$, we note that $\mathcal{S}$, $\mathcal{S} \oplus p(v)$, and $\mathcal{S} \ominus p(v)$ differ only in whether or not vertex v has property p. One way to leverage this is to first compute and cache $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})$ independent of v. We then only need to calculate one summation in order to compute $\mathsf{dc}(v)$.

Proposition 1. *Consider a social network* $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$*, a GAP* Π*, and a property* p*. Let* $\phi = \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})$ *and* v *be a vertex in* V*. Then,*

$$\mathsf{dc}(v) = \begin{cases} \sum_{v' \in V - \{v\}} \phi(p(v')) - \sum_{v'' \in V - \{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \ominus p(v)})(p(v'')) & \textit{if } p \in \mathsf{VL}(v) \\ \sum_{v' \in V - \{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \oplus p(v)})(p(v')) - \sum_{v'' \in V - \{v\}} \phi(p(v'')) & \textit{if } p \notin \mathsf{VL}(v) \end{cases}$$

Proof. Consider a vertex $v \in V$. If v has property p, then, by definition of $\mathcal{S} \oplus p(v)$, we have that $\phi = \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \oplus p(v)})$ and the first equation holds. Likewise, if v does not have the property p, then, by definition of $\mathcal{S} \ominus p(v)$, we have that $\phi = \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \ominus p(v)})$ and then the second equation also follows. □

p-monotonic GAPs, defined below, are a class of GAPs to which we can apply optimization techniques. The *dependency graph* of a GAP Π is a directed graph $dep(\Pi)$ whose vertices are the predicate symbols in Π. There is an edge from a predicate symbol q to a predicate symbol p iff there is a rule in Π where p occurs in the head and q occurs in the body. We say that q *can reach* p if there exists a path from q to p in $dep(\Pi)$. If q can reach p and vice versa, then we say that p and q are *mutually recursive*. We use $M_{\Pi,p}$ to denote the set of predicate symbols that are mutually recursive with p. We define $R_{\Pi,p}$ as the set of predicate symbols that can reach p and appear in the head of some rule of Π. Note that $M_{\Pi,p} \subseteq R_{\Pi,p}$.

Definition 5 (p-monotonic GAP)**.** *We say that a rule $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$ is* monotonic *iff μ_0 is a monotonic function*[3]*. Given a GAP Π and a property p, we say that Π is p*-monotonic *iff, for every rule r in Π, when the head predicate symbol is in $R_{\Pi,p}$ then r is monotonic.*

We now introduce the p-interfered predicate set—intuitively, it is the set of predicate symbols that may impact the values assigned to ground atoms of the form $p(v)$ in the least fixed point.

Definition 6 (p-interfered predicate set)**.** *Given a GAP Π and a property p, the p*-interfered predicate set *is defined as follows:*

$$I_{\Pi,p} = \begin{cases} M_{\Pi,p} & \textit{if } \Pi \textit{ is } p\textit{-monotonic} \\ R_{\Pi,p} & \textit{if } \Pi \textit{ is not } p\textit{-monotonic} \end{cases}$$

The GAP Π of Example 2.1.2.2 is hiv-monotonic with $I_{\Pi,hiv} = \{hiv\}$. The following GAP

[3] If μ_0 is a constant, it is considered to be monotonic. Moreover, if $\mu_0 = f(\ldots)$, then μ_0 is monotonic if f is monotonic, i.e., if $x_i \leq y_i$ for all $1 \leq i \leq arity(f)$, then $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$.

Π is *not* p*-monotonic* because of the second rule and $I_{\Pi,p} = \{p, q\}$.

$$p(V) : 0.5 \times X \times W \leftarrow \mathsf{fr}(V, V') : W \wedge p(V') : X$$

$$q(V) : (1 - X) \times W \leftarrow \mathsf{fr}(V, V') : W \wedge p(V') : X$$

$$p(V) : 0.9 \times X \times W \leftarrow \mathsf{fr}(V, V') : W \wedge q(V') : X$$

Given a GAP Π and a property p, we define

$$\begin{aligned} \Pi_p &= \{r \in \Pi \ \mid \text{the head predicate symbol of } r \text{ belongs to } R_{\Pi,p}\}, \\ \Pi_p^* &= \{r \in \Pi_p \mid \text{every predicate symbol } q \text{ in the body of } r \text{ is s.t. } q \notin I_{\Pi,p}\}. \end{aligned}$$

Proposition 2. *Consider an SN $\mathcal{S}$, a GAP Π, and a property p. Let $\psi = \mathsf{lfp}(\Pi_{\mathcal{S}} \cup \Pi_p^*)$. Then, $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})(p(v)) = \mathsf{lfp}((\Pi_p - \Pi_p^*) \cup \{A : \psi(A) \mid A \in \mathcal{A}\})(p(v))$ for every vertex v of $\mathcal{S}$.*

Proof. All the rules in $\Pi - \Pi_p$ have a predicate in the head atom that cannot reach predicate p, and then these rules do not affect the value of $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})(p(v))$. As we are interested in computing the diffusion centrality for property p, we can ignore these rules in the computation. Moreover, rules in Π_p can be partitioned into two sets of rules, namely Π_p^* and $(\Pi_p - \Pi_p^*)$, and by definition of Π_p^*, we have that rules in $(\Pi_p - \Pi_p^*)$ "depend on" rules in Π_p^* because an atom having predicate symbol $q \in I_{\Pi,p}$ may appear in the head of a rule in Π_p^* and in the body of a rule in $(\Pi_p - \Pi_p^*)$, but not vice versa. Thus, the rules in $(\Pi_p - \Pi_p^*)$ do not contribute to the computation of the least fixed point of the program Π_p^* and then the value of ψ can be pre-computed. $\square$.

To compute $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})(p(v))$, Proposition 2 says that *(i)* we do not need to consider rules in $\Pi - \Pi_p$ (they do not affect the value assigned to atoms of the form $p(v)$ in the least fixed point); and *(ii)* we can compute ψ as a preprocessing step, use it to define facts, and do not need to consider again the rules in Π_p^* (however, we can benefit from this only when Π is p-monotonic).

The last optimization step we propose is called *network filtering*, which consists of reducing the given SN by removing vertices (and the relative incoming and outgoing edges) that do not play any role in the diffusion process, i.e., those vertices that can never receive or transmit diffusive property p from/to other vertices in the SN via any rule in the considered diffusion model.

Example 2.1.8. *Consider the diffusion model Π_{hiv} from Example 2.1.2.2:*

$$\mathsf{hiv}(V) : 0.9 \times X \times W \leftarrow \mathsf{sp}(V, V') : W \wedge \mathsf{hiv}(V') : X$$

$$\mathsf{hiv}(V) : 0.4 \times X \times W \times W' \leftarrow \mathsf{fr}(V, V') : W \wedge \mathsf{sp}(V', V'') : W' \wedge \mathsf{hiv}(V'') : X$$

$$\mathsf{hiv}(V) : 0.6 \times X \times W \times W' \leftarrow \mathsf{sp}(V, V') : W \wedge \mathsf{sp}(V', V'') : W' \wedge \mathsf{hiv}(V'') : X$$

Suppose v is a vertex in an SN having no sp *relations. Moreover, suppose none of v's friends have* sp *relations. Then, v is an "unnecessary" vertex (for the purpose of computing DC) because there is no rule $r \in \Pi_{hiv}$ by which the vertex v can receive the property hiv or transmit hiv to other vertices.*

Given a GAP Π, a property p, and a rule $r \in \Pi$, we define $relbody(r) = \{AA \mid AA$ is an annotated atom in the body of r and its predicate symbol is not in $R_{\Pi,p}\}$. Vertex v of an SN $\mathcal{S}$ *activates* a rule $r \in \Pi$ iff there exists a ground rule $r' \in grd(r)$ such that:

1. for every $AA \in relbody(r')$, $\Pi_{\mathcal{S}} \models AA$;

2. if $A : \mu$ is the head of r', then $\mu > 0$; and

3. v appears in an annotated atom of the body of r'.

Definition 7 (Necessary and unnecessary vertices)**.** *Let $\mathcal{S}$ be an SN, Π be a GAP, and p a property. A vertex of $\mathcal{S}$ is* necessary *if it activates a rule of Π_p, otherwise it is* unnecessary.

The filtering of a social network eliminates all unnecessary vertices (along with their incoming/outgoing edges).

Definition 8 (Network Filtering)**.** *Let $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ be an SN, Π a GAP, p a property, and U the set of unnecessary vertices. The* filtering *of $\mathcal{S}$ (w.r.t. Π and p) is the SN $\mathcal{S}' = (V', E', \mathsf{VL}', \omega')$ where:*

1. $V' = V \setminus U$*;*

2. $E' = E \setminus \{\langle u, v, q\rangle \in E \mid u \in U \vee v \in U\}$*;*

3. $\mathsf{VL}'(v) = \mathsf{VL}(v)$ *for all* $v \in V'$*;*

4. $\omega'(e) = \omega(e)$ *for all* $e \in E'$*.*

The filtering of an SN is useful because unnecessary vertices have a DC of zero. The DC of necessary vertices can be computed on the (smaller) filtered SN.

Proposition 3. *Let $\mathcal{S}$ be an SN, Π be a GAP, p a property, and $\mathcal{S}'$ the filtering of $\mathcal{S}$. For every vertex v of $\mathcal{S}$,*

1. if v is unncessary, then $\mathsf{dc}_{\Pi,p,\mathcal{S}}(v) = 0$*;*

2. if v is necessary, then $\mathsf{dc}_{\Pi,p,\mathcal{S}}(v) = \mathsf{dc}_{\Pi,p,\mathcal{S}'}(v)$*.*

Proof. 1. If a vertex v is unnecessary, it cannot activate any rule in Π, thus, independently from the fact whether it has the diffusion property p or not, it does not give any contribution in diffusing p to other vertices (see item 2 of the definition of "activation"). It follows that

$\sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \oplus p(v)})(p(v')) = \sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \ominus p(v)})(p(v')) =$

$\sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})(p(v'))$

and then $\mathsf{dc}_{\Pi,p,\mathcal{S}}(v) = 0$.

2. If a vertex v is necessary, all the rules it activates contain necessary vertices, and then we have that

$\sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \oplus p(v)})(p(v')) = \sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}' \oplus p(v)})(p(v'))$,

$\sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S} \ominus p(v)})(p(v')) = \sum_{v' \in V-\{v\}} \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}' \ominus p(v)})(p(v'))$.

It follows that $\mathsf{dc}_{\Pi,p,\mathcal{S}}(v) = \mathsf{dc}_{\Pi,p,\mathcal{S}'}(v)$. ☐

Below we present another class of GAPs, called p-*dwindling*, for which our HyperLFP algorithm has a faster convergence to the least fixed point.

Definition 9 (p-dwindling GAP)**.** *Suppose p is a property. A GAP Π is p-*dwindling *iff for every ground rule $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \wedge \ldots \wedge A_n : \mu_n$ in $grd(\Pi)$ s.t. q is the predicate symbol of A_0 and $q \in I_{\Pi,p}$, it is the case that $\mu_0 \leq \mu_i$ for every $1 \leq i \leq n$ s.t. the predicate symbol of A_i is in $I_{\Pi,p}$.*

For the Flickr, Jackson-Yariv, and SIR models (see Ap.1.1) we consider in our experimental evaluation, we can state the following properties.

Proposition 4. *The Flickr model is p-monotonic and p-dwindling. The Jackson-Yariv model is p-monotonic but not p-dwindling. The SIR model is neither p-monotonic nor p-dwindling.*

Proof.

1. Clearly, the function $\mu_{v',v} \times \mu_p \times \mu_q \times dp_F$ in the head atom annotation of the unique rule of the Flickr model is monotonic, thus the GAP describing the Flickr model is p-monotonic. Moreover, the GAP is also p-dwindling because the value of the function $\mu_{v',v} \times \mu_p \times \mu_q \times dp_F$ is less than or equal to any value that μ_p can assume as the annotations $\mu_{v',v}$ and μ_q and the constant dp_F can assume only values between 0 and 1.

2. Also the GAP describing the Jackson-Yariv model is p-monotonic. In fact, the function $\frac{b_i}{c_i} \times r(\sum_j E_j) \times \frac{\sum_j w_j}{\sum_j E_j} \times w_q \times dp_{JY}$ in the head of its only rule is monotonic as the term $\frac{b_i}{c_i} \times r(\sum_j E_j) \times \frac{1}{\sum_j E_j} \times dp_{JY}$ is constant. However, the GAP describing the Jackson-Yariv model is not p-dwindling because of the term $\sum_j \omega_j$ in the head atom function.

3. The GAP describing the SIR model is not p-monotonic because of the terms $(1 - R)$ and $(1 - R')$ in the head atom annotation function of the first rule (which is $(1 - R) \times \mu^e_{v',v} \times$

$\mu^p_{v'} \times (1 - R') \times \mu^q_v \times dp_{_{SIR}})$. Moreover, the GAP is not p-dwindling. To see this, it is sufficient to note that, because of the presence of the terms $(1 - R)$ and $(1 - R')$ in the head atom annotation function of the first rule, we cannot say that the value assumed by this function is always less than or equal to R or R' (remember that $I_{\Pi_{SIR},p} = \{p, r_1, r_2\}$). $\square$

2.1.4.2 The HyperLFP Algorithm

In this section, we propose an efficient hypergraph-based algorithm, *HyperLFP*, to compute the least fixed point used for diffusion centrality computation. As HyperLFP uses hypergraphs, we first define hypergraphs.

Definition 10. *A directed hypergraph is a pair $\langle V, H\rangle$ where:*

1. *V is a finite set of vertices.*
2. *H is a finite set of directed hyperedges. A hyperedge is a pair $\langle S, t\rangle$ where S is a (possibly empty) set of vertices, called* source set*, and t is a vertex, called* target vertex*. Given a hyperedge $h \in H$ we use $S(h)$ to denote its source set and $t(h)$ to denote its target vertex.*

We now define a hypergraph that captures how a property p diffuses through an SN $\mathcal{S}$ according to a GAP Π. The hypergraph does not depend on which vertices have property p in the original SN, but depends only on Π and the structure of $\mathcal{S}$ in terms of edges and vertex properties other than p. *Therefore, given a GAP Π and an SN $\mathcal{S}$, the diffusion hypergraph has to be computed only once and can be used with different assignments of a property p to the vertices and edges of* $\mathcal{S}$. In addition, the hypergraph allows us to eliminate (ground) diffusion rules that are useless for the purpose of computing the least fixed point.

Definition 11 (Enabled Rule)**.** *Consider an SN $\mathcal{S}$, a GAP Π, and a property p. Let $\varphi = \mathsf{lfp}(\Pi_{\mathcal{S}} \cup \Pi^*_p)$. A rule $r \in grd(\Pi - \Pi^*_p)$ is* enabled *iff $\varphi(A) \geq \mu$ for every annotated atom $A : \mu$ in the body* *of r whose predicate symbol is not in $I_{\Pi,p}$.*

Intuitively, enabled rules are the ground rules that can affect the diffusion of p (directly or indirectly) in the least fixed point computation. Proposition 2 is applied in Definition 11 via the computation of $\varphi = \mathsf{lfp}(\Pi_{\mathcal{S}} \cup \Pi_p^*)$.

Example 2.1.9. *Consider the GAP Π_{hiv} of Example 2.1.2.2 and the SN of Example 2.1.1 (cf. Figure 2.1). The following ground instance of the second rule is enabled:* $\mathsf{hiv}(d) : 0.4 \times 0.08 \leftarrow \mathsf{fr}(d,a) : 0.8 \wedge \mathsf{sp}(a,c) : 0.1 \wedge \mathsf{hiv}(c) : 1$. *Notice that the atom* $\mathsf{hiv}(c)$ *does not play any role in determining whether or not the rule is enabled.*

Definition 12 (Diffusion Hypergraph $\mathcal{H}(\mathcal{S},\Pi,p)$)**.** *Consider an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP Π, and a property p. The hyperedge associated with a ground rule $r \in grd(\Pi)$ whose head annotated atom is of the form $p'(v) : \mu$ is defined as $\langle \{p''(v_i) \mid p''(v_i) : \mu_i$ is in the body of r and $p'' \in I_{\Pi,p}\}, p'(v)\rangle$ and is denoted by $hedge(r)$. The* diffusion hypergraph *$\mathcal{H}(\mathcal{S},\Pi,p)$ is a triple $\langle N, H, W\rangle$ such that:*

1. *$\langle N, H\rangle$ is a directed hypergraph with $N = \{q(v) \mid q \in I_{\Pi,p}$ and $v \in V\}$, and $H = \{hedge(r) \mid r$ is an enabled ground rule of Π whose head predicate symbol is in $I_{\Pi,p}\}$,*

2. *W is a function such that for each $h \in H$ and matrix $M[1...|I_{\Pi,p}|][1..|V|]$ of real values in $[0,1]$, $W(h, M)$ is the head annotation of the ground rule r satisfying the following two properties: (i) $hedge(r) = h$, and (ii) for every atom $q(v)$ appearing in S(h), $M[q][v]$ is equal to the annotation of $q(v)$ in r.*

Example 2.1.10. *Figure 2.2 shows the diffusion hypergraph for the GAP Π_{hiv} of Example 2.1.2.2 and the social network of Example 2.1.1.*

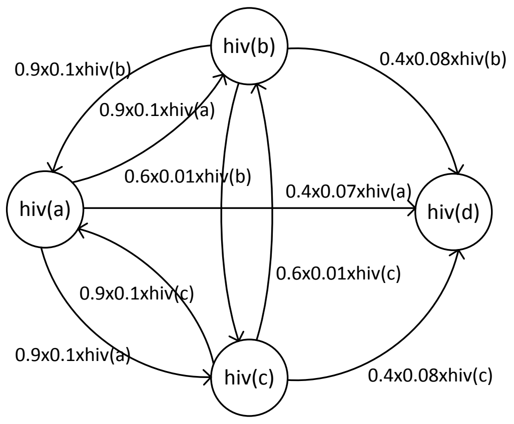


Fig. 2.2: A diffusion hypergraph.

The rough idea of the HyperLFP algorithm (Algorithm 1) is that hyperedges that propagate a value greater than zero are kept in a max-heap and those propagating higher values are visited first; the max-heap is updated as propagation unfolds.

For all $q \in I_{\Pi,p}$ and $v \in V$, $M[q][v]$ is the initial value of the ground atom $q(v)$ and is given as input, $U[q][v]$ keeps track of the hyperedges having $q(v)$ in their source set and is also given in input, and $M'[q][v]$ is the current assignment to $q(v)$. Specifically, $M'[q][v]$ is initially set to M and then iteratively updated by the algorithm. C keeps track of the highest values propagated by hyperedges that were added to $MaxHeap$. At each iteration of the **while** loop on lines 5–18, a pair $\langle h, w\rangle$ with maximum w is retrieved from $MaxHeap$. If $M'[q][v]$ is less than w, it is set to w, otherwise another hyperedge is retrieved from $MaxHeap$. If w is assigned to $M'[q][v]$, then hyperedges that can be affected by this are inspected (**for each** loop on lines 10–18). Only the hyperedges having $t(h)$ in the source set are inspected. For each of them, if no hyperedge added to $MaxHeap$ propagated a higher value (line 13), then if Π is p-monotonic the hyperedge is added to $MaxHeap$ (along with the value it propagates), otherwise it is added to $MaxHeap'$. The reason for this is that when Π is p-monotonic we can retrieve hyperedges in descending order of their weights even if their values were derived from different iterations of $\mathbf{T}_\Pi$. However, if Π is not p-monotonic, the iterations of $\mathbf{T}_\Pi$ must be executed one after the other without mixing the values derived at each of them. So, when Π is not p-monotonic, hyperedges are retrieved in descending order of their weight for each iteration of $\mathbf{T}_\Pi$. When $\mathit{MaxHeap}$ is empty, $\mathit{MaxHeap}'$ is assigned to $\mathit{MaxHeap}$. M' is returned if both heaps are empty.

If a GAP Π is p-dwindling and p-monotonic, then HyperLFP ensures that when a value w is assigned to a ground atom $q(v)$, then w is the final value for $q(v)$ in the least fixed point; hence the hyperedge that propagated w as well as any other hyperedge having $q(v)$ as a target atom no longer needs to be considered in order to see if a new higher value can be assigned to $q(v)$.

Proposition 5. *The worst-case time complexity of Algorithm HyperLFP is* $O(|N| + \frac{1}{\alpha} \cdot |H| \cdot (\log|H| + U_{max} \cdot (S_{max} + \log|H|)))$*, where* $U_{max} = \max_{v \in V, q \in I_{\Pi,p}}\{|\{h \mid h \in H \wedge q(v) \in S(h)\}|\}$*,* $S_{max} = \max_{h \in H}\{|S(h)|\}$*, and* α *is the minimum value obtained at line 9 for* $(w - M'[q][v])$.

Proof. The cost of making two copies of the matrix M at Lines 1-2 is $O(|N|)$. The the loop at Line 5 is executed $|H|$ times as at each iteration we remove an hyper-edge from $MaxHeap$ (Line 6). Within this loop, the predominant costs are the cost of deleting the maximum from $MaxHeap$ (Line 6), which is $O(\log|H|)$, and the cost of the loop at Line 10. This loop is executed for each hyper-edge $h' \in U[q][v]$ whose number is U_{max} in the worst-case and at each iteration the cost of computing the function W at Line 11 is $O(S_{max})$ in the worst-case, while the cost of adding $\langle h', w' \rangle$ either to $MaxHeap$ (Line 16) or $MaxHeap'$ (Line 18) is $O(\log|H|)$. Thus, the cost of executing Lines 5-19 is $O(|N| + |H| \cdot (log|H| + U_{max} \cdot S_{max}))$. Finally, the loop at Line 3 is executed $\frac{1}{\alpha}$ times in the worst-case as 1 is the maximum growth the annotation of an atom can have and α is the minimum increment step. Thus, the complexity of HyperLFP algorithm is

$O(|N| + \frac{1}{\alpha} \cdot |H| \cdot (\log |H| + U_{max} \cdot (S_{max} + \log |H|)))$. □

Algorithm 1 HyperLFP

```
Input: For a social network $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP $\Pi$, and a property $p$,
       two matrices $U[1...|I_{\Pi,p}|][1..|V|]$ and $M[1...|I_{\Pi,p}|][1..|V|]$,
       a max-heap $MaxHeap$, the diffusion hypergraph $\mathcal{H}(\mathcal{S}, \Pi, p) = \langle N, H, W \rangle$
Output: $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})(q(v))$ for all $v \in V$ and $q \in I_{\Pi,p}$
 1: $C \leftarrow$ copy of M;
 2: $M' \leftarrow$ copy of M;
 3: while $MaxHeap \neq \emptyset$ do
 4:     $MaxHeap' \leftarrow \emptyset$;
 5:     while $MaxHeap \neq \emptyset$ do
 6:         $\langle h, w \rangle \leftarrow deleteMax(MaxHeap)$;
 7:         Let $q(v) = t(h)$;
 8:         if $M'[q][v] < w$ then
 9:             $M'[q][v] \leftarrow w$;
10:             for each $h' \in U[q][v]$ do
11:                 $w' \leftarrow W(h', M')$;
12:                 Let $q'(v') = t(h')$;
13:                 if $w' > C[q'][v']$ then
14:                     $C[q'][v'] \leftarrow w'$;
15:                     if $\Pi$ is $p$-monotonic then
16:                         Add $\langle h', w' \rangle$ to $MaxHeap$;
17:                     else
18:                         Add $\langle h', w' \rangle$ to $MaxHeap'$;
19:     $MaxHeap \leftarrow MaxHeap'$;
20: return $M'$;
```

2.1.4.3 The HyperDC Algorithm

The HyperDC algorithm (Algorithm 2) initializes M_{init}, U, and $MaxHeap_{init}$ by calling Algorithm 3 (line 1), and then uses them to compute $\mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}})$ (lines 2–4). After that, the diffusion centrality of each vertex is computed (lines 6–25). Specifically, the value of atom $p(v)$ is incorporated into M_{init}, and $MaxHeap$ is updated accordingly (lines 7–16). Then, the least fixed point is computed using the updated M_{init} and $MaxHeap$ (lines 17–18). Finally, the diffusion centrality of vertex v is computed and the value of $p(v)$ in M_{init} is restored (lines 19–25). In this last step, Proposition 1 is leveraged. In fact, if $p \in \mathsf{VL}(v')$, the positive summand of the definition of DC has already been computed in lines 2–4 and what is being computed is the negative summand. In this case, the positive summand is equal to $sum_p - F[p][v]$ and the negative summand is equal to sum'_p, so the diffusion centrality of vertex v is $(sum_p - F[p][v]) - sum'_p$. If $p \notin \mathsf{VL}(v')$, the negative summand has been computed already in lines 2–4. In this case, the diffusion centrality of vertex v is $sum'_p - (sum_p - F[p][v])$.

Proposition 6. *The worst-case time complexity of Algorithm HyperDC is* $O(|V| \cdot (|N| + \frac{1}{\alpha} \cdot |H| \cdot (\log|H| + U_{max} \cdot (S_{max} + \log|H|))))$*, where* $U_{max} = \max_{v \in V}\{|\{h \mid h \in H \wedge v \in S(h)\}|\}$, $S_{max} = \max_{h \in H}\{|S(h)|\}$*, and* α *is defined as in Proposition 5.*

Proof. By leveraging Proposition 1, the HyperDC algorithm computes one least fix point for each node, so its worst-case time complexity is given by the number of vertices ($|V|$) times the worst-case time complexity of the HyperLFP algorithm (which is $O(|N| + \frac{1}{\alpha} \cdot |H| \cdot (\log|H| + U_{max} \cdot (S_{max} + \log|H|)))$) called at line 17. □

Algorithm 2 HyperDC

```
Input: An SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP $\Pi$, a property $p$,
       the diffusion hypergraph $\mathcal{H}(\mathcal{S}, \Pi, p) = \langle N, H, W \rangle$
Output: $\{\langle v, \mathsf{dc}(v) \rangle \mid v \in V\}$
 1: $\langle M_{init}, U, \mathit{MaxHeap}_{init} \rangle \leftarrow \mathsf{InitializeHyperDC}(\mathcal{S}, I_{\Pi,p}, \mathcal{H})$;
 2: $\mathit{MaxHeap} \leftarrow$ copy of $\mathit{MaxHeap}_{init}$;
 3: $F \leftarrow \mathsf{HyperLFP}(U, M_{init}, \mathit{MaxHeap}, \mathcal{H}(\mathcal{S}, \Pi, p))$ ;
 4: $sum_p \leftarrow \sum_{v \in V} F[p][v]$;
 5: $Result \leftarrow \emptyset$;
 6: for each $v \in V$ do
 7:     $\mathit{MaxHeap} \leftarrow$ copy of $\mathit{MaxHeap}_{init}$;
 8:     if $p \in \mathsf{VL}(v)$ then
 9:         $M_{init}[p][v] \leftarrow 0$;
10:     else
11:         $M_{init}[p][v] \leftarrow 1$;
12:     for each $h \in U[p][v]$ do
13:         remove $\langle h, w \rangle$ from $\mathit{MaxHeap}$;
14:         Let $p'(v') = t(h)$;
15:         if $W(h, M_{init}) > M_{init}[p'][v']$ then
16:             Add $\langle h, W(h, M_{init}) \rangle$ to $\mathit{MaxHeap}$;
17:     $M \leftarrow \mathsf{HyperLFP}(U, M_{init}, \mathit{MaxHeap}, \mathcal{H}(\mathcal{S}, \Pi, p))$;
18:     $sum'_p \leftarrow \sum_{v' \in V, v' \neq v} M[p][v']$;
19:     if $p \in \mathsf{VL}(v)$ then
20:         $\mathsf{dc}(v) \leftarrow (sum_p - F[p][v]) - sum'_p$;
21:         $M_{init}[p][v] \leftarrow 1$;
22:     else
23:         $\mathsf{dc}(v) \leftarrow sum'_p - (sum_p - F[p][v])$;
24:         $M_{init}[p][v] \leftarrow 0$;
25:     Add $\langle v, \mathsf{dc}(v) \rangle$ to $Result$;
26: return $Result$;
```

Algorithm 3 InitializeHyperDC

```
Input: A social network $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$,
       a $p$-interfered predicate set $I_{\Pi,p}$,
       a diffusion hypergraph $\mathcal{H} = \langle N, H, W \rangle$.
Output: $M[1..|I_{\Pi,p}|][1..|V|]$, $U[1..|I_{\Pi,p}|][1..|V|]$, $MaxHeap$
 1: $n = |I_{\Pi,p}|$; $m = |V|$;
 2: $M[1..n][1..m]$; $U[1..n][1..m]$; $MaxHeap \leftarrow \emptyset$;
 3: $M[q][v] \leftarrow 0$, $U[q][v] \leftarrow \emptyset$ for all $q \in I_{\Pi,p}$ and $v \in V$;
 4: for each $v \in V$ do
 5:     for each $q \in \mathsf{VL}(v) \cap I_{\Pi,p}$ do
 6:         $M[q][v] \leftarrow 1$;
 7: for each $h \in H$ do
 8:     Add $\langle h, W(h, M) \rangle$ to $MaxHeap$;
 9:     for each $q(v) \in S(h)$ do
10:         $U[q][v] \leftarrow U[q][v] \cup \{h\}$;
11: return $\langle M, U, MaxHeap \rangle$;
```

A brief note is in order about how the techniques in this section yield better scalability. Compared to our past work [123], HyperDC is approximately 100 times faster than that in [123]. In particular, the U and $MaxHeap_{init}$ structures used in HyperDC are built just once. This yields a speedup of approximately 5x. In addition, the filtering based on Proposition 3 yields a further speedup of 20x, leading to a total speedup of 100x.

2.1.5 *Experimental Evaluation*

This section contains a detailed report on our experiments. We compared the runtime and spread generated by diffusion centrality against classical centrality measures (Section 2.1.5.2), using networks with up to around 265K vertices and 440K edges. The networks used here were not as large as possible because some classical algorithms were unable to handle these sizes.

We implemented the HyperLFP (Algorithm 1) and HyperDC (Algorithm 2) in Java. To compute degree, eigenvector, PageRank, closeness, and betweenness centrality we used the Java Universal Network/Graph Framework (JUNG)[4]. All experiments were run on an Intel Xeon @ 2.40 GHz, 24 GB RAM.

2.1.5.1 Experimental Setup

Social Networks. Our experiments used several real-word social networks summarized in Table 2.2.

Network	Description	Type	# Vertices	# Edges	Avg. Degree	Density
BlogCatalog	friendship	undirected	10,312	333,983	64.78	6.28E-03
e-mail Enron	e-mail communications	undirected	36,692	183,831	10.02	2.73E-04
Douban	friendship	undirected	154,907	654,188	8.45	5.45E-05
wiki-Vote	Wikipedia vote relationships	directed	7,115	103,689	14.57	2.05E-03
soc-Epinions	Trust relationships	directed	75,879	508,837	6.71	8.84E-05
email-EuAll	e-mail communications	directed	265,214	420,045	1.58	5.97E-06

Tab. 2.2: Non-Game Social Networks used in the experiments.

We also considered an additional online game dataset called STEAM [22] which contains friendship relations (represented as directed edges) between players (vertices). Each player has several vertex properties and we selected country, group(s), games played and total time played per game. We extracted 10 subnetworks from the whole STEAM dataset by choosing different games and selecting, for each game, all players who played and all edges among the players. The features of the extracted networks are reported in Table 2.3.

Diffusion Models. We ran experiments with a conditional probability model (hereafter referred to as the “Flickr model” [**?**]), the Jackson-Yariv tipping model [115], and the SIR (susceptible, infec-

[4] jung.sourceforge.net

Social Network	# Vertices	# Edges	Avg. Degree	Density
GAME8690	4,083	21,447	5.25	1.29E-03
GAME50510	28,872	115,254	3.99	1.38E-04
GAME6850	52,879	164,702	3.11	5.89E-05
GAME1500	66,571	303,240	4.56	6.84E-05
GAME24420	82,377	437,554	5.31	6.45E-05
GAME11450	89,942	327,258	3.64	4.05E-05
GAME17300	122,467	441,657	3.61	2.94E-05
GAME420	1,307,335	12,431,564	9.51	7.27E-06
GAME220	2,030,579	20,596,331	10.14	5.00E-06

Tab. 2.3: STEAM networks used in the experiments.

tious, removed) model of disease spread [10]. [184] has shown that these three diffusion models can be expressed as GAPs (along with many others). We generalized these diffusion models by adding an additional condition $q(u) : \mu$ in rule bodies. When all vertices have the property q, then we have the original diffusion models. The q-condition determines when the diffusion process can happen. In our experiments, we compared DC with other classical centrality measures by varying the percentage of vertices in the network with property q. The diffusion models are reported in Section Ap.1.1, while Table 2.4 summarizes their main features (see Proposition 4).

Model	$I_{\Pi,p}$	p-monotonic	p-dwindling
Flickr model	$\{p\}$	yes	yes
Jackson-Yariv model	$\{p\}$	yes	no
SIR model	$\{p, r_1, r_2\}$	no	no

Tab. 2.4: Features of the Flickr, Jackson-Yariv, and SIR models.

2.1.5.2 Diffusion Centrality vs. Classical Centrality Measures

As described earlier, the Flickr, Jackson-Yariv, and SIR diffusion models assume that some set of vertices in the network have property p and that only vertices satisfying some property q can spread p. For our STEAM data, we were able to use known properties of the vertices for q but we were not able to do this for the other data sets. We therefore report our results comparing DC with classical measures in two subsections.

Before getting into the details of the very extensive experiments we conducted comparing diffusion centrality with classical centrality measures, we summarize the high level conclusion of these comparative experiments.

1. The runtime of HyperDC is significantly better than betweenness and closeness centrality and comparable with the others.

2. The spread achieved by diffusion centrality is almost always significantly better than those achieved by classical centrality measures.

2.1.5.3 STEAM Data

We used all but the last two (very large) STEAM data sets (Table 2.3) for the comparative analysis as some classical centrality measures could not complete even on the first seven STEAM data sets. Consistently with the data, we set the percentage δ_p of vertices that *initially* have property p to 0% and varied the percentage δ_q of vertices having property q by using real properties of vertices in the STEAM data. In the STEAM data, q was taken to be the playing time of users in the game and was assigned as follows: we sorted the players in descending order according to playing time and assigned q to the first δ_q users. We varied $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$. The last two STEAM subnetworks, which are substantially larger, were used for the evaluation of the CBAF algorithm (cf. Section 3.2.4).

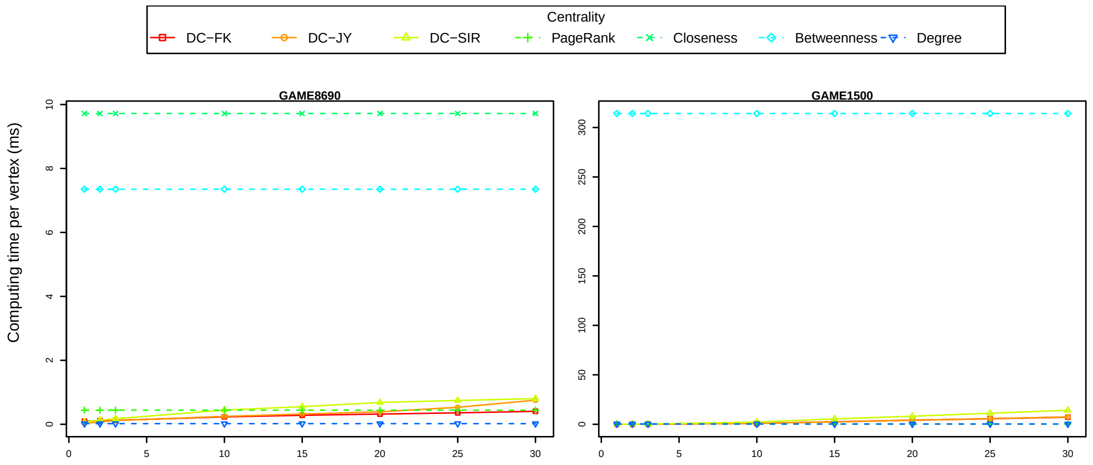
Runtime. We compared the time to compute DC w.r.t. the three diffusion models against the time to compute classical centrality measures. Figure 2.3a shows how the runtimes vary w.r.t. δ_q for two representative STEAM games (Ap.1.2 shows all the games). Of the 7 STEAM games we tested in this experiment, closeness centrality computation finished only in the smallest case (GAME8690)—this is shown in Figure 2.3. HyperDC's runtime is much faster than betweenness and closeness centrality for all the STEAM subnetworks and for all values of δ_q. Even when $\delta_q = 30\%$, the runtime of diffusion centrality is still low.

Figure 2.3b zooms in on the cases where δ_q varies from 1% to 5% and shows the runtimes for HyperDC, degree centrality, and PageRank. HyperDC is faster than PageRank over networks GAME8690 and GAME50510, and also faster than degree over network GAME8690.[5] However, when the number of vertices increases, HyperDC becomes slower than PageRank and degree for all three diffusion models. For instance, we see in GAME1500 (Figure 2.3b) that a crossover occurs when $\delta_q = 3\%$ for the SIR model but not for the Jackson-Yariv model. In fact, across the 7 STEAM games, there are some cases when HyperDC is faster than degree centrality and PageRank and some cases where the opposite is true with no clear discernible pattern.

We also checked p-values of T-test with the runtimes of HyperDC and other centrality. The values are shown in Appendix Ap.2. The results show that HyperDC runtime is significantly different from the runtime of Betweenness centrality for various δ_q. In most cases, the p-values are less than 0.05 except three cases for Jacson-Yariv model when δ_q is 20, 25 and 30% (The p-values for the cases are $0.058, 0.065$ and 0.069.). The results for PageRank and degree centrality

[5] Though it may seem surprising that HyperDC sometimes beats degree centrality, the reason for this is that when δ_q is low (below 3%), HyperDC only needs to compute diffusion centrality for a smal number of vertices. However, when δ_q is larger, this is not the case. Likewise, PageRank computation can also be slower than HyperDC when the network density is high.

(a) STEAM Data: Runtime of HyperDC algorithm for Exact Diffusion Centrality computation with Flickr, Jackson-Yariv, and SIR diffusion models compared with Classical Centrality Measures Averaged Per Vertex when $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$.



(b) STEAM Data: Runtime of HyperDC algorithm for Exact Diffusion Centrality computation with Flickr, Jackson-Yariv, and SIR diffusion models compared with Classical Centrality Measures Averaged Per Vertex when $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%\}$. Detailed view of part (a).

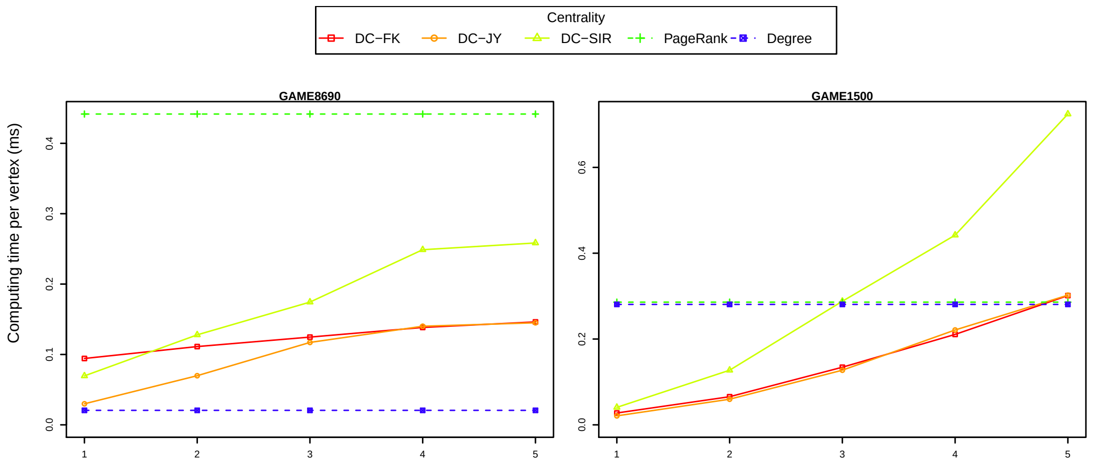


Fig. 2.3: STEAM Data Runtime

are interesting. For Flickr and SIR models, the general trend is similar. When δ_q is small, 1% (and 2%), and relatively high ($\geq 10\%$), the difference between HyperDC runtime and the runtime of the two centrality measures is significant. But when δ_q is $3-5\%$ for Flickr model and $2-3\%$ for SIR model, the p-values show that the difference is not significant. For Jackson-Yariv model, the p-values show that the difference is not significant in all vairous δ_q.

Runtime of HyperDC Tends to be Linear. We note from Proposition 6 that in the worst case, the complexity of HyperDC is nonlinear. In order to assess actual runtime characteristics of HyperDC, we ran an experiment on the STEAM data when the number of vertices increases and as δ_q varies. We used the networks from GAME 50510, 6850, 11450, and 17300 which all have a similar average degree (in the 3-4 range). Figure 2.4 shows the number of vertices in the network on the x-axis and the average runtime per vertex on the y-axis. Different curves show varying values of δ_q. Each upper-lower pair of graphs uses a different diffusion model (Flickr, Jackson-Yariv, and SIR). The upper figure shows δ_q ranging from 1-5% while the lower figure shows δ_q ranging from 5-30%. We see that irrespective of the diffusion model used and the value of δ_q, HyperDC runs in linear time in practice.

Spread. In this section, we compare the diffusion of property p when we choose the top-k central vertices according to HyperDC vs. using classical centrality measures on the STEAM data. In each case, the top-k vertices are called *seeds*. We vary k from 10 to 100 in steps of 10. The *spread* w.r.t. a given set of seeds is the expected number of vertices with property p (after diffusion) assuming the seeds have property p minus the expected number of vertices with property p (after diffusion) in the original social network. This difference is normalized.

Our spread experiments are presented in two ways. In the first, we show how spread varies by averaging over the number of seeds for fixed δ_q values. In the second, we do the opposite.

Spread Experiments Averaged over Varying k values for specific δ_q values. We varied δ_q over the set $\{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$. For each selection of δ_q, we considered every value of k from the set $\{10, 20, \ldots, 100\}$. Then for a fixed $\delta_q, k,$ *Diffusion-Model* triple,

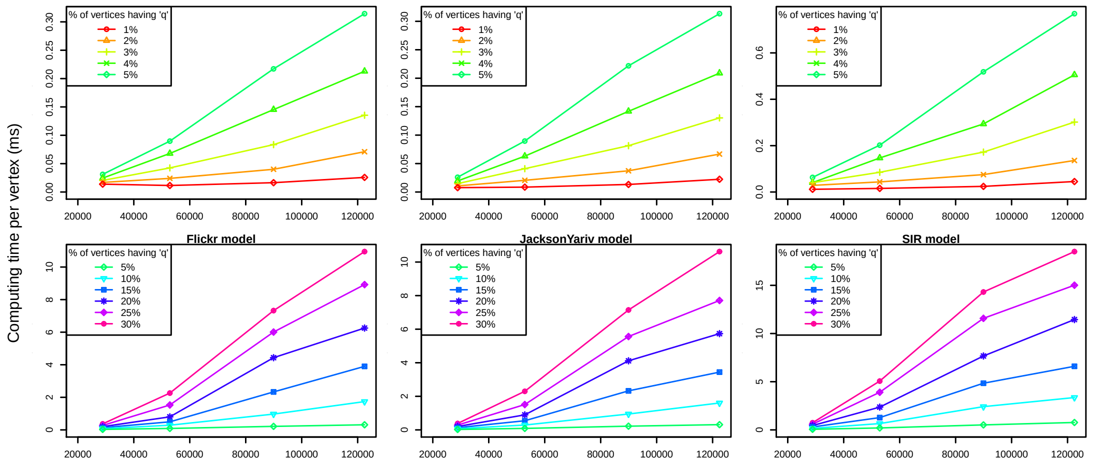


Fig. 2.4: [

STEAM Data: Average Runtime (per vertex) of HyperDC]STEAM Data: Average Runtime (per vertex) of HyperDC as the number of vertices is increased as as δ_q is varied using Flickr, Jackson-Yariv, and SIR models. The figure shows that in practice, HyperDC runs in linear time, irrespective of the value of δ_q and the diffusion model used.

we computed the ratio of the spread using HyperDC and the best spread achieved by any of the classical centrality measures. Table 2.5 reports the average of these ratios. Thus, any ratio greater than 1 in Table 2.5 shows that HyperDC achieves a higher spread than all of the classical centrality measures.

For the Flickr and Jackson-Yariv diffusion models, HyperDC always achieved a better spread than all classical centrality measures. In the case of the SIR model, on average, HyperDC achieves better spreads than classical centrality measures as long as $\delta_q \leq 15\%$ — at 20%, they are even, and at 25-30%, classical centrality measures achieve a better spread.

Not surprisingly, this spread ratio decreases as δ_q increases because when δ_q is large, diffusion occurs due to network structure rather than due to diffusion rules.

Interestingly, spread ratios for the Flickr model are very large compared to those for the Jackson-Yariv and SIR models. This is because the expected number of vertices which have

Tab. 2.5: Spread over STEAM networks for different δ_q values.

		STEAM Data: Average ratio of the spread generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different δ_q.									
Model	Network	1%	2%	3%	4%	5%	10%	15%	20%	25%	30%
FKModel	GAME8690	19.4	2.4	2.0	1.8	1.9	1.9	1.6	1.5	1.4	1.3
	GAME50510	18.1	22.5	23.8	26.5	19.5	5.0	3.6	3.0	2.7	2.4
	GAME6850	436.6	654.2	11.6	10.4	10.7	8.1	6.4	5.6	4.1	3.2
	GAME1500	677.6	89.9	48.3	49.9	46.6	48.8	50.0	56.2	55.5	3.1
	GAME24420	125.6	6.5	6.1	6.3	5.7	4.9	3.1	2.3	1.9	1.7
	GAME11450	512.0	585.5	804.3	8.9	9.7	6.9	5.5	5.1	4.9	3.7
	GAME17300	288.7	363.5	6.6	6.7	6.8	5.4	5.1	4.4	3.7	2.8
	Average	296.9	246.4	129.0	15.8	14.4	11.6	10.8	11.2	10.6	2.6
JYModel	GAME8690	2.8	1.7	1.4	1.2	1.3	1.2	1.1	1.1	1.1	1.2
	GAME50510	11.0	5.0	5.1	4.3	4.0	2.7	1.9	1.5	1.1	1.0
	GAME6850	8.1	5.8	4.9	4.5	3.6	2.5	2.1	1.8	1.7	1.6
	GAME1500	8.8	6.7	4.6	3.7	3.6	2.3	2.0	1.9	1.7	0.9
	GAME24420	5.4	3.3	2.8	2.6	2.2	1.6	1.4	1.3	1.3	1.3
	GAME11450	10.9	7.7	6.2	5.1	4.2	2.9	2.2	1.9	1.7	1.6
	GAME17300	9.0	4.5	2.4	2.2	2.2	1.6	1.2	1.2	1.2	1.2
	Average	8.0	5.0	3.9	3.4	3.0	2.1	1.7	1.5	1.4	1.3
SIRModel	GAME8690	1.2	0.8	0.8	0.8	0.8	0.8	0.9	0.9	0.8	0.8
	GAME50510	10.1	2.6	3.0	2.2	2.2	1.4	1.1	1.0	0.9	0.9
	GAME6850	4.9	3.6	2.8	2.6	2.5	1.8	1.7	1.4	1.3	1.2
	GAME1500	5.7	3.3	2.2	2.4	2.1	1.7	1.2	1.1	1.0	0.8
	GAME24420	2.0	1.4	1.5	1.5	1.4	1.1	0.8	0.8	0.8	0.8
	GAME11450	6.7	4.4	2.7	1.7	1.8	1.4	1.1	1.1	1.0	0.9
	GAME17300	4.3	2.1	1.0	1.1	1.2	0.7	0.6	0.6	0.6	0.6
	Average	5.0	2.6	2.0	1.8	1.7	1.3	1.1	1.0	0.9	0.9

property p after diffusion is much higher in the case of the Flickr model than in the other two cases.

Spread Experiments Averaged over Varying δ_q values for specific k values. Here we selected values of k as before and averaged over different possible values of δ_q drawn from the set $\{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$. Table 2.6 summarizes the results. As in the previous case, a ratio exceeding 1 implies that HyperDC outperforms all classical centrality measures.

On average, for all possible values of k and for all three diffusion models, HyperDC achieves a better spread than all the classical centrality measures. In fact, with a few exceptions

Tab. 2.6: [

STEAM Data: Average ratio of spread]STEAM Data: Average ratio of the spread generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different k.

		Average ratio of the DC spread to the best spread of other centrality measures for different k.									
Model	Network	10	20	30	40	50	60	70	80	90	100
FlickrModel	GAME8690	16.5	2.1	2.2	2.1	2.1	2.0	2.0	2.1	2.1	2.1
	GAME50510	48.1	6.8	9.0	10.5	7.7	8.6	9.2	9.8	10.3	7.2
	GAME6850	41.4	67.3	86.1	101.2	114.9	127.9	138.5	149.6	157.8	166.2
	GAME1500	364.0	94.3	58.4	67.8	74.2	81.0	88.3	93.5	99.9	104.7
	GAME24420	48.2	72.2	4.4	5.2	5.5	5.7	5.5	5.7	6.0	5.7
	GAME11450	70.9	116.4	155.1	187.7	216.8	241.5	265.2	287.8	310.0	95.3
	GAME17300	63.3	101.0	134.8	164.0	187.8	8.0	8.0	8.5	8.9	9.4
	Average	93.2	65.7	64.3	76.9	87.0	67.8	73.8	79.6	85.0	55.8
JYModel	GAME8690	1.9	1.5	1.3	1.2	1.3	1.4	1.4	1.4	1.4	1.4
	GAME50510	3.3	3.6	3.9	3.7	4.1	3.7	3.9	4.0	3.7	3.8
	GAME6850	2.6	3.3	3.7	3.7	3.8	3.5	3.8	4.0	4.2	4.0
	GAME1500	3.1	3.3	3.9	3.9	4.0	3.7	3.7	3.6	3.5	3.5
	GAME24420	2.2	2.2	2.2	2.3	2.4	2.4	2.5	2.4	2.4	2.3
	GAME11450	3.0	4.3	4.7	4.5	4.7	4.5	4.6	4.8	4.8	4.6
	GAME17300	2.1	1.9	2.5	2.8	2.8	2.8	3.0	3.0	2.9	3.0
	Average	2.6	2.9	3.2	3.2	3.3	3.1	3.3	3.3	3.3	3.2
SIRModel	GAME8690	1.2	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.9	0.9
	GAME50510	2.1	2.0	2.4	2.4	2.8	2.5	2.7	3.0	2.7	2.7
	GAME6850	2.2	2.4	2.5	2.4	2.4	2.2	2.3	2.5	2.5	2.4
	GAME1500	2.5	2.4	2.6	2.2	2.2	2.1	2.1	1.9	1.8	1.8
	GAME24420	1.4	1.4	1.4	1.3	1.3	1.1	1.1	1.1	1.0	1.0
	GAME11450	2.1	2.8	2.4	2.3	2.2	2.2	2.1	2.2	2.2	2.2
	GAME17300	1.3	1.2	1.3	1.4	1.3	1.2	1.3	1.3	1.3	1.4
	Average	1.8	1.9	1.9	1.8	1.9	1.7	1.8	1.8	1.8	1.8

in the SIR model, HyperDC achieves a better spread than all centrality measures.

Interestingly, the spread ratio for GAME8690 is consistently the lowest according to each of three diffusion models and in each setting of both Tables 2.5 and 2.6. This game has just 4083 vertices (so it is very small and dense). Our choice of q is based on the amount of playing time of a player and there is strong correlation between that and the number of friends. As a consequence, having multiple seeds does not greatly increase diffusion of property p because of overlaps between the vertices that may be influenced by the seeds.

2.1.5.4 Non-Game Social Network Data

In this section, we perform experiments similar to those reported above on the networks in Table 2.2. For these networks, the data did not have associated vertex properties. We looked at two cases.

Case 1. We randomly selected $\delta_p = 0.1\%$ of the vertices in the network to have a synthetic property p (in 5 different runs). In each run, we varied $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$ and selected $\delta_q\%$ of the vertices to have a synthetic property q.

Case 2. We randomly selected $\delta_q = 3\%$ of the vertices and varied $\delta_p \in \{0.1\%, 0.2\%, 0.3\%, 0.4\%, 0.5\%\}$ and associated synthetic properties p and q with vertices as above.

Runtime. Figure 2.5a shows how runtime varies w.r.t. δ_q for Case 1 above. The networks are sorted from left to right according to the number of vertices, with directed networks in the first row and undirected networks in the second row. As in the case of our experiments with the STEAM data, closeness and betweenness centrality are time consuming compared to the other algorithms (including HyperDC). Figure 2.5b zooms in on the cases where δ_q varies from 1% to 5% (with closeness and betweenness centrality removed as they are too slow).

HyperDC with the Flickr model is faster than PageRank in the majority of cases considered, and sometimes faster than eigenvector centrality. HyperDC with the SIR model is faster than PageRank for directed networks when $\delta_q \leq 4\%$. As in the case of the STEAM data described earlier, this is because the network filtering step can eliminate many vertices. For the Flickr and SIR models, the runtimes increase slightly as δ_q increases because of the higher diffusion that takes place. However, as in the case of the STEAM data, HyperDC with the Jackson-Yariv model often takes the most time.

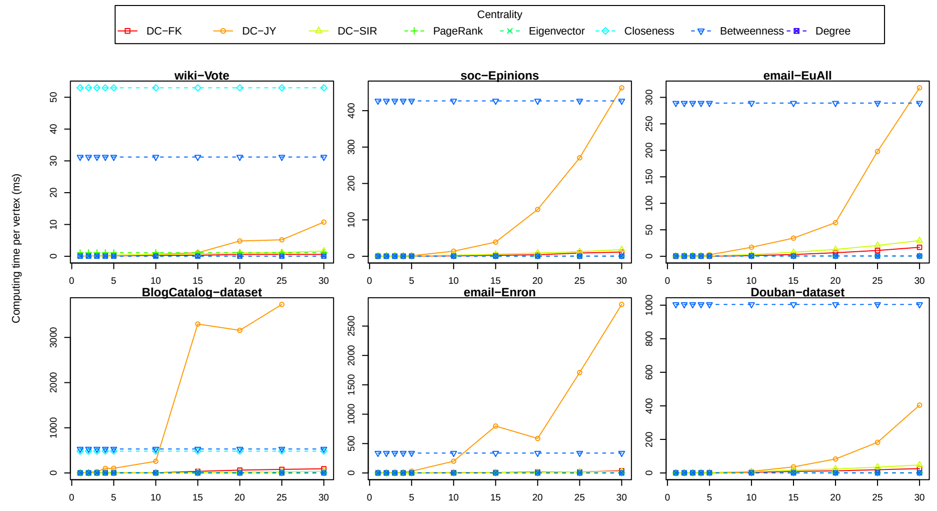
Figure 2.6 shows how runtime varies as δ_p varies (*Case 2*). We note that the runtime for closeness and betweenness centrality are very high (worse by 1 to 3 orders of magnitude) and hence we do not report runtimes for them. HyperDC's runtime for the Flickr and SIR models do not vary much with δ_p. HyperDC with the Flickr is faster than Pagerank in almost all networks and settings. Compared to degree centrality, HyperDC with Flickr exhibits competitive runtimes, being faster in about half the settings considered and comparable or slightly worse in the others. HyperDC with the SIR is faster than Pagerank in all data sets except the Douban data set. As in the case of the STEAM data, HyperDC with the Jackson-Yariv is the worst performer w.r.t. runtime (excluding betweenness and closeness centrality which we eliminated earlier due to their very poor performance).

Spread. In both Cases 1 and 2, on average, the ratio of spread generated by HyperDC to the spread generated by the best classical measure exceeds 1. As in the case of the STEAM data, the best ratios are for the Flickr model. Due to space constraints we show 4 tables in Ap.1.3.

2.1.5.5 Summary of Experiments

In this section, we conducted two experiments. The first compares HyperDC (which exactly computes diffusion centrality) with classical centrality measures. The experiment conclusively shows that: The runtime of HyperDC is significantly better than betweenness and closeness centralities and comparable with PageRank, eigenvector centrality, and degree centrality. The spread achieved by diffusion centrality is almost always significantly better than those achieved by classical centrality measures.

(a) Non-Game Social Network Data: Runtime (ms) per vertex when varying δ_q from 1% to 30% and $\delta_p = 0.1\%$.



(b) Non-Game Social Network Data: Runtime (ms) per vertex when varying δ_q from 1% to 5% and $\delta_p = 0.1\%$. Zoomed in view of Figure 2.5a.

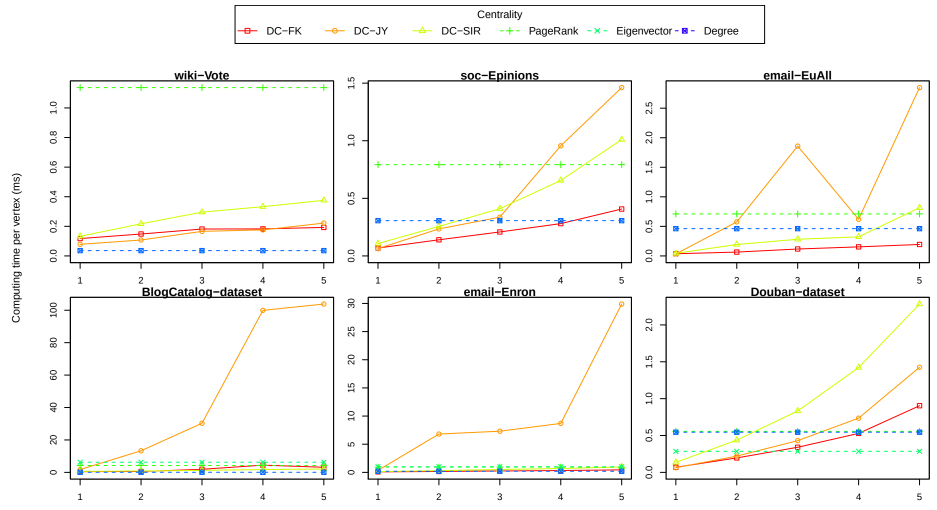


Fig. 2.5: Runtime (ms) per vertex vs. δ_q.

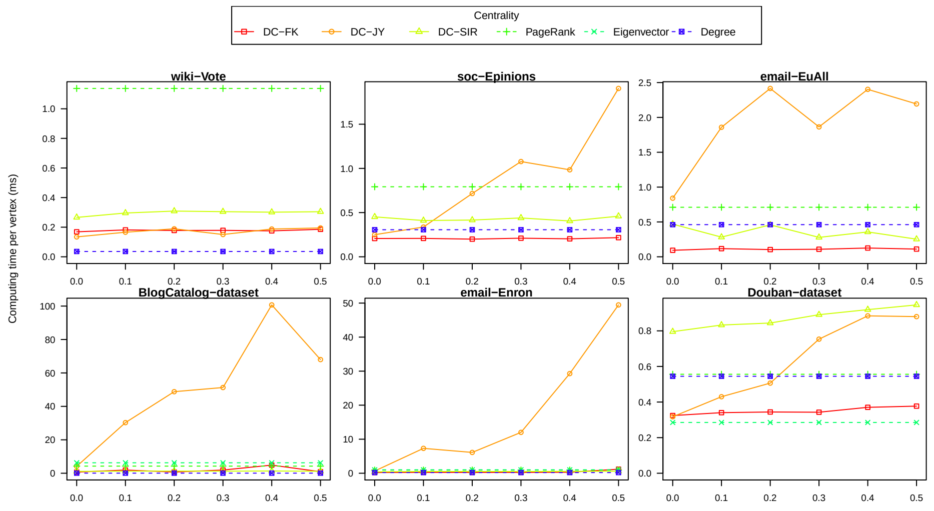


Fig. 2.6: [

Non-Game Social Network Data: Runtime (ms) per vertex when varying δ_p]Non-Game Social Network Data: Runtime (ms) per vertex when varying δ_p from 0.1% to 0.5% with $\delta_q = 3\%$.

2.2 Covertness Centrality

2.2.1 Introduction

There are certainly several important applications where users want to have low centrality according to classical centrality measures, but still retain the ability to communicate with a given set of vertices. For example: (i) Terrorists or criminals communicating through online social networks may need to communicate with a certain set of vertices, i.e. their terror or criminal network, while appear "common", i.e. looking similar to many other vertices in the network so that they cannot be easily distinguished from other vertices. (ii) A booming industry is in online marketing on platforms like Twitter where companies offer to "insert" certain types of messaging into the network for a client. In such cases, the marketers want to get the message out without "standing" out as spreaders of a marketing message. We call actors who wish to satisfy (i) and (ii) *covert actors*, i.e. actors who want to communicate well with certain other actors, but who wish to stay "below the

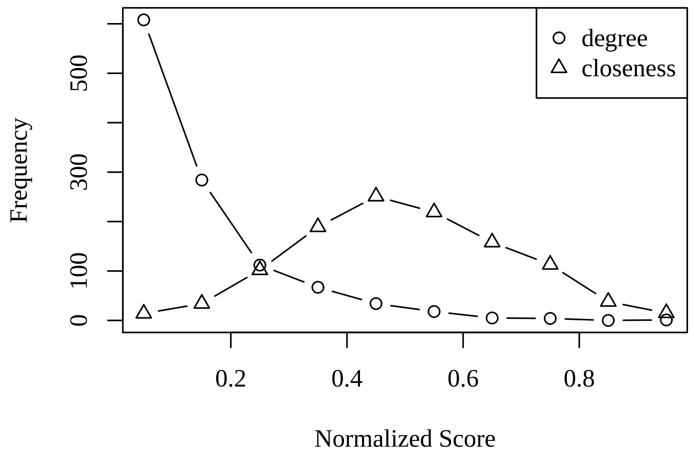


Fig. 2.7: [

]Histograms of the degree and closeness centrality scores (normalized to the interval [0,1]) for the URV dataset (see Table 2.7).

radar."

Measuring which network positions are "not common" and which ones are "common" is not straightforward. In most social networks there are a lot more vertices with a low degree than vertices with a high degree, as most social networks are known to be scale-free. But low values are not always dominating. Closeness centrality scores are usually normally distributed (or follow at least another bell shaped distribution) as shown in Fig. 2.7 [6]. So it takes more to measure how well a vertex hides in the crowd than looked at how low its centrality scores are.

In this section, we define a new centrality measure, denoted as *Covertness Centrality* (CC for short), that captures the intuitions. Unlike classical centrality measures which start solely with a graph as input, CC assumes that we are given as input, both a graph *and* a set $\mathcal{C}$ of classical centrality measures. Intuitively, think of $\mathcal{C}$ as being a set of centrality measures that a covert actor thinks an adversary might use to identify key players in a network. We split the definition of CC

[6] These distributions for the respective centrality measures hold true for all of our evaluation networks listed in Tab. 2.7

into two parts - how “common” a vertex is in a network w.r.t. given $\mathcal{C}$ and how well the vertex can communicate with a given set of nodes in the network according to a centrality measure.

Section 2.2 is organized as follows: In Section 2.2.2, we discuss requirements that a definition of “common-ness” should be satisfied as well as possible ways to define covertness centrality. Next, in Section 2.2.3 and 2.2.4, we propose sampling methods to compute covertness centrality and evaluate the proposed covertness centrality measures and algorithms.

2.2.2 Covertness Centrality Definition

Our notion of covertness centrality is a combination of commonness and communication potential. Commonness means hiding in a crowd of equal or similar actors and communication potential is the ability to efficiently communicate in order to achieve the objective of the group. In this section, we will discuss how to construct a covertness centrality measure from this two components.

2.2.2.1 Commonness

As introduced, commonness should measure how well an actor hides in a crowd of similar actors. Let $CM(a)$ denote the (as yet undefined) commonness of an actor a. Intuitively, we want $CM(a) > CM(b)$ if and only if knowledge of properties of the structural position of actor a reveals less information about a than knowledge of properties of the structural position of actor b reveals on b. This means, that if the size of the crowd actor a is hiding in is larger than the crowd actor b is hiding in, a’s covertness centrality has to be higher.

In terms of conditional probability we can describe what we want to measure as follows. If we want to search for an actor a and we know that this actor has the property I, what is the probability that a randomly picked actor with property I is actor a? In this work, we are focusing solely on network data. That means, no other information on actors are available than the edges connecting them. Therefore, the possible information on a person are properties of his or her position in the network. Moreover, we assume that this information is encoded via a user-specified set $\mathcal{C}$ of centrality measures.

Before we can give a formal definition of covertness centrality, we need to introduce some definitions. Let $G = (V, E)$ be a graph where V denotes a set of vertices and $E \subset V \times V$ and set of edges connecting pairs of vertices. Furthermore, let $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ be an ordered set of centrality measures on graph G where C_i is a function from V to $\mathbb{R}$.

We can regard an actor w as similar to an actor v with respect to a centrality measure C_i, when $C_i(w)$ and $C_i(v)$ are "similar". We define what values of C_i can be regarded as similar based on the variance of the values of the centrality measure. Let σ_i be the standard deviation of centrality scores $C_i(v)$ where $v \in V$. Then, the k-dimensional interval of centrality values similar to the one of vertex v (v's similarity neighborhood) is

$$\begin{aligned} I_v = {} & [C_1(v) - \alpha\sigma_1, C_1(v) + \alpha\sigma_1] \times \\ & [C_2(v) - \alpha\sigma_2, C_2(v) + \alpha\sigma_2] \times \\ & \ldots \times \\ & [C_k(v) - \alpha\sigma_k, C_k(v) + \alpha\sigma_k] \,. \end{aligned}$$

where $\alpha \geq 0$ is a parameter that defines the range of similar values. A user can define what α should be.

Furthermore, let $X(v) = (C_1(v), C_2(v), \ldots, C_k(v))$ be the vector of centrality measures for vertex v and let $\sigma(v) = (\sigma_1(v), \sigma_2(v), \ldots, \sigma_k(v))$ be the vector of the standard deviations for all centrality measures in $\mathcal{C}$.

As we have a finite set of actors, the distinct centrality vectors that are similar to a vertex v

w.r.t. $\mathcal{C}$ are

$$D_{\mathcal{C}}(v) = \{X | C_i(w) = X_i, i \in \{1..k\}, w \in V \wedge X \in I_v\}.$$

The probability that a randomly chosen vertex has the centrality vector X is

$$\begin{aligned} f_{\mathcal{C}}(X) =& P(C_1(v) = X_1 \wedge \ldots \wedge C_k(v) = X_k) \\ =& \frac{|\{v | C_1(v) = X_1 \ldots \wedge C_k(v) = X_k\}|}{|V|}. \end{aligned}$$

However, in our context, we are not interested in the probability that a randomly chosen vertex has a specific centrality vector but in the probability that some other randomly chosen vertex has the same centrality vector X as a given vertex. So we define

$$g_{\mathcal{C}}(X) = \max \left\{ \frac{|\{v | C_1(v) = X_1 \ldots \wedge C_k(v) = X_k\}| - 1}{|V| - 1}, 0 \right\}.$$

Given these definitions, we can define the commonness $CM(v)$ of a vertex $v \in V$ with respect to a non-empty set $\mathcal{C}$ of base centrality measures in various ways. However, it is desirable (though not strictly necessary) that all specific definitions of CM satisfy certain properties.

Property 1. *Optimal Hiding: If all vertices are equal w.r.t. all centrality measures, the vertices are indistinguishable and so the hiding is optimal. In this case the commonness has to be* 1 *for all vertices.*

$$\forall v, w \in V, C \in \mathcal{C} \quad C(v) = C(w) \Rightarrow \forall u \in V : CM(u) = 1.$$

Optimal hiding gives the upper limit for commonness as there can be no better hiding than in the crowd that consists of all other actors.

Property 2. *No hiding: Given a definition of similarity, if a vertex is not similar to any other vertex w.r.t. any centrality measure, the commonness of this vertex should be 0.*

$$\forall w \neq v \in V, C \in \mathcal{C} \quad C(v) \nsim C(w) \Rightarrow CM(v) = 0$$

This property defines the lower bound for commonness and simply means that if an actor cannot hide at all, s/he is maximally exposed and should get the lowest possible commonness score.

Property 3. *If one measure in the set of base centrality measures assigns the same score to all vertices, removing this measure from the set should have no effect on the commonness scores.*

$$\exists x \forall v \in V \quad \overline{C}(v) = x \Rightarrow CM_{\mathcal{C}}(v) = CM_{\mathcal{C}\setminus\overline{C}}(v)$$

This axiom basically says that if a centrality measure $C \in \mathcal{C}$ is not able to distinguish between the vertices (i.e. it assigns the exact same centrality score to all vertices), than it should have no influence on the commonness score.

Given this desired properties, we can define $\underline{\text{specific}}$ commonness measures and see how well they align with these properties.

Definition 13 (Common-ness Measure CM_1)**.** *We can define commonness as the sum of the squared distances separately for each dimension:*

$$CM_1(\mathcal{C}, v) = 1 - \frac{\sum_{C\in\mathcal{C}} \left(1 - \sum_{X\in D_{\{C\}}(v)} g_{\{C\}}(X)\right)^2}{k} \tag{2.1}$$

Definition 14 (Common-ness Measure CM_2)**.** *Another way to define commonness is the fraction of all vertices that are similar to a vertex v in all considered dimensions:*

$$CM_2(\mathcal{C}, v) = \sum_{X\in D_{\mathcal{C}}(v)} g_{\mathcal{C}}(X). \tag{2.2}$$

If we think of the deviation from no informational value (that is when all actors have the same centrality value) as an error, we can define commonness as the sum of the squared error over all dimensions as in Definition 13. However, if the centrality measures are uncorrelated, measuring the averageness of an actor independently for each measure can lead to undesired results. For example, an actor might appear to be totally average w.r.t. all measures considered independently, but if the sets of actors he is similar to are large, but do not overlap for the different centrality measure, there is no crowd he can hide in. To account for this, in Definition 14 the joint probability distribution of all centrality measures is used.

We now investigate whether the two definitions have the desired properties. We start with CM_1.

Proposition 7. *CM_1 satisfies properties (1) and (2) but not (3).*

Proof. It follows immediately for the definition that CM_1 satisfies Property 1. If for a centrality measure all vertices have the same value x, then for all centrality measures C there is a X so that for all vertices v $D_{\{C\}}(v) = X$ and $f(X) = 1$. It follows immediately $CM_1(v) = 1-(1/1) = 0$. Property 2 is satisfied as well. If for a measure C and a vertex v for all other vertices $w \in V$ $C(w) \notin I_v$ (this is the definition of similarity $\sim$ of CM_1), then $D_{\{C\}}(v)$ is empty. So, the sum over all X in $D_{\{C\}}$ is 0, the whole fraction becomes $k/k = 1$ and $CM(v) = 0$. Finally, lets have a look at Property 3 for CM_1. Let us assume we have two centrality measures C_1 and C_2, whereof C_2 is non-distinguishing. For C_1 the term $\sum_{X \in D_{\{C\}(v)}} g_{\{C\}}(X)$ will have some value $a < 1$ for every v while the term get 0 for C_2 (compare discussion of Property 2). So $CM_1(\{C_1\}, v) = 1 - a/1$ while $CM_1(\{C_1, C_2\}, v) = 1 - (a+1)/2$. Since $a < 1$ follows $CM_1(\{C_1\}, v) \neq CM_1(\{C_1, C_2\}, v)$ and CM_1 does not satisfy Property 3.

Proposition 8. *CM_2 satisfies all 3 properties (1)–(3).*

Proof. The proof that CM_2 satisfies Property 1 is similar to that for CM_1. If for some set of centrality measures $\mathcal{C}$ all vertices have the identical centrality vector X, for every $v \in V$ the only similar vector $D_C(v)$ is X. As all vectors are identical $g(x) = 1$ and so $CM_2(\mathcal{C}, v) = 1$ for every v. Likewise, if $D_{\{C\}}(v)$ is the empty set when there are not other vertices similar to a vertex v, i.e. there is no $w \neq v \in V$ with $C(w) \in I_v$. From $D_{\{C\}}(v) = \emptyset$ follows $CM_2(\cdot\,, v) = 0$ and so CM_2

satisfies Property 2. To proof that Property 3 is satisfied goes as follows. Let C_n be a measure that assign the same score s to every vertex. If we add C_n to $\mathcal{C}$ we only extend the centrality vectors and intervals by one dimension. As every vertex has the same centrality score of C_n, for every vector $Y = (y_1, \ldots, y_k)$ in $D_{\mathcal{C}}(v)$ there is a corresponding vector $Y_n = (y_1, \ldots, y_k, s)$ in $D_{\mathcal{C} \cup C_n}(v)$. As $X = (x_1, \ldots, x_k)$ we be extended to $X_n = (x_1, \ldots, x_k, s)$ it follows that $g_{\mathcal{C}}(X) = g_{\mathcal{C} \cup C_n}(X_n)$ holds because of the correspondence between X and X_n.

Thus, CM_2 satisfies all three properties we want a commonness measure to have, while CM_1 satisfies only the first two desired properties. As a consequence, CM_2 seems to be the epistemologically better choice.

2.2.2.2 Communication Potential

The communication potential should reflect the ability to communicate and cooperate to achieve a common objective. This vague statement makes no point about which communication and co-operation options are important to achieve the common goal. If only in-group connections are important for achieving the group's objective, we define the the communication potential based on a centrality measure D and the group $\hat{V} \subset V$. Let $\hat{G} = (\hat{V}, \hat{E})$ be the induced subgraph of G given by the group $\hat{V}$. Then the communication potential is $CP_1(v) = D_{\hat{G}}(v)$, i.e. the centrality score of v on the graph $\hat{G}$ that is "of interest". In a criminal or terrorist network application, $\hat{V}$ might consist of vertices that an investigative agency has already uncovered - people they know are suspicious for one reason or another — and the agency wants to "uncover" the rest of the covert network. However, if the ability to communicate with people outside the group is important as well, the entire graph G can be used to measure the communication potential: $CP_2(v) = D_G(v)$.

Calculating CP_1 requires knowledge of the group that a vertex is trying to communicate with — a factor that can be problematic in the real world. Therefore, we can only calculate and compare CP_1 and so the covertness centrality of the members of a group to each other. CP_1 is undefined for actors that are not group members. So a covertness centrality measure based on CP_1 is suitable for explaining organizational structures of terror/criminal groups when $\hat{V}$ is known through other processes. In contrast, a covertness centrality score based on CP_2 is defined for every actor in a network. It can be used to answer generic questions such as "Where would criminals try to hide?"

2.2.2.3 Covertness Centrality

In Section 2.2.2.1 and 2.2.2.2 we discussed definitions of the two components of covertness centrality: commonness and communication potential. There is no unique "best way" to combine these two measure into one — any combination method would reflect a different judgment on the importance of common-ness and communication. Let us assume that the communication potential CP is normalized in a reasonable way to the interval $[0, 1]$ like CM is. A possible family of functions is

$$CC(v, \tau, \lambda) = \begin{cases} 0 & , CM(v) < \tau \\ \lambda CM(v) + (1 - \lambda)CP(v) & , CM(v) \geq \tau \end{cases} \tag{2.3}$$

For $\tau = 0$, CC is a classic trade-off between the two requirements. Additionally requiring a minimum level τ of commonness would reflect an understanding that the communication potential is irrelevant as long as the threat of being identified is too high. For the case of criminal networks, the actual assessment of the trade-off is influenced by the type of illegal activity a groups pursues. Morselli et al. [161] discuss that criminal for-profit organizations put more emphasis on efficiency, while ideological groups like terrorists are primarily concerned with security.

2.2.3 CC Computation using Sampling Methods

Calculating the exact commonness of a vertex requires calculating all base centralities for all vertices. While the computational effort to calculate some centrality measures is low, e.g. degree centrality, other centrality measures have a high runtime complexity, e.g. closeness and betweenness centrality. Calculating the exact commonness for large graphs is a very time-consuming or practical impossible task. However, if we are not interested in the covertness centrality of all vertices but only for some, we can speed up the calculation.

We could estimate the probability $g_{\mathcal{C}}(X)$ that a random vertex has a given centrality score from a sample of vertices $S \subset V$. That means, we replace $g_{\mathcal{C}}(X)$ with

$$\tilde{g}_{\mathcal{C}}(X) = \min\left\{\frac{|\{v \in S | C_1(v) = X_1 \ldots \wedge C_k(v) = X_k\}| - 1}{|S| - 1}\right\}. \tag{2.4}$$

If we expect that the centrality scores follow a power-law distribution (as they e.g. usually do for degree and betweenness centrality [90]), this sampling strategy might be problematic. We can expect random sampling – and hence $\tilde{g}_{\mathcal{C}}(X)$ – to tend to underestimate the size of the neighborhood for vertices in the long tails of power-law distributions when the sample size is low. In a similar but less significant way, centrality measures involving other distributions (e.g. closeness centrality distributions which are usually bell-shaped) can be affected when large parts of the score range have very low probabilities.

Let $\hat{f}_{\mathcal{C}}$ be the estimated joint probability distribution for the set of centrality measure $\mathcal{C}$. The continuous equivalent of CM_1 (Definition 13) is

$$\widehat{CM}_1(\mathcal{C}, v) = 1 - \frac{\sum_{i=1}^{k}\left(1 - \int_{C_i(v)-\alpha\sigma_i}^{C_i(v)+\alpha\sigma_i} \hat{f}_{\{C_i\}}(X)dX\right)^2}{k}. \tag{2.5}$$

and the equivalent of CM_2 (Definition 14) is

$$\widehat{CM_2}(\mathcal{C}, v) = \int_{C_1(v)-\alpha\sigma_1}^{C_1(v)+\alpha\sigma_1} \dots \int_{C_k(v)-\alpha\sigma_k}^{C_k(v)+\alpha\sigma_k} \hat{f}_{\mathcal{C}}(X) dX_k \dots dX_1 \tag{2.6}$$

The equations are identical to their discrete counterparts except for the calculation of the estimated number of vertices in the interval around v. The equivalence goes directly back to the derivation of the integral as the approximation of a stepwise function with infinite steps.

As we can assume that many centrality measures have a power-law distribution and are at least to some degree correlated to the degree centrality, an improved sampling technique might also improve the accuracy of heuristic algorithms. Instead of simple random sampling, systematic sampling [150] based on the degree of the vertices seems to be a good idea. Simple random sampling means we randomly select vertices of V. For systematic sampling, we order all vertices by degree. From a population of n vertices, we then select k vertices by taking every n/k-th vertex starting from a randomly select start vertex among the first n/k-th vertices.

To summarize, the computation of commonness is based on the size of the similarity neighborhood of a vertex, which can be computed in several ways:

1. Exact computation: Empirical distribution of all vertices in V
2. Heuristic computation based on simple sampling:
    (a) Empirical distribution[7]
    (b) Estimated distribution[8]
3. Heuristic computation based on systematic sampling:
    (a) Empirical distribution

[7] i.e. we compute the exact number of similar vertices based on the empirical distribution of the centrality scores

[8] i.e. we compute the expected number of similar vertices based on the probability density of the estimated distribution of the centrality scores

(b) Estimated distribution

Algorithm 4 shows the pseudo-code to compute covertness centrality for a set of vertices S based on the empirical distributions of a set $\mathcal{C}$ of base centrality measures. The function $sample$ could be either a simple random or systematic sampling or could return all vertices V if we want to compute exact scores. The communication potential measure D can be any centrality measure, $\hat{G}$ is an arbitrary subgraph of G (including $\hat{G} = G$) to compute D on, CM is a commonness function like those given in Eq. 2.1, 2.2, 2.5 and 2.6. The covertness function CC combines commonness and communication potential as in Eq. 2.3.

Though the Covertness Centrality Algorithm (Algorithm 4) can take any distribution as input, we note that the computation of commonness based on estimated distributions requires a method to detect the type of distribution of centrality measure as well as the parameters of the distribution — due to space constraints, we do not discuss 2(b) and 3(b) above.

Algorithm 4 Covertness Centrality

Input: Graph $G = (V, E)$, set of vertices S, set of centrality measures $\mathcal{C}$, communication potential measure D, sampling function $sampling$, commonness function CM, covertness function CC, subgraph $\hat{G}$

Output: covertness centrality scores list<Float> $covertness$

```
1: GetBaseCentralityScores
2: for $v \in sample(V) \cup S$ do
3:     for $C \in \mathcal{C}$ do
4:         $scores[v][C] \leftarrow C(G, v)$
5: GetCovertnessCentralityScores
6: for $v \in S$ do
7:     $cm \leftarrow CM(v, scores)$
8:     $cp \leftarrow D(v, \hat{G})$
9:     $covertness[v] \leftarrow CC(cm, cp)$
10: covertness
```

2.2.3.1 Algorithmic Complexity

The covertness of actors is especially of interest for the analysis of large networks as only then is there potentially a large crowd to hide in. Therefore, the computational complexity of algorithms to compute covertness centrality is important. To compute the exact commonness of a vertex, the base centrality score of all vertices for all centrality measures in $\mathcal{C}$ must be computed and the most expensive one will determine the compound complexity. Eigenvector and shortest-path based centrality measures have at least a quadratic run-time. However, for most measures, approximation algorithms have been developed. For a discussion of centrality measure computation see [116].

Let n denote the number of vertices in a graph. Calculating the exact commonness has a complexity equal to the sum of the base centrality measures plus the time to determine the neighborhood of a vertex. If all the centrality vectors are stored in a range tree whose creation takes $O(n(\log n)^{k-1})$, retrieving the similarity neighborhood of a vertex takes $O((\log n)^k + t)$ where t is the number of retrieved neighbors [25]. To calculate the covertness centrality of all vertices only for all distinct centrality vectors, a neighborhood search is required. In most cases, the dominating factor in the complexity of common-ness will originate from the calculation of the base centrality measures, e.g. when the set of base centralities includes a shortest-path based measure.

For some base centrality measures, sampling can speed up the calculation of commonness. Instead of calculating the empirical distribution of the centrality of all vertices, the empirical distribution of a random sample can be used as shown (see Equation 2.4). This is beneficial for all centrality measures where the individual centrality scores can be calculated independently for each vertex. In the case of betweenness centrality, all source shortest paths have to be computed for getting the score of one vertex and so individual score computation provides almost no benefit.

For eigenvector centrality, computing the score of only one vertex is impossible as the scores of the vertices are interdependent. If all scores have to be calculated, sampling would only decrease the complexity of the similarity neighborhood search. For closeness centrality however, calculating the scores of only a sample of vertices can dramatically decrease the runtime because in this case the single source shortest path problem has to be solved for each and every vertex.

As discussed in Section 2.2.3 the simple strategy of using the empirical distribution of a random sample will probably tend to underestimate the size of the neighborhood for vertices in the long tails of power-law distributions. Estimating the parameter of a distribution based on the sample is likely to provide higher accuracy for the same sample size. A second advantage would be that instead of having to determine the similarity neighborhood of each vertex, we just need to calculate the integral in Equation 2.5 or 2.6. For a graph with n vertices, a sample size of e.g. $\log n$ would decrease the effort for getting the required closeness centrality scores from $O(n^3)$ to $O(n^2 \log n)$. We will evaluate required sample sizes in Section 2.2.4. The required sample size depends on the sampling technique and the desired accuracy.

Systematic sampling instead of simple random sampling would require an additional $O(n \log n)$ to sort the vertices. For example, determining the closeness centrality score of one vertex takes $O(n^2)$ time — so the advanced sampling technique would pay off if the same accuracy is achieved with a minimal smaller sample size.

2.2.4 Evaluation

In this evaluation, we analyze the properties of the covertness centrality measures as well as the algorithms to calculate them. For this evaluation, we use the testbed of real-world networks listed in Table 2.7.

Tab. 2.7: Real-world evaluation datasets. The Youtube networks are snowball samples from the original dataset. URV is the abbreviation for Universitat Rovira i Virgili.

Network	#Vertices	#Edges	Type	Reference
URV	1133	10902	e-mail	[103]
Youtube 40k	39998	85793	friendship	[158]
Youtube 60k	59998	151481	friendship	[158]

2.2.4.1 Measures

First, we study the results of the two different commonness definitions. We have already discussed the fact that CM_1 might have a problem when the different centralities in $\mathcal{C}$ are not correlated because vertices that are very exposed w.r.t. one measure might still end up with high commonness scores. This is what we see in Fig. 2.8. Vertices with low closeness centrality scores have low CM_2 scores because of the low probability density of closeness in that area. However, these vertices have high CM_1 scores as they have many vertices in their similarity neighborhood for other base measures. That compensates for their rare closeness values.

Commonness is strongly negatively correlated to the base centrality measure degree, closeness, eigenvector and betweenness centrality. In Fig. 2.9 we see for $\lambda = 1$ (i.e. covertness is measured by common-ness only) that the distribution density increases for higher common-ness scores. For lower values of λ, i.e. for an increasing contribution of communication potential to covertness centrality, the distribution of covertness gets closer to that one of the communication potential measure. For $\lambda = 0.25$ the covertness centrality distribution is already very similar to the closeness centrality distribution (compare Fig. 2.7).

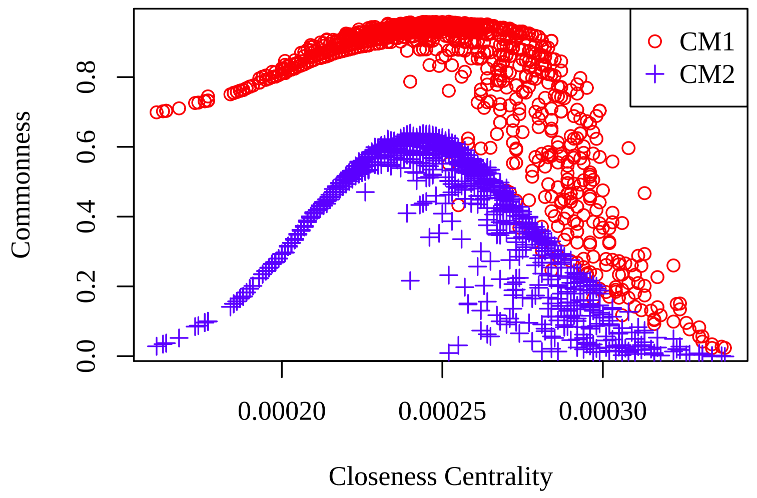


Fig. 2.8: Scatter plot of the commonness scores according to CM_1 and CM_2 (based on the four base centrality measures degree, closeness, betweenness and eigenvector centrality and the URV dataset) in relation to closeness centrality.

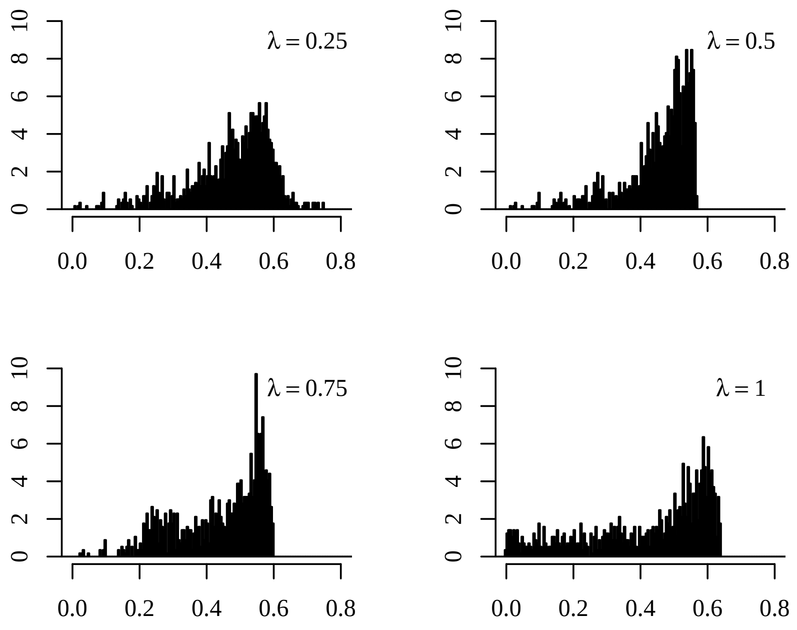


Fig. 2.9: Distribution of CC scores depending on different λ values when CM_2 (based on degree, betweenness, closeness and eigenvector centrality) is used as the measure of commonness and closeness centrality is used to measure CP and computed base on the whole graph.

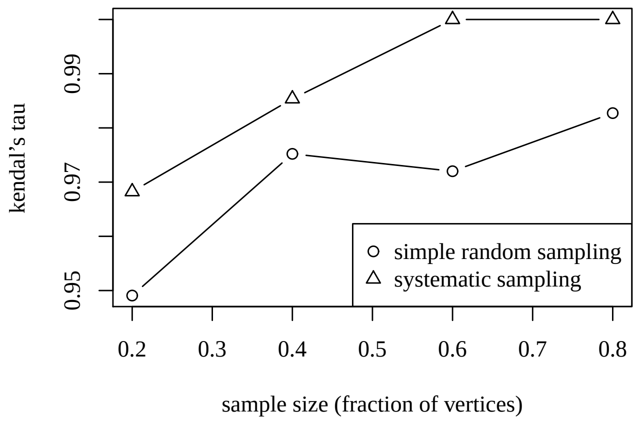


Fig. 2.10: Comparison of the rank correlation between the exact algorithm and the two sampling algorithm with different sampling methods for the URV dataset.

2.2.4.2 Compute Time

We implemented CM_1 and CM_2 in Python and evaluated the performance on a standard desktop machine. The scores of the base centralities were read from a file to evaluate the performance of the covertness computation only. The runtime scales linearly with the number of vertices with compute times of 0.1, 2, and 3 seconds respectively for our three test networks.

Beside the exact commonness algorithm we also studied the heuristic computation of the sizes of the similarity neighborhoods based on sampling. Fig. 2.10 shows for both sampling methods the Kendall τ rank correlation [129] of the commonness scores to the exact scores (with no sampling) is very high — well over 0.95. To achieve the same accuracy than simple random sample, systematic sampling requires a significantly lower sample size. More detailed results are shown in Table 2.8 and Tab. 2.9.

Tab. 2.8: Accuracy of sampling for different sampling methods and networks for CM_1 measured with Kendal's τ.

Network	Simple			Systematic		
	20%	40%	60%	20%	40%	60%
URV	0.965	0.981	0.981	0.966	0.985	1.000
YouTube 40k	0.975	0.992	0.997	0.979	0.996	1.000
YouTube 60k	0.981	0.995	0.997	0.990	0.996	1.000

Tab. 2.9: Accuracy of sampling for different sampling methods and networks for CM_2 measured with Kendal's τ.

Network	Simple			Systematic		
	20%	40%	60%	20%	40%	60%
URV	0.880	0.929	0.940	0.876	0.965	0.976
YouTube 40k	0.983	0.985	0.998	0.993	0.992	0.996
YouTube 60k	0.992	0.989	0.996	0.995	1.000	0.998

2.3 Related Work

Several centrality measures have been proposed in graph theory and social network analysis: *degree centrality* [83, 163], *betweenness centrality* [37, 82],*PageRank* [**?**], *closeness centrality* [177, 21], and *eigenvector centrality* [36]. Different variants of such centrality measures have been proposed as well [42, 71, 4, 3, 172, 100]. Game-theoretic centrality measures have been proposed in [114, 191].

Diffusion Centrality. One of the main differences between all of these centrality measures and diffusion centrality is that the former only take the structure of the network into account, ignoring properties of vertices as well as properties and weights of edges. As a consequence, any "semantics" embedded in the network is completely neglected—for instance, two completely different social networks (in terms of properties assigned to vertices and edges) that have the same topology are treated in the same way.

Another drawback of the aforementioned centrality measures is that they do not take into account the diffusive property with respect to which a vertex is considered "influential" or "central", as well as the model describing how such a property propagates. However, person A can have the highest centrality in terms of spread of support for Republicans, while person B can be the central player in terms of spread of support for conserving gorillas, so it becomes crucial to take into account the property with respect to which vertices are considered influential.

As discussed in Section 2.1 and shown in our experimental evaluation, neglecting this information can cause serious problems that diffusion centrality overcomes, being able to take into account both the semantics aspects of social networks and the diffusion model of interest. To the best of our knowledge, classical centrality measures either do not capture these aspects or simply ignore them.

There has been extensive work in reasoning about diffusion in social networks. One well-known and extensively studied problem is *influence maximization*, that is, the problem of identifying a subset of vertices in a social network that maximizes the spread of influence. The problem was first posed in [72] where diffusion processes were modeled as Markov random fields. The problem was formulated as an optimization problem in the seminal paper [127], which mainly focuses on two propagation models: the independent cascade and the linear threshold models. [127] shows that the influence maximization problem is NP-hard and provides an approximation greedy algorithm (with approximation guarantees).

The algorithm proposed in [127] is extremely expensive and does not scale to large social networks. To overcome this limitation, there have been numerous proposals aimed at achieving better performance under the independent cascade and linear threshold models [128, 133, 143, 53, 54, 117, 93, 95, 56, 55, 96, 198, 43, 135]. Algorithms for the Susceptible-Infected-Susceptible (SIS) model have proposed as well [134, 178, 112].

All of the approaches reported above for the influence maximization problem consider restricted diffusion models which suffer from various limitations when it comes to real-world propagation as no properties and vertex/edge conditions are considered. In contrast, our approach deals with very general diffusion models and takes social network semantic properties into account. The language we consider can express several well-known diffusion models such as the independent cascade, the linear threshold, the SIS and SIR models, the Jackson-Yariv model, and many others. Being able to accurately model the diffusion process and incorporate the semantic aspects of networks is fundamental for correct analysis of real-world diffusion phenomena.

Another well-studied related problem in computer science is the *target set selection* problem [119, 146, 58]. This problem assumes a deterministic tipping model and seeks to find a set of vertices of a certain size that optimizes the final number of adopters. Although approximation algorithms for this problem have been discovered, there is no evidence that they scale well for large datasets. Moreover, these approaches focus on specific diffusion models and neglect networks' semantics.

Recently, after our paper on diffusion centrality [123], a few pieces of work have observed that no individual can be a universal influencer, and influential members of the network tend to be influential only in one or some specific domains of knowledge, highlighting that the research on social influence has surprisingly largely overlooked this aspect [19, 47, 193]. Several theories in sociology [136, 99] show that the effect of the social influence from various angles may be different. This has led to the extension of the classic independent cascade and linear threshold models to be "topic-aware" [19, 47] thus considering propagation with respect to a particular topic. Approaches along this line but tailored for the microblogging setting have been proposed in [202, 166]. Still, the diffusion models considered by these works are limited, as well as the characteristics of the vertices and the edges of social networks.

A recent survey on different issues related to information diffusion in social networks is [102]. A general framework to solve social network diffusion optimization problems that can take a broad class of diffusion models as input has been proposed in [184], where diffusion models are expressed via generalized annotated logic programs [131].

Covertness Centrality. The literature on different types of dark, hidden or covert networks of criminal or terrorist groups is extensive. However, most existing literature deals with the network among the members of a covert group only. In contrast, the situation we deal with in Section 2.2 is a large network (e.g. phone, e-mail) in which a small fraction of actors constitute a covert sub-network in a larger (mostly innocent) population.

Our work is inspired by previous work on the conflict between the efficiency of communication in a terrorist group and the attempt to maintain secrecy. Lindelauf et al. [147] analyzed the trade-off between secrecy and information. Secrecy is defined by the exposure probability and the link detection probability. The exposure probability is the probability that a group member gets identified. Link detection probability is the probability that a covert actor is identified by following a link of an already identified covert actor. The link detection probability refers to what Gutfraind [104] denote as cascade resilience. In their case, a link is established if the resulting communication efficiency outweighs the costs of potentially being identified by following that link. In contrast, we see the costs of establishing a link with respect to the exposure an actor receives through it. So our approach is different, as for us covertness does not mean having the least possible number of connections but having connections that appear unsuspicious.

There has been a small amount of work on detecting covert cells within networks. In such a case, a group of individuals tries to stay hidden within a large network. However, most past works on covert cell detection do not study the relationship between centrality measures and covert cell structure. For instance, [20] discusses groups that try to camouflage communications - but the idea that an adversary trying to uncover such covert groups may look at centrality measures is not considered. In addition, there is work on detecting anomalous positions in a network - however, an anomaly may occur not because someone is trying to hide while maintaining communications with cell members but because an individual is in a location in the network that has skewed statistical properties. Conversely, covertness is not anomalous - an individual trying to stay ”off the radar” is making an explicit attempt to look ”normal” within the network and our notion of common-ness represents an explicit attempt by a ”bad guy” to not look anomalous. Thus, the two concepts are very different.

2.4 Conclusion

Centrality is a fundamental concept in social network analysis. Each centrality measure quantifies the relative importance of vertices in social networks w.r.t. a different analysis purpose. Many centrality measures have been extensively studied and widely used in social influence analysis. But there is no centrality measure to measure the influence of vertices with considering explicitly diffusion process of diffusive properties over social networks. So, in Section 2.1, we propose the novel concept of *Diffusion Centrality* and show how it can be computed with respect to diffusion models expressed via generalized annotated programs (GAPs). We then present the HyperDC algorithm that exactly computes diffusion centrality vertices w.r.t. any GAP-expressible diffusion models. We conduct a very detailed experimental study on several real-world social networks. The experiments compare the runtime and spread generated via the HyperDC algorithm with classical centrality measures. Our results show that HyperDC runs very fast, comparable in time to PageRank (and sometimes even degree centrality) for the Flickr and SIR models – but slightly slower for the Jackson-Yariv model, while producing much better spread for all three diffusion models.

In Section 2.2, we discuss the problem of constructing a measure that reflects covertness of a vertex that combines elements of "common-ness" as well as communication efficiency in a social network. Being covert in a network means hiding in a crowd of similar actors. It is motivated by attempt to analyze how malicious actors in a network behave if they are aware that the network is being monitored via a set $\mathcal{C}$ of centrality measures. Previous work on covertness focused on how actors as parts of small covert networks optimize their position within the network only. We extend those work by the broader view on large networks where covert network are embedded in. We present several ways to construct a covertness centrality measure as well as ways to compute the scores of such measures. Given that the base centrality scores are known covertness centrality can be computed efficiently. If the base scores are unknown, we show that with suitable sampling techniques it is not necessary to determine the exact similarity neighborhoods. We develop an algorithm to compute covertness centrality of nodes using different types of sampling methods, showing that these sampling methods are highly correlated to methods without sampling and that they can be computed relatively efficiently with small sample sizes.

Note. The material in Section 2.2 has been published in [165]. The authors agree that I made a significant contribution and may use this work in my thesis.

Chapter 3: Network Simplification

For very huge networks, it is not feasible to apply social network analysis techniques directly and exactly. Due to the reason, several network sparsification and simplifications techniques have been developed in order to help social influence analysis in such large social networks. But the techniques do not consider diffusion process explicitly and semantic aspects (vertex and edge properties) of social networks. Therefore, we develop new network coarsening techniques that approximately preserves both structural and semantic properties of social networks with respect to diffusion process (of a given diffusion model). In Section 3.1, first, we formulate a novel Graph Coarsening Problem to find a succinct representation of any graph while preserving key characteristics for diffusion processes on that graph. We then provide a fast and effective near-linear-time (in nodes and edges) algorithm COARSENET. Then, in Section 3.2, we develop methods to coarsen a network and propose a heuristic algorithm called "Coarsened Back and Forth" or CBAF to compute the top-k vertices (having the highest diffusion centrality w.r.t. p) which is extended from COARSENET to support various diffusion process and semantica aspects of networks in coarsening step mainly.

3.1 Fast Influence-based Coarsening for Large Networks

3.1.1 Introduction

The unprecedented popularity of online social networking websites, such as Facebook, Google+, Flickr, and YouTube, has made it possible to analyze real social networks. Word of mouth marketing and viral marketing strategies have evolved to take advantage of this network structure by utilizing network effects. Similarly, understanding large-scale epidemiological datasets is important for designing effective propagation models and containment policies for public health. The sheer size of today's large social networks makes it challenging to perform sophisticated network analysis.

Given a propagation graph, possibly learnt from cascade analysis, is it possible to get a smaller nearly diffusion-equivalent representation for it? Getting a smaller equivalent graph will help multiple algorithmic and data mining tasks like influence maximization, immunization, understanding cascade data and data compression. In this paper, we study a novel graph coarsening problem with the aim of approximating a large social network by a much smaller graph that approximately preserves the network structure. Our primary goal is to find a compact representation of a large graph such that diffusion and propagation processes on the large graph can be studied by analyzing the smaller representation. Intuitively, most of the edges in a real network are relatively unimportant; hence we propose characterizing and "contracting" precisely such edges in a graph to obtain a coarse representation.

The main contributions of this paper are:

(a) *Problem Formulation:* We carefully formulate a novel *Graph Coarsening Problem* (GCP) to find a succinct representation of a given social network so that the diffusion characteristics of the network are mostly preserved.

(b) Efficient Algorithms: We develop COARSENET, an efficient (near-linear time) and effective algorithm for GCP, using careful approximations. We show that due to our novel scoring technique, the coarsened graph retains most of the diffusive properties of the original network.

(c) Extensive Experiments: We show that COARSENET is able to coarsen graphs *up to* 90% without much loss of key information. We also demonstrate the usefulness of our approach via a number of interesting applications. A major application we consider in this work is that of influence maximization in the Independent Cascade model. We propose a framework CSPIN that involves coarsening the graph and then solving influence maximization on the smaller graph to obtain high quality solutions. As the coarsened graph is much smaller than the original graph, the influence maximization algorithm runs orders of magnitude faster on the coarsened graph. Further using real cascade data from Flixster, we show how GCP can potentially help in understanding propagation data and constructing non-network surrogates for finding nodes with similar influence.

The rest of the paper is organized as follows: Section 5.7 gives related work and Section 3.1.2 briefly gives the notation and explains some technical preliminaries. Section 3.1.3 provides a formal definition of the Graph Coarsening Problem that we introduce, while Section 3.1.4 presents our approach and solution. In Section 3.1.5, we show how our coarsening framework can be applied to solve influence maximization on large networks. Finally, Section 4.7 gives experimental results while we conclude in Section 4.9.

3.1.2 Preliminaries

Table 3.1 gives some of the notation.

Tab. 3.1: Symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (bold upper case)
$\vec{a}, \vec{b}, \ldots$	column vectors
a_j or $a(j)$	j^{th} element of vector a
n	number of vertices in the graphs
m	number of edges in the graphs
α	the reduction factor
$\lambda_{\mathbf{G}}$	first eigenvalue (in absolute value) of adjacency matrix of graph $\mathbf{G}$
$\vec{u}_{\mathbf{G}}, \vec{v}_{\mathbf{G}}$	Right and left first eigenvectors (for $\lambda_{\mathbf{G}}$) of adjacency matrix $\mathbf{G}$
IC Model	The Independent Cascade Model
GCP	Graph Coarsening Problem (see Definition 17)
COARSENET	Our algorithm for GCP

IC Model. A social network is a directed, weighted graph $G = (V, E, w)$. Usually each vertex $v \in V$ represents an individual of the network and edges represent influence relationships between these individuals. The Independent Cascade (IC) model is a popular diffusion model used to model the way influence propagates along the edges of a social network. In this setting, a vertex $v \in V$ is called *active* if it has been influenced and *inactive* otherwise. Once an inactive vertex becomes active, it always stays active, i.e. we focus only on *progressive* models. Given a seed set $S \subset V$ of initially active vertices, the Independent Cascade model proceeds in discrete time steps as follows. At time step t, let S_t denote the set of vertices activated at time t. Every vertex $u \in S_t$ is given a *single* chance to activate each currently inactive neighbor v with probability of success $w(u, v)$ independently of all other interactions. If u succeeds, then v becomes active at time $t + 1$. This diffusion process continues until no more activations are possible. The *influence spread* of seed set S, denoted by $\sigma(S)$, is the expected number of activated vertices at the end of the process.

3.1.3 Problem Formulation

Motivated by the fact that in any real network, *most* edges and vertices are not important (due to the heavily skewed degree distributions), we propose a graph coarsening problem which involves pruning away precisely such edges (and vertices). We aim to coarsen the graph to obtain a much smaller representation which retains the diffusive properties. We *coarsen* a graph by successively merging adjacent node pairs. We attempt to quickly find "good" edges which have little effect on the network's diffusive properties. At first glance, this seems impossible as the diffusive properties of a graph are highly dependent on the connectivity of the vertices and edge weights. Further, determining which node pairs to merge and analyzing the effect of merging two nodes on diffusion are non-trivial. Informally, we study the following problem in this paper:

Definition 15 (Informal Problem)**.**

INPUT: *Weighted graph* $G = (V, E, w)$ *and a target fraction* $0 < \alpha < 1$

GOAL: *Coarsen* G *by repeatedly merging adjacent node pairs to obtain a weighted graph* $H = (V', E', w')$ *such that*

- $|V'| = (1 - \alpha)|V|$
- *Graph* H *approximates graph* G *with respect to its diffusive properties*

Role of Eigenvalues. In order to address the informal problem described above, we need a tractable way to characterize the diffusive properties of a network. Recent work [169] shows that for almost any propagation model (including the IC model), important diffusion characteristics (in particular the so-called epidemic threshold) of a graph (after removing self loops) are captured by the spectrum of the graph, specifically, by the first eigenvalue of the adjacency matrix. Thus it is natural to believe that if the first eigenvalue of the coarsened graph H (its adjacency matrix) is close to that of the original graph G, then H indeed approximates G well. Although the work of [169] deals with undirected graphs, their findings are also applicable to strongly connected directed graphs.

Merging node pairs. To explicitly formulate the problem in Definition 15, we also need to define what happens when a node pair is merged (i.e. an edge is contracted) in a weighted graph. More precisely, after merging neighboring vertices a and b to form a new node c, we need to determine the *new* edge weights of all incoming and outgoing edges of c. In order to maintain the diffusive properties of the network, we need to reweight the new edges appropriately.

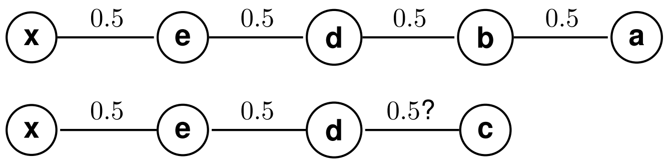
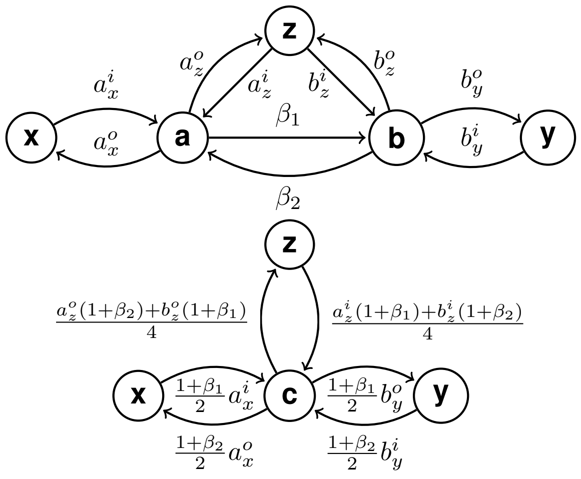


Fig. 3.1: Why reweight?

To see why this is crucial, consider Figure 3.1. Assume that the IC model is being run. Suppose we need to pick the two best seeds (i.e. two nodes with the maximum influence spread as defined in the previous Section) from the top 5-vertex chain. Further assume that the graph is undirected and each edge has the same weight $\beta = 0.5$. Clearly, vertices b and e are the best. If we merge vertices $\{a, b\}$, we get the bottom 4-vertex chain. To still match the original solution, we correspondingly want $\{c, e\}$ to be the best seed-set in the new chain—but if edge $\{d, c\}$ remains the same weight, any of the pair of vertices $\{e, c\}$ or $\{x, d\}$ are the best seed sets in the 4-vertex chain. This motivates the need to reweight suitably so that new coarsened graph still retains the original characteristics.

The main insight is that if we select c as a seed, we are in-effect intending to choose only *one* of vertices a and b to be seeded (influenced); which suggests that the likelihood of d being influenced from c is *either* 0.5 or 0.25 (corresponding to when a or b is chosen respectively). Hence the weight of edge (c, d) should be modified to reflect this fact.

We propose the following solution: Suppose $e = (a, b)$ is contracted and a and b are merged together to form "supervertex" c (say). We reweight the edges adjacent to a and b while coarsening so that the edges now represent the average of the transmission probabilities via a or b. So in our example of Figure 3.1, edge $\{c, d\}$ would have weight 0.375 (average of 0.5 and 0.25). Further, we can verify that in this case $\{e, c\}$ will be the best seed-set, as desired.



z
a_z^o
a_z^i
b_z^i
b_z^o
a_x^i
a_x^o
β_1
b_y^o
b_y^i
x
a
b
y
β_2
z
$\frac{a_z^o(1+\beta_2)+b_z^o(1+\beta_1)}{4}$
$\frac{a_z^i(1+\beta_1)+b_z^i(1+\beta_2)}{4}$
x
c
y
$\frac{1+\beta_1}{2}a_x^i$
$\frac{1+\beta_1}{2}b_y^o$
$\frac{1+\beta_2}{2}a_x^o$
$\frac{1+\beta_2}{2}b_y^i$

Fig. 3.2: Reweighting of edges after merging node pairs

Extending the same principle, Figure 3.2 shows the general situation for any candidate node pair (a, b) and how a merge and re-weight (= contract) operation will look like. More formally, our contract operation is as follows:

Definition 16 (Merging node pairs)**.** *Let $Nb^i(v)$ (respectively $Nb^o(v)$) denote the set of in-neighbors (resp. out-neighbors) of a vertex v. Let $v^i_u = w(u,v)$ and $v^o_u = w(v,u)$ denote the weight of the corresponding edges. If the node pair(a,b) is now contracted to a new vertex c, and $w(a,b) = \beta_1$ and $w(b,a) = \beta_2$, then the new edges are weighted as -*

$$c^i_t = \begin{cases} \dfrac{(1+\beta_1)a^i_t}{2} & \forall t \in Nb^i(a) \backslash Nb^i(b) \\[2ex] \dfrac{(1+\beta_2)b^i_t}{2} & \forall t \in Nb^i(b) \backslash Nb^i(a) \\[2ex] \dfrac{(1+\beta_1)(a^i_t) + (1+\beta_2)(b^i_t)}{4} & \forall t \in Nb^i(a) \cap Nb^i(b) \end{cases}$$

$$c^o_t = \begin{cases} \dfrac{(1+\beta_2)a^o_t}{2} & \forall t \in Nb^o(a) \backslash Nb^o(b) \\[2ex] \dfrac{(1+\beta_1)b^o_t}{2} & \forall t \in Nb^o(b) \backslash Nb^o(a) \\[2ex] \dfrac{(1+\beta_2)(a^o_t) + (1+\beta_1)(b^o_t)}{4} & \forall t \in Nb^o(a) \cap Nb^o(b) \end{cases}$$

Graph Coarsening Problem. We are now ready to state our problem formally. Motivated by the connections between the diffusive and spectral properties of a graph, we define the following Graph Coarsening Problem to find the set of node pairs which when merged (according to Definition 16) lead to the least change in the first eigenvalue. Further, since a vertex cannot influence itself, we assume without loss of generality that the graph G has no self loops.

Definition 17 (Graph Coarsening Problem)**.**

INPUT: *Directed, strongly connected, weighted graph $G = (V, E, w)$ without self loops and a target fraction $0 < \alpha < 1$*

OUTPUT: $E^* = \arg\min_{E' \subset E, |E'| = \alpha|V|} |\lambda_G - \lambda_{G'}|$, *where G' is obtained from G by merging all node pairs in E'.*

A related problem is Edge Immunization [195] that asks for a set of edges whose *removal* leads to the *greatest drop* in the first eigenvalue. In contrast, GCP seeks to find a set of edges whose *contraction* (Definition 16) leads to the *least change* in the first eigenvalue. The Edge Immunization problem is known to be NP-hard [195].

3.1.4 Our Solution

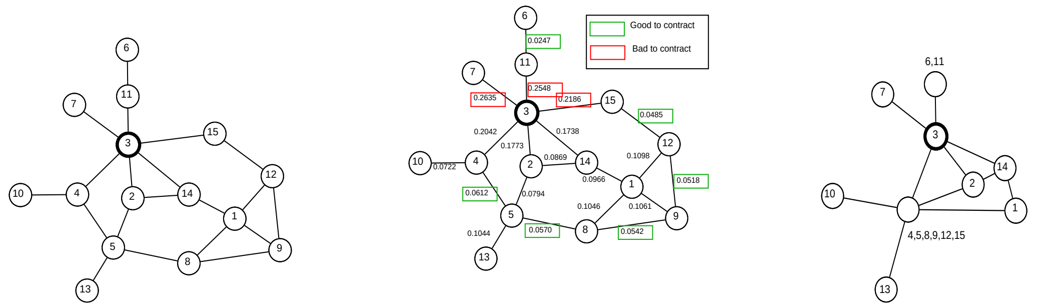
As obvious algorithms to GCP are clearly exponential, we propose a greedy heuristic that repeatedly merges a node pair which minimizes the change in the first eigenvalue. Let $G_{-(a,b)}$ denote the graph G after merging nodes a and b (and incorporating the re-weighting strategy), and λ_G denote the first eigenvalue of the adjacency matrix of G. We define the score of a node pair(a, b) as follows -

Definition 18 (Score)**.** *Given a weighted graph $G = (V, E, w)$ and an adjacent node pair(a, b), $score(a, b)$ is defined by:*

$$score(a, b) = |\lambda_{G_{-(a,b)}} - \lambda_G| = \Delta\lambda_{(a,b)}$$

Intuitively, if $score(a, b) \approx 0$, it implies that edges (a, b) and (b, a) do not play a significant role in the diffusion through the graph and can thus be contracted. Figure 3.3 shows an example of our approach.

Naïve Algorithm: The above intuition suggests the following naïve algorithm for selecting node pairs for merging. At each stage, calculate the change in the eigenvalue due to merging each adjacent node pair, choose the node pair leading to the least change, merge the chosen nodes, and repeat until the graph is small enough. An implementation for this, even using the Lanczos algorithm for eigenvalue computation for sparse graphs, will be too expensive, taking $O(m^2)$ time. Can we compute (maybe approximately) the scores of each node pair faster?



(a) Original Network

(b) Assigning scores to each edge

(c) Coarsened Network

Fig. 3.3: Our approach on an example network. All edges have weight 0.5 in the original graph. We do not show the new edge weights in the coarsened graph for clarity.

Main Idea: We use a matrix perturbation argument to derive an expression for the change in eigenvalue due to merging two adjacent nodes. Using further information about the specific perturbations occurring due to merging two adjacent nodes, we show that the change in the eigenvalue can be approximated well in constant time. Thus, we obtain a linear ($O(m)$) time scheme to estimate the score of every pair of adjacent nodes.

3.1.4.1 Score Estimation

Let a and b denote the two neighboring vertices that we are trying to score. We assume that the first eigenvalue of the graph λ_G and the corresponding right and left eigenvectors $\vec{u}, \vec{v}$ are precomputed. Further since the graph G is strongly connected, by the Perron-Frobenius theorem, the first eigenvalue λ_G and the eigenvectors $\vec{u}$ and $\vec{v}$ are all real and have positive components. When it is clear from the context, we drop subscripts G and (a, b). In the proofs that follow $\lambda = \lambda_G$ and $\Delta\lambda = \Delta\lambda_{(a,b)}$ as there is no ambiguity. Let $\mathbf{A}$ denote the adjacency matrix of the graph. Further as $\vec{u}$ denotes the eigenvector of $\mathbf{A}$, let $u_a = u(a)$ denote the component of $\vec{u}$ corresponding to vertex a. Merging nodes changes the dimensions of the adjacency matrix $\mathbf{A}$ which we handle by viewing merging nodes a, b as adding b's neighbors to a and isolating node b.

Approximation 9 provides an equation for the change in the eigenvalue by a matrix perturbation argument. Proposition 10 and Proposition 11 show how our reweighting strategy helps us to approximate the $score(a, b)$ in constant time.

Approximation 9. *The change in eigenvalue $\Delta\lambda$ can be approximated by* $\Delta\lambda = \dfrac{\vec{v}^T \Delta\mathbf{A}\vec{u} + \vec{v}^T \Delta\mathbf{A}\Delta\vec{u}}{(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u})}$ *where $\Delta\mathbf{A}$ denotes an infinitesimally small change in the adjacency matrix $\mathbf{A}$ and $\Delta\vec{u}$ denotes the corresponding change in the eigenvector $\vec{u}$.*

JUSTIFICATION. By the definition of an eigenvalue and eigenvector of a matrix, we have

$$\mathbf{A}\vec{u} = \lambda\vec{u} \tag{3.1}$$

$$\vec{v}^T\mathbf{A} = \vec{v}^T\lambda \tag{3.2}$$

Perturbing all values of (3.1) infinitesimally, we get

$$(\mathbf{A} + \Delta\mathbf{A})(\vec{u} + \Delta\vec{u}) \approx (\lambda + \Delta\lambda)(\vec{u} + \Delta\vec{u})$$

$$\mathbf{A}\Delta\vec{u} + \Delta\mathbf{A}\vec{u} + \Delta\mathbf{A}\Delta\vec{u} \approx \lambda\Delta\vec{u} + \Delta\lambda\vec{u} + \Delta\lambda\Delta\vec{u}$$

Premultiplying by $\vec{v}^T$ and using (3.1) and (3.2),

$$\Delta\lambda(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u}) \approx \vec{v}^T\Delta\mathbf{A}\vec{u} + \vec{v}^T\Delta\mathbf{A}\Delta\vec{u}$$

$$\Delta\lambda \approx \frac{\vec{v}^T\Delta\mathbf{A}\vec{u} + \vec{v}^T\Delta\mathbf{A}\Delta\vec{u}}{(\vec{v}^T\vec{u} + \vec{v}^T\Delta\vec{u})} \tag{3.3}$$

□

Using expression (3.3) along with prior knowledge about the perturbations to the adjacency matrix $\mathbf{A}$ and the eigenvector $\vec{u}$, we obtain an expression for computing the score of the node pair.

Proposition 10 (Score Estimate)**.** *Under Approximation 9, the score of a node pair $score(a, b)$ can be approximated as*

$$\Delta\lambda_{(a,b)} = \frac{-\lambda\left(u_a v_a + u_b v_b\right) + v_a \vec{u}^T \vec{c^o} + \beta_2 u_a v_b + \beta_1 u_b v_a}{\vec{v}^T\vec{u} - (u_a v_a + u_b v_b)}$$

(ignoring second order terms).

Proof. Approximation 9 provided an expression for $\Delta\lambda$ in terms of the change in the adjacency matrix and the eigenvector. Now $\Delta\mathbf{A}$, i.e., change in the adjacency matrix can be considered as occurring in three stages namely (i) Deletion of a, (ii) Deletion of b, (iii) Insertion of c. Assume that c is inserted in place of a. Thus we obtain,

$$\begin{aligned}\Delta\mathbf{A} = &- \left(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T\right) - \left(\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T\right) \\ &+ \left(\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T\right)\end{aligned} \tag{3.4}$$

where $\vec{e_v}$ denotes a vector with a 1 in the v^{th} row and 0 elsewhere. Further, as we modify only two rows and columns of the matrix, this change $\Delta\mathbf{A}$ is very small.

Also, deletion of vertices a and b cause a^{th} and b^{th} components of $\vec{u}$ and $\vec{v}$ to be zero. $\Delta\vec{u}$, i.e, change in the eigenvector $\vec{u}$ can thus be considered as setting u_a and u_b to zero, followed by small changes to other components and to u_a due to addition of c. Thus we obtain,

$$\Delta\vec{u} = -u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta} \tag{3.5}$$

Although $\Delta\vec{u}$ cannot be considered as small, we assume that the changes $\vec{\delta}$ after setting u_a and u_b components to zero are very small.

Substituting for $\Delta\mathbf{A}$, we get

$$\begin{aligned}\vec{v}^T\Delta\mathbf{A}\vec{u} = \vec{v}^T(&-(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T) - (\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T \\ &+ (\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T))\vec{u}\end{aligned}$$

Since $\vec{v}^T\vec{e_a} = v_a$ and similarly,

$$\begin{aligned}\vec{v}^T\Delta\mathbf{A}\vec{u} = &-u_a\vec{v}^T\vec{a^i} - v_a\vec{a^o}^T\vec{u} - u_b\vec{v}^T\vec{b^i} - v_b\vec{b^o}^T\vec{u} \\ &+ u_a\vec{v}^T\vec{c^i} + v_a\vec{c^o}^T\vec{u}\end{aligned}$$

But $\vec{v}^T\vec{a^i} = \lambda v_a$ and $\vec{a^o}^T\vec{u} = \lambda u_a$ and similarly,

$$\vec{v}^T\Delta\mathbf{A}\vec{u} = -2\lambda\left(u_a v_a + u_b v_b\right) + u_a\vec{v}^T\vec{c^i} + v_a\vec{c^o}^T\vec{u} \tag{3.6}$$

Now using (3.4) and (3.5) consider,

$$\vec{v}^T \Delta\mathbf{A}\Delta\vec{u} = \vec{v}^T \Delta\mathbf{A}(-u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta})$$

Since $\Delta\mathbf{A}$ and $\vec{\delta}$ are both very small, we ignore the second order term $\vec{v}^T\Delta\mathbf{A}\vec{\delta}$.

$$\begin{aligned}\Rightarrow \vec{v}^T \Delta\mathbf{A}\Delta\vec{u} &= \vec{v}^T \Delta\mathbf{A}(-u_a\vec{e_a} - u_b\vec{e_b}) \\ &= \vec{v}^T(-(\vec{a^i}\vec{e_a}^T + \vec{e_a}\vec{a^o}^T) - (\vec{b^i}\vec{e_b}^T + \vec{e_b}\vec{b^o}^T \\ &\quad + (\vec{c^i}\vec{e_a}^T + \vec{e_a}\vec{c^o}^T))(-u_a\vec{e_a} - u_b\vec{e_b})\end{aligned}$$

Since self loops do not affect diffusion in any way, we can assume without loss of generality that G has no self loops. Further, simplifying using definitions of eigenvalue we get,

$$\begin{aligned}\vec{v}^T \Delta\mathbf{A}\Delta\vec{u} &= \lambda(u_a v_a + u_b v_b) + \beta_2 u_a v_b \\ &\quad + \beta_1 u_b v_a - u_a \vec{v}^T \vec{c^i}\end{aligned} \tag{3.7}$$

Ignoring small terms, we also have,

$$\vec{v}^T \Delta\vec{u} = \vec{v}^T(-u_a\vec{e_a} - u_b\vec{e_b} + \vec{\delta}) = -(u_a v_a + u_b v_b) \tag{3.8}$$

Substituting (3.6),(3.7) and (3.8) in Approximation 9, we get

$$\Delta\lambda = \frac{-\lambda\,(u_a v_a + u_b v_b) + v_a \vec{u}^T\vec{c^o} + \beta_2 u_a v_b + \beta_1 u_b v_a}{\vec{v}^T\vec{u} - (u_a v_a + u_b v_b)}$$

□

Note that every term in this expression is a simple product of scalars, except for the $\vec{u}^T\vec{c^o}$ term. We now show that even $\vec{u}^T\vec{c^o}$ can in fact be expressed in terms of scalars and can thus be computed in constant time.

Proposition 11. *Using the re-weighting scheme as defined in Definition 16, if c denotes the new vertex created by merging nodes $\{a, b\}$ and $\vec{c^o}$ denotes the out-adjacency vector of c, $\vec{u}^T\vec{c^o} =$*

$\frac{(1+\beta_2)}{2}(\lambda u_a - \beta_1 u_b) + \frac{(1+\beta_1)}{2}(\lambda u_b - \beta_2 u_a)$ *where* β_1 *is the weight of edge* (a, b) *and* β_2 *is the weight of the edge* (b, a).

Proof. Let $X = Nb^o(a) \setminus Nb^o(b), Y = Nb^o(b) \setminus Nb^o(a), Z = Nb^o(a) \cap Nb^o(b)$. Since, c is adjacent only to neighbors of a and b, we have

$$\vec{u}^T \vec{c^o} = \sum_{t \in X} u_t c_t^o + \sum_{t \in Y} u_t c_t^o + \sum_{t \in Z} u_t c_t^o + u_c W$$

where W is the weight of a self loop added at c. Note that a self loop does not affect diffusion in any way (as a node can not influence itself). We use a self loop only in the analysis so as to compute the scores efficiently.

As per our reweighting scheme (See Definition 16)

$$\begin{aligned} \vec{u}^T \vec{c^o} = &\sum_{t \in X} \frac{(1+\beta_2)}{2} u_t a_t^o + \sum_{t \in Y} \frac{(1+\beta_1)}{2} u_t b_t^o \\ &+ \sum_{t \in Z} (\frac{(1+\beta_2)}{4} a_t^o + \frac{(1+\beta_1)}{4} b_t^o) u_t + u_c W \end{aligned} \tag{3.9}$$

But, by definition of eigenvalues, we know that

$$\lambda u_a = \sum_{t \in V} u_t a_t^o = \sum_{t \in X} u_t a_t^o + \sum_{t \in Z} u_t a_t^o + u_b \beta_1$$

$$\begin{aligned} \sum_{t \in X} u_t a_t^o &= \lambda u_a - \sum_{t \in Z} u_t a_t^o - \beta_1 u_b \\ &= \lambda u_a - a^o(Z) - \beta_1 u_b \end{aligned} \tag{3.10}$$

where $a^o(Z) = \sum_{t \in Z} u_t a_t^o$

Similarly, we get

$$\sum_{t \in Y} u_t b_t^o = \lambda u_b - b^o(Z) - \beta_2 u_a \tag{3.11}$$

Substituting Equations (3.10), (3.11), in (3.9),

$$\begin{aligned}\vec{u}^T\vec{c^o} &= \frac{(1+\beta_2)}{2}(\lambda u_a - a^o(Z) - \beta_1 u_b) \\ &+ \frac{(1+\beta_1)}{2}(\lambda u_b - b^o(Z) - \beta_2 u_a) \\ &+ \frac{(1+\beta_2)}{4}a^o(Z) + \frac{(1+\beta_1)}{4}b^o(Z) + u_c W\end{aligned}$$

We now choose $W = (-\frac{(1+\beta_2)}{4}a^o(Z) - \frac{(1+\beta_1)}{4}b^o(Z))/u_c$, so that we get

$$\vec{u}^T\vec{c^o} = \frac{(1+\beta_2)}{2}(\lambda u_a - \beta_1 u_b) + \frac{(1+\beta_1)}{2}(\lambda u_b - \beta_2 u_a)$$

□

Corollary 1. *Given the first eigenvalue λ and corresponding eigenvectors $\vec{u}, \vec{v}$, the score of a node pair $score(a, b)$ can be approximated in constant time.*

Proof. Substituting for $\vec{u}^t\vec{c^o}$ in Proposition 10 using Proposition 11, we obtain an expression for $score(a, b)$ that is composed entirely of scalar terms. Thus we can estimate the edge score in constant time. □

3.1.4.2 Complete Algorithm

Using the approximation described in the previous section, we assign a score to every pair of adjacent nodes of the graph. We then sort these node pairs in ascending order of the absolute value of their scores. Intuitively, we would like to merge a node pair if it has minimal score. Given an upper bound of α, the graph is then coarsened by contracting αn node pairs one by one in this order ignoring any pairs that have already been merged. We give the pseudo-code of our algorithm COARSENET in Algorithm 5.

Lemma 1 (Running Time)**.** *The worst case time complexity of our algorithm is $O(m\ln(m) + \alpha n n_\theta)$ where n_θ denotes the maximum degree of any vertex at any time in the coarsening process.*

Algorithm 5 Coarsening Algorithm - COARSENET(G, α)

```
Input: A directed, weighted graph $G$=($V$,$E$,$w$),
    a reduction factor $\alpha$
Output: Coarsened graph $G^{\alpha}_{coarse}$=($V'$,$E'$,$w'$)
 1: $i = 0$
 2: $n = |V|$
 3: $G' = G$
 4: for each adjacent pair of nodes $a, b \in V$ do
 5:     Compute $score(a, b)$ using Section 3.1.4.1
 6: $\pi \leftarrow$ ordering of node pairs in increasing order of $score$
 7: while $i \leq \alpha n$ do
 8:     $(a, b) = \pi(i)$
 9:     $G' \leftarrow Contract_{G'}(a, b)$
10:     $i$++
11: return $G^{\alpha}_{coarse} = G'$
```

Proof. Computing the first eigenvalue and eigenvector of the adjacency matrix of the graph takes $O(m)$ time (for example, using Lanczos iteration assuming that the spectral gap is large). As shown in Section 3.1.4.1, each node pair can be assigned a score in constant time. In order to score all m adjacent pairs of nodes of the graph, we require linear i.e. $(O(m))$ time. The scored node pairs are sorted in $O(m \ln(m))$ time. Merging two nodes (a, b) requires $O(deg(a) + deg(b)) = O(n_\theta)$ time. Since we each merge at most αn pairs of nodes, the merging itself has time complexity $O(\alpha n n_\theta)$.

Therefore, our worst-case time complexity is $O(m \ln(m) + \alpha n n_\theta)$. $\square$

3.1.5 Sample Application: Influence Maximization

The eigenvalue based coarsening method described above aims to obtain a small network that approximates the diffusive properties of the original large network. As an example application, we now show how to apply our graph coarsening framework to the well studied influence maximization problem. Recall that given a diffusion model (IC model in our case) and a social network, the influence maximization problem is to find a small seed set of k nodes such that the expected number of influenced nodes is maximized.

Since we have designed our coarsening strategy such that nodes and edges important for diffusion remain untouched, we expect that solving influence maximization on the coarsened graph is a good proxy for solving it on the much larger original network. The major challenge in this process is to determine how to map the solutions obtained from the coarsened graph back onto the vertices of the original network. But due to the carefully designed coarsening strategy which tries to keep important, candidate vertices unmerged, we observe that a simple random pull back scheme works well in practice.

More formally, we propose the following multi-stage approach to solve influence maximization:

1. **Coarsen** the social network graph G by using Algorithm 5 to obtain a much smaller graph G_{coarse}. Let $\mu : V \to V_{coarse}$ denote a mapping from vertices of the original graph to those of the coarsened graph.
2. **Solve** the influence maximization problem on G_{coarse} to get k vertices $s_1, \ldots, s_k$ in the coarsened graph that optimize the desired objective function. We can use any off-the-shelf algorithm for influence maximization in this step. Since G_{coarse} is much smaller than G,

traditional algorithms for influence maximization can provide high quality solutions in little time.

3. **Pull back** the solutions on to the vertices of the original graph. Given a seed s_i in G_{coarse}, we need to select a vertex $v \in \mu^{-1}(s_i)$ from G as a seed. Multiple strategies can be considered here such as $v = \mathbf{argmax}_{u\in\mu^{-1}(s_i)}(\sigma(u))$ where $\sigma(u)$ is the expected influence by seeding u. However, thanks to our careful coarsening framework, we show that a simple strategy of selecting a seed uniformly at random from $\mu^{-1}(s_i)$ for every seed s_i performs very well in practice.

Algorithm 6 describes our strategy to solve influence maximization problems. Note that a similar strategy can be applied to study other problems based on diffusion in networks.

Algorithm 6 CSPIN: Influence Maximization Framework

```
Input: A weighted graph $G$=($V$,$E$,$w$), the number of seeds $k$, a reduction factor $\alpha$
Output: A seed set $S$ of $k$ seeds
1: $G^{\alpha}_{coarse}$, $\mu \leftarrow$ COARSENET($G$,$\alpha$)      (See Algorithm 5)
2: $s'_1, s'_2, \ldots, s'_k \leftarrow$ InfluenceMaximization($G^{\alpha}_{coarse}$, $k$)
3: for $i = 1, \ldots, k$ do
4:     $s_i \leftarrow$ random sample from $\mu^{-1}(s'_i)$
5: return $S = \{s_1, s_2, \ldots, s_k\}$
```

3.1.6 Experimental Evaluation

We performed several experiments to show the effectiveness of COARSENET algorithm and also the GCP framework for cascade analysis.

Tab. 3.2: Datasets: Basic Statistics

Dataset	#Vertices	#Edges	Mean Degree
Flickr_small	500,038	5,002,845	20.01
Flickr_medium	1,000,001	14,506,356	29.01
Flickr_large	2,022,530	21,050,542	20.82
DBLP	511,163	1,871,070	7.32
Amazon	334,863	1,851,744	11.06
Brightkite	58,228	214,078	7.35
Portland	1,588,212	31,204,286	39.29
Flixster	55,918	559,863	20.02

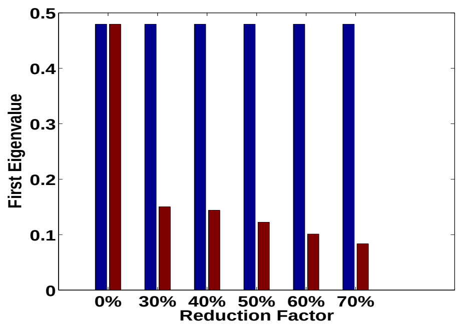
Datasets All experiments were conducted on an Intel Xeon machine (2.40 GHz) with 24GB of main memory[1]. We used a diverse selection of datasets from different domains to test our algorithm and framework (see Table 3.2). These datasets were chosen for their size as well as the applicability to the diffusion problem. COARSENET was tested on data from Flickr, DBLP, Amazon, Brightkite and Portland epidemiology data. In the Flickr data, vertices are users, and links represent friendships [49]. In the DBLP data, vertices represent authors and edges represent co-authorship links. Brightkite is a friendship network from a former location-based social networking service provider Brightkite. In the Amazon dataset, vertices are products and an edge represents that the two products are often purchased together. The Portland dataset is a social contact graph of vertices representing people and edges representing interactions—it represents a synthetic population of the city of Portland, Oregon, and has been used in nation-wide small-

[1] Code at: http://www.cs.vt.edu/~badityap/CODE/coarsenet.tgz

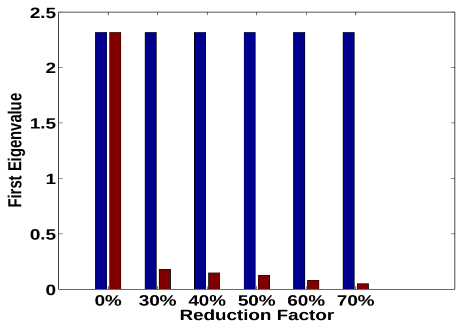
pox studies [77]. Finally, we also used a *real* cascade dataset Flixster[2], where cascades of movie ratings happen over a social network.

3.1.6.1 Performance for the GCP problem

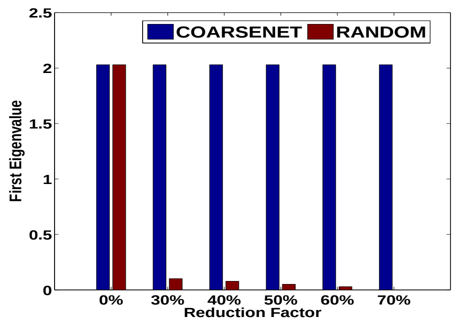
We want to measure the performance of COARSENET algorithm on the GCP problem. In short, we can coarsen up to 70% of node-pairs using COARSENET, and still retain almost the same eigenvalue.



a) Amazon



(b) DBLP



(c) Brightkite

Fig. 3.4: Effectiveness of COARSENET for GCP. λ vs α for COARSENET and RANDOM. COARSENET maintains λ values.

.

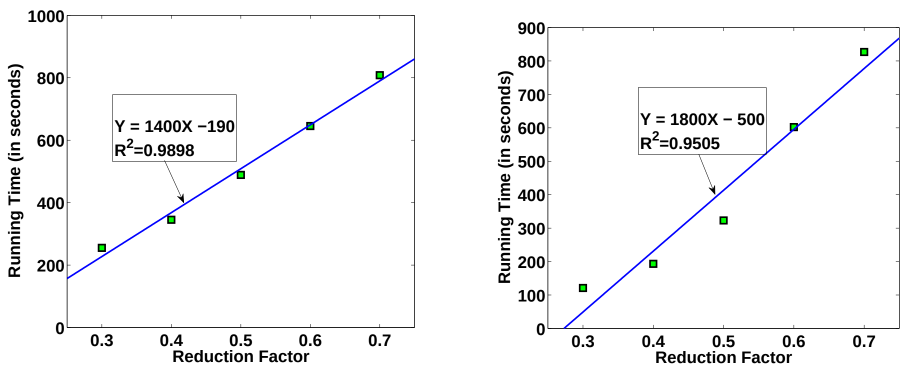
[2] http://www.cs.ubc.ca/~jamalim/datasets/

3.1.6.1.1 Effectiveness As a baseline we used RANDOM, a random node-pair coarsening algorithm (randomly choose a node-pair and contract), used in some community detection techniques. Figure 3.4 shows the values of λ as the reduction factor α increases when we ran COARSENET and RANDOM on three datasets (we set a weight of 0.02 for this experiment). We observed that in all datasets, as the reduction factor α increases, the values of λ barely change for COARSENET, showing that the diffusive properties are maintained even with almost 70% contraction; while RANDOM destroyed the eigenvalue very quickly with increasing α. This shows that (a) large graphs *can* in fact be coarsened to large percentages while maintaining diffusion; and (b) COARSENET effectively solves the GCP problem. As we show later, we apply the GCP problem and COARSENET on a detailed sample application of influence maximization.

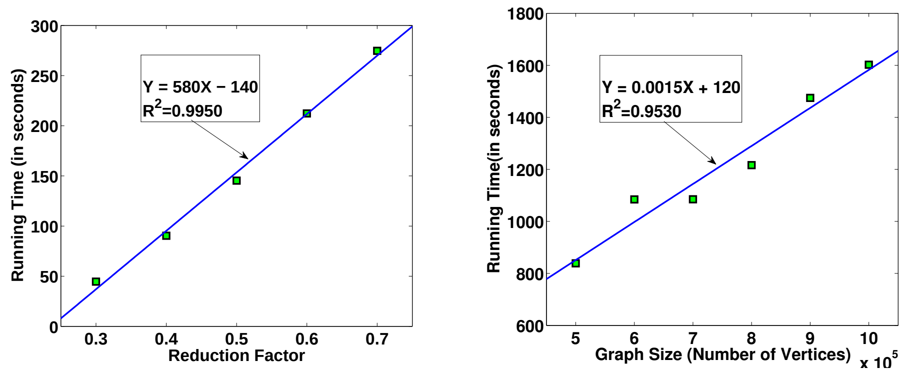
3.1.6.1.2 Scalability Figure 3.5 shows the running times of COARSENET w.r.t. α and n. To analyze the runtime of COARSENET with respect to graph size (n), we extracted 6 connected components (with 500K to 1M vertices in steps of 100K) of the Flickr_large dataset. As expected from Lemma 1, we observe that in all datasets, as the reduction factor α increases, the running time increases linearly (figures also show the linear-fit, with R^2 values), and scale near-linearly as the size of the graph increases. This demonstrates that COARSENET is scalable for large datasets.

3.1.6.2 Application 1: Influence Maximization

Here we demonstrate in detail a concrete application of our GCP problem and COARSENET algorithm to diffusion-related problems. We use the well-known Influence Maximization problem. The idea as discussed before is to use the Coarsen-Solve-Project CSPIN framework (see Section 3.1.5). In short we find that we obtain $300\times$ speed-up on large networks, while maintaining the quality of solutions.



(a) Amazon (b) DBLP



(c) Brightkite (d) Flickr (Varying sizes)

Fig. 3.5: Scalability of COARSENET for GCP. (a,b,c) Linear w.r.t. α. (d) Near-linear w.r.t. size of graph.

Propagation probabilities: Since accurate propagation probabilities for these networks are not available, we generate propagation probabilities according to two models following the literature.

- **Uniform:** Each edge is assigned a low propagation probability of 0.02. In most real social networks, the propagation probabilities are known to be low. For example, [49] find that the propagation probability in the Flickr network is about 1-2%.
- **Trivalency:** We also test on the trivalency model studied in [54]. For every edge we choose a probability uniformly at random from the set $\{0.1, 0.01, 0.001\}$ which correspond to the edge having high, medium and low influence respectively.

Algorithms and setup: We can use any off-the-shelf algorithm to solve Inf. Max. problem on the smaller coarsened network. Here, we choose to use the fast and popular PMIA [54] algorithm. We then compared the influence spreads and running-times of the CSPIN framework with the plain PMIA algorithm to demonstrate gains from using GCP.

3.1.6.2.1 Effectiveness **Quality of solution (Influence spread).** In all experiments, the influence spread generated by our CSPIN approach is within 10% of the influence spread generated by PMIA. In some cases, we even perform slightly better than PMIA. Figure 3.6(a) shows the expected spread obtained by selecting $k = 1000$ seeds on five datasets. For these experiments, the percentage of edges to merged is set at 90% and we use the uniform propagation model.

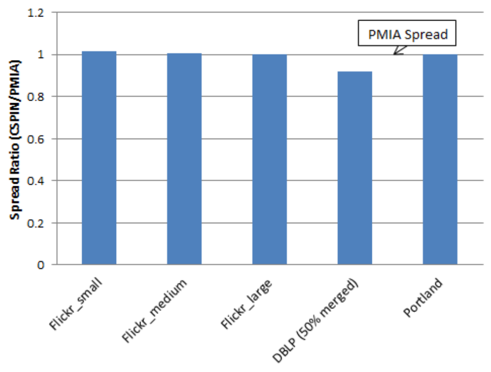
Quality w.r.t α. We find that we can merge up to 95% of the edges while still retaining influence spread. As more edges are merged, the coarsened graph is smaller; so the superseeds in G^{α}_{coarse} can be found faster and thus we expect our running time to decrease. We ran tests on the Flickr_medium dataset for 1000 seeds and varied α from 80% to 95%. Figure 3.6(b) shows the ratio of the expected influence spread obtained by CSPIN to that obtained by PMIA is almost 1 with varying α.

Quality of solution: Effect of unbiased random pullback.

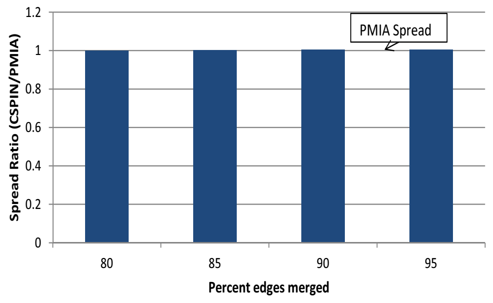
Tab. 3.3: Insensitivity of CSPIN to random pullback choices : Expected influence spread does not vary much.

#Trials	Maximum Spread	Minimum Spread	Coefficient of variation ($\frac{\sigma}{\mu}$)
100	58996.6	58984.8	5.061×10^{-5}

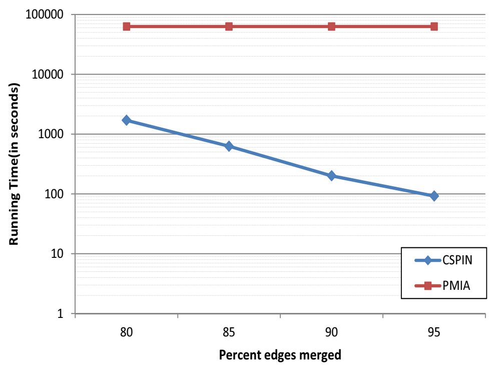
COARSENET groups nodes which have similar diffusion effects, hence choosing *any one* of the nodes randomly inside a group will lead to similar spreads (hence we do the random pullback in



(a) Spread ratio



(b) Spread ratio vs α on Flickr_medium

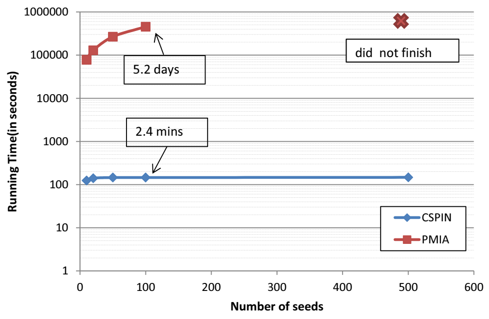


(c) Running times vs α on Flickr_medium

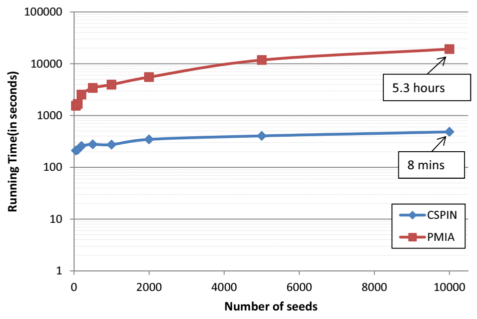
Fig. 3.6: Effectiveness of CSPIN. Ratio of influence spread between CSPIN and PMIA for (a) different datasets; (b) varying α. (c) Running time vs α.

CSPIN). Note we do not claim that these groups belong to link-based communities—only that their diffusive effects are similar. To demonstrate this, we performed 100 trials of the random pullback phase for the Flickr_small graph. For these trials, 1000 superseeds were found by coarsening 90% of the edges. In each trial, we use these same superseeds to find the 1000 seeds independently and uniformly at random. Table 3.3 shows that the coefficient of variation of the expected spread is only 5.061×10^{-5}.

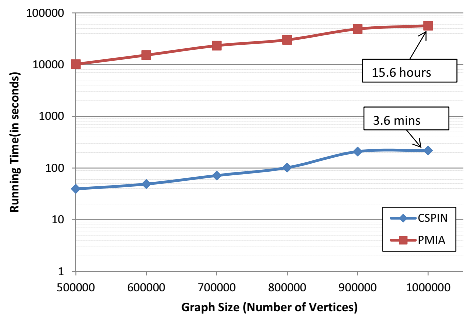
3.1.6.2.2 Scalability **Scalability w.r.t number of seeds (k).** As the budget k increases, we see dramatic performance benefits of CSPIN over PMIA. We run experiments on Flickr_small, and Portland by setting $\alpha = 90\%$, and k varied from 0.01% to 1% of $|V|$. Figure 3.7(a,b) shows the total running times (including the coarsening). Due to lack of space we show only the results for the trivalency model (the uniform case was similar). In all datasets, as k increases, the running time of CSPIN increases very slowly. Note that we get *orders of magnitude* speed-ups: e.g. on Flickr PMIA takes *more than 10 days* to find 200+ seeds, while CSPIN runs in 2 minutes.



(a) Flickr_small (trivalency model)



(b) Portland (trivalency model)



(c) Flickr (Varying sizes)

Fig. 3.7: Scalability of CSPIN. (a,b) vs k; (c) vs size of graph. CSPIN gets increasing orders-of-magnitude speed-up over PMIA.

Scalability w.r.t α. We can see that the running time also drops with increased coarsening as seen in Figure 3.6(c).

Scalability w.r.t n. We ran CSPIN on the components of increasing size of Flickr_large with

$k = 1000$ and $\alpha = 90\%$. Figure 3.7(c) plots the running times: CSPIN obtains a speed up of around $250\times$ over PMIA consistently.

3.1.6.3 Application 2: Diffusion Characterization

We now briefly describe how the GCP problem can help in understanding cascade datasets in an exploratory setting.

Methodology: We used a Flixster dataset, where users can share ratings of movies with friends. There is a log-file which stores ratings actions by each user: and a cascade is supposed to happen when a person rates the same movie soon after one of her friends. We use the methodology of [94] to learn influence probabilities of a IC-model over the edges of the friendship network from the traces. We then coarsen the resulting directed graph using COARSENET to $\alpha = 50\%$, and study the formed groups (supernodes). Note that this is in contrast to the approaches where the group information is supplied by a graph-partitioning algorithm (like METIS), and then a group-based IC model is learnt. The base network had $55,918$ nodes and $559,863$ edges. The trace-log contained about 7 *million* actions over $48,000$ movies. We get 1891 groups after removing groups with only one node, with mean group size 16.6 with the largest group having 22061 nodes (roughly 40% of nodes).

Distribution of movies over groups: Figure 3.8 shows the histogram of the # of groups reached by the movie propagations (following [155], we assume that a movie reaches a group if at least 10% of its nodes rated that movie). We show only the first 100 points of the distribution. We observe that a very large fraction of movies propagate in a small number of groups. Interestingly we observe a *multi-modal* distribution, suggesting movies have multiple scales of spread.

Groups through the lens of surrogates: An important point to note is that our groups *may not* be link-based communities: we just ensure that nodes in a group have the same diffusive properties.

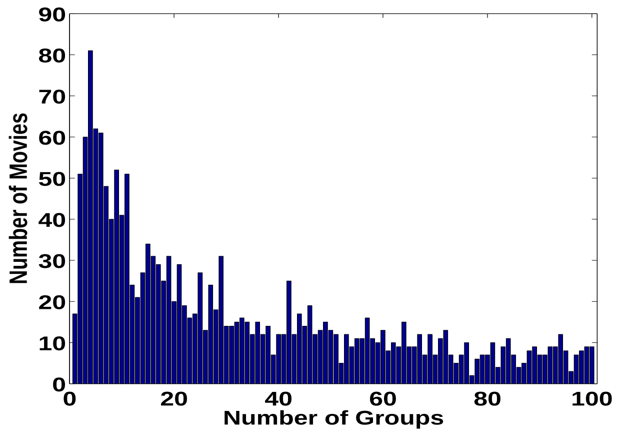


Fig. 3.8: Distribution of # groups entered by movie traces.

We validated this observation in the previous section (Table 3.3). Hence a natural question is if groups found in Flixster have any other natural structure (e.g. demographics)—if they do, we can get a non-network external *surrogate* for similar diffusive characteristics. Fortunately, the Flixster does contain a couple of auxiliary features for its users (like ID, Last Login, Age). We calculated the Mean Absolute Error (MAE) for 'Age' *inside* each group, and compared it with the MAE across groups. We found that the average MAE inside the group is very small (within 2 years) compared to a MAE of almost 8 outside, which implies that ages are concentrated within groups and can act as surrogates for diffusive characteristics.

Note. The material in Section 3.1 has been published in [171]. The authors agree that I made a significant contribution and may use this work in my thesis.

3.2 CBAF Algorithm for Approximating kDCP

The goal of this section is to develop our Coarsened Back and Forth (CBAF) algorithm to approximately compute the top-k diffusion centrality vertices in huge social networks where it is not feasible to compute DC for all vertices.

Top-k Diffusion Centrality Problem (kDCP). Given a $0 < k < |V|$, the top-k diffusion centrality problem consists of finding a set T of k vertices of $\mathcal{S}$ having the highest DC, i.e. the DC of every vertex in T is greater than or equal to the DC of every vertex in $V - T$.

The basic idea is to coarsen the original social network $\mathcal{S}$ into a smaller network $\mathcal{S}'$ which tries to preserve the "diffusive behavior" of $\mathcal{S}$. The top-k vertices are then computed over $\mathcal{S}'$ and the solution is mapped back to a subgraph of $\mathcal{S}$ on which we again compute the top-k vertices. Given a social network $\mathcal{S}$, a GAP Π, and a property p, CBAF performs the following steps:

1. Compute the filtering $\mathcal{S}'$ of $\mathcal{S}$;
2. Coarsen $\mathcal{S}'$ by merging its vertices so as to obtain a new social network $\mathcal{S}_C$ and a mapping from the vertices of $\mathcal{S}'$ to the vertices of $\mathcal{S}_C$;
3. Compute a set T_C of top-k vertices of $\mathcal{S}_C$;
4. Use T_C to compute an approximate set of top-k vertices of $\mathcal{S}$.

The first step has already been described in Section 2.1.4.1, while the other steps will be detailed in the rest of this section. The notations of the following sections are summurized in Section Ap.1.4.

3.2.1 Social Network Coarsening

This section proposes a new semantical coarsening technique that reduces network size by merging together vertices that are similar, while trying to preserve the structural and semantic properties of the network w.r.t. a property p. The coarsening process involves the following issues: *(i)* how to select "similar" vertices to be merged together, *(ii)* how to assign properties to a merged vertex and to its edges after merging, and *(iii)* how to compute edge weights between merged vertices. These issues are addressed in the following subsections.

3.2.1.1 Vertex Similarity

CBAF can work with any mechanism to determine whether two vertices are similar. Throughout this paper, we assume that given any vertex v, there is a set $SIM(v) \subseteq V$ of vertices that are similar to it. Such a function SIM can obviously be defined in many ways. We provide one such way of defining SIM that takes the diffusion model for p into account (as well as the social network structure).

Consider an SN $\mathcal{S}$, a GAP Π, and a property p. Given two vertices u and v of $\mathcal{S}$, we write $u \sim v$ iff u and v activate the same set of rules of Π (cf. Section 2.1.4.1 for the the notion of "activation"). Using this equivalence relation, we define $SIM_{\Pi}(v) = \{u \in V \mid u \sim v\}$.

Obviously, many other definitions are also possible, but this is the one used in our experiments.

Consider the social network in Figure 3.9 and the diffusion model Π_{hiv} of Example 2.1.2.2.

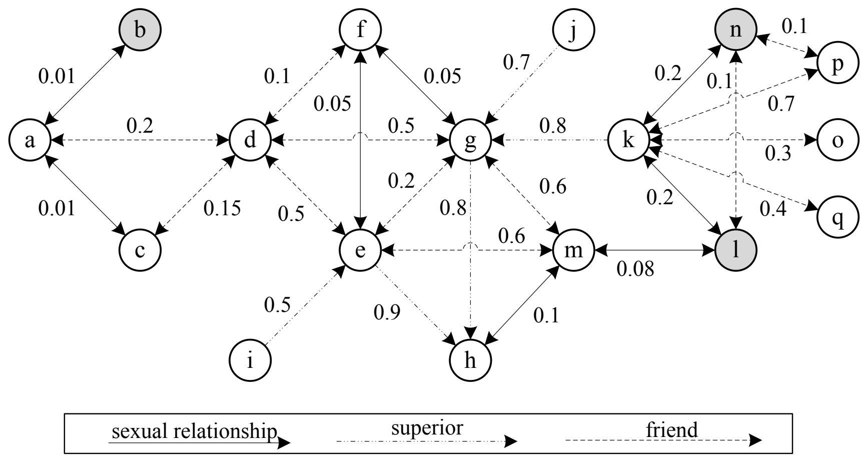


Fig. 3.9: An HIV social network. Shaded vertices denote people with HIV.

The set of vertices that activate rules r_1, r_2, and r_3 are, respectively:

$$\{a, b, c, e, f, g, h, k, l, m, n\}$$

$$\{a, b, c, d, e, f, g, h, k, l, m, n, o, p, q\}$$

$$\{a, b, c, e, f, g, h, k, l, m, n\}$$

Thus, $SIM_{\Pi}(x) = \{a, b, c, e, f, g, h, k, l, m, n\}$ for all $x \in \{a, b, c, e, f, g, h, k, l, m, n\}$ and $SIM_{\Pi}(x) = \{d, o, p, q\}$ for all $x \in \{d, o, p, q\}$. Notice that vertices i and j are unnecessary.

3.2.1.2 Vertex Merging

We now define how to merge similar vertices. When a set of vertices is merged into a new vertex v, we have to specify vertex properties of v, as well as associated edge properties/weights.

Definition 19 (Vertex properties merging)**.** *Let $\mathcal{S}$ be an SN and $\{v_1, \ldots, v_n\}$ be a subset of the vertices of $\mathcal{S}$ (to be merged). For each $p \in$ VP, let $g_p : \{0,1\}^n \to \{0,1\}$ be any associative and commutative function. Then, the new vertex has the set of properties*

$$mergeVP(\{v_1, \ldots, v_n\}, \mathsf{VL}) = \{p \in \textit{VP} \mid g_p(\vartheta(v_1, p), \ldots, \vartheta(v_n, p)) = 1\}$$

where $\vartheta(v_i, p) = 1$ *if* $p \in \mathsf{VL}(v_i)$*, otherwise* $\vartheta(v_i, p) = 0$.

The above definition assumes the existence of a function g_p that takes the values (0 or 1) of the p-property of the vertices being merged and combines them into a single 0 or 1 value denoting whether the merged vertex has property p or not.

Consider the social network shown in Figure 3.9 where the only vertex property is hiv. Some examples for the function g_{hiv} can be $g'_{\mathsf{hiv}}(x_1, \ldots, x_n) = \min_{1 \le i \le n} x_i$, computing the property intersection, or $g''_{\mathsf{hiv}}(x_1, \ldots, x_n) = \max_{1 \le i \le n} x_i$, computing the property union. Suppose we want to merge the vertices in the set $V' = \{l, m, n\}$. If we use g'_{hiv}, the resulting merged vertex will not have any property as $mergeVP(V', \mathsf{VL}) = \emptyset$, while if we use g''_{hiv}, the resulting merged vertex will have the set of properties $mergeVP(V', \mathsf{VL}) = \{\mathsf{hiv}\}$. We can also use a "majority" function where the new vertex has the property p if the majority of the vertices in V' have the property p.

We now address the problem of assigning edges to the new merged vertex. Let $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ be an SN and $V' = \{v_1, \ldots, v_n\}$ be a subset of V to be merged into a new vertex v'. For each edge $(v, u, ep) \in E$ such that either $v \in V'$ and $u \in V \setminus V'$ or $u \in V'$ and $v \in V \setminus V'$, the new vertex v' will have the outgoing edge (v', u, ep) if $v \in V'$, or the incoming edge (v, v', ep) if $u \in V'$. For instance, consider the SN in Figure 3.9 and suppose we are merging vertices e and g into a new vertex e'. Then, the new edges involving vertex e' are shown in Figure 3.10.

We now define how to assign a weight to an arbitrary set of edges (having the same label) in $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$.

Definition 20 (Edge weighting)**.** *Let* $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ *be an SN and* $ep \in \mathsf{EP}$*. Moreover, let* $\{e_1, \ldots, e_m\} \subseteq (V \times V \times \{ep\})$ *be an arbitrary set of edges between vertices in* V *labeled with*

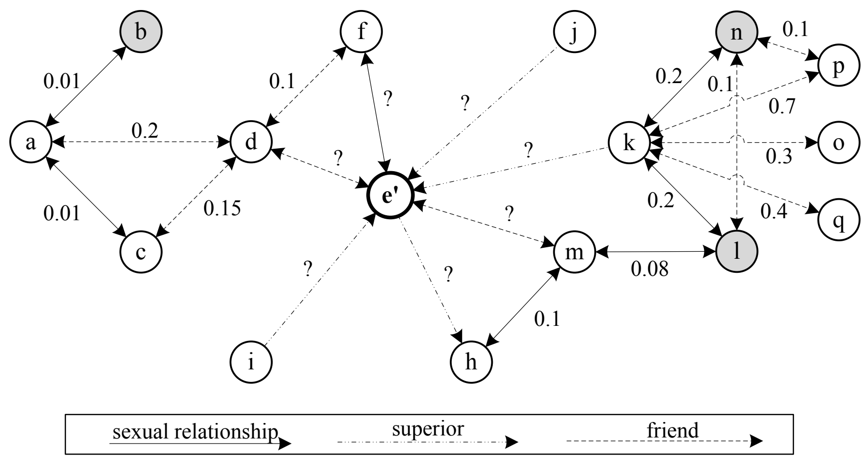


Fig. 3.10: Social network of Figure 3.9 after merging vertices e and g into vertex e'.

ep, and $g_{ep} : (0,1]^m \rightarrow (0,1]$ *be any associative and commutative function. Then, we define*

$$weight(\{e1,\ldots,e_m\}) = g_{ep}(\omega^*(e_1),\ldots,\omega^*(e_m))$$

where $\omega^*(e) = \omega(e)$ *if* $e \in E$*, otherwise* $\omega^*(e) = 0$.

Given two sets of vertices V' and V'' and an edge property $ep \in \mathsf{EP}$, we define the set $possEdges(V', V'', ep)$ as the set of all possible edges having property ep that can exist from vertices in V' to vertices in V'', i.e. $possEdges(V', V'', ep) = \{(v', v'', ep) \mid v' \in V' \wedge v'' \in V''\}$. Finally, if v' is a new vertex obtained by merging a set V' of vertices and $e' = (v, v', ep)$ is a new incoming edge then its weight is computed as $weight(possEdges(\{v\}, V', ep))$, while, if $e'' = (v', v, ep)$ is a new outgoing edge, then its weight is computed as $weight(possEdges(V', \{v\}, ep))$.

3.2.1.3 Social Network Coarsening

We are now ready to give the definition of social network coarsening.

Definition 21 (Social network coarsening)**.** *Let* $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ *be an SN and* θ *be a real number in* $(0,1]$ *called the* contraction factor*. A* coarsening *of* $\mathcal{S}$ *is an SN* $\mathcal{S}' = (V', E', \mathsf{VL}', \omega')$

together with an onto mapping $\pi : V \to V'$ satisfying the following properties:

- $|V'| \leq \theta \cdot |V|$*;*
- $E' = \{\langle \pi(v_1), \pi(v_2), ep \rangle \mid \langle v_1, v_2, ep \rangle \in E \wedge \pi(v_1) \neq \pi(v_2)\}$*;*
- $\mathsf{VL}'(v') = \mathit{mergeVP}(\{v \in V \mid \pi(v) = v'\}, \mathsf{VL})$ *for each* $v' \in V'$*;*
- $\omega'(e') = \mathit{weight}(\mathit{possEdges}(\pi^{-1}(v_1), \pi^{-1}(v_2), ep))$*, for each* $e' = \langle \pi(v_1), \pi(v_2), ep \rangle \in E'$*.*

Thus, coarsening a social network $\mathcal{S}$ yields a new social network $\mathcal{S}'$ together with a mapping π from the vertices of $\mathcal{S}$ to the vertices of $\mathcal{S}'$ such that *(i)* the number of vertices of $\mathcal{S}'$ is smaller than that of $\mathcal{S}$ by a factor θ; *(ii)* if there was an edge between two vertices u and v in $\mathcal{S}$, and u has been merged into a new vertex u' in $\mathcal{S}'$ while v has been merged into a new vertex $v' \neq u'$ in $\mathcal{S}'$, then there is an edge between u' and v' in $\mathcal{S}'$; *(iii)* the vertex properties of each merged vertex of $\mathcal{S}'$ are assigned using function $\mathit{mergeVP}$; and *(iv)* the weights of the edges of $\mathcal{S}'$ are assigned by function weight.

Consider an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$ and let $\mathcal{S}' = (V', E', \mathsf{VL}', \omega')$ and $\pi : V \to V'$ be a coarsening of $\mathcal{S}$. Given $V'' \subseteq V'$, with a slight abuse of notation, we denote by $\pi^{-1}(V'')$ the set $\{v \in V \mid \pi(v) \in V''\}$.

Algorithm 7 CoarsenSN

Input: A social network $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP Π, a contraction factor $\theta \in (0, 1]$, a similarity function SIM, a neighbors threshold $\rho \in (0, 1]$,

, a vertex properties merging function $mergeVP$, and an edge $weight$ function.

Output: A coarsening $\mathcal{S}' = (V', E', \mathsf{VL}', \omega')$ and $\pi : V \to V'$ of $\mathcal{S}$.

```
1:  $\mathcal{S}' = (V', E', \mathsf{VL}', \omega') = (V, E, \mathsf{VL}, \omega)$;
2:  $\pi(v) \leftarrow v$ for all $v \in V$;
3:  $R \leftarrow V'$, $cont \leftarrow true$;
4:  **while** $((|V'| > \theta \cdot |V|) \wedge cont)$ **do**
5:      Randomly select a vertex $v \in R$;
6:      $M \leftarrow getMergingSet(v, \mathcal{S}', \Pi, SIM, \rho)$; $R \leftarrow R \setminus (M \cup \{v\})$;
7:      **if** $(M \neq \emptyset)$ **then**
8:          $\mathsf{VL}'(v) \leftarrow mergeVP(\{v\} \cup M, \mathsf{VL}')$;
9:          $\pi(v') \leftarrow v$ for all $v' \in \pi^{-1}(v)$; $\pi(v') \leftarrow v$ for all $v' \in M$;
10:         $V' \leftarrow V' - M$; $\langle E', \omega' \rangle \leftarrow UpdateEdges(E', \omega', v, M, weight, \pi)$;
11:     **if** $(R = \emptyset)$ **then**
12:         $cont \leftarrow false$;
13: **return** $\langle \mathcal{S}', \pi \rangle$;
14:
15: *getMergingSet*$(v, \mathcal{S}', \Pi, SIM, \rho)$
16:     $U =$ all neighbors of vertex $v$ in $\mathcal{S}'$;
17:     $U' = U \cap SIM(v)$;
18:     Randomly select a set $M \subseteq U'$ of size $|M| = \rho \cdot |U'|$ ;
19:     **return** $M$;
20:
21: *updateEdges*$(E', \omega, v, M, weight, \pi)$
22:     $E'' = \{\langle \pi(v_1), \pi(v_2), ep \rangle \mid \langle v_1, v_2, ep \rangle \in E' \wedge \pi(v_1) \neq \pi(v_2)\}$;
23:     **for each** $e' = \langle \pi(v_1), \pi(v_2), ep \rangle \in E''$ **do**
24:         $\omega'(e') \leftarrow weight(possEdges(\pi^{-1}(v_1), \pi^{-1}(v_2), ep))$ ;
25:     **return** $\langle E'', \omega' \rangle$;
```

Algorithm 7 is a general algorithm for coarsening a social network. It starts by initializing the mapping function π as the identity function. In each iteration, a vertex v in the current SN is randomly selected, and the set of vertices to be merged with it is computed by the function $getMergingSet$. This function computes the set U' of all v's neighbors that are similar to v according to the similarity function SIM received as input, and returns a subset M of U' whose size is a percentage ρ of $|U'|$. If the set of vertices M is not empty, vertex v is merged with M, otherwise a new vertex v is randomly selected. If v is merged with M, the new vertex properties of v are computed using the $mergeVP$ function, the mapping π is updated, and the set of edges is updated by the function $UpdateEdges$. This function first computes the new set of edges so that if there was an edge between a vertex in $M \cup \{v\}$ and another vertex u, now there is an edge between v and u, while the new edge weight is computed by using the function $weight$ received in input. The algorithm's iterations stop when either the number $|V'|$ of vertices in the current SN is less than or equal to $\theta \cdot |V'|$, where θ is the contraction factor establishing the desired size of the coarsened network, or it is not possible to merge any other vertex. Of course, Algorithm 7 can be used with our similarity function SIM_{Π}.

3.2.2 Adapting the DC Definition to the Coarsened Network

We now adapt diffusion centrality to the case of coarsened networks. This intermediate step will later be used in the computation of diffusion centrality for vertices of the original network.

When a set of vertices M in the original network $\mathcal{S}$ is merged into a single vertex v' in the coarsened one, v' represents the network $\mathcal{S}_{v'}$ consisting of all vertices in M and all edges among them from the original network. Thus, even if two vertices v' and v'' have the same diffusion centrality on the coarsened network, the actual diffusion of the property of interest in the original network among the vertices belonging to $\mathcal{S}_{v'}$ and $\mathcal{S}_{v''}$ may be different and depends on the properties of the two subnetworks $\mathcal{S}_{v'}$ and $\mathcal{S}_{v''}$ (number of vertices, edges, density, etc.). To take this into account, we assign a weight to each vertex v' in the coarsened network representing the importance of vertex v' w.r.t. to the original network.

Definition 22 (Diffusion centrality on a coarsened network)**.** *Let $\mathcal{S}$ be a social network, $\mathcal{S}_C = (V_C, E_C, \mathsf{VL}_C, \omega_C)$ be a coarsening of $\mathcal{S}$ with vertex mapping π, and mvw be a function assigning a weight to each vertex in V_C. Then, the diffusion centrality of a vertex v in the coarsened SN is defined as*

$$\begin{aligned}\mathsf{dc}'_{\Pi,p,\mathcal{S}_C}(v) \quad = \quad & \Sigma_{v' \in V_C - \{v\}} \big(mvw(v') \cdot \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}_C \oplus p(v)})(p(v')) \big) - \\ & \Sigma_{v'' \in V_C - \{v\}} \big(mvw(v'') \cdot \mathsf{lfp}(\Pi \cup \Pi_{\mathcal{S}_C \ominus p(v)})(p(v'')) \big)\end{aligned}$$

Function mvw, which we call *merged vertex weight* function, can be defined in several ways. Below we provide three alternative definitions. Consider an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, and let $\mathcal{S}_C$ and π be a coarsening of $\mathcal{S}$. Given a vertex v of $\mathcal{S}_C$,

- $mvw_0(v) = 1$. In this case, the original definition of DC is used.

- $mvw_1(v) = |V_v| \times \frac{|E_v|}{|V_v|^2 - |V_v|} = \frac{|E_v|}{|V_v| - 1}$,

  where $V_v = \pi^{-1}(\{v\})$ and $E_v = \{\langle v_1, v_2, ep \rangle \mid v_1, v_2 \in V_v \wedge \langle v_1, v_2, ep \rangle \in E\}$. In this case, the weight of v is given by the number of vertices in the original SN that were merged to form v multiplied by the density of the (sub)network $\mathcal{S}_v$ represented by v. The idea behind this is that if $\mathcal{S}_v$ has high density and many vertices, the weight of v should be high.

- $mvw_2(v) = \ln(mvw_1(v)) + 1$.

  Here, we consider the fact that the diffusion of the property p can rapidly decrease according to the distance of vertices in V_v from the diffusion source vertex v. Thus, $mvw_2(v)$ reduces the effect of $mvw_1(v)$ on the diffusion centrality by taking its natural log plus 1.

Algorithm 8 CBAF (Approximate Top-k)

```
Input: An SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a GAP $\Pi$, a property $p$, an integer $k$,
    a set of options $opts$ as described in Table 3.4.
Output: An approximate set of top-$k$ vertices.
1: $\mathcal{S}' = (V', E', \mathsf{VL}', \omega') \leftarrow networkFiltering(\mathcal{S}, \Pi, p)$
2: $\langle \mathcal{S}_C = (V_C, E_C, \mathsf{VL}_C, \omega_C), \pi \rangle \leftarrow CoarsenSN(\mathcal{S}', \Pi, opts)$
3: $T_C \leftarrow computeTopK(\mathcal{S}_C, k, mvw)$ % Use HyperDC
4: $T_C = T_C \cup nbrs(T_C, d_C, \mathcal{S}_C)$
5: $V_I = \pi^{-1}(T_C) \cup nbrs(\pi^{-1}(T_C), d_I, \mathcal{S})$
6: $\mathcal{S}_I = (V_I, E_I, \mathsf{VL}_I, \omega_I) \leftarrow$ SN induced from $\mathcal{S}$ by the vertices in $V_I$
7: return $computeTopK(\mathcal{S}_I, k, mvw_0)$
```

3.2.3 CBAF: Approximately Solving the kDCP Problem

In this subsection we show how to approximately compute the top-k diffusion centrality vertices in a social network $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$. First, we introduce some definitions.

Definition 23 (Induced social network)**.** *Given an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, the SN* induced *from $\mathcal{S}$ by a set of vertices $V_I \subseteq V$ is $\mathcal{S}_I = (V_I, E_I, \mathsf{VL}_I, \omega_I)$, where $E_I = \{\langle v_1, v_2, ep\rangle \mid \langle v_1, v_2, ep\rangle \in E \wedge v_1, v_2 \in V_I\}$, $\mathsf{VL}_I(v) = \mathsf{VL}(v)$ for all $v \in V_I$, and $\omega_I(e) = \omega(e)$ for all $e \in E_I$.*

Given an SN $\mathcal{S} = (V, E, \mathsf{VL}, \omega)$, a set of vertices $T \subseteq V$, and a positive integer d, we denote by $nbrs(T, d, \mathcal{S})$ the set of the neighbors of the vertices in T at distance at most d. If $d = 0$, then $nbrs(T, d, \mathcal{S}) = \emptyset$.

$\theta \in (0, 1]$	contraction factor	mvw	merged vertex weight function
SIM	similarity function	d_C	neighbors distance for extending
$\rho \in (0, 1]$	neighbors threshold		the set T_C
$mergeVP$	vertex properties merging function	d_I	neighbors distance for extending
$weight$	edge weight function		the set $\pi^{-1}(T_C)$

Tab. 3.4: Input options $opts$ for Algoritm 8.

We are now ready to present our CBAF algorithm shown in Algorithm 8. CBAF takes as input a social network $\mathcal{S}$, a GAP Π, a property p, an integer k, and a set of options $opts$ as described in Table 3.4. It returns an (approximate) set of top-k diffusion centrality vertices over $\mathcal{S}$. The first step of the algorithm filters out the original SN $\mathcal{S}$ by removing all unnecessary vertices, obtaining the network $\mathcal{S}'$ (line 1). Then $\mathcal{S}'$ is coarsened into a smaller social network $\mathcal{S}_C$ (line 2), and the exact set T_C of top-k vertices over $\mathcal{S}_C$ is computed (line 3) by running HyperDC to find the diffusion centrality of all vertices in $\mathcal{S}'$ and choosing the top-k. At this point the set T_C is extended with its neighbors in $\mathcal{S}_C$ at distance at most d_C so that the following computation is not biased by the vertices in T_C (line 4). Next, the above set of vertices is mapped back to the vertices of the original SN $\mathcal{S}$ and is extended with its neighbors on $\mathcal{S}$ at distance at most d_I obtaining the set of vertices V_I (line 5). After that, the social network $\mathcal{S}_I$ induced from the vertices in V_I on $\mathcal{S}$ is computed (line 6), and the algorithm returns the approximate set of top-k vertices over $\mathcal{S}$ as the exact set of top-k diffusion centrality vertices computed over $\mathcal{S}_I$ (line 7). Note that CBAF first evaluates diffusion centrality of all vertices in the coarsened network $\mathcal{S}'$ (which is typically small) and then diffusion centrality of all vertices in a subgraph of the original network corresponding to the neighborhood of vertices associated with the solution found in line 3 of the CBAF algorithm. This is typically a small fraction of the vertices in the original social network $\mathcal{S}$.

3.2.4 CBAF Algorithm: Performance Experiments

We implemented CBAF (Algorithm 8) in Java and tested scalability, runtime, and spread of CBAF with networks of up to 2M vertices and 20M edges (Section 3.2.4). In this section, we describe experiments we performed to compare CBAF with HyperDC in terms of both runtime and spread. Simply put, these experiments show that CBAF almost always achieves the same spread as HyperDC with a runtime that is always lower (with the correct choice of settings) than HyperDC—moreover, sometimes it takes less than half the runtime of HyperDC.

Input Options for CBAF. There are a number of input options for CBAF that influence its performance. In order to find the best possible input options, we ran extensive experiments in which 1944 different candidate option sets were considered. The three option sets that achieved a good balance between runtime and spread are reported in Table 3.5—these were the options we used in our experiments with CBAF.

label	d_C	d_I	$weight$	$mergeVP$	mvw	ρ
OP_1	1	0	max	$majority$	mvw_1	1.0
OP_2	1	0	max	$intersect$	mvw_1	1.0
OP_3	1	0	$average$	$majority$	mvw_0	1.0

Tab. 3.5: Best Input Options for CBAF

In all experiments, we used the vertex similarity function SIM_Π defined earlier. The contraction factor θ was chosen from the set $\{0.2, 0.3, 0.4\}$.

CBAF Evaluation Measures. We used two measures to evaluate CBAF. The time ratio is simply the ratio of time taken by CBAF vs. HyperDC. The spread ratio is the ratio of spread according to CBAF (assuming the top-k vertices have property p) vs. that according to HyperDC.

We experimentally compared HyperDC and CBAF over 4 datasets — these included the two largest STEAM networks (GAME420 with 1.3M vertices and 12.43M edges, GAME220 with 2.03M vertices and 20.59M edges) and the 2 largest non-game social networks (Email-Eu and Douban) together with the Flickr, Jackson-Yariv, and SIR diffusion models. In all the experiments we set $\delta_p = 0$ and $\delta_q = 5\%$.

Tables 3.6 and 3.7 show the experimental comparisons between CBAF and HyperDC over the Email-Eu and Douban networks, using the options of Table 3.5, with $k = 100$, and $\theta \in \{0.4, 0.3, 0.2\}$.

For the huge networks GAME420 and GAME220 we ran the experiments with $\theta = 0.4$ and $k \in \{0.01\%, 0.05\%, 0.10\%\}$. In Table 3.8 we present the results only for the network GAME220, which is the largest one, because the results for GAME420 are quite similar to the ones for the GAME220 network.

Runtime. We note that option OP_3 consistently yields the fastest runtimes for CBAF. The time ratios are always less than one in all networks and for all diffusion models with the exception of the Jackson-Yariv model in the huge networks with $k = 0.10\%$. In particular, on the Douban network with the SIR model the time ratio is less than 50%, while delivering an almost perfect spread ratio of 0.99. On the Email-Eu network using the Jackson-Yariv model, it runs in just over 50% of the time taken by HyperDC. On the huge GAME220 network using the Flickr model it runs in under 60% of the time taken by HyperDC. CBAF does not work well for the Jackson-Yariv model in the two huge networks because of the high average degree of these networks.

Spread. In all cases, all three options yield approximately the same spread. In the huge networks and the Douban network, the spread ratio is always close to one. On the Email-Eu network, the spreads range from 0.8-0.9 for the Jackson-Yariv and SIR models. However, for the Flickr model, the spread is lower, mostly in the 0.6-0.7 range. The reason for this is that the Email-Eu network has a very low average degree which leads to merged vertices (in the coarsened networks) representing only small sets of vertices of the original network so that the induced network is small and not representative enough to compute approximate top-k vertices well.

Summary of Experiments. We compares HyperDC with CBAF. We show that CBAF almost always achieves the same spread as HyperDC with a runtime that is always lower (with the correct choice of settings) than HyperDC - moreover, sometimes, it takes less than half the runtime of HyperDC.

Tab. 3.6: Results over the Douban network.

Model	Option set	θ	$ratio_{time}$	$ratio_{spread}$
Flickr	OP_1	0.4	0.67	0.96
		0.3	0.68	0.96
		0.2	0.74	0.96
	OP_2	0.4	0.70	0.94
		0.3	0.69	0.94
		0.2	0.70	0.94
	OP_3	0.4	0.63	0.96
		0.3	0.62	0.97
		0.2	0.64	0.97
JY	OP_1	0.4	1.04	0.93
		0.3	1.00	0.93
		0.2	1.12	0.93
	OP_2	0.4	1.00	0.98
		0.3	1.03	0.98
		0.2	1.04	0.98
	OP_3	0.4	0.93	0.93
		0.3	0.91	0.94
		0.2	0.91	0.94
SIR	OP_1	0.4	0.79	1.00
		0.3	0.81	1.00
		0.2	0.81	1.00
	OP_2	0.4	0.79	1.00
		0.3	0.81	1.00
		0.2	0.82	1.00
	OP_3	0.4	0.46	0.99
		0.3	0.48	0.99
		0.2	0.47	0.99

Tab. 3.7: Results over the Email-eu network.

Model	Option set	θ	$ratio_{time}$	$ratio_{spread}$
Flickr	OP_1	0.4	1.27	0.69
		0.3	1.26	0.63
		0.2	1.37	0.64
	OP_2	0.4	1.16	0.61
		0.3	1.17	0.57
		0.2	1.19	0.59
	OP_3	0.4	0.83	0.61
		0.3	0.82	0.67
		0.2	0.83	0.62
JY	OP_1	0.4	0.72	0.82
		0.3	0.77	0.81
		0.2	0.73	0.81
	OP_2	0.4	0.63	0.82
		0.3	0.68	0.82
		0.2	0.66	0.83
	OP_3	0.4	0.52	0.82
		0.3	0.55	0.82
		0.2	0.66	0.83
SIR	OP_1	0.4	0.93	0.90
		0.3	0.85	0.91
		0.2	0.88	0.88
	OP_2	0.4	0.69	0.91
		0.3	0.72	0.91
		0.2	0.72	0.92
	OP_3	0.4	0.72	0.90
		0.3	0.76	0.89
		0.2	0.76	0.91

Tab. 3.8: Results over the GAME220 network.

Model	k ($\|seeds\|/\|V\|$)	Option set	$ratio_{time}$	$ratio_{spread}$
Flickr	0.01%	OP_1	0.61	0.98
		OP_2	0.70	0.98
		OP_3	0.52	0.99
	0.05%	OP_1	0.69	0.97
		OP_2	0.76	0.99
		OP_3	0.45	0.99
	0.10%	OP_1	0.60	0.99
		OP_2	0.85	0.99
		OP_3	0.60	0.94
JY	0.01%	OP_1	1.09	0.98
		OP_2	1.10	0.97
		OP_3	0.95	0.97
	0.05%	OP_1	1.25	0.99
		OP_2	1.11	0.98
		OP_3	0.95	1.00
	0.10%	OP_1	1.18	0.95
		OP_2	1.19	0.98
		OP_3	1.04	1.00
SIR	0.01%	OP_1	0.96	0.97
		OP_2	1.13	1.02
		OP_3	0.66	0.97
	0.05%	OP_1	1.19	1.00
		OP_2	1.23	0.99
		OP_3	0.72	0.98
	0.10%	OP_1	1.05	0.98
		OP_2	1.07	0.98
		OP_3	0.64	1.00

3.3 Related Work

The idea of coarsening a network for some task is not new, and has been used extensively in the popular community detection techniques (METIS [125] and GRACLUS [69]): nevertheless, they use different metrics for coarsening like cut-based, flow-based or heavy-edge matching-based conditions. In contrast we study diffusion-based metrics, and do not aim to find communities.

The related problem of graph sparsification has also been well studied in the theory community under the notion of "spanners" [75]. A spanner is a sparse subgraph that maintains the pairwise distances between all nodes within a multiplicative or additive factor. Fung et al. [84] study the cut-sparsifier problem which asks for a sparse weighted subgraph such that the weight of all cuts is maintained within a small multiplicative factor. Graph sparsification for influence analysis has emerged as a new tool for analyzing large networks. Mathioudakis et al. [151] propose an algorithm to find the sparse backbone of an influence network. The major difference is that graph sparsification *removes* edges (so the nodes stay the same), while we *coarsen* and *contract* node-pairs to reduce the graph. Another line of very recent work [155] tries to learn influence models at community-scale, using groups supplied by graph-partitioning algorithms like METIS. Our work is related in the sense that we also aim to 'group' nodes, but not based on link-based communities, instead automatically based on nodes' diffusion characteristics. In that sense we believe our work provides a complementary viewpoint: learn models directly at the node level, and then try to group them appropriately automatically.

The rest of the related work can be categorized into Epidemic Thresholds, Influence Maximization, Other Optimization problems, and General Information Diffusion.

Epidemic Thresholds. The classical texts on epidemic models and analysis are May and Anderson [11] and Hethcote [111]. Much research in virus propagation focuses on the so-called epidemic threshold, i.e. determining the conditions under which an epidemic will not break out. Widely-studied epidemiological models include *homogeneous models* [17, 154, 11] which assume that every individual has equal contact with others in the population. While earlier works [130, 168] focus on some specific types of graph structure (e.g., random graphs, power-law graphs, etc), Chakrabarti et al. [50] and Ganesh et al. [88] found that, for the flu-like SIS model, the epidemic threshold for any arbitrary graph depends on the leading eigenvalue of the adjacency matrix of the graph. Prakash et al. [169] further extended the result to a broad class of epidemic models.

Influence Maximization: The influence maximization problem was introduced by Domingos and Richardson [174]. Kempe et al. [127] formulated it as a combinatorial optimization problem under the Independent Cascade Model, proved it is NP-Hard and gave a simple $1 - 1/e$ approximation based on the submodularity of expected spread of a set of starting seeds. Numerous follow-up papers have looked at speeding-up the algorithm (e.g., [143, 97, 53, 133, 54]).

Other Optimization Problems. Another related problem is immunization, i.e, the problem of finding the best vertices for removal to stop an epidemic, with effective immunization strategies for static and dynamic graphs [109, 196, 38]. Other such problems where we wish to select a subset of '*important*' vertices on graphs, include 'outbreak detection' [143] and finding most-likely starting points ('culprits') of epidemics [142, 170].

General Information Diffusion. There is a lot of research interest in studying dynamic processes on large graphs, (a) blogs and propagations [101, 137, 127], (b) information cascades [27, 91, 98] and (c) marketing and product penetration [176]. These dynamic processes are all closely related to virus propagation. General algorithms for information diffusion based optimization include [184].

3.4 Conclusions

We propose influence-based coarsening as a fundamental operation in the analysis of diffusive processes in large networks. Based on the connections between influence spread and spectral properties of the graph, we propose a novel Graph Coarsening Problem and provide an effective and efficient heuristic called COARSENET. By carefully reweighting the edges after each coarsening step, COARSENET attempts to find a succinct representation of the original network which preserves important diffusive properties. We then describe the CSPIN framework to solve influence maximization problems on large networks using our coarsening strategy. Experimental results show that CSPIN indeed outperforms traditional approaches by providing high quality solutions in a fraction of the time. Finally we show that our COARSENET framework can also be used for examining cascade datasets in an exploratory setting. We observe that in our case study the nodes merged together form meaningful communities in the sense of having similar diffusive properties which can serve as surrogates using external demographic information.

In addition, we present a very novel (but approximate) Coarsening Back and Forth (CBAF) algorithm that allows us to take a huge social network, reduce it to a manageable size, solve the problem of finding the top-k vertices on the coarsened network, and then pull the results back onto the original. We conduct a very detailed experimental study on several real-world social networks. The experiments looks at the scalability of CBAF, showing that it almost always has a lower runtime than HyperDC, while achieving high spreads. In particular, CBAF was tested on networks with over 2M vertices and over 20M edges and achieved acceptable runtime.

Chapter 4: STUN:Querying Spatio-Temporal Uncertain (Social) Networks [1]

4.1 Introduction

There are now numerous ongoing efforts to infer spatial, temporal, and other properties of social networks. Fig. 4.1 shows a small example of a Facebook style graph that represents various friend relationships, organized relationships showing who organized a particular event, and attended relationships showing who attended certain events. In such applications, edge relationships can be quantified via three types of labels characterizing the relationship. **Uncertainty** arises when applications infer edges from Facebook style social networks, e.g. inferring whether A and B are friends by analyzing the original Facebook friend relationship. **Time** arises even in original Facebook friend relationships where the friendship begins at a certain time and may end at a certain time. **Space** arises when applications infer the location of an individual or an event which has a Facebook page.

The friend edge in Fig. 4.1 from Jon to Liz labeled $(1, [1, 20])$ says that Jon friended Liz at time 1 upto time 20 (current time) and the certainty that the friendship exists is 1. The friend edge from Jim to Ed labeled $(0.7, [10, 20])$ on the other hand may be an inferred friendship link saying Jim is a friend of Ed with 70% certainty and the friendship is believed to have started at time 10 and continued to the current time (20).

[1] The material appearing in the chapter is based on the paper "STUN:Querying Spatio-Temporal Uncertain (Social) Networks" [**?**]

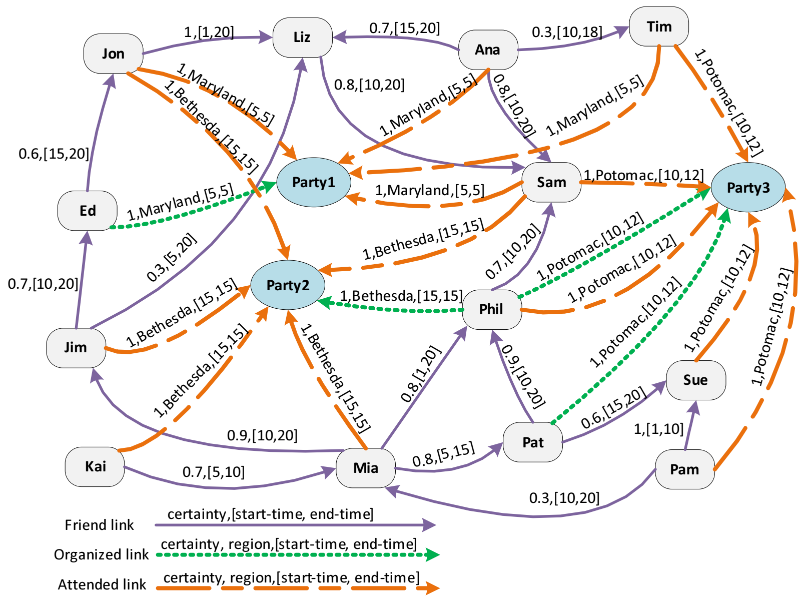


Fig. 4.1: A STUN knowledge base. The region for all friend edges is **S**

In this chapter, we present the theoretical framework underlying Spatio-Temporal Uncertain Networks (STUN), together with prototype algorithms, index structures, and a system that supports some (but certainly not all) aspects of spatio-temporal uncertainty in networks. First we formally define STUN knowledge bases and their semantics in Section 4.2. Section 4.3 defines STUN subgraph matching queries and answers. Section 4.4 describes an index structure and a query processing algorithm that uses that index structure in order to process STUN subgraph matching queries. The index hierarchically decomposes the network as well as space-time simultaneously and the query processing algorithm uses this index to effectively prune search. Section 4.5 introduces STUN ranking queries which attempts to identify important nodes in a STUN network, extending the well-known PageRank algorithm. In Section 4.6 we provide specific proposals for applying certainty, time, and space to ranking queries. Finally, Section 4.7 describes our prototype implementation, together with experiments showing the scalability of our approach.

4.2 STUN Syntax and Semantics

STUN manipulates graph data consisting of a set of vertices, and a set of labeled edges. These are then augmented with certainty, space, and time attributes. Thus in Figure 4.1 the individuals and parties are the vertices and the types of links, namely friend, organized and attended are the labels. Formally, we write $\mathcal{V}$ for the set of vertices and $\mathcal{L}$ for the set of labels, with $\mathcal{V} \cap \mathcal{L} = \emptyset$. Let **SNT** be a set of triples of the form (v, l, v'), where $v, v' \in \mathcal{V}$ and $l \in \mathcal{L}$.

In our STUN framework we add uncertainty, space, and time information to such triples. Uncertainty is added by the use of a certainty factor $c \in [0, 1]$ for each triple to form a *STUN quadruple* $(v, l, v'; c)$. For example, the quadruple $(jim, \textsf{friend}, ed; 0.7)$ represents the information that "Jim" is a friend of "Ed" with certainty 0.7.

We provide spatial information in terms of regions. We assume the existence of a spatial reference system $\mathbf{S} \subseteq [0, M] \times [0, N]$, with $M, N \in \mathbb{R}$ that contains a set of two-dimensional real-valued points. A *space point* is a member of **S** that represents an exact position in the reference system. A *region* is a set of space points in the reference system that is not necessarily connected. We use $\mathcal{R}$ for the set of regions. Throughout this paper, and in any realistic system, we assume that regions are "named" and that the names are used rather than the corresponding sets of points.

We provide time information in terms of time intervals. We assume the existence of a temporal reference system **T** that is a set of non-negative real numbers. A *time point* is a member of **T** that represents an exact time in the reference system. An *interval* is a pair $[st, et]$, with $st, et \in \mathbf{T}$ and $st \leq et$, that expresses a specific period in the reference system. Given a time interval $T = [st, et]$, we write $|T|$ for $et - st$ and $t \in T$ iff $st \leq t \leq et$.

We represent spatial and temporal information as a pair, written $[R, T]$, called a *STUN annotation*, where R is a region (or region name) and T is a time interval. Thus we will annotate each STUN quadruple with its corresponding spatial and temporal attributes. In Fig. 4.1, we use the STUN-annotation $[Bethesda, (15, 15)]$ to denote that "Phil" organized "Party2" at time 15 in the region named "Bethesda".

Definition 24 (STUN Tuple)**.** *A STUN tuple is an expression of the form* $(v, l, v'; c) : [R, T]$ *where* $(v, l, v'; c)$ *is a STUN quadruple and* $[R, T]$ *is a STUN annotation.*

The STUN tuple $(Phil, \textsf{organized}, Party2; 1) : [Bethesda, (15, 15)]$ says that "Phil" organized "Party2" with certainty 1 and the event occurred at time 15 at some location within the region "Bethesda", but we do not know exactly where.

A *STUN knowledge base* KB is a finite set of STUN tuples. A graph representation of an example STUN knowledge base is shown in Fig. 4.1. Note that for convenience, in Fig. 4.1 for the organized and attended links we indicate the certainty factor, region, and time interval; while for friend links we indicate the certainty and time interval only; the region is taken to be **S**.

Next we move to the STUN semantics. There are two key concepts. An *interpretation* assigns a real number between 0 and 1 to every triple in **SNT** for every point in space and time. An interpretation *satisfies* a STUN tuple if it assigns a large enough certainty value to a STUN tuple for the given region and interval. Formally, a *STUN interpretation* **I** is a mapping $\mathbf{SNT} \times \mathbf{S} \times \mathbf{T} \to [0, 1]$. **I** satisfies a STUN tuple $(snt; c) : [R, T]$ iff $\forall t \in T$, $\exists p \in R$ such that $\mathbf{I}(snt, p, t) \geq c$. Thus we require that for every time value in the interval, there is a point in the region for which the interpretation assigns a value at least as big as c to the triple. A STUN knowledge base is *consistent* if there is an interpretation that satisfies all its STUN tuples.

It should be noted that, as things stand, using this semantics makes every STUN knowledge base consistent because the interpretation that maps everything to 1 satisfies all the STUN tuples. In order to make the concept of consistency correspond better to real-world situations, we may introduce integrity constraints.

As in this chapter we do not deal with consistency checking, we do not define a formal syntax for STUN integrity constraints. We just indicate what such a constraint might look like applying a logic formalism to the STUN knowledge base of Fig. 4.1. Here the notation $\leftarrow F$ indicates that formula F is not allowed to hold in a STUN interpretation.

Consider first

$$\leftarrow (v, \ell, v'; c) : [R, T], (v, \ell, v'; c') : [R, T], c < c'.$$

This integrity constraint states that there cannot be two links that are identical in all respects except that the certainty values are different.

Next, consider that all organized labels going to the same vertex have the same time interval. As an integrity constraint we can write this as

$$\begin{aligned} \leftarrow \quad & (v, \mathsf{organized}, v''; c) : [R, T], \\ & (v', \mathsf{organized}\, v''; c') : [R', T'], \\ & T \neq T', c > 0, c' > 0. \end{aligned}$$

A similar constraint holds for the attended labels as well as for regions. However, the corresponding constraint for friend

$$\begin{aligned} \leftarrow \quad & (v, \mathsf{friend}\, v''; c) : [R, T], \\ & (v', \mathsf{friend}\, v''; c') : [R', T'], \\ & T \neq T', c > 0, c' > 0. \end{aligned}$$

does not hold for v = “Kai” v' = “Pam”, v” = “Mia”, $c = 0.7$, $c' = 0.3$, $R = R' = \mathbf{S}$, $T = [5, 10]$, and $T' = [10, 20]$.

4.3 STUN Subgraph Matching Queries

Using a STUN knowledge base involves asking queries. For this purpose we need to define the syntax of such queries. In this section we deal with queries that match subgraphs. Recall first that the STUN syntax already contains the set of vertices: $\mathcal{V}$, labels: $\mathcal{L}$, regions: $\mathcal{R}$, and intervals: $\mathcal{I}$. These are the domains of interest: $\mathcal{D} = \{\mathcal{V}, \mathcal{L}, \mathcal{R}, \mathcal{I}\}$ where $\mathcal{R} = 2^{\mathbf{S}}$ and $\mathcal{I} = \{[st, et] \,|\, st, et \in \mathbf{T},\ st \leq et\}$. The elements of these domains are the constants. For each domain $\sigma \in \mathcal{D}$ we assume the existence of a set VAR_σ of variables ranging over σ. Variables are expressed by strings beginning with "?". We denote the set of all variables as *VAR*. A *term* is a member of $\bigcup_{\sigma \in \mathcal{D}} \sigma \cup VAR$, that is, a term is a constant or variable.

For queries that specify spatial and temporal constraints we need a set of predicate symbols. Queries need the capability of expressing spatial concepts such as "inside" and temporal concepts such as "after". We assume the existence of a set $\mathbf{P}$ of predicate symbols representing spatio/temporal relationships. Each predicate symbol $ps \in \mathbf{P}$ has an associated arity, $arity(ps) \in \mathbb{N}$, a *signature* $sign(ps) =< \sigma_1, \ldots, \sigma_n >$ with $arity(ps) = n$ such that $\forall i \in [1, n]\ \sigma_i \in \mathcal{D}$, and a *denotation* $den(ps) \subseteq \sigma_1 \times \cdots \times \sigma_n$.

For example, the current prototype STUN system supports the spatial predicates inside, overlap, intersect, equal, and disjoint, and the temporal predicates after, before, overlap, and during. For the inside predicate, we have $arity(\textsf{inside}) = 2$, $sign(\textsf{inside}) = < \mathcal{R}, \mathcal{R} >$, $den(\textsf{inside}) = \{(r, s)\}$ where the region r is inside the region s.

A *STUN SM-query* is a subgraph matching query which is formally composed of two parts: the graph part and the constraint part. For the graph part we define the concept of an *SM-query graph tuple* as a STUN tuple where a variable may be used for each element. Formally, an SM-query graph tuple has the form $(v, l, v'; c) : [R, T]$ where $v, v' \in \mathcal{V} \cup VAR_{\mathcal{V}}$, $l \in \mathcal{L} \cup VAR_{\mathcal{L}}$,

$c \in [0,1]$, $R \in \mathit{VAR}_{\mathcal{R}}$, and $T \in \mathit{VAR}_{\mathcal{I}}$. An *SM-query constraint* is an expression of the form $ps(t_1,\ldots,t_n)$ where $ps \in \mathbf{P}$, $arity(ps) = n$, $sign(ps) = \{\sigma_1, \ldots, \sigma_n\}$, and $\forall i \in [1,n]$ $t_i \in \sigma_i \cup \mathit{VAR}_{\sigma_i}$. Thus an SM-query constraint applies one spatial or temporal predicate. An SM-query constraint is *ground* iff none of its terms are in *VAR*. A ground SM-query constraint $ps(t_1,\ldots,t_n)$ is *true* iff $(t_1,\ldots,t_n) \in den(ps)$.

Next we define the concept of a STUN SM-query and query answer.

Definition 25 (STUN SM-Query)**.** *A* STUN SM-query *q is a pair (G_q, C_q) where G_q is a set of SM-query graph tuples and C_q is a set of SM-query constraints.*

The output variables of an SM-query may be just a subset of those present in the query. To clarify this issue we underline the names of the variables that are required to be in the output.

We write two SM-queries in graph form in Fig. 4.2. The one on top asks for all people $\underline{?I}$ who attended a party in Maryland at timepoint 5, with certainty at least 0.5. This query can be written as the STUN SM-query $q = (G_q, C_q)$, where $G_q = \{(\underline{?I}, \text{attended}, ?P; 0.5) : [?s, ?t]\}$ and $C_q = \{inside(?s, Maryland), during(?t, [5,5])\}$. The second query is more

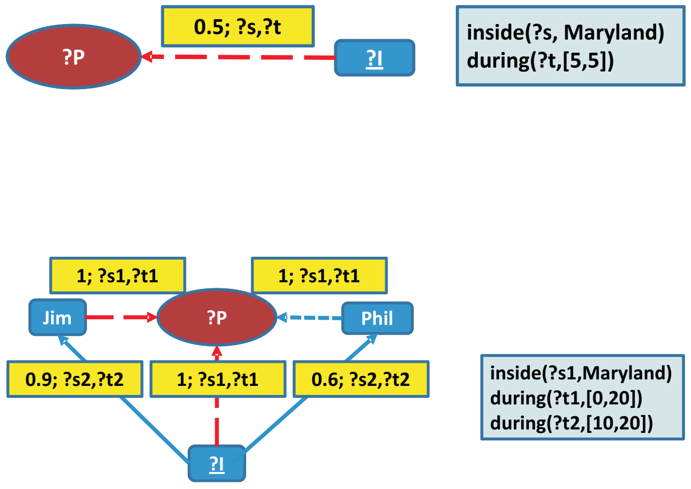


Fig. 4.2: Examples of STUN SM-queries

complicated. It asks for all people $\underline{?I}$ who have been a friend of Jim in the time interval $[10, 20]$ with certainty at least 0.9 as well as a friend of Phil in the same time interval with certainty at least 0.6 and who attended a party in Maryland organized by Phil that occurred during the time interval $[0, 20]$ and was attended by Jim. This query can be written as $q = (G_q, C_q)$, where $G_q = \{(\underline{?I}, \textsf{attended}, ?P; 1.0) : [?s1, ?t1]), (\underline{?I}, \textsf{friend}, Jim; 0.9) : [?s2, ?t2]), (Jim, \textsf{attended}, ?P; 1.0) : [?s1, ?t1]), (\underline{?I}, \textsf{friend}, Phil; 0.6) : [?s2, ?t2]), (Phil, \textsf{organized}, ?P; 1.0) : [?s1, ?t1])\}$ and $C_q = \{inside(?s1, Maryland), during(?t1, [0, 20]), during(?t2, [10, 20])\}$.

To define the concept of an answer to a STUN SM-query we start with substitutions. A *substitution* θ maps variables to ground terms and each ground term to itself.

Possible substitutions for the first query of Example 4.3 are:

- $\theta_1 = \{\underline{?I}/$"Jon", $?P/$"Party1", $?s/$"Maryland", $?t/[5, 5]\}$;
- $\theta_2 = \{\underline{?I}/$"Sam", $?P/$"Party1", $?s/$"Maryland", $?t/[5, 5]\}$;
- $\theta_3 = \{\underline{?I}/$"Sam", $?P/$"Party1", $?s/$"Potomac", $?t/[10, 15]\}$;
- $\theta_4 = \{\underline{?I}/$"Ana", $?P/$"Sue", $?s/$"Maryland", $?t/[5, 5]\}$.

For the second query we could have:

- $\theta_5 = \{\underline{?I}/$"Mia", $?P/$"Party2", $?s1/$"Bethesda", $?t1/[15, 15], ?s2/$"Bethesda", $?t2/[1, 20]\}$;
- $\theta_6 = \{\underline{?I}/$"Mia", $?P/$"Sue", $?s1/$"Bethesda", $?t1/[15, 15], ?s2/$"Bethesda", $?t2/[1, 20]\}$.

We denote the application of θ to a term x as $x\theta$. Likewise, we denote the application of θ to all of the terms appearing in an expression X as $X\theta$.

Definition 26 (STUN SM-Query Answer)**.** *A substitution θ is an* answer *to a STUN SM-query $q = (G_q, C_q)$ w.r.t. a STUN knowledge base KB iff*

- $\forall (v, l, v'; c) : [R, T] \in G_q$, $\exists c' \geq c$ *s.t.* $(v\theta, l\theta, v'\theta; c') : [R\theta, T\theta] \in KB$*, and*

- $\bigwedge_{c_q \in C_q} c_q\theta$ *is true.*

Substitutions θ_1 and θ_2 of Example 4.3 are answers to the first query of Example 4.3, whereas θ_3 and θ_4 are not. Likewise, θ_5 is an answer to the second query, and θ_6 is not.

4.4 Indexing and SM-Query Answering

In order to efficiently answer SM-queries we use an index that exploits the type of information stored in a STUN knowledge base. This section gives the details of the indexing algorithm and how it is used to answer SM-queries. The STUN index is a tree each of whose nodes occupies a disk page and represents a portion of the STUN knowledge base (i.e., of the corresponding graph). In addition, each inner node "captures" the subgraph represented by its two children. Each level of the index tree represents a partition not only of the whole graph, but also a partition of the entire spatial reference system **S** and the entire temporal reference system **T**. The latter two are induced by the graph partition. Before describing these in detail, we need to introduce the notions of k-merges of a graph, and minimal bounding rectangles (and time intervals).

In a k-*merge*, multiple graphs are mapped into a new graph containing at most k vertices – every vertex (resp. edge) in the "merged" graph represents one or more vertices (resp. edges) in the original graph. Let $G = (V, E)$ be a graph. Consider graphs $G_i = (V_i, E_i)$, with $i \in [1, f]$, such that for all $i \in [1, f]$ $V_i \subseteq V$, $k \leq max_{i \in [1,f]}(|V_i|$, and $E_i \subseteq E$. The k-merge of $G_1, G_2, \ldots, G_f$ w.r.t. G is a graph $G_m = (V_m, E_m)$ such that (*i*) $|V_m| = k$; (*ii*) there is a *merge mapping* $\mu : \bigcup_{i \in [1,f]} V_i \rightarrow V_m$ (given a vertex $v \in V_m$, we call the vertices which are mapped to v the *represented* vertices of v, $rep(v) = \{v' \in \bigcup_{i \in [1,f]} V_i \mid \mu(v') = v\}$), and (*iii*) $e_m = (v_1, v_2) \in E_m$ iff $\exists v'_1 \in rep(v_1), v'_2 \in rep(v_2)$ such that $e = (v'_1, v'_2) \in E$.

Graph G_L associated with the left child L of the root of the tree in Fig. 4.3 is a 3-merge of graphs G_1, G_2 associated with the children of L. In particular, (*i*) vertex 1 of G_L represents vertex 1 of G_1, (*ii*) vertex 2 of G_L represents vertices 2 of G_1 and 1 of G_2, and (*iii*) vertex 3 of G_L represents vertices 3 of G_1 and $2, and 3$ of G_2.

In order to handle space and time we use the well known notion of *minimum bounding rectangles* (*MBR*s) of space regions and define *minimum bounding intervals* (*MBI*s) of time intervals. The MBR of a region R is a quadruple $MBR(R) = (min(X), max(X), min(Y), max(Y))$ where $X = \{x|(x,y) \in R\}$ and $Y = \{y|(x,y) \in R\}$.[2] The *minimal bounding interval* (MBI) of a set of time intervals TI is the time interval $MBI(TI) = [st, et]$ where $st = min(\{st'|[st', et'] \in TI\})$ and $et = max(\{et'|[st', et'] \in TI\})$.

Each of the nodes in a STUN index tree represents a subgraph KB' of KB. We employ a vertex- and edge-*weighted undirected graph* for representing KB', denoted $WUG(KB')$, where

- the set of vertices is that of KB';
- two vertices have at most 1 edge connecting them;
- there is an edge between two vertices iff there is at least 1 edge between them in KB';
- each edge e is annotated with an MBR (denoted $e[MBR]$) and an MBI ($e[MBI]$).

The notion of k-merge is straightforwardly extended to this kind of graph. For instance, the WUG associated with the leftmost left node of the tree in Fig. 4.3 is shown below it – vertices "Ed" and "Party1" are mapped to node 1 in the leftmost inner node.

[2] Given a set of MBRs M, $MBR(M) = (min(X), max(X), min(Y), max(Y))$ where $X = \{x|(x, MX, mY, MY) \in M\} \cup \{x|(mX, x, mY, MY) \in M\}$ and $Y = \{y|(mX, MX, y, MY) \in M\} \cup \{y|(mX, MX, mY, y) \in M\}$.

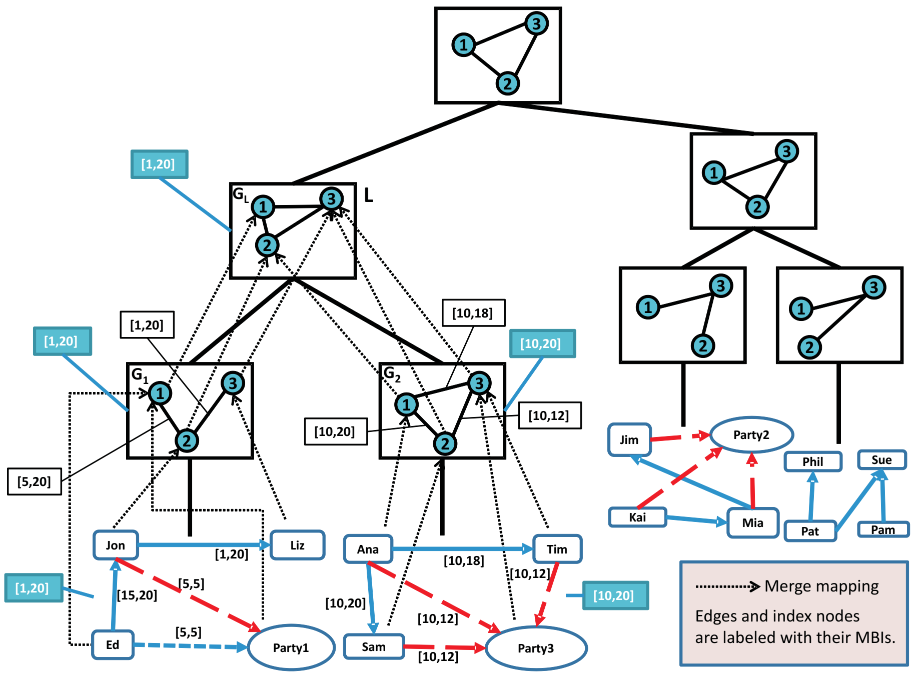


Fig. 4.3: STUN index for the knowledge base in Fig. 4.1

Definition 27 (STUN Index)**.** *Given a STUN knowledge base KB, a* STUN index $\mathcal{I}_{KB}$ of order *k ($k \geq 2$) is a balanced tree such that:*

- *Each node occupies a disk page and represents a subgraph of KB (by storing its WUG).*
- *The graphs at leaf nodes represent a vertex partition of $WUG(KB)$.*
- *If node N is the parent of nodes $N_1, N_2, \ldots, N_f$, then the graph G_N at node N is a k-merge of the graphs $G_{N_1}, G_{N_2}, \ldots, G_{N_f}$ w.r.t. $WUG(KB)$.*
- *For each node N, $N[MBR]$ (resp. $N[MBI]$) is the MBR (MBI) of the regions (intervals) associated with the STUN tuples in the subgraph of KB represented by G_N.*

Fig. 4.3 shows a STUN index of order 3 with $f = 2$. For clarity, the figure shows only the left subtree of the root in detail. In particular, the figure shows the merge mappings, the MBIs of leaf and inner nodes, and the MBIs of subgraph edges.

The tree structure of the STUN index is similar to that of DOGMA [40]. However, when building a STUN index, we wish to maximize the "quality" of the graph partitions represented by each level of the index tree by also taking into account the spatio-temporal annotations on the STUN tuples. In principle, each index node should represent a subset of the STUN graph that has (*i*) few cross edges with the subgraphs in other nodes at the same level; (*ii*) small overlap with the regions in the subgraphs in other nodes at the same level; (*iii*) small overlap with the time intervals in the subgraphs in other nodes at the same level.

Fig. 4.4 shows the buildSTUNIndex algorithm that, given a STUN knowledge base KB, builds a STUN index $\mathcal{I}_{KB}$. Lines 1–7 build the initial WUG that represents KB. Vertex weights are initialized to 1 (line 3). Edge weights are initialized by means of any suitable distance function δ that measures the spatio-temporal distance between two neighbor vertices (line 5). The MBR (resp., MBI) associated with each edge $e = (v, v')$ of the WUG is computed by looking at the regions (resp., intervals) of the STUN tuples connecting v and v' in KB (lines 6–7). Lines 8–11 iteratively apply Algorithm coarsenGraph which builds a new, smaller WUG along with an appropriate vertex mapping μ (all WUG's and μ's are considered global variables of the algorithms). The iteration ends when the current WUG is small enough to be stored (together with the vertex mapping) on a single disk page. Finally, on lines 12–14 the STUN index is built and returned, by applying Algorithm buildTree on the empty root of the index tree and the coarsest WUG.

Procedure coarsenGraph starts by building a copy G' of the given graph G and an identity merge mapping between the two (lines 1–2). Then, until the size of G' becomes half of that of G, the procedure repeatedly chooses two vertices v, m of G' and merges v into m. Vertex v is chosen at random and m is its closest neighbor in terms of edge weight (ties are broken at random). The edge between v and m is denoted as g. First, the procedures adds the vertex weight of v to that of m (line 7). We then consider two cases: (1) v has neighbors other than m; (2) v has no neighbor

```
Algorithm buildSTUNIndex
Input: STUN knowledge base $KB$
Output: STUN index $\mathcal{I}_{KB}$
1  $G_0 \leftarrow WUG(KB)$ with uninitialized weights, $MBR$'s and $MBI$'s
2  $\mu_0 \leftarrow$ empty mapping
3  for all $v \in V_0$, $v\text{-}weight(v) \leftarrow 1$
4  for all $e = (v, v') \in E_0$
5      $e\text{-}weight(e) \leftarrow \delta(v, v')$
6      $e[MBR] \leftarrow MBR(\{MBR(R) \,|(v, l, v'; c) : [R, T] \in KB\})$
7      $e[MBI] \leftarrow MBI(\{T \,|\, (v, l, v'; c) : [R, T] \in KB\})$
8  $i \leftarrow 0$
9  while $G_i, \mu_i$ occupy more than 1 disk page
10     $i \leftarrow i + 1$
11     $G_i, \mu_i \leftarrow$ coarsenGraph($G_{i-1}, v\text{-}weight, e\text{-}weight$)
12 $root(\mathcal{I}_{KB}) \leftarrow$ empty node $R$
13 buildTree($R, i, G_i$)
14 return $\mathcal{I}_{KB}$
```

Fig. 4.4: Algorithm buildSTUNIndex

```
      Procedure coarsenGraph
      Input: $G = (V, E)$, weight functions $v\text{-}weight$, $e\text{-}weight$
      Output: Coarsened graph $G'$, merge mapping $\mu$

1-2   $G' \leftarrow G$, $\mu \leftarrow$ mapping from each vertex in $V$ to its copy in $V'$
3     while $|V'| > \frac{|V|}{2}$
4-5     $v \leftarrow$ uniformly random chosen vertex from $V'$, $n \leftarrow$ number of neighbors of $v - 1$
6       $m \leftarrow$ neighbor of $v$ with minimum edge weight
7       $v\text{-}weight(m) \leftarrow v\text{-}weight(m) + v\text{-}weight(v)$, $g \leftarrow (v, m)$
9       if $n > 0$
10        for all $e = (v, u) \in E'$ with $u \neq m$
11          if $\exists f = (m, u) \in E'$
12            $e\text{-}weight(f) \leftarrow e\text{-}weight(f) + e\text{-}weight(e) + \frac{e\text{-}weight(g)}{n}$
13            $f[MBR] \leftarrow MBR(\{f[MBR], e[MBR], g[MBR]\})$
14            $f[MBI] \leftarrow MBI(\{f[MBI], e[MBI], g[MBI]\})$
15          else
16            $f \leftarrow (m, u)$
17            $e\text{-}weight(f) \leftarrow e\text{-}weight(e) + \frac{e\text{-}weight(g)}{n}$
18            $f[MBR] \leftarrow MBR(\{e[MBR], g[MBR]\})$
19            $f[MBI] \leftarrow MBI(\{e[MBI], g[MBI]\})$
20            $E' \leftarrow E' \cup \{f\}$
21          $E' \leftarrow E' \setminus \{e\}$
22      else
23        $n' \leftarrow$ number of neighbors of $m - 1$
24        for all $e = (m, u) \in E'$
25            $e\text{-}weight(e) \leftarrow e\text{-}weight(e) + \frac{e\text{-}weight(g)}{n'}$
26            $e[MBR] \leftarrow MBR(\{e[MBR], g[MBR]\})$
27            $e[MBI] \leftarrow MBI(\{e[MBI], g[MBI]\})$
28-30   $E' \leftarrow E' \setminus \{g\}$, $V' \leftarrow V' \setminus \{v\}$, $\mu(\mu^{-1}(v)) \leftarrow m$
31    return $G', \mu$
```

Fig. 4.5: Algorithm coarsenGraph

```
Procedure buildTree
Input: Tree node $N$, level $i$, $WUG\ G = (V, E)$
1  $G_N \leftarrow G$
2  $N[MBR] \leftarrow MBR(\{e[MBR] \mid e \in E\})$
3  $N[MBI] \leftarrow MBI(\{e[MBI] \mid e \in E\})$
4  if $G$ is not a subgraph of $WUG(KB)$
5    $\{G_1, \ldots, G_f\} \leftarrow$ graphPartition($G$)
6    for all $j \in [1, f]$
7      add an empty node $C_j$ as a child of $N$
8      $G_{C_j} \leftarrow$ induced subgraph in $G_{i-1}$ by vertex set $\{v \mid \mu_i(v) \in V_{G_j}\}$
9      buildTree($C_j, i-1, G_{C_j}$)
```

Fig. 4.6: Algorithm buildTree

other than m.

In case (1), the procedure considers each of v's other neighbors, denoted as u, in turn (lines 9–21). The edge between v and u is denoted as e, and it is removed after the merging process (line 21). If there is already an edge between u and m, its weight is increased by e's weight and a portion of g's weight; moreover, the MBR and MBI of the edge between u and m is extended with those of e and g (lines 11–14). If no edge between u and m exists, the procedure adds a new such edge to the graph. The weight of this edge is the sum of e's weight and a portion of g's weight, whereas the MBR and MBI of the edge are computed based on those of e and g (lines 16–20).

In case (2), the procedure adds an equal portion of g's weight to the edges connecting m to its neighbors, and extends the MBRs and MBIs of these edges with that of g (lines 22–27).

Finally, g and v are removed from the graph and the merge mapping is updated (lines 28–30).

Procedure buildTree starts by assigning the $WUG\ G$ to the index node N (which is at the

i-th level of the tree) and computing the corresponding MBR and MBI (lines 1–3). Then, if N is not a leaf node, the procedure employs any external graph partitioning algorithm graphPartition that, given a vertex- and edge-weighted graph, partitions its vertex set into f sets such that the total weight of all edges crossing the sets is minimized, and the total vertex weights are as close as possible (line 5). Finally, for each of these f vertex sets, a child node is added to N, the subgraph induced in G_{i-1} by the vertex set is assigned to N (lines 6–8), and the procedure calls itself recursively to build the child tree (line 9). Note that obviously G, being at the i-th level of the tree, is always coarser than G_{i-1}.

We conclude the description of our algorithms with the following remarks:

1. As a further improvement, during the coarsening phase we keep track of which vertices and edges are "original", i.e., present in the WUGs at leaf nodes. For each original edge, we store the set of labels associated with the STUN tuple represented by that edge (such sets can therefore be shared among different nodes of the index). Thus, if an original vertex only has original edges, we can retrieve the labels associated with these edges by just looking at the current index node.

2. We apply the notion of *internal partition distance* from [40] to reduce the size of the sets of candidate ground terms during query answering. In particular, for each index node N and each vertex v in G_N, we compute and store the distance of v from the outside of G_N. This way, besides restricting candidates to immediate neighbors of a vertex, we can apply additional constraints that take into account "longer-range" distance requirements among query vertices.

3. We employ a spatio-temporal vertex distance function that looks at the neighborhood of the two vertices and measures the "amount" of space and time the vertices share with each other

with respect to their neighborhoods. Moreover, the function weighs the "contributions" of the various edges with their associated certainty level. Given an edge (v, v') in a STUN graph, we define

$$\begin{aligned} \delta(v,v') = \quad & \alpha \cdot cT(v,v') \cdot \left(\tfrac{1}{nT(v)} + \tfrac{1}{nT(v')}\right) + \\ & \beta \cdot cS(v,v') \cdot \left(\tfrac{1}{nS(v)} + \tfrac{1}{nS(v')}\right) \end{aligned}$$

where

$$\begin{aligned} cT(v,v') &= \sum_{(v,l,v';c):[R,T] \in KB} \left[c \cdot \mathit{length}(T)\right], \\ nT(v) &= \sum_{(v,l,v';c):[R,T] \in KB} \left[c \cdot \mathit{length}(T)\right], \\ cS(v,v') &= \sum_{(v,l,v';c):[R,T] \in KB} \left[c \cdot \mathit{area}(R)\right], \\ nS(v) &= \sum_{(v,l,v';c):[R,T] \in KB} \left[c \cdot \mathit{area}(R)\right], \end{aligned}$$

functions $length$ and $area$ have their common meaning, and $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$.

4. As our graph partitioning algorithm, we employ *GGGP* [126].

4.4.1 SM-Query Answering Algorithm

Fig. 4.7 shows our answerQuery algorithm for answering STUN SM-queries using the STUN index. The algorithm has a recursive, depth-first structure and it maintains a set of candidate substitutions R_x for each variable x in the query. At each recursive invocation, it greedily chooses the next variable w to find substitutions for (line 6), by taking the one which has the smallest number of candidates (ties are broken at random). If there is no candidate for w, no further extension of the current substitution can be found and the algorithm backtracks (line 7). Otherwise, for each candidate m in R_w, it builds a new substitution with $w \to m$ and new candidate sets from the current ones (lines 8–13) by restricting the set of candidates to the "valid" neighbors of m, i.e., those that satisfy the constraints in the query, and calls itself recursively.

```
Algorithm answerQuery
Input: STUN SM-query $q = (G_q, C_q)$, substitution $\theta$,
          candidate sets $\{R_x\}$
Output: Set $A$ of substitutions
Global variables: STUN index $\mathcal{I}_{KB}$, set $A$ initialized to $\emptyset$

1   if $\theta$ maps every variable in $q$ to a ground term then
2     add $\theta$ to $A$ and end-algorithm
3   if $\theta = \emptyset$ then
4     for each variable $x$ in $q$
5       $R_x \leftarrow$ ALL // placeholder for all the vertices in $KB$
6   $w \leftarrow$ variable in $q$ having minimum $|R_w|$
7   if $R_w = \emptyset$ then end-algorithm
8   for each $m \in R_w$
9     $\theta' \leftarrow \theta \cup (w \rightarrow m)$
10    for each variable $x$ in $q\theta'$
11      $R'_x \leftarrow R_x$
12    for each variable $v$ linked to $w$ by a tuple in $G_q$
13      $R'_v \leftarrow R_v \cap$ validNeighbors$(q, \theta, R_v, m)$
14    answerQuery$(q\theta', \theta', \{R'_x\})$
```

Fig. 4.7: Algorithm answerQuery

```
Procedure validNeighbors
Input: STUN SM-query $q$, substitution $\theta$, ground term $m$
Output: Set $M$ of valid neigbhors of $m$
1  $\{(v_1, N_1), \ldots, (v_h, N_h)\} \leftarrow$ followMergeMappings$(m)$
2  $L \leftarrow N_1$
3  for each $i \in [2, h]$
4    $L \leftarrow L \cup$ findLeafNodes$(q, (v_i, N_i), (v_{i-1}, N_{i-1}))$
5  $M \leftarrow$ retrieveNeighbors$(m, L)$
6  filterInternalPartitionDistance$(q\theta, M)$
7  return $M$
```

Fig. 4.8: Algorithm validNeighbors

```
Procedure findLeafNodes
Input: STUN SM-query $q$,
       (vertex, node) pairs $(v, N), (v_e, N_e)$
Output: Set $L$ of leaf nodes of $\mathcal{I}_{KB}$
1  $D \leftarrow$ child nodes of $N$ except $N_e$
2  if $D = \emptyset$ then return $\{N\}$
3  $L \leftarrow \emptyset$
4  for each $N_d \in D$
5    if conn$(v, N, N_d)$ and checkSTConstraints$(N_d, C_q)$ then
6      $L \leftarrow L \cup$ findLeafNodes$(q, (\perp, N_d), \perp)$
7  return $L$
```

Fig. 4.9: Algorithm findLeafNodes

The validNeighbors procedure starts by extracting the list of index nodes that connect the leaf node containing m to the root node, along with all the vertices that represent m in these nodes (line 1). Observe that we only need the leaf node itself if m does not have edges that cross the boundaries of the leaf node. The algorithm then retrieves all the leaf nodes that may contain valid neighbors of m (and are not already in memory), extracts these neighbors (applying the appropriate restrictions on labels), and filters out those that do not satisfy internal partition distance constraints (lines 3–6).

The findLeafNodes procedure, given an index node N (along with the vertex v that represents m in N) and one of its child nodes, recursively finds the other child nodes N_d such that the subtree rooted in N_d may contain valid neighbors of m (lines 4–6). To achieve this, it uses (*i*) a procedure conn that checks whether a vertex in N_d is mapped to v or to one of its neighbors in N, and (*ii*) a procedure checkSTConstraints that checks whether the spatio-temporal constraints in the query can be satisfied by considering the $N_d[MBR]$ region and the $N_d[MBI]$ interval.

4.5 STUN Ranking and SR-Queries

In this section, we show how to extend the well-known PageRank algorithm [39] to STUN databases, thus taking into account certainty, space, and time. This then allows users to pose STUNRank queries which allow them to find important nodes in STUN graphs.

We start by briefly reviewing the PageRank algorithm, which is a way to give scores to vertices based on the stationary distribution of a random walk on a directed graph. Vertices score higher if they are visited more often this way.

Basic PageRank starts with a graph $G = (V, E)$ where $|V| = n$. We can assume that the vertices (pages) are numbered from 1 to n and use v to refer both to a vertex and its numerical value. A random walk is described by an $n \times n$ transition matrix M^{pr} where $m^{pr}[v, v'] = \frac{1}{out(v')}$ iff vertex v' has an outgoing edge to vertex v; 0 otherwise. We write $out(v')$ for the outdegree of v'. The Web surfing interpretation is that $m^{pr}[v, v']$ is the probability of a random surfer following a link from page v' to page v.

The *PageRank* (column) $n \times 1$ *vector* $\mathbf{p}^{pr}$ is initialized to $\mathbf{p}_0^{pr} = \mathbf{1}/n$. So initially all pages get the same rank. Then the transition matrix is applied iteratively as

$$\mathbf{p}_{i+1}^{pr} \leftarrow d \cdot M^{pr}\mathbf{p}_i^{pr} + (1-d) \cdot \mathbf{1}/n$$

where d is a real value in $[0,1]$ called the *damping factor*, $\mathbf{1}$ is an $n \times 1$ vector of all 1's, and $\mathbf{1}/n$ is called the *teleport vector*. The Web surfing interpretation is that after the j-th iteration, $\mathbf{p}_j^{pr}[v]$ is the probability of a random surfer being at page v after following j links and $1-d$ is the probability of "teleporting" to a random page or, equivalently, to introduce a new random surfer at a random page. Usually, the values of $\mathbf{p}^{pr}$ converge to acceptable error limits after a few hundred iterations. Moreover, M^{pr} is usually very sparse because the average number of non-zero elements in a column is the average outdegree of the vertices in the network.

The original PageRank formulation assumes that the PageRank of a vertex is divided equally among the out-neighborhood of the vertex. We define a transition matrix that also takes into account space, time, and uncertainty. Moreover, we use the *personalized* variant of PageRank [108] that has a non-uniform distribution of the values in the teleport vector (instead of $\mathbf{1}/n$).

In our *STUNRank* implementation users can specify how to take into account

(1) space, time, and uncertainty;

(2) the relative importance of vertices and labels;

(3) which labels, regions, time intervals, and minimum certainty level to consider.

(1) involves general concepts, while typically (3) depends on a query. Hence we define two versions of *rank flow functions*: *query-unaware* and *query-aware*. In more detail, for (1) users may define 3 *query-unaware* rank flow functions:

- $\mathcal{F}_s(v, v')$ evaluates how the space-based STUNRank of vertex v flows to vertex v';

- $\mathcal{F}_t(v, v')$ evaluates how the time-based STUNRank of vertex v flows to vertex v';
- $\mathcal{F}_c(v, v')$ evaluates how the certainty-based STUNRank of vertex v flows to vertex v'.

For (2) users may define a function $score : \mathcal{V} \cup \mathcal{L} \rightarrow \mathbb{N}$ that expresses the relative importance of vertices and labels. This function can in turn be used in the specification of the rank flow functions.

For (3), we introduce the *query-aware* versions of the rank flow functions. They have a third argument that is a *STUN ranking query* (*SR-query*) of the form

$$q = \langle L_q, R_q, T_q, \tau_q \rangle$$

where L_q is a set of labels, R_q is a set of regions, T_q is a set of time intervals, and τ_q is a certainty threshold. The meaning of q is that we consider only the portion of the graph restricted to L_q, R_q, and T_q – moreover, when looking at certainty values, we consider only those above τ_q. We require any user-provided set of functions be such that, for any $k \in \{s, t, c\}$, $\mathcal{F}_k(v, v') = \mathcal{F}_k(v, v', q_{all})$ with $q_{all} = \langle \mathcal{L}, \mathcal{R}, \{[min(\mathbf{T}), max(\mathbf{T})]\}, 0 \rangle$.

4.5.1 Computing STUNRanks

Recall from earlier that the PageRank vector $\mathbf{p}^{pr}$ is generally computed using $\mathbf{p}^{pr}_{i+1} \leftarrow d \cdot M^{pr} \mathbf{p}^{pr}_i + (1 - d) \cdot \mathbf{1}/n$. In the computation of the STUNRank vector we modify this computation to

$$\mathbf{p}_{i+1} \leftarrow d \cdot M \mathbf{p}_i + (1 - d) \cdot \mathbf{t}.$$

Hence we need to define M and $\mathbf{t}$. The new definition of $\mathbf{t}$ uses the function $score$ as follows:

$$\mathbf{t}[v] = \frac{score(v)}{\Sigma_{v' \text{in} KB} score(v')}.$$

M is defined in two different ways depending on how the iteration is done: *combine-first* uses a single matrix, whereas *combine-last* builds four different matrices then combines the resulting rank vectors. We allow users to provide weights $\alpha, \beta, \gamma \in [0,1]$ with $\alpha + \beta + \gamma \leq 1$ to specify the relative importance of the spatial, temporal, and certainty based components.

Table 4.1 shows the combine-first and combine-last definitions of transition matrices, iterations, and STUNRank vectors in both the query-unaware and query-unaware cases. Note that, in the query-aware case, besides employing the query-aware versions of the $\mathcal{F}$ functions, we modify the definitions in order to also restrict the computation of the PageRank matrix based on the given query $q = \langle L_q, R_q, T_q, \tau_q \rangle$.

4.6 Proposed Rank Flow Functions

In the previous section we stated that users may specify various rank flow functions. In this section we give our proposed rank flow functions along with the intuition behind them.

We start by defining query-unaware functions. For the space-based STUNRank we define

$$\mathcal{F}_s(v, v') = \frac{\sum_{(v,l,v';c):[R,T] \in KB} [|R| \cdot score(l)]}{\sum_{(v,l,v'';c):[R,T] \in KB} [|R| \cdot score(l)]}.$$

The intuition is that the STUNRank flowing from a vertex v to each of its out-neighbors v' is proportional to the total amounts of space occupied by the regions annotating the edges from v to v'. Regions are also weighted by the score of edge labels.

Similarly, for the time-based STUNRank we define

$$\mathcal{F}_t(v, v') = \frac{\sum_{(v,l,v';c):[R,T] \in KB} [|T| \cdot score(l)]}{\sum_{(v,l,v'';c):[R,T] \in KB} [|T| \cdot score(l)]}.$$

where the STUNRank flowing from a vertex v to each of its out-neighbors v' is proportional to the total length of the time intervals annotating the edges from v to v'.

Then again, for the certainty-based STUNRank we define

		Transition matrices	*Iterations*	*STUNRank vector*
Query-unaware	*Combine-first*	$m[v,v'] = \pi \cdot m^{pr}[v,v'] + \alpha \cdot \mathcal{F}_s(v',v) + \beta \cdot \mathcal{F}_t(v',v) + \gamma \cdot \mathcal{F}_c(v',v)$	$\mathbf{p}_{i+1} \leftarrow d \cdot M\mathbf{p}_i + (1-d) \cdot \mathbf{t}$	$\mathbf{p}$
	Combine-last	$\forall k \in \{s,t,c\}$, $m^k[v,v'] = \mathcal{F}_k(v',v)$	$\mathbf{p}^{pr}_{i+1} \leftarrow d \cdot M^{pr}\mathbf{p}^{pr}_i + (1-d) \cdot \mathbf{t}$ $\forall k \in \{s,t,c\}$, $\mathbf{p}^k_{i+1} \leftarrow d \cdot M^k\mathbf{p}^k_i + (1-d) \cdot \mathbf{t}$	$\mathbf{p} = \pi \cdot \mathbf{p}^{pr} + \alpha \cdot \mathbf{p}^s + \beta \cdot \mathbf{p}^t + \gamma \cdot \mathbf{p}^c$
Query-aware	*Combine-first*	$m[v,v'] = \pi \cdot m^{pr}_q[v,v'] + \alpha \cdot \mathcal{F}_s(v',v,q) + \beta \cdot \mathcal{F}_t(v',v,q) + \gamma \cdot \mathcal{F}_c(v',v,q)$	$\mathbf{p}_{i+1} \leftarrow d \cdot M\mathbf{p}_i + (1-d) \cdot \mathbf{t}$	$\mathbf{p}$
	Combine-last	$\forall k \in \{s,t,c\}$, $m^k[v,v'] = \mathcal{F}_k(v',v,q)$	$\mathbf{p}^{pr}_{i+1} \leftarrow d \cdot M^{pr}_q\mathbf{p}^{pr}_i + (1-d) \cdot \mathbf{t}$ $\forall k \in \{s,t,c\}$, $\mathbf{p}^k_{i+1} \leftarrow d \cdot M^k\mathbf{p}^k_i + (1-d) \cdot \mathbf{t}$	$\mathbf{p} = \pi \cdot \mathbf{p}^{pr} + \alpha \cdot \mathbf{p}^s + \beta \cdot \mathbf{p}^t + \gamma \cdot \mathbf{p}^c$

Tab. 4.1: Definitions of transition matrices, iterations, and STUNRank vectors, where (*i*) $\pi = 1-\alpha-\beta-\gamma$, and (*ii*) given an SR-query q, M^{pr}_q is the PageRank matrix computed by only considering edges $(v', l, v''; c) : [R,T]$ such that $l \in L_q$

$$\mathcal{F}_c(v,v') = \frac{\sum_{(v,l,v';c):[R,T] \in KB} [c \cdot score(l)]}{\sum_{(v,l,v'';c):[R,T] \in KB} [c \cdot score(l)]}.$$

so the STUNRank flowing from a vertex v to each of its out-neighbors v' is proportional to the total certainty annotating the edges from v to v'.

Fig. 4.10 shows a graphical representation of the query-unaware combine-first transition matrix for the example of Fig. 4.1 obtained with:

- label and vertex scores set to 1;
- $\alpha = \beta = \gamma = 0.25$;
- "Bethesda" and "Potomac" regions in "Maryland" and their areas set to 49, 100, and 10,000, respectively.

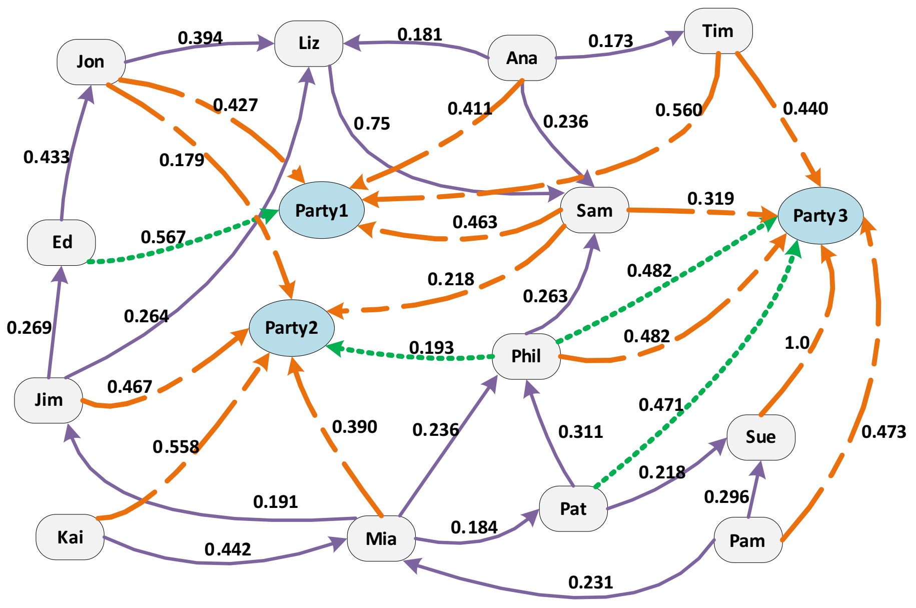


Fig. 4.10: Graphical representation of a query-unaware combine-first transition matrix for the example of Fig. 4.1

Given an SR-query $q = \langle L_q, R_q, T_q, \tau_q \rangle$, we define the query-aware STUNRank flow functions shown in Fig. 4.11. The difference with the query-unaware functions is the use of L_q for all of them, R_q for the space-based function, T_q for the time-based function, and τ_q for the certainty-based function.

$$\mathcal{F}_s(v, v', q) = \frac{\sum_{(v,l,v';c):[R,T]\in KB, l\in L_q} \left[|R \cap (\cup_{R'\in R_q} R')| \cdot score(l)\right]}{\sum_{(v,l,v'';c):[R,T]\in KB, l\in L_q} \left[|R \cap (\cup_{R'\in R_q} R')| \cdot score(l)\right]}$$

$$\mathcal{F}_t(v, v', q) = \frac{\sum_{(v,l,v';c):[R,T]\in KB, l\in L_q} [|\{t \in T \mid \exists T' \in T_q, t \in T'\}| \cdot score(l)]}{\sum_{(v,l,v'';c):[R,T]\in KB, l\in L_q} [|\{t \in T \mid \exists T' \in T_q, t \in T'\}| \cdot score(l)]}$$

$$\mathcal{F}_c(v, v', q) = \frac{\sum_{(v,l,v';c):[R,T]\in KB, l\in L_q, c\geq\tau_q} [c \cdot score(l)]}{\sum_{(v,l,v'';c):[R,T]\in KB, l\in L_q, c\geq\tau_q} [c \cdot score(l)]}$$

Fig. 4.11: Proposed query-aware similarity functions

Fig. 4.12 shows a graphical representation of the query-aware combine-first transition matrix for the example of Fig. 4.1 obtained with:

- label and vertex scores set to 1;
- $\alpha = \beta = \gamma = 0.25$;
- "Bethesda" and "Potomac" regions in "Maryland" and their areas set to 49, 100, and 10,000, respectively;
- $L_q = \{\textsf{friend,attended}\}$;
- $R_q = \{\text{Bethesda}\}$;
- $T_q = [3, 11]$;
- $\tau_q = 0.5$.

Figs. 4.13 and 4.14 list the vertices of the example of Fig. 4.1 ordered according to the

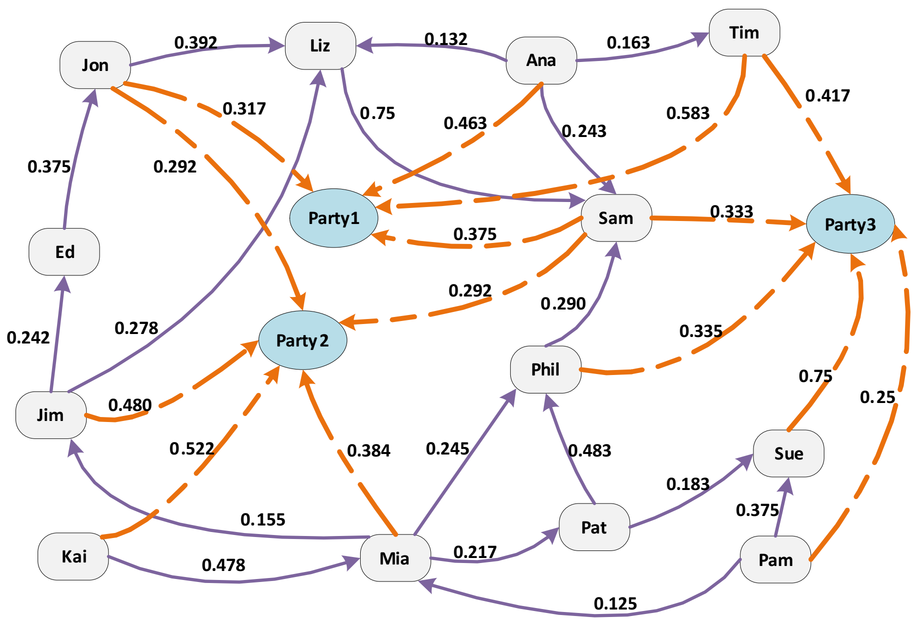


Fig. 4.12: Graphical representation of a query-aware combine-first transition matrix for the example of Fig. 4.1

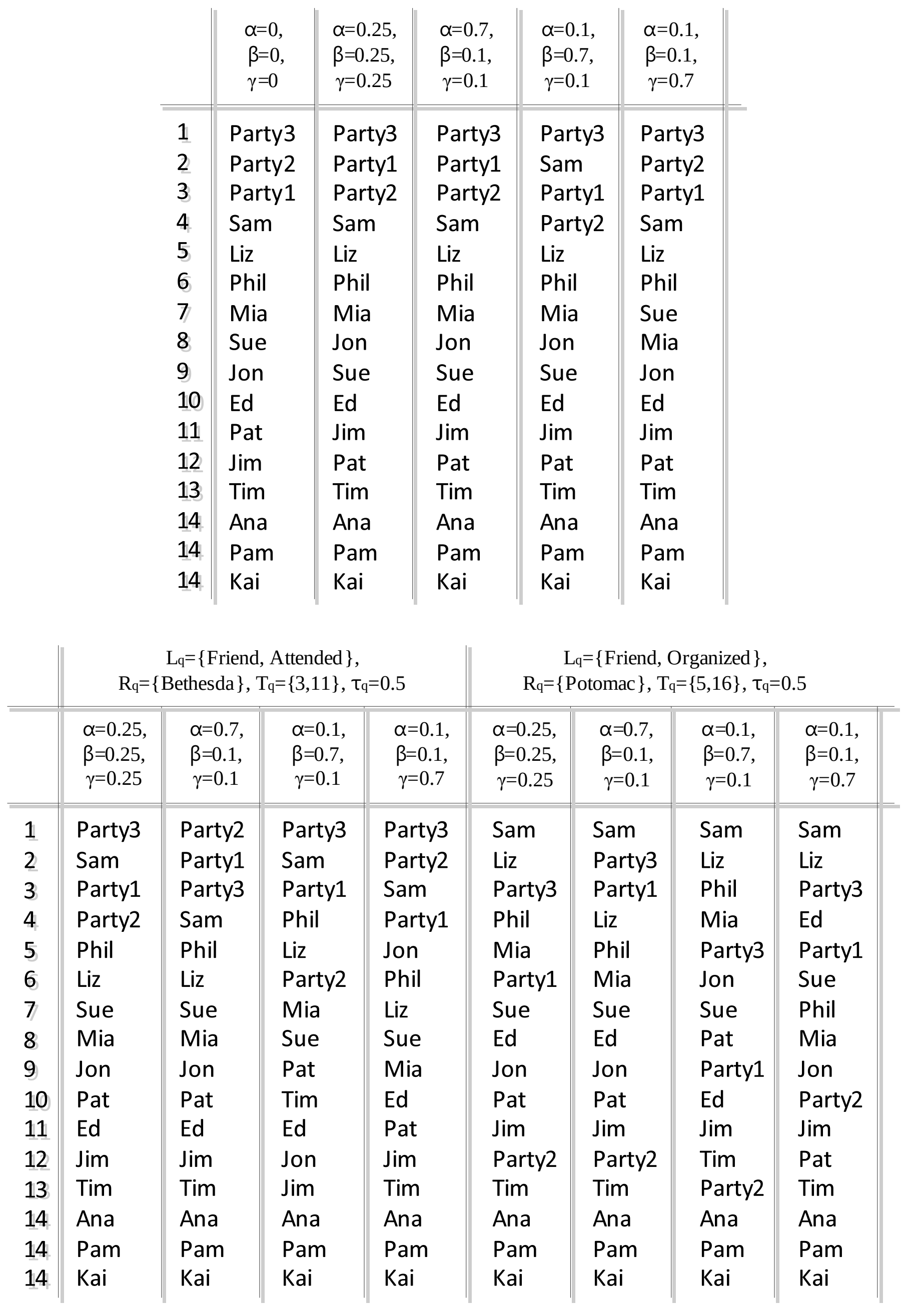
STUNRanks obtained by employing our proposed rank flow functions in the combine-first (Fig. 4.13) and combine-last (Fig. 4.14) cases. In each of the figures, we also compare the results obtained in the query-unaware case with those obtained with two different SR-queries.

It can easily be noticed that the "Party" vertices have the higher ranks in almost all of the query-unaware combinations, with the "Sam" vertex among the top-3 vertices in the cases where the temporal component has higher weight. Moreover, in the presence of a query, ranks are heavily influenced by the query components, as expected. For instance, with $L_q = \{\text{friend,organized}\}$, vertex "Sam" ranks top-1 in almost all cases, because its outgoing links do not match the query, thus the fraction of STUNRank it receives from its incoming links can only decrease due to the combination with the teleport vector.

	$\alpha=0$, $\beta=0$, $\gamma=0$	$\alpha=0.25$, $\beta=0.25$, $\gamma=0.25$	$\alpha=0.7$, $\beta=0.1$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.7$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.1$, $\gamma=0.7$
1	Party3	Party3	Party3	Party3	Party3
2	Party2	Party1	Party1	Sam	Party1
3	Party1	Party2	Party2	Party1	Party2
4	Sam	Sam	Sam	Party2	Sam
5	Liz	Liz	Liz	Liz	Liz
6	Phil	Phil	Mia	Phil	Phil
7	Mia	Mia	Phil	Mia	Sue
8	Sue	Sue	Sue	Jon	Mia
9	Jon	Jon	Jon	Sue	Jon
10	Ed	Ed	Ed	Ed	Ed
11	Pat	Jim	Jim	Jim	Jim
12	Jim	Pat	Pat	Pat	Pat
13	Tim	Tim	Tim	Tim	Tim
14	Ana	Ana	Ana	Ana	Ana
14	Pam	Pam	Pam	Pam	Pam
14	Kai	Kai	Kai	Kai	Kai

	L_q={Friend, Attended}, R_q={Bethesda}, T_q={3,11}, $\tau_q=0.5$				L_q={Friend, Organized}, R_q={Potomac}, T_q={5,16}, $\tau_q=0.5$			
	$\alpha=0.25$, $\beta=0.25$, $\gamma=0.25$	$\alpha=0.7$, $\beta=0.1$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.7$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.1$, $\gamma=0.7$	$\alpha=0.25$, $\beta=0.25$, $\gamma=0.25$	$\alpha=0.7$, $\beta=0.1$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.7$, $\gamma=0.1$	$\alpha=0.1$, $\beta=0.1$, $\gamma=0.7$
1	Party3	Party2	Party3	Party3	Sam	Party3	Sam	Sam
2	Party2	Party1	Sam	Party2	Party3	Party1	Liz	Liz
3	Party1	Party3	Party1	Party1	Liz	Sam	Phil	Party3
4	Sam	Sam	Phil	Sam	Party1	Liz	Party3	Party1
5	Liz	Liz	Liz	Phil	Phil	Mia	Mia	Ed
6	Phil	Phil	Party2	Liz	Mia	Phil	Jon	Sue
7	Sue	Mia	Mia	Jon	Sue	Sue	Sue	Phil
8	Mia	Sue	Sue	Sue	Jon	Ed	Party1	Mia
9	Jon	Jon	Pat	Mia	Ed	Jon	Pat	Jon
10	Pat	Pat	Tim	Ed	Pat	Pat	Ed	Party2
11	Ed	Ed	Ed	Pat	Jim	Jim	Jim	Jim
12	Jim	Jim	Jim	Jim	Party2	Party2	Tim	Pat
13	Tim	Tim	Jon	Tim	Tim	Tim	Party2	Tim
14	Ana	Ana	Ana	Ana	Ana	Ana	Ana	Ana
14	Pam	Pam	Pam	Pam	Pam	Pam	Pam	Pam
14	Kai	Kai	Kai	Kai	Kai	Kai	Kai	Kai

Fig. 4.13: STUNRanks obtained with combine-first in the query-unaware (top) and query-aware (bottom) cases. Label and vertex scores are set to 1. The damping factor is set to 0.85



	α=0, β=0, γ=0	α=0.25, β=0.25, γ=0.25	α=0.7, β=0.1, γ=0.1	α=0.1, β=0.7, γ=0.1	α=0.1, β=0.1, γ=0.7
1	Party3	Party3	Party3	Party3	Party3
2	Party2	Party1	Party1	Sam	Party2
3	Party1	Party2	Party2	Party1	Party1
4	Sam	Sam	Sam	Party2	Sam
5	Liz	Liz	Liz	Liz	Liz
6	Phil	Phil	Phil	Phil	Phil
7	Mia	Mia	Mia	Mia	Sue
8	Sue	Jon	Jon	Jon	Mia
9	Jon	Sue	Sue	Sue	Jon
10	Ed	Ed	Ed	Ed	Ed
11	Pat	Jim	Jim	Jim	Jim
12	Jim	Pat	Pat	Pat	Pat
13	Tim	Tim	Tim	Tim	Tim
14	Ana	Ana	Ana	Ana	Ana
14	Pam	Pam	Pam	Pam	Pam
14	Kai	Kai	Kai	Kai	Kai

	L_q={Friend, Attended}, R_q={Bethesda}, T_q={3,11}, τ_q=0.5				L_q={Friend, Organized}, R_q={Potomac}, T_q={5,16}, τ_q=0.5			
	α=0.25, β=0.25, γ=0.25	α=0.7, β=0.1, γ=0.1	α=0.1, β=0.7, γ=0.1	α=0.1, β=0.1, γ=0.7	α=0.25, β=0.25, γ=0.25	α=0.7, β=0.1, γ=0.1	α=0.1, β=0.7, γ=0.1	α=0.1, β=0.1, γ=0.7
1	Party3	Party2	Party3	Party3	Sam	Sam	Sam	Sam
2	Sam	Party1	Sam	Party2	Liz	Party3	Liz	Liz
3	Party1	Party3	Party1	Sam	Party3	Party1	Phil	Party3
4	Party2	Sam	Phil	Party1	Phil	Liz	Mia	Ed
5	Phil	Phil	Liz	Jon	Mia	Phil	Party3	Party1
6	Liz	Liz	Party2	Phil	Party1	Mia	Jon	Sue
7	Sue	Sue	Mia	Liz	Sue	Sue	Sue	Phil
8	Mia	Mia	Sue	Sue	Ed	Ed	Pat	Mia
9	Jon	Jon	Pat	Mia	Jon	Jon	Party1	Jon
10	Pat	Pat	Tim	Ed	Pat	Pat	Ed	Party2
11	Ed	Ed	Ed	Pat	Jim	Jim	Jim	Jim
12	Jim	Jim	Jon	Jim	Party2	Party2	Tim	Pat
13	Tim	Tim	Jim	Tim	Tim	Tim	Party2	Tim
14	Ana	Ana	Ana	Ana	Ana	Ana	Ana	Ana
14	Pam	Pam	Pam	Pam	Pam	Pam	Pam	Pam
14	Kai	Kai	Kai	Kai	Kai	Kai	Kai	Kai

Fig. 4.14: STUNRanks obtained with combine-last in the query-unaware (top) and query-aware (bottom) cases. Label and vertex scores are set to 1. The damping factor is set to 0.85

4.7 Implementation and Experiments

We developed a prototype implementation of the STUN system in about 11,600 lines of Java code. We ran three sets of experiments to assess the performance of the STUN system when dealing with both subgraph matching and ranking queries.

4.7.1 Results with SM-Queries

In the first set of experiments, we used data from YouTube [158] where the vertices included individuals and groups, and the edges included follows, group membership, and co-located edges. We randomly assigned time intervals to the first two kinds of edges. To assign regions, we first selected 20% of the groups and randomly associated regions to them. Then, we added a co-located edge between two members of such groups if there was a nonempty temporal overlap between their memberships – this temporal overlap and the region associated with the group were assigned the edge. All the experiments were run on a dual-core 2.8GHz CPU with 8G of RAM running Windows 7.

The objective of our experiments was to determine the scalability of the STUN index by varying three quantites: the size of the graph (number of edges), the complexity (number of edges and number of variable vertices) of queries, and the number of constraints in the query. Graphs of a specific size were selected from the YouTube graph by randomly choosing a connected set of edges till we reached the desired number of edges. Query graphs were also randomly generated.

Fig. 4.15 shows three graphs in which we varied the number of edges of the STUN graph (100K, 500K, 1M), the number of constraints in the query (0 through 5) and the number of edges and variable vertices in the query. Though we ran far more experiments than shown in the figure, due to space limitations, we show three classes of queries – ones with 3 edges and only 1 variable vertex, ones with 5 edges and 2 variable vertices, and ones with 12 edges and 5 variable vertices. Every single data point shown in Fig. 4.15 was obtained by running 200 queries.

The results shows that (*i*) as the number of constraints in a query increases, STUN's query processing time decreases – this is because additional constraints enable us to prune more of

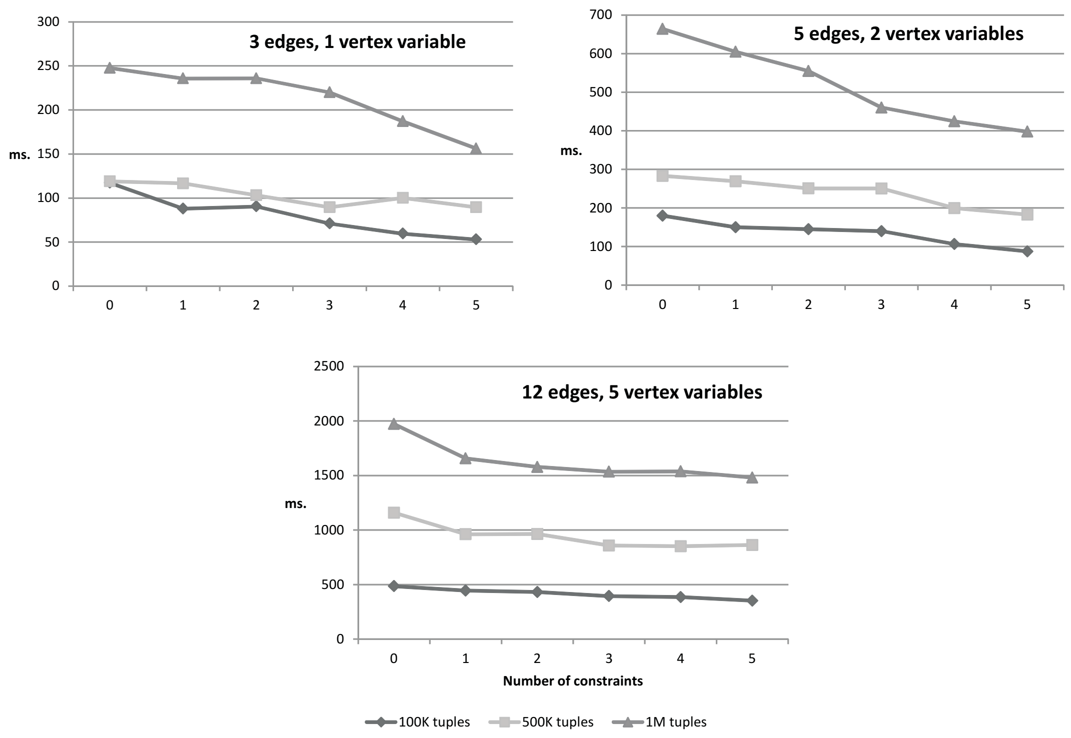


Fig. 4.15: Query execution time versus number of spatial and temporal constraints

the search space; (*ii*) as expected, the time taken to process queries increases with the query complexity and the amount of data. However, even with 5 variables in the query (corresponding to a 5-way join in a relational database), we are able to process queries in a 1M edge graph in 1.5-2 seconds – and for simpler queries, the query processing time can often be well under 0.5 seconds.

Fig. 4.16 shows the same data in the previous figure – only this graph shows how the three types of queries mentioned above perform as we vary the size of the STUN knowledge base. We see that the query processing time increases slightly super-linearly with the size of the database, though the slope of the graph increases with the complexity of the query.

4.7.2 Results with STUNRank and SR-Queries

The objective of our second set of experiments was to determine the compute time performance of STUN ranking in the query-unaware case when using the rank flow functions defined in Sec-

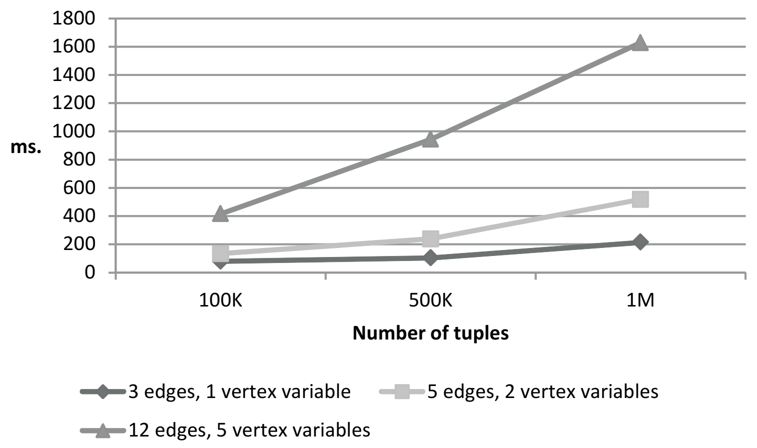


Fig. 4.16: Query execution time versus size of the STUN knowledge base

tion 4.6. For these experiments, we varied the size of the graphs and their average outdegree. We used datasets from YouTube. Graphs of specific sizes (50K, 100K, 150K, and 200K vertices) were selected from the YouTube graph by randomly choosing a connected set of edges till we reached the desired number of vertices. We generated 30 datasets for each size. In these data sets, we assigned time intervals and regions in the same way as the datasets used in the first set of experiments, and in addition we randomly assigned a spatial region to 50% of follow edges. We fixed the parameters $\alpha = 0.25$, $\beta = 0.25$, and $\gamma = 0.25$, and set all label scores and vertex scores to 1. Numbers were averaged over 5 independent runs with the maximum number of iterations set to 100 and a tolerance (difference between two consecutive rank vectors) set to 10^{-5}. All the experiments were run on an Intel Xeon@2.40 GHz, 24 GB RAM.

Fig. 4.17 reports the overall computation times needed to compute STUNRanks versus dataset size. The results show that, in general, STUN scales gracefully with respect to dataset size, and that most of the computation is devoted to the construction of the transition matrices. The most important difference between the combine-first and the combine-last scenarios is the time needed to perform the iterations – as expected, iterations take around 4 times more for the combine-last scenario.

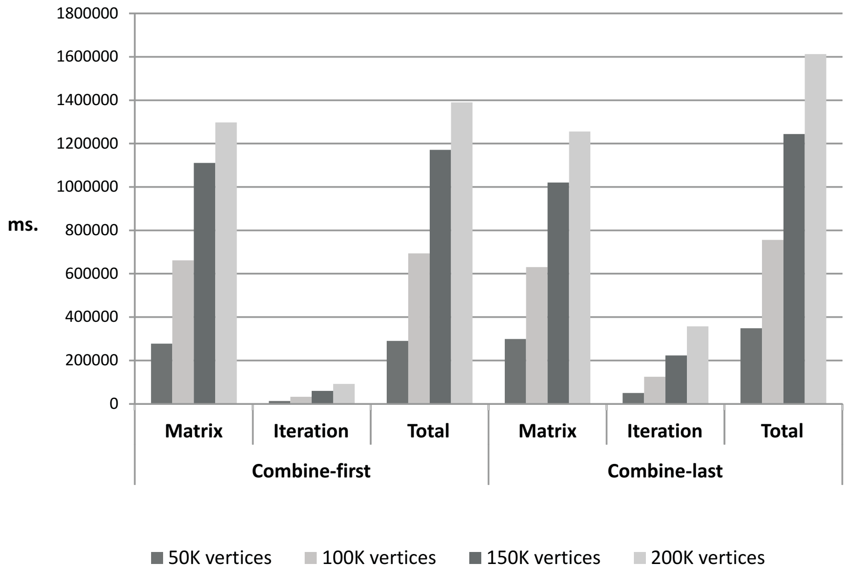


Fig. 4.17: Overall STUNRank computation times in the query-unaware case

Figs. 4.18 and 4.19 report the computation times measured when using each of the datasets, versus the average outdegree of the datasets. The results clearly show that, while iteration times naturally depend on the size and average outdegree of the graph, the construction of the transition matrices is much more influenced by the specific characteristics of each different dataset.

In the third set of experiments, we assessed the performance in the query-aware case, by varying the "coverage" of the R_q and T_q components of an SR-query ($area(R_q)/area(\mathbf{S})$ and $length(T_q)/length(\mathbf{T})$) in the $[0.01, 0.6]$ range. We randomly generated 10 datasets of 100K vertices each and 100 queries for each value of the coverage and each dataset. We assigned time intervals and regions in the same way as for the datasets used in the first set of experiments. We fixed the parameters $\alpha = 0.25$, $\beta = 0.25$, $\gamma = 0.25$, $L_q = \{\mathsf{follows}\}$ and $\tau_q = 0.5$, and set all labels and vertex scores to 1. Numbers were averaged over 100 independent runs per each value of the coverage and each dataset with a maximum number of iterations of 100 and a tolerance equal to 10^{-5}. These experiments were also run on an Intel Xeon@2.40 GHz, 24 GB RAM.

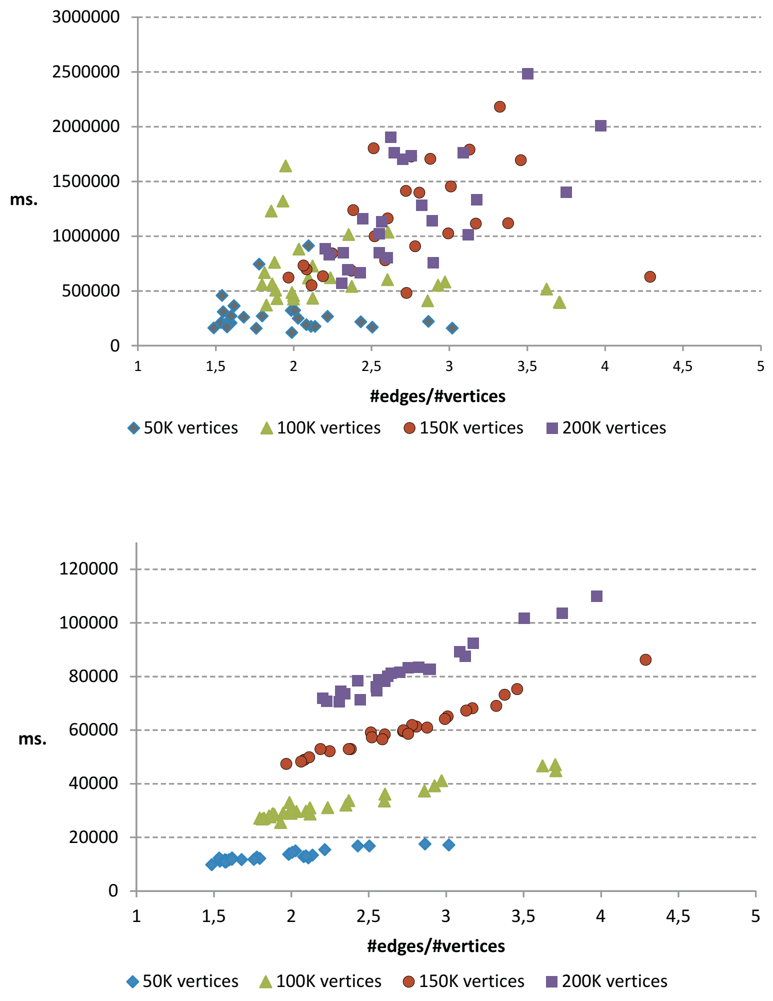
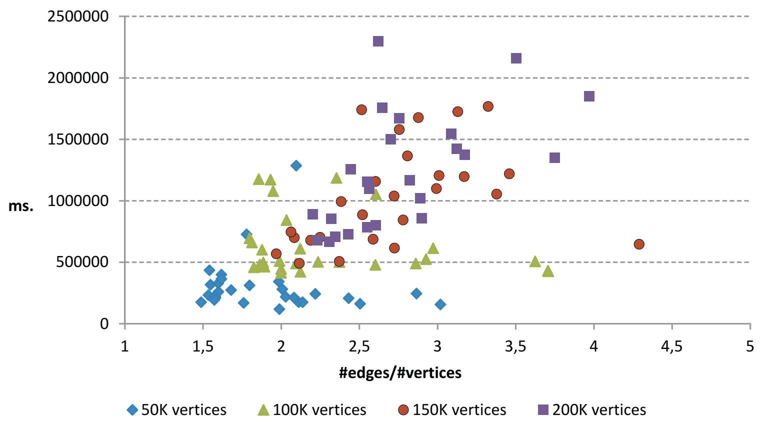


Fig. 4.18: Matrix (top) and iteration (bottom) computation time for various network densities in the combine-first query-unaware case

Fig. 4.20 reports the matrix building time in the combine-last case when varying query coverage, averaged over the whole set of average outdegree values employed. The results show that the performances are rather stable, with an increase of just 19.2% when moving from the minimum to the maximum coverage value.

Finally, Fig. 4.21, which reports iteration time in both the combine-first and combine-last scenarios versus coverage for two specific values of average outdegree (1.8 and 3.6), shows that the performances are basically constant in the combine-first case, whereas in the combine-last



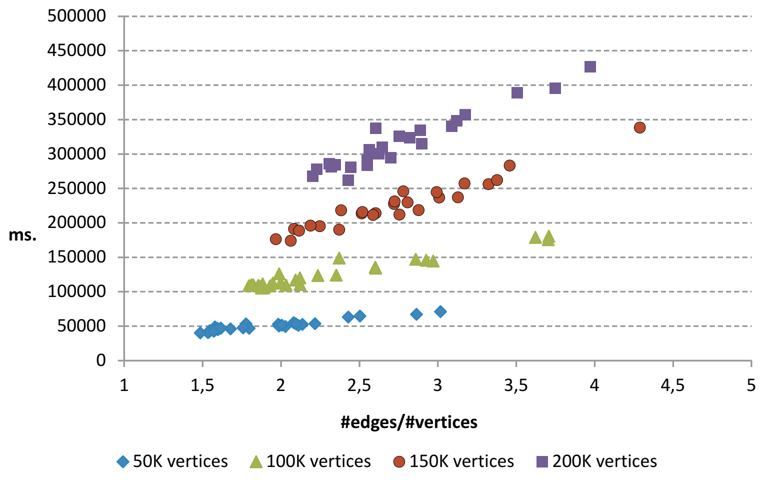


Fig. 4.19: Matrix (top) and iteration (bottom) computation time for various network densities in the combine-last query-unaware case

case, computation time increases at most 35% when moving from the minimum to the maximum coverage value.

4.8 Related Work

To date, the only system capable of performing subgraph matching in the presence of space, time, and uncertainty when they are all present at the same time is the Annotated RDF system [197] in which edges can be annotated with any complete lattice of truth values, including, in princi-

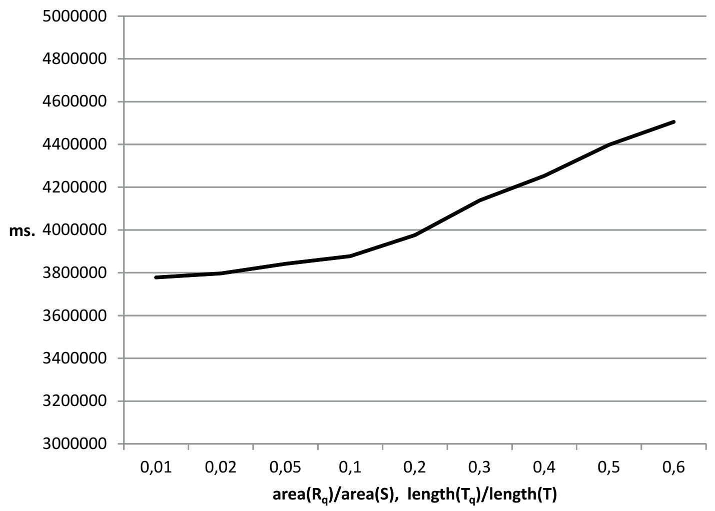


Fig. 4.20: Average matrix computation time in the combine-last query-aware case

ple, triples of spatial regions, time intervals, and uncertainty. Annotated RDF query processing algorithms are all *in memory* algorithms. However, as shown in [197][Table II], even for a graph database with just 549K edges from the ChefMoz data set used in those experiments, it takes 1.243 seconds *in memory*, while with just over 1M edges (1.234M edges) on the GovTrack data set, it takes almost 18 seconds to process simple queries. In contrast to their *in memory* times which one would expect to be fast, our on-disk times for comparable queries on 500K edge data is under 200ms for the first two types of queries shown in Fig. 4.15. Our work here is also closely related to the DOGMA work [40] in which we previously developed an index structure to store RDF graphs on disk to support subgraph matching. In this paper, we show how such a framework can be augmented with four quantities – constraints, space, time, and uncertainty – and show that we can still answer complex queries on social network data efficiently.

Many approaches to graph storage and querying in the Semantic Web community use relational databases as their back-end [194], for instance by inferring the relational schema of the given RDF data [185], or using a triple based denormalized relational schema [203]. Other RDF databases rely on classical index structures, such as the B-tree and its generalizations [106].

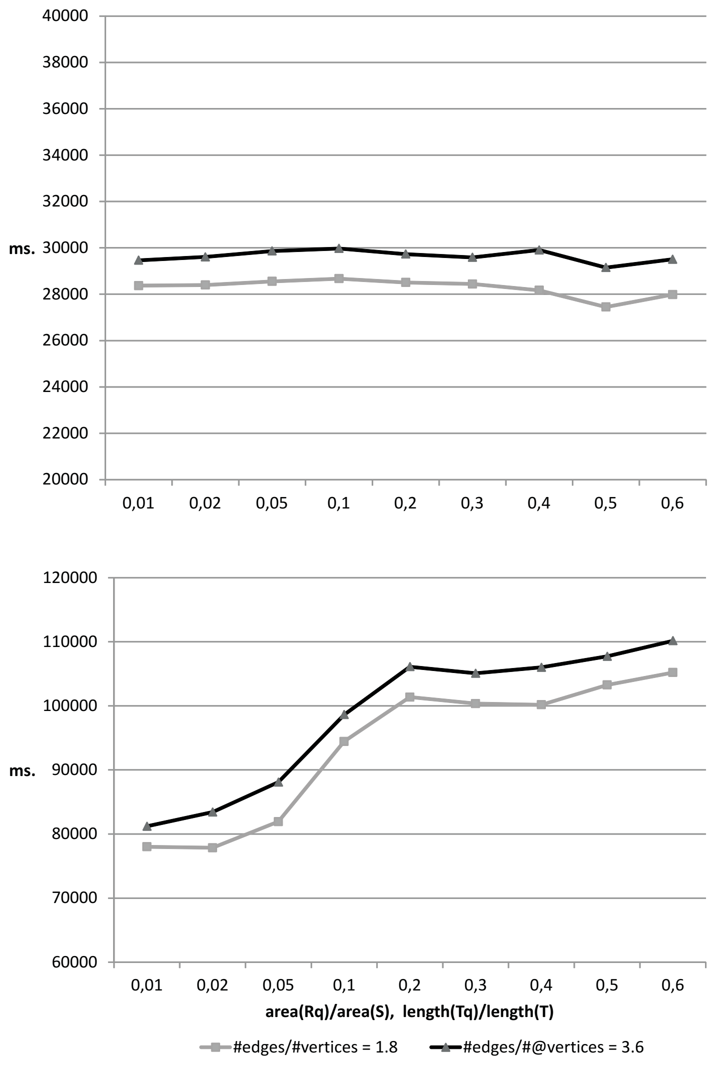


Fig. 4.21: Iteration time in the combine-first (top) and combine-last (bottom) query-aware case

Existing work on general subgraph matching (*e.g.*, [57, 70, 205, 206, 207, 208]) typically employ heuristics to predict the cost of answering strategies based on statistics about the dataset and the current state of query processing. For instance, [208] transforms vertices into points in a vector space, thus converting queries into distance-based multi-way joins over the vector space. In particular, [57] proposes a two-step join optimization algorithm based on a cluster based join index. *GADDI*, proposed in [205], employs a structural distance based approach and a dynamic matching scheme to minimize redundant calculations. In [206] the authors propose *SUMMA*, which improves over GADDI and employs more advanced indices. The algorithm in [207] employs an aggressive pruning strategy based on an index storing label distributions. In [70], the authors propose an index over *sets* of data graphs with the objective of enabling efficient pruning over graphs with tens of thousands of vertices. In contrast, our work focuses on query answering over *single*, very large graphs that contain edges that may be labeled with a combination of space, time, and uncertainty – something none of these papers address.

The PageRank algorithm and its many variants have the general objective of appropriately ranking linked objects in various scenarios. They have received a tremendous deal of attention in the last decade [15, 26, 35, 51, 60, 108, 141, 162, 39]. The initial problem formulation [39] has been extensively studied in order to improve its feasibility and performances in fastly growing networks [16, 26, 35, 81, 141, 39]. Many works have also been targeted at exploiting the flexibility of the personalized variant of PageRank [16, 26, 51, 81, 108, 141]. For instance, [51] presented an approach targeted at proximity search in entity-relationship graphs. The main intuition was to compute and index "sketchy" random walk fingerprints for a subset of vertices, which were then adaptively loaded to various resolutions to form approximate personalized PageRank vectors. Finally, some work has been devoted to the idea of exploiting "context" information to make the PageRank of a vertex flow non-uniformly to its out-neighborhood [15, 51, 60, 162]. For instance, [15] presented a variant of PageRank where link *weights* were computed based on their relative position in the Web page, associated tags, and length of the anchor text. In [60], a dynamic query-dependent ranking technique was proposed that, given a link- and vertex-labeled directed graph, applied a flow-based model to dynamically rank vertices with respect to any of the original labels. [162] employed a given clustering of Web pages and gave different weights to each link by looking at the positions of the connected vertices with respect to the clusters. To the best of our knowledge, none of the approaches proposed in the past takes into account spatio-temporal, certainty, and vertex/label importance information to derive suitable PageRank-based vertex rankings in complex graphs.

4.9 Conclusions

With the growing importance of social networks, there are now many efforts to learn properties of such networks. For instance, learning relationships between people is a complex problem, and there is much work in identifying links between individuals [132]. However, such links are almost always uncertain. Likewise, when we look at real-world social networks like Facebook, there is spatial and temporal information often present. For instance, a page associated with an event being organized on FaceBook has an associated location and an associated time.

In this chapter, we extend classical social network data management with spatial, temporal, and uncertain attributes, leading to the STUN framework. We propose a theoretical model for STUN as well as a STUN subgraph matching query language that extends classical graph matching queries with space, time, and uncertainty. Furthermore, we develop an index structure to store STUN databases. The graph structure, time, and space are hierarchically decomposed into a single index. We also develop a STUN subgraph matching algorithm that effectively utilizes this index, pruning away large parts of the search space. Finally, we introduce STUNRank queries where users can ask queries about important nodes in a STUN database. STUNRank queries come in two flavors – query-aware and query-unaware. Moreover, we provide two classes of STUNRank definitions – *combine-first* and *combine-last*. Additionally, our framework allows a user to specify the relative importance of space, time, uncertainty, and traditional network flows in his query, allowing the user greater expressive power. Finally, we report on experiments using real-world YouTube data showing that our algorithms work well in practice.

Chapter 5: Malware Infection Prediction

5.1 Introduction

The goal of this research is to develop methods to predict the percentage of hosts in a given population (we use country in our experiments) that will be infected by a particular piece of malware, given some historical data about the malware and the hosts, but with no information whatsoever on how the hosts are connected together. This is made even more challenging by the fact that not all truly infected machines are actually detected to be infected (by say using some anti-virus software). This problem is an important one with immediate applications in web and cyber security. For example, a better prediction of the number of infections in a country will enable anti-virus companies and security firms to better deploy patches and safety measures to counter threats.

In order to achieve this, we make several contributions. First, we define a very novel set of features that are related to the ability of hosts to detect malware and patch vulnerabilities. In order to achieve this, we present a novel host-malware bipartite graph and a bi-fix-point algorithm to compute these features. These lead to a feature-based prediction model (FBP). Then, building upon the well-known SIR-model of disease spread, we develop an epidemiologically-inspired predictive model called DIPS in which each host is either in a detected, infected, patched, or susceptible state. We define the model and show how to learn the parameters of the model in a data-driven way. We also define a variant of DIPS called DIPS-EXP. The outputs of these models, as well as outputs of past work on predicting epidemic spreads, are then fed into three different ensemble models: ESM_0, $\text{ESM}_1$and FBP^{+}_{Funnel}. We study the relative predictive accuracies of all of these models. On split-sample 10-fold cross validation tests, ESM_0 provides the best performance, significantly outperforming past work by large margins, irrespective of whether we use root mean squared error (RMSE), normalized RMSE (NRMSE), or Pearson Correlation Coefficient (PCC) as our measure. All the experiments use large-scale extensive real-data from Symantec's Worldwide Intelligence Network Environment (WINE) data set.

The paper is organized as follows. We describe our dataset and set-up in the next section. Section 5.3 describes statistical features we develop for our task and the feature based prediction model (FBP). In Section 5.4, we propose a novel non-linear time-series model (DIPS). The ensemble models (ESM) to combine two suggested models (FBP and DIPS) is described in Section 5.5. In Section 5.6 we present our empirical data analysis including the prediction task. We review related work in Section 5.7, and we discuss the implications of our results in Section 5.8.

5.2 Dataset and Set-up

Symantec's WINE data is collected from real-world hosts running Symantec anti-virus software. Users of Symantec's product line have a choice of opting-in to report telemetry about the security events[1] (e.g. executable file downloads, virus detection) that occur on their hosts. The WINE dataset contains information about the hosts, files of the hosts and malware recognized on a file of a host.

We selected the 100 most frequently detected malware from the WINE data and extracted infection information about hosts infected by these malware using the following records.

Anti-virus telemetry. Anti-virus (AV) telemetry records detection history of known malware over hosts. From each record, we extracted the detection time and the hash (MD5 and SHA2) of the

[1] The events included in WINE are representative of events that Symantec observes around the world [167].

malicious file detected. Each record indicates that the anti-virus has blocked an attack that may have resulted in an infection.

Binary reputation. The binary reputation data records all binary executable files – benign or malicious – that were downloaded/copied on end-hosts worldwide. From each record, we extracted the creation time of the malicious file in the host (infection time) and the country where the host was when the file was infected.

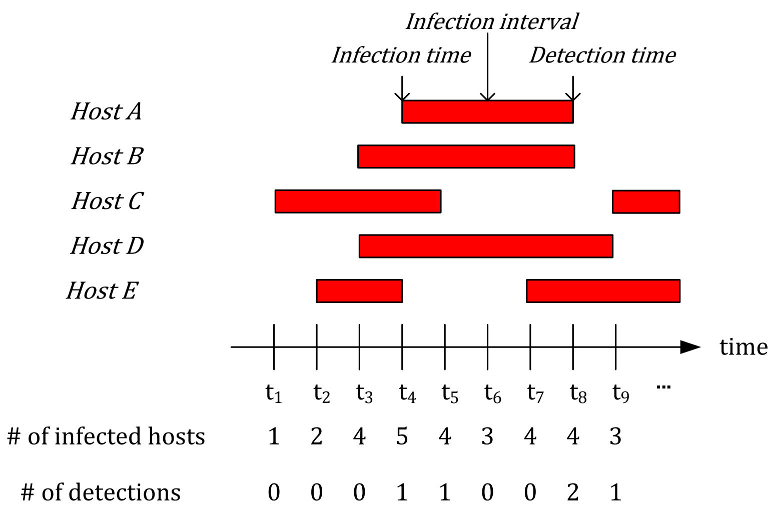
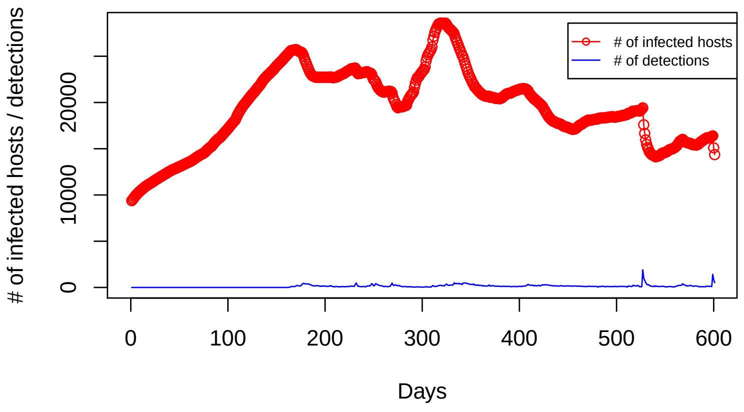

Infection interval
Infection time
Detection time
Host A
Host B
Host C
Host D
Host E
time
t1 t2 t3 t4 t5 t6 t7 t8 t9 ...
# of infected hosts 1 2 4 5 4 3 4 4 3
# of detections 0 0 0 1 1 0 0 2 1

Fig. 5.1: Examples of infections/infection intervals

We note that an infection interval is the period from the infection time to the detection time of a malware in a host (See Figure 5.1). When the length of the infection interval is larger than 0, we call it an infection.[2] The number of infections over the 100 most frequently detected malware during 640 days was 3,170,669 (for the US alone, it was 989,263).[3] The infection patterns are very different (See Figure 5.2). We adopt the following approach to address this problem: (a) we first carefully define and extract novel domain-based features from the WINE datasets including

[2] If the anti-virus software blocked a file containing a malware immediately when the files appeared, then the detection record is not an infection in our framework.

[3] Note that we do not know the *number* of infected hosts that have not been detected.



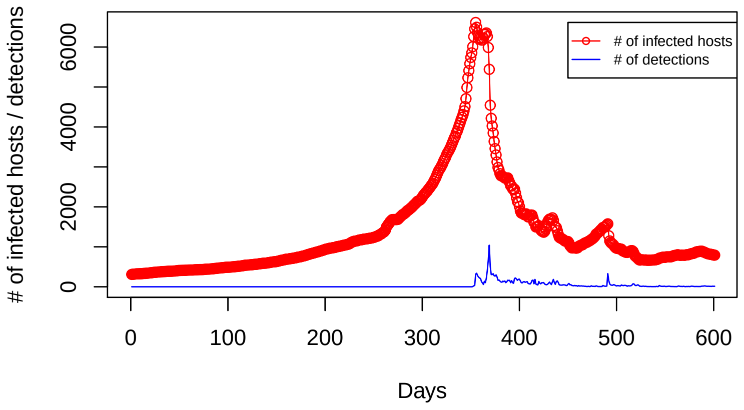


Fig. 5.2: Examples for infection patterns of 2 malware in the US in our dataset. Number of infected hosts (bold red) and number of detection (blue) vs time (in days). Note that the number of detection are a fraction of the number of infected hosts and that both time-series are highly non-linear.

measures of hosts' detection ability and incompetence — we call this feature-based prediction (FBP) ; (b) we then construct a novel non-linear temporal model called DIPS (Detected, Infected, Patched, and Susceptible) based on epidemiological principles, and then (c) utilize sound regression methods to link both the features as well as the epidemiology-based models to the *dependent* variable (the number of malware infections). Because the inputs of FBP as well as multiple DIPS variants and FUNNEL[153] are themselves predictors that feed into yet another predictor, our overall framework is an ensemble prediction method.

*5.3 Feature-based prediction (*FBP*)*

In a very dynamic environment like malware infections and patching, it is not possible to predict infection patterns solely based on infection trends. Some amount of domain knowledge has to be incorporated (for example, when the number of detected infections are small). Hence as a first step, we build a feature-based prediction model (FBP), where given past history we extract meaningful features of hosts and malware. Essentially these features encode the defending abilities of hosts and infection properties of the malware as relative scores.

Next, we describe the domain-based features we constructed for our analysis in more detail. We define $\mathcal{H}$ and $\mathcal{M}$ as the set of host and the set of malware, respectively. For each host $h \in \mathcal{H}$, $\mathcal{F}(h)$ denotes the set of files of h. For $f \in \mathcal{F}(h)$, t_f^h is the time stamp representing the time when f appears on h. For each host $h \in \mathcal{H}$, the malware detection set $dH(h)$ of h is a set of tuples of the form (f, m, t) where $f \in \mathcal{F}(h)$, $m \in \mathcal{M}$ and t is the time stamp when the file f infected by the malware m is discovered on the host h. By using the the malware detection sets for each host, we define the set of host detection $dM(m)$ of the malware m as $dM(m) = \{(f, h, t) | h \in \mathcal{H}, f \in \mathcal{F}(m), (f, m, t) \in dH(h)\}$.

Vulnerabilities: Let V be the set of all vulnerabilities. Given a malware $m \in \mathcal{M}$ the set $Vu(m)$ is the set of all vulnerabilities used by this malware. For each vulnerability $v \in V$, two time stamps are important [28]:

- **Anti-virus signatures released.** Once the vulnerability is disclosed, anti-virus vendors release new signatures for ongoing attacks and created heuristic detection for the exploit. After this point, the attacks can be detected on end-hosts with updated A/V signatures. We denote this as t_v^s. We can retrieve this information from Symantec website or the National

Vulnerabilities Database (NVD)[4] maintained by NIST (National Institute of Standards and Technology of the US Government).

- **Patch released.** On the disclosure date, or shortly afterward, the software vendor releases a patch for the vulnerability. After this point, the hosts that have applied the patch are no longer susceptible to the exploit. We denote this as t_v^p. This information is readily available from NVD.

5.3.1 Incompetence

Given a host $h \in \mathcal{H}$ and a tuple $(f, m, t) \in dH(h)$, we define two classes of features: hosts' incompetence in detecting malware and incompetence in patching (DI and PI, respectively). For each, we consider an absolute case and a relative case (i.e ADI, API, RDI and RPI). These four measures measure the incompetence of a host in detecting and patching the malware m associated with file f. The only difference between the relative case and absolute case is that the absolute case does not consider the time t_v^s for each vulnerability v contained in a malware.

$$\begin{aligned}
ADI(h,f,m,t) = &\quad t - t_f^h \\
RDI(h,f,m,t) = &\quad \max_{v \in Vu(m)} \max(0, t - \max(t_v^s, t_f^h)) \\
API(h,m) = &\quad (\max_{(f,m,t) \in dH(h)} t) - (\min_{(f,m,t) \in dH(h)} t_f^h) \\
RPI(h,m) = &\quad \max_{v \in Vu(m)} \max(0, (\max_{(f,m,t) \in dH(h)} t \\
&\quad - \max(t_v^p, (\min_{(f,m,t) \in dH(h)} t_f^h))))
\end{aligned}$$

For instance, absolute detection incompetence, $ADI(h, f, m, t)$ of host h w.r.t. a file f of malware m and the time t when f was detected by host h is simply $(t - t_f^h)$, i.e. the time that passed from the detection time to the time of original infection. The relative detection incompetence $RDI(h, f, m, t)$ looks at all possible vulnerabilities v that malware m exploits. For each such

[4] https://nvd.nist.gov/

vulnerability, it looks at the time elapsed between when the file f was detected by host h and when either the file infected the host or when an updated anti-virus signature could have detected it. The longer this time gap, the more incompetent the host. The absolute and relative (API, RPI) patching incompetence are similarly defined.

Estimation by using only the WINE *datasets.* The relative detection and patch incompetence formulas can be rewritten as:

$$
\begin{aligned}
RDI(h,f,m,t,) &= \max(0, t - \max((\min_{v \in Vu(m)} t_v^s), t_f^h))) \\
RPI(h,m) \quad &= \max(0, (\max_{(f,m,t) \in dH(h)} t - \\
&\quad \max((\min_{v \in Vu(m)} t_v^p), (\min_{(f,m,t) \in dH(h)} t_f^h))))
\end{aligned}
$$

For a malware m, $t^s(m) = \min_{v \in Vu(m)} t_v^s$ denotes the earliest time at which a host downloaded an anti-virus signature for the vulnerability (we call this "time signature"), showing that he might have become aware of the malware, and $t^p(m) = \min_{v \in Vu(m)} t_v^p$ denotes the earliest time the host applied a patch for a vulnerability for malware m (we call this "patch signature"). Both these quantities are recorded by WINE. Then, the two measures becomes the following:

$$
\begin{aligned}
RDI(h,f,m,t,) = &\quad \max(0, t - \max(t^s(m), t_f^h)) \\
RPI(h,m) = &\quad \max(0, (\max_{(f,m,t) \in dH(h)} t) - \\
&\quad \max(t^p(m), (\min_{(f,m,t) \in dH(h)} t_f^h))))
\end{aligned}
$$

A possible way to estimate time signature and patch signature of a malware in WINE is:

$$
\begin{aligned}
t^s(m) = &\quad \min_{v \in Vu(m)} t_f^m \\
\approx &\quad \min_{h \in \mathcal{H}, (f,m,t) \in dH(h)} t \\
t^p(m) = &\quad \min_{v \in Vu(m)} t_v^p \\
\approx &\quad \min_{h \in \mathcal{H}, (,m',t') \in dH(h)} \max_{(f,m,t) \in dH(h)} t
\end{aligned}
$$

Average Incompetence. Average/relative detection/patching incompetence of a host h can be defined by aggregation.

$$\begin{aligned} AADI(h) = & \quad \frac{\sum_{(f,m,t)\in dH(h)} ADI(h,f,m,t)}{|dH(h)|} \\ ARDI(h) = & \quad \frac{\sum_{(f,m,t)\in dH(h)} RDI(h,f,m,t)}{|dH(h)|} \\ AAPI(h) = & \quad \frac{\sum_{m\in\{m'|(,m',)\in dH(h)\}} API(h,m)}{|\{m'|(,m',)\in dH(h)\}|} \\ ARPI(h) = & \quad \frac{\sum_{m\in\{m'|(,m',)\in dH(h)\}} RPI(h,m)}{|\{m'|(,m',)\in dH(h)\}|} \end{aligned}$$

For instance, average patch incompetence (AAPI) is obtained by adding up the average patch incompetence of the host w.r.t. all malware files that it patched, and dividing this by the number of such malware files associated with that host.

5.3.2 Features on Bipartite Host-Malware (HM) Graphs

Given a training set of historical WINE data, we can derive a bipartite graph whose vertices are hosts and malware, respectively. An edge links a host and malware if the training data shows that the malware has infected the host. Edges can be weighted with quantities like $AD(h,m)$ or $RDI(h,m)$ as defined earlier in this section.

We now define some novel features using HM-graphs. The absolute detection ability (ADA) of a host h captures the host's ability to detect malware, while the absolute detection hardness (ADH) of a malware m captures the difficulty in detecting that malware. Clearly, these two quantities are closely intertwined. We define them below in a mutually recursive manner.

$$\begin{aligned} ADA(h) & \quad = \textstyle\sum_{(f,m,t)\in dH(h)} w_{12}(h,f,m,t)\cdot ADH(m) \\ ADH(m) & \quad = \textstyle\sum_{(f,h,t)\in dM(m)} w_{21}(m,f,h,t)\cdot ADA(h) \end{aligned}$$

where $h \in \mathcal{H}, m \in \mathcal{M}$ and

$$\begin{aligned} w_{12}(h,f,m,t) = & \quad \frac{\frac{1}{ADI(h,f,m,t)}}{\sum_{(f',h',t')\in dM(m)} \frac{1}{ADI(h',f',m,t')}} \\ w_{21}(m,f,h,t) = & \quad \frac{ADI(h,f,m,t)}{\sum_{(f',m',t')\in dH(h)} ADI(h,f',m',t')} \end{aligned}$$

By way of explanation, consider the definition of $ADA(h)$. To do this, we look at every (f,m,t) triple in the training set. If h can detect m, then the harder m is to detect (captured by $ADH(m)$ in the formula), the better h's detection abilities are. Thus, when considering a specific (f, m, t) triple, the definition of $ADA(h)$ multiplies $ADH(m)$ by a quantity w_{12} which is the relative competence (inverse of absolute detection incompetence $ADI(h, f, m, t)$). A similar rationale applies to the definition of ADH. As in the previous section, we can define relative variants, RDA, RDH as well.

$$\begin{aligned} RDA(h) &= \sum_{(f,m,t)\in dH(h)} w_{12}(h, f, m, t) * RDH(m) \\ RDH(m) &= \sum_{(f,h,t)\in dM(m)} w_{21}(m, f, h, t) * RDA(h) \end{aligned}$$

where $h \in \mathcal{H}, m \in \mathcal{M}$ and

$$\begin{aligned} w_{12}(h, f, m, t) = & \frac{\frac{1}{RDI(h,f,m,t)}}{\sum_{(f',h',t')\in dM(m)} \frac{1}{RDI(h',f',m,t')}} \\ w_{21}(m, f, h, t) = & \frac{RDI(h,f,m,t)}{\sum_{(f',m',t')\in dH(h)} RDI(h,f',m',t')} \end{aligned}$$

5.3.2.1 *Fix-point Computation and Convergence Result*

As ADA, ADH are mutually recursively defined, one may wonder if they are well-defined. We show below, an iterative fix-point computation procedures that describes how this may be achieved. We now define the iterations ADA^i, ADH^i as follows. Let $ADA^0(h) = \frac{1}{|\mathcal{H}|}$, $ADH^0(m) = \frac{1}{|\mathcal{M}|}$ for $h \in \mathcal{H}, m \in \mathcal{M}$,then for iteration $i > 0$,

$$\begin{aligned} ADA^i(h) &= \sum_{(f,m,t)\in dH(h)} w_{12}(h, f, m, t) * ADH^{i-1}(m) \\ ADH^i(m) &= \sum_{(f,h,t)\in dM(m)} w_{21}(m, f, h, t) * ADA^{i-1}(h) \end{aligned}$$

We can capture the above computations as a bi-fix-point algorithm, Algorithm 9 that maintains two lists ADA, ADH. The entry $ADA(h)$ specifies the value of the ADA quantity computed thus far - similarly for the entry $ADH(m)$. These two lists are manipulated. We assume the existence of an equivalence relation $\equiv$ on ADA and ADH lists, telling us when two ADA lists and two ADH lists are essentially the same. For instance, we can decide that ADA lists ℓ_1, ℓ_2 are similar whenever a measure of distance, such as cosine distance or Jaccard distance, between the two vectors is below a threshold.

Algorithm 9 BiFixpoint

```
Input: $\mathcal{H}, \mathcal{M}, T$ (*$T$ is a training set *)
Output: $ADA, ADH$
 1: forall $h \in \mathcal{H}$, $ADA(h) \leftarrow \frac{1}{|\mathcal{H}|}$ (* initialize *)
 2: forall $m \in \mathcal{M}$, $ADH(m) \leftarrow \frac{1}{|\mathcal{M}|}$
 3: change ← true;
 4: while change do
 5:     $ADA'(h) \leftarrow \sum_{(f,m,t) \in dH(h)} w_{12}(h, f, m, t) * ADH(m)$
 6:     $ADH(m) \leftarrow \sum_{(f,h,t) \in dM(m)} w_{21}(m, f, h, t) * ADA(h)$
 7:     if $ADA' \sim ADA$ and $ADH' \equiv ADH$ then
 8:         change ← false
 9:     else
10:         $ADA \leftarrow ADA'$ and $ADH \leftarrow ADH'$
11: return $ADA, ADH$
```

The following result shows that this algorithm is guaranteed to converge.

Theorem 1. *For any sets* $(\mathcal{H}, \mathcal{M}, T)$, $BiFixpoint(\mathcal{H}, \mathcal{M}, T)$ *is guaranteed to converge and the returned solution is unique.*

Proof. The recursive equations describing RDA and RDH have the same structure of those of a bipartite Markov Chain where the first partition is the set of hosts $\mathcal{H}$ and the second is the set of malware $\mathcal{M}$. In fact we can rewrite the equations in the following way $RDA(h) = \sum_{m \in \{m' | (,m',) \in dH(h)\}} p_{12}(h, m) * RDH(m)$ and $RDH(m) = \sum_{h \in \{f | (,h,) \in dM(m)\}} p_{21}(m, h) * RDA(h)$ where $p_{12}(h, m) = \sum_{(f,m,t) \in dH(h)} w_{12}(h, f, m, t)$ and $p_{21}(m, h) = \sum_{(f,h,t) \in dM(m)} w_{21}(m, f, h, t)$. Because of the definition of the weights w_{12} and w_{21}, it follows that for each $m \in \mathcal{M}$, $\sum_{h \in \{h' | (,h',) \in dM(m)\}} p_{12}(h, m) = 1$ and for each $h \in \mathcal{H}$, $\sum_{m \in \{m' | (,m',) \in dH(h)\}} p_{21}(m, h) = 1$.

In addition both the $p_{1,2}(h,m)$ (with $m \in \mathcal{M}$ and $h \in \{h'|(,h',) \in dM(m)\}$) and $p_{2,1}(m,h)$ (with $h \in \mathcal{H}$ and $m \in \{m'|(,m',) \in dH(h)\}$) are greater than zero. Moreover, by construction, if we have an edge (h,m) also the edge (m,h) exists, both with value (probability) greater than zero. The last statement implies that for each pair of nodes (a,b) in each connected components in this Markov chain the probability to reach a from b (and vice versa) after a finite number of steps is greater than zero. This implies that, for each connect component, the associate transition matrix is regular, it follows that the $BiFixpoint(\mathcal{H},\mathcal{M},T)$ algorithm converges and the returned solution is unique. □

5.3.3 Patch Ability

Exactly analogous to the measures defined above, we can define two patch abilities: absolute and relative. The absolute patch ability (APA) and absolute patch hardness (APH) are defined in the following way:

$$\begin{aligned} APA(h) &= \textstyle\sum_{m\in\{m'|(,m',)\in dH(h)\}} w_{12}(h,m)\cdot APH(m) \\ APH(m) &= \textstyle\sum_{h\in\{h'|(,h',)\in dM(m)\}} w_{21}(m,h)\cdot APA(h) \end{aligned}$$

where $h \in \mathcal{H}$, $m \in \mathcal{M}$, and

$$\begin{aligned} w_{12}(h,m) = &\quad \frac{\frac{1}{API(h,m)}}{\sum_{h''\in\{h'|(,h',)\in dM(m)\}}\frac{1}{API(h'',m)}} \\ w_{21}(m,h) = &\quad \frac{API(h,m)}{\sum_{m''\in\{m'|(,m',)\in dH(h)\}} API(h,m'')} \end{aligned}$$

The relative patch ability (APA) and relative patch hardness (APH) are defined in the following way:

$$\begin{aligned} RPA(h) &= \textstyle\sum_{m\in\{m'|(,m',)\in dH(h)\}} w_{12}(h,m) * RPH(m) \\ RPH(m) &= \textstyle\sum_{h\in\{h'|(,h',)\in dM(m)\}} w_{21}(m,h) * RPA(h) \end{aligned}$$

where $h \in \mathcal{H}$, $m \in \mathcal{M}$ and

$$w_{12}(h,m) = \frac{\frac{1}{RPI(h,m)}}{\sum_{h'' \in \{h' | (,h',) \in dM(m)\}} \frac{1}{RPI(h'',m)}}$$
$$w_{21}(m,h) = \frac{RPI(h,m)}{\sum_{m'' \in \{m' | (,m',) \in dH(h)\}} RPI(h,m'')}$$

In order to compute the patch ability we can use the $BiFixpoint(\mathcal{H}, \mathcal{M}, T)$ algorithm with opportune modifications according the specific equations. By following the same proof methodology of Theorem 1, is possible to prove that the resulting algorithm converge and the returned solution is unique.

5.3.4 Collaborative Features

Let H be a set of all hosts and H' be a subset of H. We use a memory based collaborative filtering approach to produce features whose meaning is close to the number of malware infections in a specific set of hosts H', i.e. all the hosts of a specific country. The basic intuition under this collaborative filtering is that if host h is infected by malware m and host h' similar to h, then h' could also be infected by m. We assume the existence of a function $sim : H \times H \to [0,1]$ that similarity between hosts. Given a host $h \in H'$, the percentage of infected hosts in H (weighted by the similarity) by malware m is

$$infected(h,m) = \frac{\sum_{h' \in H, \exists (,m,) \in dH(h')} sim(h,h')}{|H|}$$

where $\exists(,m,) \in dH(h')$ is used to check if h is infected by m. Intuitively $infected(h,m)$ estimates the possibility that the host will be infected by m. In fact, the greater the percentage of hosts similar to h infected by m, the greater is the possibility (or probability) that h will be infected by m. Since we want to predict the expected number of infected hosts in H', our resulting variable is the following:

$$\sum_{h \in H'} infected(h,m)$$

If we assume that $infected(h, m)$ is a correct estimation of the probability that the host h will be infected by m, the value $\sum_{h \in H'} infected(h, m)$ estimates the expected number of infected host in H'. Since H can be huge (millions of hosts), we consider a reduced subset $\tilde{H}$ obtained by uniformly sampling H, and then we replace H in Eq 5.3.4 by $\tilde{H}$.

Similarity Measures. We define different notions of similarity – each of these notions yields a different feature estimating the expected number of infected hosts. The definitions below show some examples.

$$sim1(h, h') = 1 - |ADA(h) - ADA(h')|$$

$$sim2(h, h') = 1 - |RDA(h) - RDA(h')|$$

$$sim3(h, h') = 1 - |APA(h) - APA(h')|$$

$$sim4(h, h') = 1 - |RPA(h) - RPA(h')|$$

$$sim5(h, h') = \\ 1 - \sqrt{\frac{(ADA(h) - ADA(h'))^2 + (APA(h) - APA(h'))^2}{2}}$$

$$sim6(h, h') = \\ 1 - \sqrt{\frac{(ADA(h) - RDA(h'))^2 + (RPA(h) - RPA(h'))^2}{2}}$$

5.3.5 The FBP Model

The FBP Model is a general framework to predict the ratio of hosts infected by malware m in country c with a given regression method r such as GPR (Gaussian Process Regression), Elastic Net, Ridge, Lasso, and etc. The dependent and independent variables for the regression method are following:

Dependent variable. The dependent variable for each country -malware pair (c, m) is the percentage of hosts in country c infected by malware m.

Independent variables (features). For each country-malware pair (c, m), we compute ADI, API, RDI, RPI, $AADI$, $ARDI$, $AAPI$, $ARPI$, ADA, RDA, APA and RPA of hosts in country

c, APH and RPH of malware m and the $collaborative\ variables$ of the hosts in c with the 6 similarity measures. Other features are the percentages of the hosts which were not detected for m in countries which are economically similar to country c.[5] Two measures, Human Development Index (HDI) and Per Capita GDP PPP of countries are used to determine the similar countries. The features are computed for each day d with respect to four different time intervals from 0, $d-60$, $d-30$, $d-10$ to d day.

Nearest Neighbors (additional feature selection) To make the learned model more robust, we also take features of the $k-$nearest neighbors. For a given country-malware pair (c,m) and similarity measure S, we find the k *most similar* country-malware pair(s) P_k to (c,m) with respect to the past infection pattern. We then use the features of P_k as well (in addition to those of (c,m)) for predicting. Various existing techniques can be used for measuring similarity between two time-series e.g. Dynamic Time Warping (DTW), Euclidean distance etc.

5.4 Time-series model based prediction (DIPS)

In this section we propose a novel non-linear time-series model called DIPS (short for Detected, Infected, Patched, Susceptible) which is a variant of the SIR model of disease spread tailored for malware. Typical infection patterns (see Figure 5.2) are highly non-linear. Hence intuitively linear approaches are ill-suited to this task. Malware spread has multiple similarities to information diffusion and epidemiology (indeed, some of the earliest papers on this topic try to model it as a disease). Inspired by these models, DIPS adapts a recent model [153] to handle malware infections. The most challenging part is to deal with difference in the *detected* and *actual* infections. We assume that in the training dataset we have the true (as well as the detected) number of

[5] We define that country C and C' are similar w.r.t. measure T when $|T(C)-T(C')|/(T(C)+(T(C'))*0.5 \leq p$ where $p = \{3\%, 6\%, 9\%, 12\%, 15\%, 18\%\}$.

infections but in the test dataset, we are only given the number of actual detection.

5.4.1 Model Parameters

Similar to fundamental models like the Susceptible-Infected-Recovered (SIR) 'mumps-like' model, our model assumes that the machine population can be in one of the following states:

- *Susceptible:* In this state, a machine is not patched and hence is vulnerable to an attack by a particular malware.
- *Infected:* In this state, the machine is now infected, and spreads the malware by infecting other susceptible hosts.
- *Detected:* In this state, we have detected that the machine is infected (by say an anti-virus software running on the machine). Note that not all infected machines may be detected as infected.
- *Patched:* As the name suggests, in this state the machine is patched, and cannot get infected.

We temporally track the number of hosts in each of these states. In addition, we assume we have a variable $DET(t)$ which is the true number of detection at time t.

5.4.2 The DIPS *Model*

We are now ready to describe the DIPS model. There are a total of N potential susceptible hosts for the malware (one may also assume that there is an inflow of additional hosts at every time tick). Each host starts out in the Susceptible state. From the susceptible state, a host may move into the Patched state (at the rate of $\theta(t)$) or the Infected state. We assume each infected host may infect a susceptible host at a certain rate (called $\beta(t)$), which is also periodic and dictated by users' habitual behavior. An infected host may move into the Detected state upon detection, after which the host may become susceptible again (if it is not patched), or be patched (at the rate of $\delta(t)$) and never get infected again — these transitions are shown in Figure 5.3.

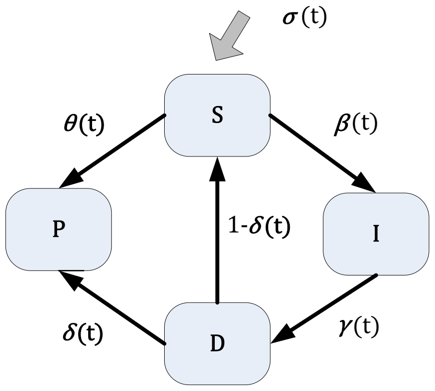


Fig. 5.3: DIPS model. D, I, P and S are the Detected, Infected state, Patched and Susceptible states

Infection Rate $\beta(t)$: This is the rate at which an infected host infects susceptible machines. We assume $\beta(t)$ has the following form:

$$\beta(t) = \beta_0(1 + P_a \cos(\frac{2\pi}{P_p}(t + P_S))) \ \ where,$$

- β_0: is the infection rate by users' (who are using the hosts) habitual behavior averaged over days
- P_p: stands for the period of a cycle. Infection data, like information diffusion data, and epidemiological data, typically shows obvious periodicity (like weekly) which we want to capture.
- P_a: represents the amplitude of the fluctuation.
- P_S: represents the phase shift in the cycle.

Patching rates $\delta(t)$ and $\theta(t)$: As mentioned above, we assume that susceptible hosts get patched at the same rate $\theta(t)$ and hosts in the detected state get patched at rate $\delta(t)$. More formally:

$$\theta(t) = \quad 0 \ (t < t_p) \quad or \ \ \theta_0 \ (t \geq t_p)$$

$$\delta(t) = \quad 0 \ (t < t_p, t_d) \quad or \ \ \delta_0 \ (t \geq t_p, t_d)$$

where t_p is the start time of the patching and t_d is the first detection time.

Error rate $\gamma(t)$: This captures the fact that the number of true detections $DET(t)$ may not exactly match the number of detection in our model (i.e. the number of hosts in the $D(t)$ state).

Finally, putting everything together, the DIPS model can be expressed using the following differential equations:

$$\begin{aligned}
S(t+1) &= S(t) - \beta(t)S(t)I(t) + (1-\delta(t))D(t) - \theta(t)S(t) \\
I(t+1) &= I(t) + \beta(t)S(t)I(t) - \gamma_0 DET(t) \\
D(t+1) &= \gamma_0 DET(t) \\
P(t+1) &= P(t) + \delta(t)D(t) + \theta(t)S(t)
\end{aligned}$$

Intuitively, $S(t), I(t), D(t), P(t)$ capture the number of hosts in a susceptible, infected, detected, and patched state, respectively.

5.4.2.0.1 The DIPS*-exp model* We have another variant of the DIPS model, called DIPS-exp where we have an inflow of additional susceptible hosts at every time-tick ($\sigma(t)$). All the equations in Equation 5.1 remain the same, except for the equation defining $S(t+1)$ which changes to:

$$S(t+1) = S(t) - \beta(t)S(t)I(t) + (1-\delta(t))D(t) - \theta(t)S(t) + \sigma(t)$$

5.4.3 Model Learning

The parameters of the DIPS model $\Theta = \{N, \beta_0, P_a, P_S, \gamma_0,$
$t_p, \theta_0, \delta_0\}$ (DIPS-exp additionally has a parameter of σ_0) need to be learned. Given two real time-sequences, $I_{true}(t)$ and $D_{true}(t)$, of the number of actual infections and detection (resp.) at time-tick t, we find the parameters which optimize the following equation:

$$\Theta^* = \arg\min_{\Theta} \sum_t (I_{true}(t) - I_{\Theta}(t))^2.$$

The learning algorithm is given in Algorithm 10. Basically, there are two phases to learn the parameters. The first step is to adjust the initial parameters to fit them on overall infection time-sequences ($I^{sum}_{true}(t)$ and $D^{sum}_{true}(t)$) and fix parameter t_p,θ_0, P_a and P_S. The learning order of the parameters is $N \rightarrow \beta_0 \rightarrow \gamma_0(\rightarrow \sigma_0) \rightarrow t_p \rightarrow \delta_0 \rightarrow \theta_0 \rightarrow P_a \rightarrow P_S$[6]. The second step is to fit the learned parameters on the specific infection pattern of country c for malware m. Here, we only update the following parameters: $N \rightarrow \beta_0 \rightarrow \gamma_0(\rightarrow \sigma_0) \rightarrow \delta_0$. The $LearnParameters(\Theta, I(t), D(t), flag)$ is the function to update the given parameter values Θ applying parameter $p \in \Theta \leftarrow \arg\min_p \sum_t (I(t) - I_\Theta(t))^2$ and to return the updated parameters, Θ' and the error, $\sum_t (I(t) - I_{\Theta'}(t))^2$. The flag represents the learning order of each step (if $flag = 1$, the first step, else the second step). Initially, $S(0) = N - D(0) - I(0), P(0) = 0, D(0) = DET(0)$, and $I(0) = I_{true}(0)$. We leverage the Levenberg-Marquardt (LM) algorithm [144] for each $argmin$ in $LearnParameters$.

5.5 Ensemble Models (ESM)

We have suggested three models: feature based prediction (FBP), and time-series model based prediction (DIPS and DIPS-exp). We now propose ensemble models, ESM which combine both of them to improve performance. We add the expected number of infections we get from the DIPS and DIPS-exp models as features for our feature prediction model. We do not use the *learned* parameters Θ^* of DIPS directly as features. This is because DIPS is highly non-linear; to take advantage of it, we use the *output* of DIPS instead.

To summarize, we extract all the features, add the output of DIPS as additional features, and then use a regression model to perform prediction. We used a variety of models like logistic regression and Gaussian process regression.

[6] The order affects the prediction performance a lot.

Algorithm 10 DIPS learning algorithm

```
Input: country $c$, malware $m$ and $X(m) = \{(c', I_{true}(t), D_{true}(t)) | c' \in Countries, I_{true}(t)$ and $D_{true}(t)$ for $c'$ and $m\}$
Output: $\Theta = \{N, \beta_0, P_a, P_S, \gamma_0, t_p, \theta_0, \delta_0, (\sigma_0)\}$
1: $\Theta \leftarrow \{N', \beta_0', P_a', P_S', \gamma_0', t_p', \theta_0', \delta_0', (\sigma_0')\}$          # Assign default parameter values
2: $I_{true}^{sum}(t), D_{true}^{sum}(t) \leftarrow \sum_{c' \in Countries, (c', I(t), D(t)) \in X(m)} I(t), D(t)$
                                                # Sum of # of infections/detection over countries w.r.t. each time
3: $err \leftarrow infinite, err' \leftarrow 0, \Theta' \leftarrow \Theta$
4: while $err > err'$ do                         # Learn all parameters
5:     $err', \Theta' \leftarrow LearnParameters(\Theta', I_{true}^{sum}(t), D_{true}^{sum}(t), 1)$
6:     if $err > err'$ then
7:         $\Theta \leftarrow \Theta', err \leftarrow err'$
8:         $err' \leftarrow 0$
9: Let $(c', I_{true}(t), D_{true}(t)) \in X(m)$ where $c'$ equals $c$
10: $\Theta' \leftarrow \Theta, err' \leftarrow 0$
11: while $err > err'$ do                        # Learn $N, \beta_0, \gamma_0(\sigma_0), \delta_0$ only
12:     $err', \Theta' \leftarrow LearnParameters(\Theta', I_{true}(t), D_{true}(t), 0)$
13:     if $err > err'$ then
14:         $\Theta \leftarrow \Theta', err \leftarrow err'$
15:         $err' \leftarrow 0$
16: return $\Theta$
```

5.6 Empirical Analyses

In this section, we empirically analyze the performance of the proposed prediction methods. Specifically, we study three questions:

- Q1: Can we predict the number of hosts in a country that are infected by a malware?
- Q2: How does predictive accuracy change with the number of infections?
- Q3: Does the prediction performance of the models depend on the number of hosts moni-

tored?

Experiment setting. We focused on top 50 most infectious malware in the 100 malware infection data extracted from WINE, and the top 40 countries (by GDP), leading to 2000 (50x40) country-malware pairs. This data includes over 1.45M unique hosts and 2.99M infections. The number of reported hosts varies from just from 634 (Iran) to 554,969 (United States). We ran the experiment over 541 days from 60th to 600th day among total 640 days. In order to ensure reliable data, we eliminated the first 60 and last 40 days of the 640-day data set.

Performance metrics. We checked RMSE (root mean squared error), NRMSE (normalized RMSE), and Pearson correlation coefficient (PCC) to evaluate performance. $NRMSE = RMSE$ $/(max(X) - min(X))$, where X is a set of ground true infection ratios of each malware-country pair over all days. Due to space constraints, we focus on NRMSE[7].

Train/Test Split. Each prediction model was trained in the first l days (on the features defined earlier) and tested for the remaining days. We used 80% of the data for training and the rest for validation.

Prediction models. We tested 7 different prediction models, FUNNEL [153][8], FBP, DIPS, DIPS-EXP, two different ESM models (ESM$_0$ and ESM$_1$) and FBP$^{+}_{Funnel}$. FUNNEL is a recently proposed time-series model which has been used to predict large scale disease patterns (and like DIPS, it too is based on epidemiological models like SEIR). ESM$_0$ refers to ESM using FBP with the expected values of the two DIPS models as additional features for each pair. ESM$_1$ uses FBP with FUNNEL and DIPS models. To estimate the performance of ESM more precisely, we also tested FBP$^{+}_{Funnel}$ which uses FBP with only FUNNEL (but without DIPS).

For FBP, we tested various regression methods such as GPR (Gaussian Process Regres-

[7] The other two measures provide similar results.

[8] The code was downloaded from http://www.cs.kumamoto-u.ac.jp/$\sim$yasuko/software.html

sion), Elastic Net, Ridge, Lasso, and etc with k more country-malware pairs whose infection patterns are similar with each country-malware. We tested with $k \in \{0, 1, 2\}$. The similarity was measured by Dynamic Time Wrapping (DTW). For FBP, DTW showed best performance in our testing among several similarity measures, such as Euclidean distance, average correlation coefficient, DTW and etc. In testing with 2000 (country,malware) pairs, FBP showed the best performance with GPR and $k = 2$ w.r.t. the average RMSE and NRMSE. The average correlation coefficients were similar for all. We did not consider $k > 2$ because the performance improvement from $k = 1$ to $k = 2$ was very trivial. In the following experiment sections, we fix k = 2 and GPR for FBP and the FBP setting was used for ESM_0, ESM_1 and FBP^{+}_{Funnel} also.

5.6.1 Performance over all countries and malware

Answering Q1 with overall performance. The overall performance of the prediction models is shown in Table 5.1. FBP is clearly better than FUNNEL with respect to all four measures, showing that the features defined in Section 5.3 are useful to predict infection patterns generally. Also, DIPS shows good prediction performance. The average MAE*, RMSE and NRMSE are 25%, 30% and 73% respectively of FUNNEL. Moreover the average correlation coefficient is more than 6 times that of FUNNEL. ESM models (FBP+DIPS) shows significant performance improvement from FBP and the best performance with respect to NRMSE. DIPS models show slightly better prediction performance w.r.t. MAE*, RMSE and Correlation coefficient than ESM_0 and ESM_1. We can also confirm that FBP^{+}_{Funnel} (FBP with FUNNEL only) and ESM_1 (ESM_0 with FUNNEL) do not show clear improvement from FBP and ESM_0. This suggests that DIPS significantly improves upon the predictive accuracy of past work. In any case we wish to point out that FUNNEL was designed and intended for disease patterns and not malware spread.

More detailed analysis: Performance of the prediction models for each country-malware

Model	MAE*	RMSE	NRMSE	C.C.
FBP	73.74	0.00170	0.179	0.12
FUNNEL	127.83	0.00269	0.226	0.08
DIPS	32.36	0.00083	0.165	0.50
DIPS-EXP	36.56	0.00096	0.223	0.48
ESM_0	39.41	0.00115	0.150	0.31
ESM_1	41.84	0.00118	0.151	0.31
FBP^{+}_{Funnel}	79.01	0.00189	0.179	0.19

Tab. 5.1: Average of MAE, RMSE, NRMSE and Correlation coefficient over 2000 pairs in testing period. Training ratio is 0.8. The MAE* values were computed with —# of ground true infected hosts - the expected # of infected hosts—. Others were based on the infection ratio of each country.

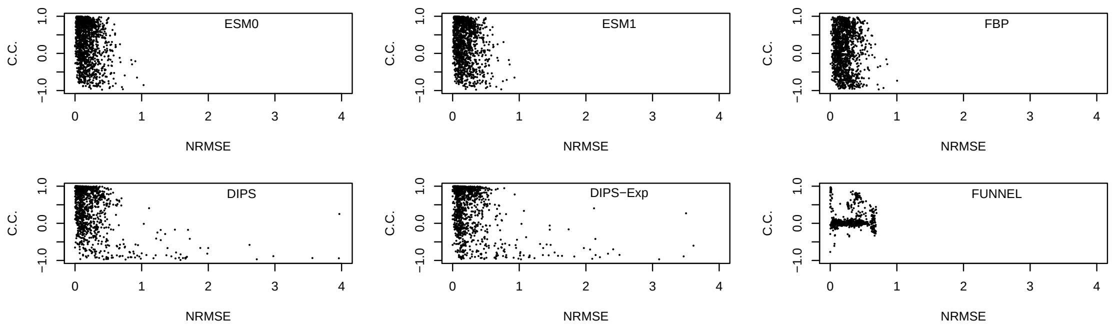
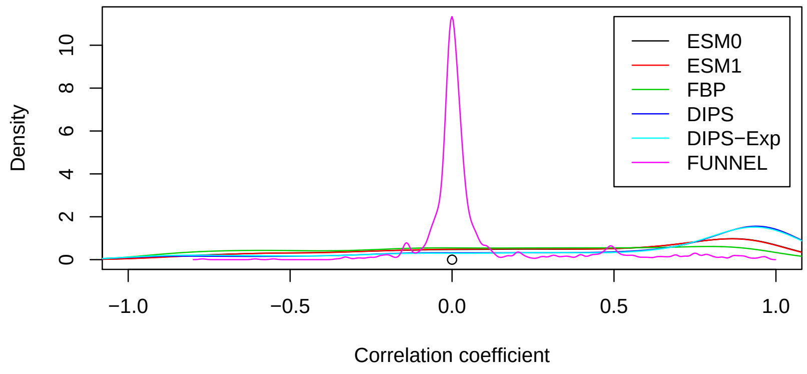


Fig. 5.4: Scatter plot for NRMSE and Correlation coefficient over 2000 pairs. Each point represents the performance for each country-malware pair.

pair is shown in Figure 5.4. DIPS models show many cases having very high correlation coefficient and very low NRMSE. But there are some cases having very high NRMSE when the correlation coefficient is very low. On the other hand, NRMSE of FBP is relatively low even when the correlation coefficient is pretty low. Interestingly, ESM models have the both strength of FBP and $DIPS$. To analyze the performance more, kernel density estimations are shown in Figure 5.5. As you see in Figure 5.5 (a), the density of ESM models and DIPS models are high in very high correlation coefficient values. In Figure 5.5 (b), the density of DIPS models and ESM models are really high with very low NRMSE values. But DIPS models have very long tail having very high NRMSE values.

(a) Correlation coefficients



(b) NRMSE

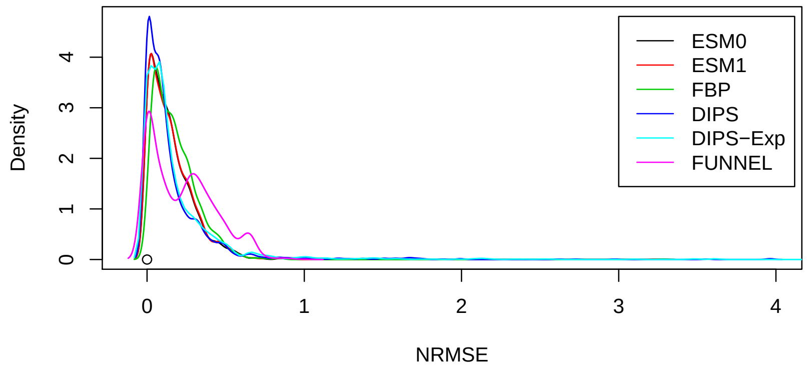


Fig. 5.5: Kernel Density of (a) Correlation coefficient and (b) NRMSE over 2000 pairs

Answering Q2. We define the infectiousness level as the average number of infected hosts over all days. Precisely, $ln(1 + Avg.\#\ infected\ hosts\ over\ days)$, here 1 is added as a default because $Avg.\#\ infected\ hosts\ over\ days$ could be less than 1. In Figure 5.6, we projected values of correlation coefficients and NRMSE over the infectiousness level w.r.t. each country-malware pair in order to check how each model works in different infectiousness levels. Each dot corresponds to a certain country-malware pair — the infectiousness level varies a lot. In terms of correlation coefficient, DIPS shows the best performance across all infectiousness levels especially with very high values in highly infectiousness levels, and ESM$_0$ follows after it. Switching the perspective to NRMSE more interestingly, DIPS still shows very good performance in general but some glitches in relatively lower infectiousness levels. This shows the performance of DIPS may be less robust when the number of infections is low. On the other hand, NRMSE of FBP is still low in the situation. This means the two models are very complementary to each other. So, ESM models show very stable and reliable performance regardless of infectiousness levels for both measures. Shortly, FUNNEL doesn't show any good performance even in highly infectiousness levels.

5.6.2 *Performance for certain countries*

Answering Q3. The number of reported hosts over countries in our data is very different. Figure 5.7 shows the performance of the models according to each country. The order of countries from left to right is the descending order w.r.t. the number of hosts in each country. DIPS models show good correlation coefficients. ESM models show stable and good performance for NRMSE over countries whose number of monitored hosts are very different. The overall prediction performance of DIPS and ESM do not depend on the size of monitored hosts except the cases when the number of monitored hosts are really small relatively. But for the countries having really small number of hosts, such like Nigeria (704) and Iran (634), the models show slightly worse

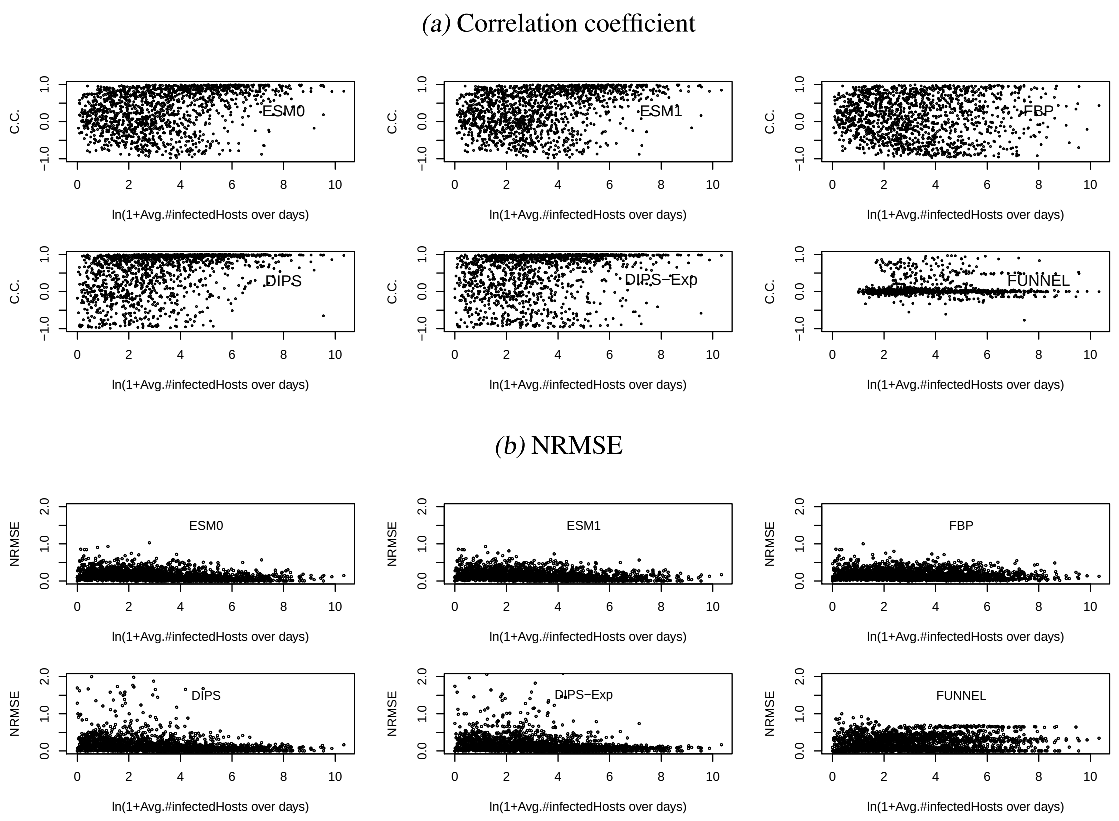


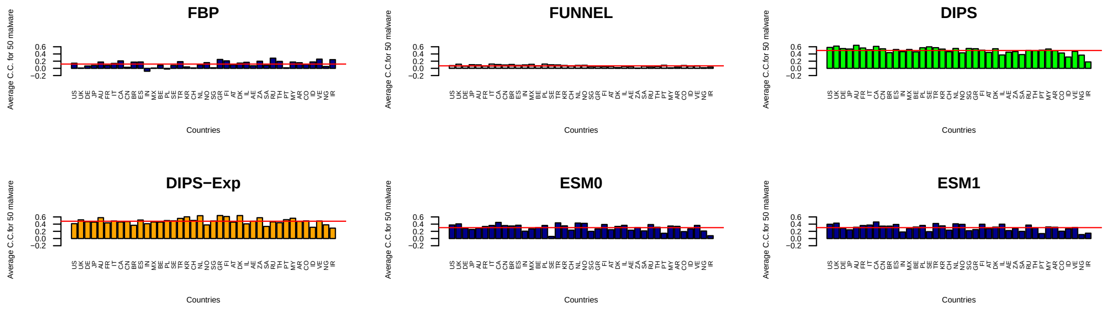
Fig. 5.6: (a) Correlation coefficient and (b) NRMSE by ln(1+ Avg.# infected hosts over all days)

correlation coefficient.

5.6.3 Summary of experiments

ESM and DIPS have low prediction errors and relatively high correlation coefficient values on average over all 2000 (country,malware) pairs. Both models have better prediction performance for the country-malware pairs having more infected hosts. For the cases having lower infectious levels, DIPS was not reliable and sometimes yielded very large errors. Interestingly, in the lower infectious levels, DIPS and FBP showed complementary performance. So, the combination of two techniques, ESM showed good and very stable performance for almost all cases except the very small number cases where DIPS was originally the best.

(a) *Correlation*



(b) *NRMSE*

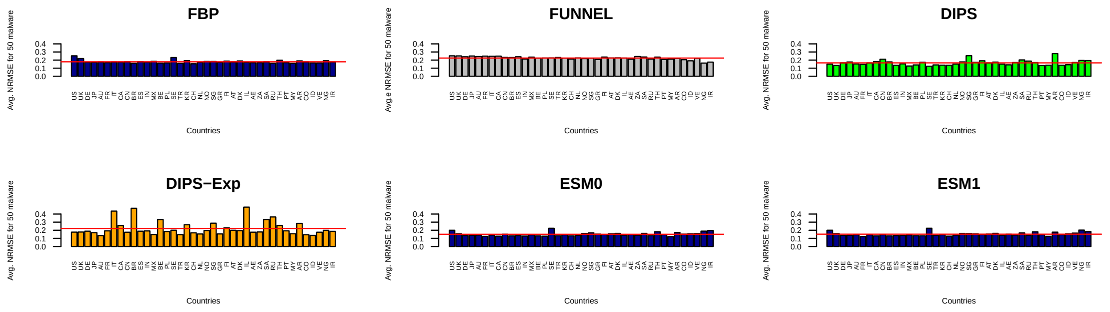


Fig. 5.7: (a) Correlation coefficient and (b) NRMSE by each country. The order from left to right is the descending order by the number of hosts over countries. Each red line is the average over countries.

5.7 Related Work

We considered related work in cyber-security, data mining and epidemiology. Much research has tried to model malware propagation. In July 2001, the Code Red worm infected 359,000 hosts on the Internet in less than 14 hours [160]. Code Red achieved this by probing random IP addresses (using different seeds for its pseudo-random number generator) and infecting all hosts vulnerable to an IIS exploit. Staniford et. al. [187, 186] analyzed the Code Red worm traces and proposed an analytic model for its propagation. They also argue that optimization like hit-list scanning, permutation scanning can allow a worm to saturate 95% of vulnerable hosts on the Internet in less than 2 seconds. Such techniques were subsequently employed by worms released in the wild, such as the the Slammer worm [159] (infected 90% of all vulnerable hosts within 10 minutes) and the Witty worm [201]. Gkantsidis et al. study the dissemination of software patches through the Windows Update service and find that approximately 80% of hosts request a patch within a day after it is released; the number of hosts drops by an order of magnitude during the second day, and is further reduced by factor of 2 in day three [89]. Additionally, [199] [138] [145] conducted measurement studies into routing instability in Broder Gateway Protocol (BGP) routers caused by catastrophic events, such as worm outbreaks or power outages. Other recent research also includes using machine learning methods like belief propagation on the file-machine graph [52] to infer files' reputations (say malicious or benign). Papalexakis et. al. [167] propose the SharkFin and GeoSplit models of spatio-temporal propagation of malware based on an analysis of the WINE data. Their system models only the total volume of malware attacks as a whole over time using a simplified model. In contrast, in this paper we model the *magnitude* of malware attacks *per machine* in context of the *machine usage* and using many more sophisticated variables. As such, our work can be also thought of as providing a fine-grained picture of malware attacks than just gross-volume.

Remotely related work also comes from the area of information diffusion. Information diffusion is the phenomena in which an action or idea becomes widely adopted due to the influence of others, typically, neighbors in some network [27, 91, 98]. There are a lot of dynamic process on graphs, all of which are related to virus propagation like blogs cascades [5, 137, 101] and information cascades [152].

Finally our DIPS model is inspired by epidemiological models. The canonical textbooks and surveys discussing fundamental epidemiological models like SIS and SIR include [111, 11]. Much work has gone into in finding epidemic thresholds (minimum virulence of a virus which results in an epidemic) for a variety of networks [17, 154, 11, 130, 87, 169].

5.8 Conclusions

In this chapter, we studied the problem of predicting the volume of actual malware infections in a country given past data. This is a challenging problem because we can only detect a fraction of malware infections. There are many potential benefits of accurately predicting malware volume, which can have implications for better anti-virus software, and targeted patching. Using Symantec's extensive malware database WINE, we investigated the infection patterns of 50 malware in more than 1.4 million hosts spread over two years in several countries. We present ESM, an ensemble based approach which carefully leverages well-designed domain-specific features (FBP) and a novel non-linear time-series model DIPS which incorporates detection and infections seamlessly. We show that our approach gives robust and stable predictions, even in case of low volume of infections, across all the countries over several metrics (correlation coefficients and NRMSE).

Chapter 6: Systemmatic banking crises prediction

6.1 Introduction

The global financial crisis has underscored the role of financial connectedness as a potential source of systemic risk and macroeconomic instability. It has also highlighted the need to better understand whether an increase in connectedness leads to a higher probability of a financial crisis. In this section, we contribute to the literature on "early warning systems" (EWS) by investigating whether measures of systemic risk, which are based on modeling the global financial system as a network, can serve as early warning indicators and improve the performance of standard crisis prediction models. In doing so, we use two commonly-employed prediction methods – a data mining algorithm and an econometric crisis incidence model.

We build on the literature that links connectivity in the global banking network ("GBN") to systemic risk and contagion that arise during systemic banking crises.[1] In particular, we compute indicators of interconnectedness based on the pattern of financial linkages worldwide and test their performance as leading crisis indicators. The EWS literature has traditionally focused on balance of payments and currency crises in emerging market countries,[2] and has identified several

[1] See [85], [64], and [8].

[2] Seminal contributions include [74], [121], [67], [122], and [24]. [23] caution that the out-of-sample performance of EWS, especially over long horizons, has been mixed. In recent years, the literature has received a new stimulus, with a focus on evaluating factors associated with recessions and growth collapses [66, 34], fiscal crises [18], and financial crises in advanced economies [113]. Credit bubbles have been identified as the most robust determinant of banking crises over very long time spans [181]. For an examination of how traditional models fared in predicting 2009

macroeconomic factors that are robustly correlated with crises. These include asset price bubbles, low international reserves, real exchange rate appreciation, hyperinflation, and excessive short-term and foreign exchange borrowing. Our approach adds to this literature by focusing on the role of financial connectedness in predicting systemic banking crises. It also speaks to the concern that complex linkages among nations and financial institutions may be a source of systemic risk and accentuate business cycle downturns [204, 188, 182].

Our strategy is two-pronged. First, we assess the ability of a wide range of network-based connectedness measures to predict crises according to a data mining technique, namely a "classification algorithm" patented by one of the authors ([189]). Classification algorithms have been used extensively in applications in the medical sciences, genetics, finance, meteorology, web research, and terrorism research. Classification algorithms allow us to evaluate the predictive ability of a large number of potential crisis indicators and to narrow down the set to the "most promising" indicators that we can use in a parsimonious regression model. Second, we employ a standard econometric model of crisis prediction – a probit – and evaluate its performance with and without this set of indicators. Probit models have been used with varying degrees of success to predict balance of payments, debt, and banking crises.[3]

Our findings indicate that network-based measures of financial connectedness help predict systemic banking crises. In particular, increases in a country's interconnectedness and decreases in the connectedness of that country's direct financial partners (or neighbors) are associated with a higher probability of crisis. We also find evidence of non-linearities – interconnectedness has a more pronounced influence on crisis probability at higher levels. The results of the classification algorithm suggest that measures of financial connectedness are useful in predicting both crisis-years and the onset of crises. This is especially true when we focus on the most recent wave of

emerging market crises, see [44].

[3] See [68], [1], [61] and [73] for reviews of the large econometric literature on EWS.

crises (2007-2010). The probit analysis reinforces these results. Estimates from a probit model that incorporates connnectedness indicators in addition to standard macroeconomic fundamentals generally outperforms, in and out of sample, a model based on fundamentals alone.

Our approach adds to a growing literature on the evolution of financial connectedness over the business cycle. In this literature, cross-country linkages are captured in many different ways. [59] build a GBN using data on cross-country debt flows and equity investments. The authors show that countries that are more connected, hence more centric, in the GBN prior to the recent crisis experienced a smaller loss of output during 2008-2009. [105] and [46] construct GBNs using granular information on bank-level exposures in the syndicated loan market. [105] finds that US recessions are associated with lower network connectivity. [46] show that the home countries of banks that are "central" in the GBN experienced lower stock market crashes during 2007-2008 compared to other countries. [86] argue that direct and indirect financial exposures can act as conduits for the transmission of banking crises internationally. [157] construct a GBN based on cross-border bank lending and securities investments and show that connectivity tends to rise before systemic banking crises and to fall afterward. This body of work underscores the importance of financial connectivity for economic performance during crises.

Few studies look directly at the question whether financial connectedness can serve as an early warning indicator. [29] show that linkages across four US financial sectors – hedge funds, banks, broker/dealers, and insurance companies – have become more significant prior to the 2007-2009 crisis. In particular, the impact of returns of banks and insurers on those of hedge funds and broker/dealers was stronger than the other way around before the crisis. This finding suggests that measures of connectedness from the returns correlation network may be useful out-of-sample indicators of systemic risk. [45] measures financial integration based on the network of inter-bank lending and borrowing in the syndicated loan market, and finds that country-level indicators of financial connectivity, which are proxies for financial integration, have predictive power for the incidence of banking crises during 1980-2007 after controlling for credit growth and other macroeconomic fundamentals.

Our work differs from these studies in several ways. First, while [29] focus on US financial institutions and hence on systemic risk in the US economy, we look at the ability of *global* financial linkages to predict systemic banking crises in a *cross-country* setting. Second, we investigate the performance of EWS augmented with measures of financial connectivity and include 2008-2010 crises (not covered in [45]). Including the latest generation of crises – which account for 74 out of 430 country-years over 1978-2010 – is important because financial connectedness may arguably have played a more important role during the most recent wave of crises compared to earlier ones. Third, while we use less granular information than [86] and [105], who employ bank-level information on syndicated loan contracts, our data gives a comprehensive view of cross-border banking *system* exposures – of which syndicated loans are only one component. Finally, we consider a wide range of network-based connectedness indicators – 27 distinct indicators – along with their lags and growth rates up to five years prior to a crisis.

To construct the GBN we use annual data on cross-border banking system exposures from the BIS locational statistics over 1978-2010. The BIS locational statistics have been used extensively to analyze geographical patterns in financial linkages [157, 107], financial integration [120], and factors associated with capital flows [33, 32]. The data represent foreign exposures of banks in BIS reporting countries vis-a-vis residents (borrowers) in all other countries. The data are representative of banking activity in each location as reporting banks typically account for more than 90 percent of total banking assets in each BIS reporting country. For each year we construct a directed GBN with countries as "nodes" and cross-border exposures as "edges," and compute network indicators using both the binary and weighted versions of this network (defined in Section

2.1).

Our analysis addresses several related, yet distinct, dimensions of systemic risk. The first one is connectedness narrowly understood as direct exposures among economic agents (such as financial systems or institutions). Connectedness matters for crisis incidence because the failure of an economic agent can lead to the failure of another agent through this direct channel. An indirect channel is contagion, by which an agent's failure leads to panic among agents not directly connected to him/her. It is difficult to empirically separate the effects of connectedness from those of contagion. In this section, we take a view of connectedness that encompasses the two concepts and refers to *both* direct and higher-order (indirect) exposures. We take this concept to the data by mapping financial linkages across countries and extracting information from the pattern of connections by means of network analysis.

The remainder of this chapter is organized as follows. In Section 6.2, we introduce the data, describe the construction of the GBN and present discuss long-term trends in financial connectivity. We also describe the classification algorithm. In Section 6.3, we present the results, ranging from simple correlations between connectivity and crises to the performance of the classification algorithm and that of the regression analysis. In Section 6.4 we conclude. Additional results are available in an online appendix.[4] In this chapter, figures and tables are in the end of this chapter.

6.2 Data and Methods

6.2.1 Data Description

We use data from the BIS bilateral locational statistics – a component of the BIS international banking statistics – that provide a comprehensive view of international banking transactions. The

[4] The online appendix is available for download from www.camelia-minoiu.com/crisis-prediction-appendix.pdf. Tables and figures cited in the text are labelled "A" for this online appendix.

data represent stocks of cross-border assets (such as loans, debt securities, and other assets) held by banking systems in 210 countries during the 1978-2010 period. Similar to balance-of-payments statistics, these data are defined on a *locational* basis, i.e., they capture exposures of reporting banks in a given location regardless of their ownership structure [30, 31]. As such, the BIS locational statistics are useful for measuring financial integration across countries and territories (such as off-shore financial centers) that report financial data. [5]

The sample comprises 29 BIS reporting countries and 181 non-reporting countries. In what follows, we refer to the former as "core" countries and to the latter as "periphery" countries. Note that the core countries comprise advanced economies, emerging market countries, and several offshore financial centers.[6],[7] The GBN is constructed solely on the basis of data from the core countries. These countries also report liabilities, which we use to impute assets of periphery countries vis-a-vis core countries. To understand how representative these data are of true global linkages, we can imagine the full network as consisting of three layers: the links among core countries, those between core and periphery countries, and those among periphery countries. Since core countries report positions to the BIS, we observe the full extent of linkages within the core. We similarly observe exposures of core countries vis-a-vis periphery countries. The liabilities of core countries vis-a-vis periphery countries are used as the claims of periphery countries vis-a-vis

[5] See Table A1 in the online appendix for a list of countries and territories included in our sample.

[6] Countries are invited to report financial data to the BIS when their financial sector becomes large. In order to minimize the effect of changes in sample composition on our results – due to countries starting to report to the BIS at different points in time, or because some countries become independent states during the sample period – we also experimented with a GBN based on data from the 13 core countries that report their claims continuously and the 175 countries that exist as independent entities since 1978, but the results were largely unchanged.

[7] We use the "core" and "periphery" labels for the nodes in the GBN based on the structure of the data rather than a formal core-periphery model. [180] shows that the international financial network underpinned by cross-country asset holdings had a core-periphery structure before the global financial crisis.

core countries, so we also observe linkages from the periphery toward the core.

It is important to note that we do not have data on linkages *among* periphery countries; in other words, within-periphery connections are missing from the GBN. This data limitation should be kept in mind when interpreting the results as our network variables may suffer from measurement error and we cannot know if these errors are systematic. To date, data availability remains a major problem in studies of global systemic risk (as discussed, for instance, in [48]). Nonetheless, ours is a first attempt to utilize core countries' liability-side data from the BIS locational statistics to obtain as comprehensive a view as possible of cross-country banking linkages.[8]

The GBN is directed: a directed edge is created from country A to country B if country A has non-zero claims vis-a-vis borrowers in country B. In the binary GBN, an edge exists if a country has non-zero banking exposures vis-a-vis another country. The data for each year between 1978 and 2010 is modeled as a separate GBN.

In the analysis we consider two variants of the GBN with different edge weighting schemes. The first one – the "baseline network," used in the chapter – refers to edge weights given by log-exposures divided by the log-product of GDPs within each pair of nodes. PPP-adjusted GDP data come from the Penn World Tables (Mark 7.1). The advantage of this weighting scheme is that banking system claims are scaled by the economic size of countries, which ensures that increases in weighted indicators of connectedness are not driven solely by economic growth. The second network – the "alternative network," for robustness tests – uses real log-exposures as edge weights.

[8] Exposures from asset- and liability-side data are not strictly comparable because the former refer to assets of banking systems in core countries vis-a-vis residents in periphery countries, while the latter refer to assets of residents in periphery countries vis-a-vis banks in core countries. This inconsistency is unlikely to affect our binary GBN, but may lead to some measurement error in the weighted GBN. For purposes of our analysis, we need to obtain as comprehensive a view of cross-country financial linkages as possible; therefore we prefer to use an imperfect measure of periphery-core linkages rather than assume that they do not exist.

(Weighted exposures are deflated using the US CPI.) The results based on the first GBN are largely robust to using the alternative network. [9]

6.2.2 Global Connectivity: Stylized Facts

We assess the extent and trend in global connectivity during 1978-2010 using selected indicators of connectedness, complementing earlier work on the topological properties of the international financial network [59, 157, 180]. (See Appendix Ap.2.1 for formal definitions of all network indicators.) The GBNs for 1980 and 2007 are shown in Figure 6.1. The visualizations depict an increase in global financial connectivity both in the full sample (top panels) and the core (lower panels), with the latter showing a significantly denser network in 2007, before the global financial crisis.

Figure 6.2 depicts network density and total exposures during 1978-2010. Network density refers to the share of observed edges in the total number of possible edges, and ranges in the full GBN between 5 percent at the beginning of the sample period and 12 percent in 2007. These figures are markedly larger in the core network, with density rising from 17 percent in 1978 to 46 percent in 2007. While network density has increased continuously over the past three decades, at an annual average rate of 3 percent, cross-border exposures have also risen, but almost three times faster (at an annual average rate of 8 percent). At the peak of network density in 2007, total cross-border exposures amounted to USD 90 trillion – about twice the size of global GDP.

Averages for our network indicators are reported in Table 6.1 at different points during the sample period. The last column in the table shows the percent increase in each network indicator between 1978 and 2010, indicating that financial integration – proxied by our rich set of interconnectedness measures – has been on the rise since 1978. Average degree and strength, representing

[9] See Tables A3, A7, Figure A5 in our online appendix.

the number of financial partners and the magnitude of cross-border banking system exposures in the GBN, exhibit the largest increases among all indicators, by a factor of 1.5 over 1978-2010.

Clustering coefficients capture the tightness of link formation around a node and are bounded between 0 and 1. The clustering coefficients developed by [78] answer the following question: Given that a node has connections to two neighbors, what is the probability that those neighbors are also connected with each other (i.e., that they form a triangle)? For the 2000s the answer is 80 to 90 percent, which implies a high tendency for nodes to form triangles. The second clustering coefficient we consider, proposed by [148], captures the effectiveness with which a node's neighbors are connected with one another and has also risen over the sample period. Overall, the rise in average clustering coefficients – by between 59 and 85 percent during 1978-2010 – suggests that not only has the number and intensity of bilateral financial links increased, but countries' financial neighbors have also become more tightly linked with one other.[10]

The next set of indicators refers to connectivity among a country's neighbors. Average nearest-neighbor degree (ANND) is the number of creditors or debtors that each country's immediate neighbors have. For instance, "out-out" ANND is the average number of creditors of a creditor country. Similarly, "in-in" ANND is the average number of debtors of a debtor country. ANNS indicators take into account the magnitude of exposures, too. All neighbor connectivity indicators until 2007 and fall in the aftermath of the crisis.

Our last indicator is the Herfindahl-Hirschmann index (HHI), a measure of market competition. Although this is not a network indicator, it helps assess the degree of portfolio diversification in the GBN. The HHI, computed from the debtor countries' perspective, is equal to the sum of squared shares a country represents in its creditors' portfolios. (For instance, if country A has 50

[10] Recent work argues that clustering coefficients based on different triangle patterns in banking networks have different implications for systemic risk [192].

percent of its liabilities vis-a-vis country B and 50 percent vis-a-vis country C, then A's HHI is equal to 0.5.) The HHI indicates a relative lack of concentration in the first half of the period, but increases during the 1990s to levels of moderate concentration. By the end of the period the HHI declines again, with debtor countries increasingly diversifying their pool of creditors. This trend coincides with a period of high economic growth coupled with increased capital account openness and financial system reform in many countries.

In sum, our network measures depict a general trend towards greater financial integration during 1978-2007 – both in terms of countries' direct connectivity and the tightness of their neighbor networks. This trend was interrupted by the global financial crisis. But was this increased financial integration higher than what would be expected conditional on the GBN's observed density? To answer this question, for each year we generate a sample of 10,000 random networks that have the same density as the GBN but we re-shuffle either (i) the location of the edges, or (ii) the edge weights while keeping edge locations fixed (for a similar approach, see [79]). We then compute, for each year, average network indicators (across nodes) in the random networks and compare them to the observed ones. [11] We find that our observed GBNs are more "interconnected" than expected, which indicates that the increase in financial integration observed during 1978-2010 is not merely the result of higher network density.[12]

6.2.3 Classification Algorithm

We first evaluate our connectedness measures' effectiveness as early warning indicators with a data mining method, namely a "classification algorithm." The 27 indicators we consider are included in the algorithm in levels lagged up to 5 years, and in growth rates computed over the past 1, 2,.., up to 5 years (for a total of 270 indicators). We set up the GBN as a matrix whose rows corre-

[11] See Figure A4 in our online appendix.

[12] We thank an anonymous referee for suggesting this comparison.

spond to country-year observations and whose columns correspond to (i) our set of network-based indicators of connectedness, and (ii) the crisis incidence variable $\texttt{crisis}$. The crisis variable takes value 1 if a systemic banking crisis occurred in the country during the specified year and 0 otherwise. (See Section 3.1 for the formal definition of systemic banking crises.) We run the algorithm on the full set of crisis-years (rather than the onset of crises) to preserve the maximum amount of variation in the data and improve the algorithms' chances to learn from it.

The output of the classification algorithm is a series of association rules that describe relationships between measures of connectedness and the probability of a crisis ([6]). Association rules are of the form "If condition C holds over the network indicators, then the crisis variable takes value 1." The condition C can take the form $\ell_1 \leq V_1 \leq u_1 \wedge \cdots \wedge \ell_n \leq V_n \leq u_n$ (a collection of sub-rules) where each V_i is a network indicator and the ℓ_i, u_i's are real numbers. (Here all network indicators have real valued attributes.)

For an association rule to be acceptable, it needs to have a high *confidence* and a high *support*. Confidence is the conditional probability $\mathbb{P}(\texttt{crisis} = 1|C)$, i.e., the share of correctly-called crises. In most applications, the confidence is required to be above a pre-specified threshold. However, confidence alone does not define "goodness" of an association rule because a rule may have a high probability (e.g., probability of 1) with C being true just once and $\texttt{crisis}$ being true on that one occasion. In addition to confidence, we use support, which is defined as $|\{r|r.\texttt{crisis} = 1 \wedge r.C\}|$ where r is a row in the matrix described above, $r.\texttt{crisis}$ is the value of the crisis entry in row r and $r.C$ is true iff row r satisfies condition C. Thus support is simply the number of times (i.e., rows) for which both C and $\texttt{crisis} = 1$ are simultaneously true for a given country-year pair. As in the case of confidence, support is required to exceed a threshold – i.e., the two conditions must co-occur sufficiently many times – in order for the association rule to be acceptable.

We also require association rules to satisfy the property that "negative probability" be low where negative probability is defined as $\mathbb{P}(\texttt{crisis} = 1|\neg C)$. This computes the probability of a crisis occurring when C is not true (the $\neg$ denotes negation). By ensuring that $\mathbb{P}(\texttt{crisis} = 1|\neg C)$ is below a pre-specified threshold, we ensure that condition C acts like a "canary in a coalmine." When C is true it predicts a crisis with high probability - when it is not true, it predicts a crisis with very low probability.

Algorithms like the one described here have been useful in a wide range of applications that involve classification problems – that is, separating data into multiple buckets (e.g., two buckets, one consisting of situations where some event occurred and one where the event did not occur). For instance, classification algorithms have been used to predict whether some patients have breast cancer [124], to explore associations among genes [65], and to predict upticks in terrorism [190]. In Internet research, classification algorithms are used to predict the category (e.g., sports vs. politics) to which a webpage belongs, whether a particular email constitutes spam, and classifying Internet traffic into various categories.

Although classification algorithms have not been used much in the economics literature, there have been applications to finance. For instance, classification algorithms have been used to predict individual stock movements [149], to classify individuals according to their credit scores [14], and to separate buyer- from seller-initiated trades [164, 13, 76]. To the best of our knowledge, this is the first application to a macroeconomics question – the prediction of systemic banking crises – one that is made possible by the availability of a large number of potential predictors obtained through network analysis. Results based on the classification algorithm are presented in Section 6.3.2.

6.3 Results: Connectivity and Systemic Banking Crises

6.3.1 Exploring the Data: Conditional Correlations

Here we explore the ability of financial connectedness indicators to predict crises. We start by examining whether network indicators display a different behavior before vs. after the onset of a crisis. Crisis incidence data come from [139]. Systemic banking crises are defined as years in which the banking sector exhibits significant distress and is subject to important policy interventions – that is, at least three out of the five following interventions are present: public guarantees, liquidity support, asset purchases, nationalizations, and large restructuring costs (for more details, see [139, 140]). The number of systemic banking crises thus defined is shown in Figure 6.3 for the 1978-2010 period. As shown in the figure, periphery countries account for most of the crises that took place prior to the mid-2000s, while systemic events are roughly equally split between core and periphery countries during 2007-2010.

To allow for the impact of connectedness on the likelihood of crises to be driven by global and country-specific factors, we start by regressing the network indicators from our baseline GBN against a full set of country and year dummies and obtain the residuals. We then plot these residuals – averaged across all country-year observations – within a window of five years around the onset of systemic banking crises. Figure 6.4 shows the evolution of selected network indicators around banking crises after controlling for unobserved heterogeneity as described above. Note that here we do not account for the impact of macroeconomic fundamentals (we do so in Section 3.3).

The three panels in Figure 6.4 present several notable patterns. In Panel 6.4a we notice that degree and strength increase steeply before the onset of a crisis and level off subsequently. The signal here is the rapid growth in interconnectedness captured by these simple network measures.

The degree of clustering also rises before banking crises and peaks 1-2 years before their onset, after which it begins to decline (Panel 6.4b). The pre-crisis decline in clustering reflects a lower probability that a given country's financial partners are connected, and suggests that there is turbulence in the network right before the onset of a crisis. What happens to a country's neighbor connectedness before crises? Panel 6.4c shows that neighbor degree and strength decline 3-4 years before the onset of a crisis. This suggests that there is link destruction among countries' higher-degree connections long before the onset of crises; and that the decline in neighbor connectivity can also be informative about the arrival of a crisis.

In the panels of Figure 6.4 we also show a dotted line that captures most of the information in our indicators. It represents the first principle component (henceforth labelled "1st PC") extracted from the variation in three groups of indicators through Principal Component Analysis (PCA). (See Appendix Ap.2.2 for details.) The first group of measures comprises degree and strength indicators. The second group includes all the clustering coefficients (both binary and weighted). The third one includes all the neighbor degree and strength indicators. The first principal component (PC) generally explains between 80 and 90 percent of the variation in the underlying indicators, and helps summarize the information from many variables into a single factor, reducing the dimensionality of our data. Working with PCs instead of the full set of variables is useful in an EWS if we wish to estimate parsimonious models.

Following this preliminary evidence that hints at the ability of financial connectedness to predict crises, we next seek to strengthen our findings using the classification algorithm and regression analysis.

6.3.2 Results from the Classification Algorithm

We ran the classification algorithm on the baseline GBN for the full period (1978-2010) and two sub-periods: 1978-2002, which covers first- and second-generation crises mainly affecting developing economies; and 2003-2010, which covers the latest wave of banking crises. Table 6.2 summarizes the in-sample performance of the algorithm in predicting crisis-years for the full sample and separately for core and periphery countries. For each sample and period, we report the number of actual crises (column 1), the number of crises predicted by the algorithm (2), support (3), precision (4), recall (5), and the number of sub-rules (6). Precision and recall are measures of the performance of the algorithm. Precision is defined as the number of correctly predicted crises divided by the number of predicted crises. It takes maximum value when are no "false alarms" (Type 1 error is 0). Recall refers to the number of correctly predicted crises divided by the number of actual crises. It attains a maximum when there are no "missed crises" (Type 2 error is 0).

For the sample of countries included in the baseline GBN there were 409 crisis-years during 1978-2010, of which more than 75 percent occurred before 2003. The algorithm produces 34 association sub-rules (Table 6.2, column 6) which we can combine into one giant rule to predict crises.[13] To give an example of several sub-rules, the algorithm yields that "A crisis will occur in a given year if (i) out-degree at $t-3$ is between 155 and 171 or (ii) the "in" BCC at $t-5$ is between 0.115 and 0.118 or (iii) in-strength at $t-5$ is between 63.5 and 66.5," etc.(See the table in Figure 6.6 for all the sub-rules generated by the algorithm for core countries during 2003-2010.) The rules can be expressed by GAP (Generalized annotated program, [131]) directly. For example, the gap rule for (i) would be simply $crisis(c,t) : 1 \leftarrow outdegree(c,t-3) : \mu \;\&\; \mu \geq 155 \;\&\; \mu \leq 171$ where c is a country and t is a target year. The set of gap rules can be a diffusion and

[13] Each association rule has the form `crisis` if C_1,..., `crisis` if C_n. These rules can be combined into one large rule `crisis` if C_1 or $\cdots$ or C_n saying that a crisis occurs if any of the conditions $C_1, \ldots, C_n$ is true.

infection model for banking crises for countries.

Over the full period, precision is very high, reaching 94 percent for the full sample and 88 percent for core countries (column 4). Recall is relatively low at 11 percent for the full period, but higher by subperiod: 94 percent for 1978-2002 and 40 percent for 2003-2010 (column 5). Precision and recall are lower for core countries compared to periphery countries during the full period, which may be partly due to the lower number of crisis-years experienced by advanced economies overall, and hence the lower ability of the algorithm to learn about them from that sample.[14] By contrast, for the 2003-2010 period which mainly contains crises for core countries, the algorithm has high precision and recall (0.85 and 0.83 respectively), thus performing quite well in identifying crisis-years for the latest wave of banking crises.[15]

The consistently low Type 1 and Type 2 errors produced by the classification algorithm add to our preliminary evidence that network-based measures of connectedness can serve as early warning indicators. When we carefully examine the sub-rules, we notice that almost of all the indicators considered are selected by the algorithm in some rule, but lagged levels show up more frequently than lagged growth rates [16]. In the next step, we use this information to construct a relatively parsimonious empirical model that uses as covariates those indicators identified by the algorithm as the most promising. Specifically, we estimate a regression-based EWS that controls for standard macroeconomic fundamentals and adds measures of financial interconnectedness. To keep the number of covariates manageable, we group the indicators into three categories – as in

[14] Another reason is that four advanced economies that experienced crises during 2007-2008 are in fact part of our periphery because we do not have cross-border exposure data for them.

[15] Notice also that the algorithm produces a larger number of sub-rules when there is less variation in the data. This is the case, for instance, in the full sample (1978-2002) and the subsample of periphery countries, which were more likely to experience crises before 2003 but are also less interconnectedness due to the absence of within-periphery edges. For these countries, network measures have less predictive content for banking crises.

[16] See Table A4 in our online appendix for the frequency with which different indicators appear in sub-rules

Figure 6.3 – and retain the first or two PCs from each group. Our rule for selecting the PCs is that they have eigenvalues greater than one.

6.3.3 Results from Regression Analysis

We estimate a benchmark binary dependent variable model in the full sample of countries over 1978-2010. The dependent variable is an indicator that takes value 1 for the onset of systemic banking crises.[17] We draw on the recent crisis prediction literature to specify a small set of key macroeconomic fundamentals that have previously been identified as leading indicators of crises. Specifically, we control for per capita income (at PPP), net foreign assets (in percentage of GDP), foreign exchange reserves (in percentage of GDP), real exchange rate (RER) misalignment, and periods of sustained capital inflows. RER misalignment is computed relative to the purchasing power parity-implied real exchange rate and is further adjusted for the Balassa-Samuelson effect, as in [175].

The net foreign asset and foreign exchange positions, as well as periods of RER overvaluation are systematically associated with increased likelihood of currency and/or external crises [121] while credit booms fueled by large capital inflows – captured in our model by a dummy variable for capital flows bonanzas [173] – play a prominent role in the run-up to banking crises [181, 7]). We also include a standard variable that allows for the possibility of contagion – a dummy variable for at least one neighbor experiencing a crisis. Our regressors are a mixture of variables that predict financial crises in general, since banking, currency, and debt crises often occur together ([122]). All regressors are lagged one year.

Estimates from the benchmark model with macroeconomic fundamentals are reported in

[17] Following [92], the dependent variable takes value 1 in the year of crisis onset, is set to missing for the subsequent four years (as countries often lack access to international capital markets in the years of the crisis), and is 0 in non-crisis years.

Table 6.3 for the probit and logit estimators. We notice that the coefficients on the macroeconomic characteristics have the expected signs across specifications: a more favorable net foreign asset position and higher foreign exchange reserves are negatively associated with the likelihood of crises. The probability of a banking crisis goes up when countries experience sustained periods of high capital inflows or an overvalued RER and hence a loss of competitiveness. Having at least one neighbor in crisis also raises the probability of the country itself experiencing a crisis – which hints at the possibility of contagion.

While our preferred specifications do not include country or year fixed effects, for completeness we also show models with them. Country fixed effects control for omitted time-invariant characteristics that may systematically drive a country's predisposition towards crises. Year fixed effects control for global shocks that may cause crises to cluster together (e.g., the 1997-1998 East Asian crises). Including these dummies leaves the coefficient estimates largely unchanged; however, we see such models as less helpful for crisis prediction. One reason is that we cannot anticipate future global shocks, so year fixed effects are unknown *ex-ante*. Another reason is that estimated country fixed effects are not informative from a policy perspective because they do not possess economic content. Furthermore, we wish to utilize all the information in our sample, including from countries that have never experienced a crisis, which would be precluded if country effects were added. As the probit vs. logit models yield similar results, in the remainder of the analysis we only estimate probits and take as our baseline the model presented in column 4.[18]

To discriminate among competing crisis prediction models we can use a metric based on the Receiver Operating Characteristic (ROC) [62, 80]. The ROC depicts the relationship between true

[18] As a general check of model performance, we run Kolmogorov-Smirnov tests for the distributions of predicted probabilities across crisis/non-crisis years. For each regression, we reject the null that the two samples of predicted probabilities are drawn from the same distribution, which suggests that the models systematically yield different predicted probabilities across the two regimes (See Table A5).

and false positives for a range of probability thresholds. The ROC lies above the 45-degree line if the model generates crisis predictions that are superior to random guessing. According to this criterion, the best model is the one that maximizes the area under the curve (AUROC). Random guessing implies an AUROC of 0.5. Looking the AUROC estimates in Table 6.3, we notice that they are in the 0.6-0.7 range, which is similar to the AUROCs obtained in recent studies of crisis incidence [181, 118].

Next, we add measures of financial interconnectedness to the benchmark model. These are the 1st and 2nd PCs extracted from our rich set of network indicators. In each specification we add the chosen PCs based on *all* the lagged levels of the indicators as well as the one-year lagged growth rates (labelled "growth"). The estimates are presented in Table 6.4. In column 1 we add the PCs for a country's own level of connectedness captured in degree, strength, and clustering. In the richest specification (column 4) we add the PCs for neighbor connectedness as well as the HHI. The coefficients on the degree/strength and clustering PCs are positive and statistically significant, which suggests that an increase in a country's own financial connectedness is associated with a higher probability of crises one year later. By contrast, the coefficient on the neighbor connectedness PC is negative and statistically significant, suggesting that a *decline* in the country's neighbor connectedness increases the probability of a crisis. A decline in neighbor connectedness may be a sign that turbulence is occurring further out in the network and is ultimately transmitted to the country in question through higher-degree connections.

How do the augmented probit models fare in terms of overall ability to predict crises? As seen in Table 6.3 (column 4), the benchmark model has an AUROC of 0.703, which rises to 0.749 for the model augmented with all connectedness measures (Table 6.4, column 4). Compared to the benchmark of a random guessing model which has an AUROC of 0.5, adding the network indicators improves the predictive performance of the model by almost 23 percent. In addition, when we test the null hypothesis that the two AUROCs are equal, we reject it with 99 percent confidence (p-value=0.001). The ROCs corresponding to these two models are shown in Figure 6.5. The figure shows that the augmented-model ROC lies above the fundamentals-only ROC, especially at high false positive rates, which are obtained by setting low probability thresholds to call crises. Such low probability cutoffs would be chosen by policy-makers who prefer a higher positive rate – i.e., to be alerted often even when there is no crisis – over missing crises. This implies that the connectedness-augmented model may be particularly useful to conservative policymakers.

Having established that the augmented probit outperforms that based solely on macroeconomic fundamentals, we next ask if there are nonlinearities in the effects of financial connectedness on the likelihood of systemic banking crises. The financial networks literature argues that connectedness has a non-monotonic effect on network stability. At low levels of connectivity, initial shocks may not have a large impact on the network's stability, but beyond a certain "threshold," such shocks – if they are large enough or sufficiently many – can lead to systemic instability [2, 9]. To examine this question, we estimate regressions that allow for our main connectedness measures to have a different impact on the likelihood of crises at high/low levels of connectedness. We do so by splicing the main variables into above/below median (for own connectedness PC and neighbor connectedness PC) and top/bottom 25th percentile (for own clustering PC). The results, shown in Table 6.5, provide some evidence of non-linearities: across specifications we notice that the positive (negative) effect on crisis probability of own (neighbor) degree and strength is *stronger* in the top range of the PC distributions. The same is the case for clustering, for which we find that increases in clustering for countries that are in the top 25th percentile of the clustering distribution are positively associated with crisis probability, but not for countries in the bottom 25th percentile of the clustering distribution.

For policymaking purposes it may be also be useful to monitor a more parsimonious set of specifications as shown in Table 6.6. These are augmented probit models that include only the 1st PC of our three groups of network measures: degree/strength, clustering, and neighbor degree/strength. To unmask any underlying heterogeneity, we split the sample by country type and also try to isolate the effects of interconnectedness on the likelihood of experiencing a crisis solely after 2007. Table 6.6 estimates suggest that during the full period an increase in clustering and a decrease in neighbor connectedness are associated with a higher probability of crisis in the subsequent year, after controlling for macroeconomic fundamentals (column 1). This full sample result is driven by core countries (column 3) while for periphery countries the coefficients on clustering and neighbor connectedness PCs retain their signs but are less precisely estimated (column 2). This loss of statistical significance is not surprising as periphery countries are necessarily less interconnected in the network due to the absence of within-periphery edges, so network indicators such as clustering and neighbor connectivity are less relevant for them. Instead, for these countries, direct linkages – both in terms of number and intensity – seem to matter for the likelihood of crisis.

To test whether interconnectedness has played a more pronounced role for the most recent wave of crises, which occurred between 2007 and 2010, we replace the dependent variable with a modified variable that takes value 1 only for post-2007 crises. This modified dependent variable allows us to examine whether the recent crises were structurally different than earlier ones. The results, shown in column 4, are telling: for the most recent crises, per capita GDP is *positively* associated with crisis probability, unlike in the full period where poorer countries were more likely to experience financial turmoil. As before, countries with higher net foreign asset positions faced a lower probability of crisis, and countries with a higher degree of RER overvaluation also had a higher probability of crisis. Finally, for the 2007-2010 wave of crises, all connectedness measures matter – higher direct *and* indirect financial linkages increase the likelihood of distress in the banking sector.

6.3.4 In- and Out-of-Sample Performance

We conclude the analysis by examining the performance of our data mining and regression approaches in predicting the most recent wave of systemic banking crises. Nineteen high-income countries and six emerging market countries experienced a crisis during 2007-2010.

We focus on crisis prediction for fourteen advanced economies listed in Table 6.7 for which we have data on all variables. We notice that the classification algorithm correctly predicts in-sample the onset of 5 out of the 14 systemic crises (column 1). The benchmark probit, which exploits only lagged information about the state of the economy, generates relatively low in-sample crisis probabilities (which is not unusual, as crises are relatively rare events), ranging from 2.6 percent for Denmark in 2008 to 10.2 percent for Greece in 2008 (column 3). For all but two countries (Portugal and the US), these probabilities increase when we augment the probit model with variables on financial connectedness. The starkest increase is for the UK, where the predicted probability of a crisis rises from 5.5 to 26.4 percent between the benchmark and augmented model. This result suggests that connectedness measures for most countries have substantial predictive content. We find this plausible given the relative importance of these countries' banking systems in the GBN.

When both the algorithm and probit model are re-run on the subsample of core countries over the 2003-2010 period, when connectedness may arguably have paid a more important role as a conduit for financial stress, both techniques yield better in-sample performance. Now the classification algorithm correctly identifies 11 of 14 onsets of crises (column 2); and the fitted crisis probabilities are larger for both the benchmark and augmented probits (columns 5-6). The augmented probit now consistently outperforms the benchmark model as it generates higher crisis probabilities for all crisis onsets during 2007-2008.

Continuing to focus on the latest wave of crises, we find that the classification algorithm has a remarkable out-of-sample performance for core countries. Table 6.8 reports the precision and recall associated with different crisis prediction windows. When $k = 1$, we look at the performance of the algorithm in predicting the onset of a crisis in the year ahead; when $k = 2$ the window expands to include the following year, etc. We notice that recall is consistently high across subsamples, which suggests that the algorithm misses few crisis onsets (columns 2, 4, and 6). However, precision – the algorithm's ability to not issue false alarms – is relatively high only for the core countries (column 1). This hints at the fact that financial connectedness, not surprisingly, is more useful for crisis prediction in core rather than periphery countries.[19]

One last piece of evidence on the out-of-sample performance of our methods is summarized in Table 6.9, which lists the crises predicted by the algorithm for the 2006-2008 period (columns 1-3); as well as the probit's out-of-sample predicted probabilities for the years 2007 and 2008 (columns 4-7). We find that the algorithm issues crisis alerts for Belgium, Ireland, The Netherlands, Portugal, Spain, and Sweden in 2006 (and even earlier for some of these countries). It issues a maximum number of alerts for the year 2007, but it does so correctly only for the US. In 2008, the algorithm correctly anticipates 6 of the 14 crises in advanced economies.[20]

Looking at the augmented probit's out-of-sample predicted probabilities, we note that they are generally small: only those for the UK and the US are larger than 10 percent (column 5). The augmented probit does not fare very well in predicting 2007 crises, with predicted crisis probabilities for the UK and the US slightly above 2 percent (column 7). For this model to issue

[19] Here we focused on the algorithm's ability to predict all crisis-years, but it also does well in predicting the onset of crises (see Table A2).

[20] The algorithm also issues false alarms during this period, e.g., predicting banking crises in South Africa in 2006, and in Canada in 2007-2008.

a crisis alert in 2008 based on macroeconomic fundamentals, the policymaker would have had to set a probability threshold of around 10 percent (column 4). With this threshold, the augmented probit would have predicted systemic banking crises in 2008 in Ireland, Spain, the UK and the US. Importantly, the out-of-sample probabilities do increase when the probit incorporates data on financial interconnectedness.

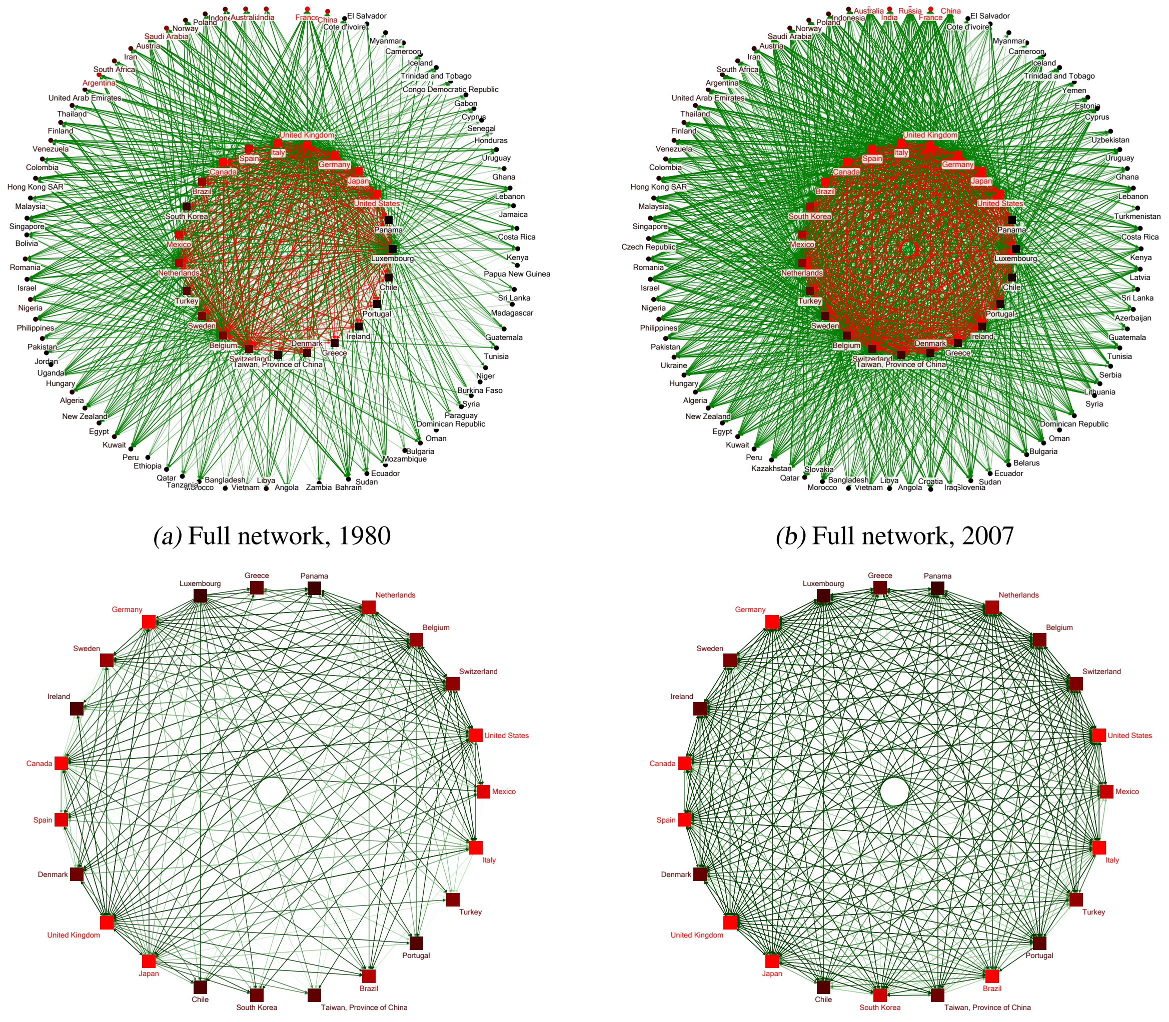
6.4 Conclusions

It is widely believed that extensive financial linkages across countries have played an important role in the severity and spread of the global financial crisis. In thischapter, we examined the performance of financial interconnectedness as an early warning indicator for systemic banking crises. To this end, we used data mining and econometric methods to examine the predictive content of a rich set of network-based connectedness measures. We constructed these measures using data on cross-border banking assets from a large sample of countries over the past four decades.

Our results suggest that financial connectedness can predict crises. Higher levels of a country's own connectedness – captured by simple indicators such as the number and intensity of its financial ties in the global banking network – are associated with a higher probability of crises. Lower connectivity among a country's direct financial partners hints at turbulence in the network and potential contagion, and it too is associated with a higher probability of crisis. We found evidence of nonlinearities in that connectedness influences the probability of crises more at higher levels of connectedness. A classification algorithm that searches for patterns in the data and associates levels and growth rates of network indicators with crisis incidence, has good ability to predict the onset of crises. A probit model of crisis prediction generally yields higher crisis probabilities, both in and out of sample, when it includes information on financial connectivity as opposed to macroeconomic factors alone.

We see these results as a first step towards exploiting the potentially rich informational content of network-based measures of financial connectivity for purposes of crisis prediction. While our findings suggest that there is a useful amount of information in the measures we considered, the list of indicators we used is by no means exhaustive. Future work could expand the scope of the analysis. Our findings could further be probed on financial networks underpinned by different types of cross-country linkages. Such work could result in improved early warning systems and more lead time for designing appropriate policy responses.

Note. The material in Section 6 has been published in [156]. The authors agree that I made a significant contribution and may use this work in my thesis.



(a) Full network, 1980

(b) Full network, 2007

(c) Core network, 1980

(d) Core network, 2007

Fig. 6.1: Global banking network in 1980 vs. 2007: Visualization of the baseline GBN for the largest 100 countries by GDP. Node color is proportional to GDP (black is lower, red is higher). Edge weight is proportional with the intensity of the edge color (from light to dark green). In panels (a)-(b), the nodes on the inner ring are core countries and the nodes on the outer ring are periphery countries; red edges connect core countries and green edges connect core with periphery countries.

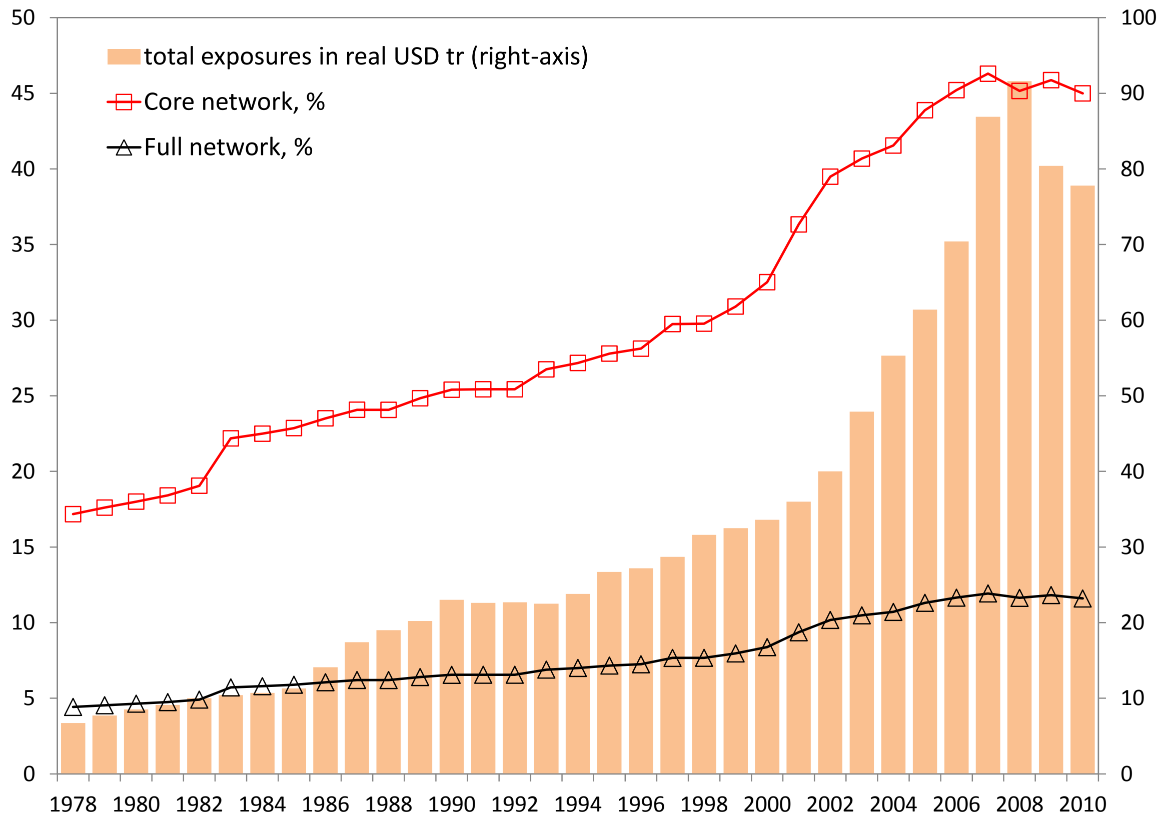


Fig. 6.2: Network density and total exposures: Network density is defined as the number of edges in the GBN divided by the number of possible edges. Total exposures, representing cross-border banking claims, are expressed in constant (2005) USD trillion.

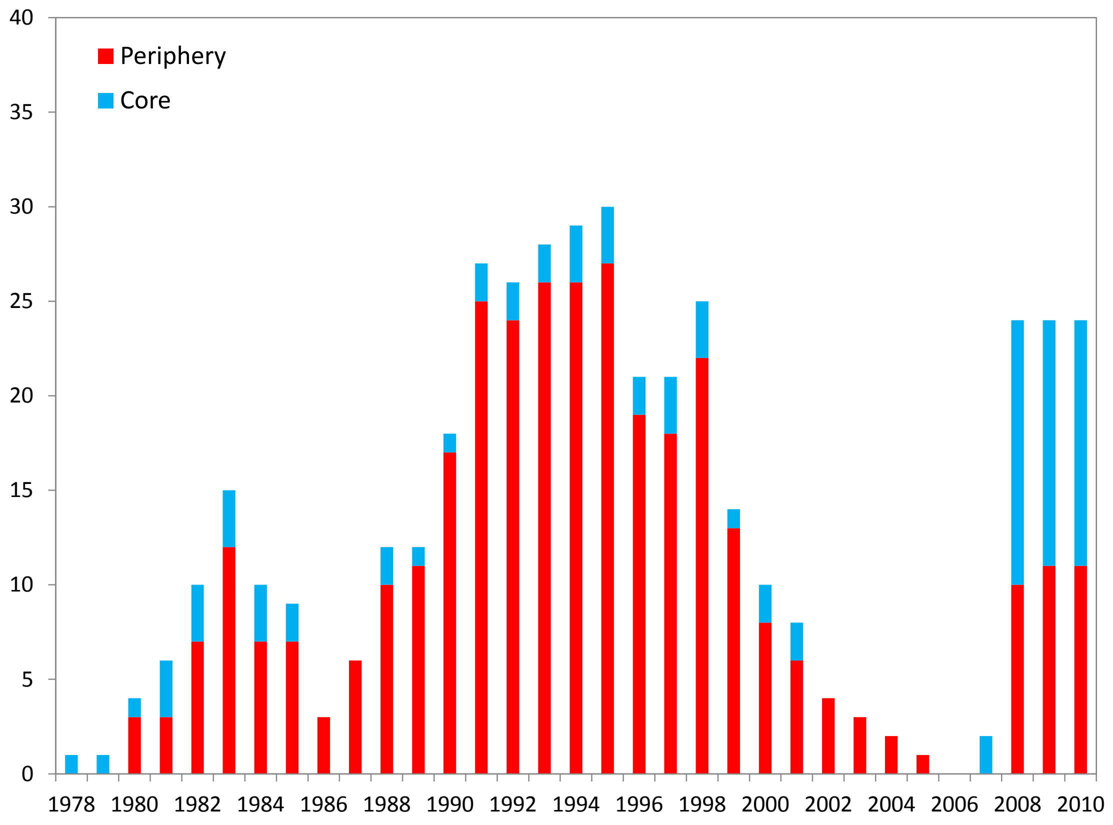
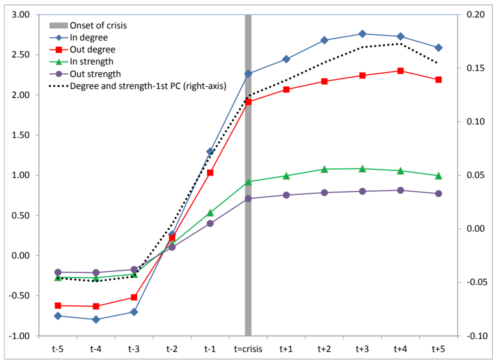
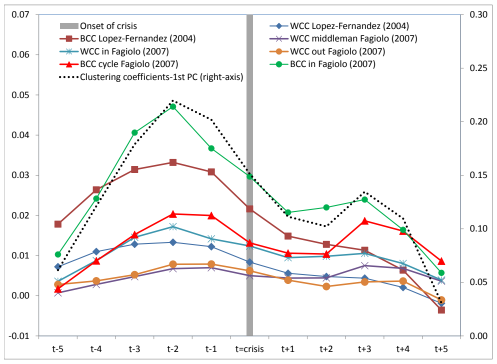


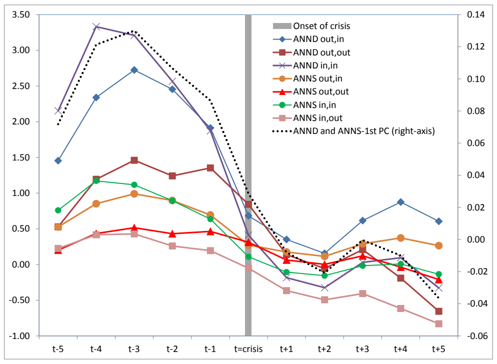
Fig. 6.3: Number of systemic banking crises by country type: Number of crises by year and type of country (periphery vs. core). "Core" countries are BIS-reporting countries. "Periphery" countries are the remaining (non-reporting) countries. Crises dates are from [139].



(a) Own connectedness: Degree and strength



(b) Own connectedness: Clustering



(c) Neighbor connectedness: Degree and strength

Fig. 6.4: Financial connectedness around systemic banking crises: The chart depicts average levels of selected network indicators during 5 years before and after systemic banking crises (t=`crisis`). The indicators are conditional on country and year fixed effects. The dotted line represents the first principal component (labeled "1st PC") extracted from each of the three groups of indicators (see Section 3.1 and Appendix B for details).

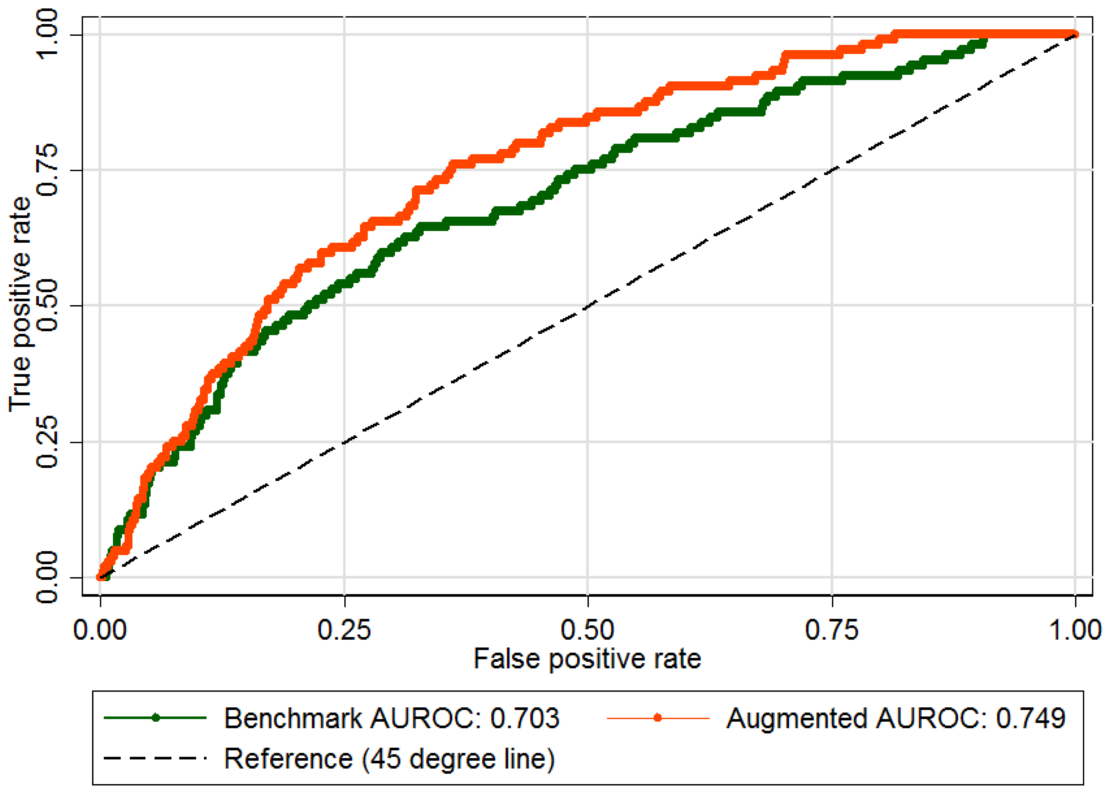


Fig. 6.5: ROCs for benchmark vs. augmented probit model: The chart shows the ROCs corresponding to the benchmark probit model (Table 6.3, column 4) vs. the probit model augmented with principal components extracted from network indicators (Table 6.4, column 4).

	1978	1985	1995	2007	2010	% increase 1978-2007
Degree and strength						
Degree-in/out	9.3	12.2	15.4	23.7	23.5	154%
Strength-in/out	3.3	4.3	5.4	8.3	8.2	152%
Binary clustering coefficients						
BCC Lopez-Fernandez 2004	0.56	0.66	0.85	0.89	0.89	59%
BCC Fagiolo 2007 (“cycle”)	0.49	0.59	0.78	0.89	0.89	84%
BCC Fagiolo 2007 (“middleman”)	0.49	0.59	0.78	0.90	0.89	82%
BCC Fagiolo 2007 (“in”)	0.53	0.62	0.78	0.88	0.88	66%
BCC Fagiolo 2007 (“out”)	0.51	0.61	0.80	0.89	0.89	75%
Weighted clustering coefficients						
WCC Lopez-Fernandez 2004	0.23	0.27	0.36	0.37	0.36	64%
WCC Fagiolo 2007 (“cycle”)	0.18	0.21	0.28	0.32	0.32	85%
WCC Fagiolo 2007 (“middleman”)	0.18	0.22	0.28	0.33	0.32	83%
WCC Fagiolo 2007 (“in”)	0.19	0.23	0.28	0.32	0.31	64%
WCC Fagiolo 2007 (“out”)	0.18	0.22	0.29	0.33	0.32	78%
Average nearest neighbor degree and strength						
ANND (out-in)	54.5	66.5	92.9	117.0	110.6	115%
ANND (out-out)	52.2	66.7	87.3	114.6	110.8	119%
ANND (in-in)	50.9	63.6	106.7	119.1	113.5	134%
ANND (in-out)	49.9	65.5	102.8	124.1	122.4	149%
ANNS (out-in)	19.9	24.6	33.5	42.0	39.5	111%
ANNS (out-out)	19.1	24.7	31.5	41.3	39.8	117%
ANNS (in-in)	18.2	22.8	37.9	43.1	40.4	136%
ANNS (in-out)	17.8	23.6	36.4	45.0	43.8	152%
Herfindahl-Hirschmann index						
HHI	0.11	0.13	0.20	0.12	0.13	12%

Tab. 6.1: Average network indicators over time: The table reports average values (across nodes) for selected network indicators in selected years.

	(1)	(2)	(3)	(4)	(5)	(6)
Sample, period	# crisis-years	# predicted crisis-years	Support	Precision	Recall	# sub-rules
Full, 1978-2010	409	49	46	0.94	0.11	34
Full, 1978-2002	329	342	309	0.90	0.94	481
Full, 2003-2010	80	35	32	0.91	0.40	18
Core, 1978-2010	87	34	30	0.88	0.34	33
Core, 1978-2002	45	59	43	0.73	0.96	76
Core, 2003-2010	42	41	35	0.85	0.83	24
Periphery, 1978-2010	322	310	283	0.91	0.88	335
Periphery, 1978-2002	284	9	9	1.00	0.03	1
Periphery, 2003-2010	38	53	38	0.72	1.00	420

Tab. 6.2: Classification algorithm in-sample performance: The table summarizes the in-sample performance of the classification algorithm. Columns 3-5 report the support (correctly predicted crises), precision (correctly predicted crises/total predicted crises), and recall (correctly predicted crises/total actual crises). Column 6 reports the number of association sub-rules.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	Probit	Probit	Probit	Probit	Logit	Logit	Logit	Logit
Log-per capita GDP	0.009	0.290	-0.006	-0.019	0.027	0.670	0.006	-0.037
	(0.026)	(0.285)	(0.030)	(0.028)	(0.059)	(0.663)	(0.064)	(0.061)
Net foreign assets/GDP	-0.092*	-0.240**	-0.085	-0.092	-0.152	-0.449*	-0.133	-0.148
	(0.055)	(0.109)	(0.054)	(0.056)	(0.118)	(0.244)	(0.115)	(0.117)
Capital inflows bonanza	0.379***	0.356***	0.353***	0.390***	0.845***	0.744***	0.730***	0.876***
	(0.095)	(0.116)	(0.103)	(0.096)	(0.209)	(0.236)	(0.224)	(0.211)
Forex reserves/GDP	-0.018***	-0.030***	-0.019***	-0.019***	-0.044***	-0.067***	-0.046***	-0.045***
	(0.005)	(0.011)	(0.005)	(0.005)	(0.012)	(0.023)	(0.014)	(0.012)
RER misalignment	1.100**	1.185*	0.994**	1.122**	2.010*	2.391	1.597	2.082**
	(0.505)	(0.653)	(0.505)	(0.504)	(1.028)	(2.061)	(0.986)	(1.015)
At least 1 neighbor in crisis				0.209**				0.489***
				(0.082)				(0.186)
Country fixed effects		yes				yes		
Year fixed effects			yes				yes	
Observations	3,949	2,319	3,139	3,949	3,949	2,319	3,139	3,949
AUROC	0.696	0.729	0.769	0.703	0.692	0.720	0.763	0.699

Tab. 6.3: Benchmark probit/logit model: The table reports the estimates of a benchmark crisis prediction model estimated on the full sample of countries over 1978-2010. The dependent variable takes value 1 for the onset of systemic banking crises. A constant is included, but the coefficient is not shown. Standard errors are clustered at the country level. *** indicates statistical significance at 1 percent, ** at 5 percent, and * at 10 percent.

	(1)	(2)	(3)	(4)
Degree and strength-1st PC	0.200***	0.176***	0.177***	0.189***
	(0.053)	(0.049)	(0.048)	(0.054)
Degree and strength, growth-1st PC	-0.003	0.002	-0.016	0.021
	(0.038)	(0.038)	(0.044)	(0.047)
Degree and strength, growth-2nd PC	-0.001	0.001	0.009	0.059
	(0.040)	(0.042)	(0.044)	(0.062)
Clustering-1st PC	0.231***	0.333***	0.348***	0.372***
	(0.074)	(0.074)	(0.074)	(0.084)
Clustering-2nd PC	0.790	0.947*	0.914*	0.935*
	(0.483)	(0.489)	(0.494)	(0.487)
Clustering, growth-1st PC	-0.003	-0.003	0.010	0.017
	(0.009)	(0.009)	(0.018)	(0.018)
Clustering, growth-2nd PC	-0.027	-0.033	-0.031	-0.002
	(0.051)	(0.053)	(0.059)	(0.059)
Neighbor connectedness-1st PC		-0.189***	-0.205***	-0.218***
		(0.049)	(0.049)	(0.051)
Neighbor connectedness, growth -1st PC			-0.048	-0.062
			(0.041)	(0.040)
Neighbor connectedness, growth -2nd PC			-0.013	-0.030
			(0.044)	(0.044)
Herfindahlindex				0.216
				(0.575)
Herfindahlindex, growth				0.004
				(0.003)
Observations	3,368	3,368	3,368	3,368
AUROC	0.723	0.742	0.744	0.749

Tab. 6.4: Augmented probit results: The table reports the estimates of the benchmark probit model augmented with principal components extracted from network indicators of financial connectivity. The models are estimated on the full sample of countries over 1978-2010. The dependent variable takes value 1 for the onset of a systemic banking crisis. All macroeconomic variables from Table 6.3 (column 4) and a constant are included, but the coefficients are as expected and not shown. Standard errors are clustered at the country level. *** indicates statistical significance at 1 percent, ** at 5 percent, and * at 10 percent.

	(1)	(2)	(3)
Degree and strength-1st PC * Above median	0.318***	0.378***	0.180**
	(0.088)	(0.104)	(0.075)
Degree and strength-1st PC * Below median	0.014	0.185**	0.188
	(0.015)	(0.075)	(0.125)
Degree and strength, growth-1st PC	-0.022	0.002	-0.016
	(0.032)	(0.039)	(0.044)
Degree and strength, growth-2nd PC	-0.027	0.002	0.009
	(0.037)	(0.043)	(0.044)
Clustering-1st PC * Above p75		0.158**	0.202*
		(0.079)	(0.105)
Clustering-1st PC * Between p25 and p75		0.001	0.017
		(0.053)	(0.054)
Clustering-1st PC * Below p25		0.066	-0.085
		(0.140)	(0.159)
Clustering-2nd PC		0.892*	0.828*
		(0.494)	(0.467)
Clustering, growth-1st PC		-0.009	0.003
		(0.011)	(0.017)
Clustering, growth-2nd PC		-0.037	-0.031
		(0.055)	(0.058)
Neighbor connectedness-1st PC * Above median			-0.171***
			(0.062)
Neighbor connectedness-1st PC * Below median			0.126
			(0.141)
Neighbor connectedness, growth-1st PC			-0.041
			(0.040)
Neighbor connectedness, growth-2nd PC			-0.015
			(0.043)
Observations	3,524	3,368	3,368
AUROC	0.720	0.729	0.751

Tab. 6.5: Augmented probit results (non-linearities): The table reports estimates of the augmented probit models presented in Table 6.4 in which we splice each of the 1st PC of the three groups of network indicators into two or three variables to detect nonlinearities. Degree/strength and neighbor degree/strength 1st PCs are spliced above and below the median; the clustering coefficient 1st PC is spliced into three variables: top 75th percentile, bottom 25th percentile, and middle portion. The dependent variable takes value 1 for the onset of a systemic banking crisis. The models are estimated over 1978-2010 on the full sample. All macroeconomic variables, the HHI, and a constant are included, but the coefficients are as expected and not shown. Standard errors are clustered at the country level. *** indicates statistical significance at 1 percent, ** at 5 percent, and * at 10 percent.

	(1)	(2)	(3)	(4)
	Full	Periphery	Core	Core, Alt DV
Log-per capita GDP	-0.069**	-0.138***	0.526	5.174***
	(0.034)	(0.041)	(0.339)	(1.926)
Net foreign assets/GDP	-0.095*	-0.105*	-0.776**	-2.492**
	(0.058)	(0.063)	(0.350)	(1.119)
Capital inflows bonanza	0.382***	0.326***	0.584**	0.474
	(0.097)	(0.108)	(0.244)	(0.581)
Forex reserves/GDP	-0.014***	-0.010**	-0.073**	-0.027
	(0.005)	(0.004)	(0.035)	(0.048)
RER misalignment	1.087**	0.949*	6.532***	5.421**
	(0.490)	(0.485)	(2.128)	(2.714)
At least 1 neighbor in crisis	0.202**	0.245**	-0.381	0.652
	(0.091)	(0.097)	(0.271)	(0.720)
Degree and strength-1st PC	0.028	0.307**	0.037	0.333**
	(0.018)	(0.131)	(0.040)	(0.132)
Clustering-1st PC	0.128**	0.034	1.027***	4.513***
	(0.055)	(0.106)	(0.244)	(1.101)
Neighbor connectedness-1st PC	-0.134**	-0.095	-1.520***	-7.683***
	(0.053)	(0.076)	(0.391)	(2.018)
Observations	3,949	3,409	540	540
AUROC	0.722	0.738	0.831	0.983

Tab. 6.6: Augmented probit results (parsimonious version): The table reports estimates of a more parsimonious version of the augmented probit model presented in Table 6.4 in which we retain only the 1st PC of each group of network indicators. In columns 1-3 the dependent variable takes value 1 for the onset of a systemic banking crisis. The models are estimated over 1978-2010 on the full sample (column 1), periphery countries (column 2) and core countries (column 3). In column 4 the dependent variable is modified to take value 1 only for the 2007-2010 crises. A constant is included, but the coefficient is not shown. Standard errors are clustered at the country level. *** indicates statistical significance at 1 percent, ** at 5 percent, and * at 10 percent.

	(1)	(2)	(3)	(4)	(5)	(6)
	Algorithm		Probit			
	Full	Core	Full		Core	
			Benchmark	Augmented	Benchmark	Augmented
Belgium	2008	2008	3.5%	11.5%	3.0%	6.4%
Denmark	2008	2008	2.6%	3.9%	1.4%	2.4%
Germany		2008	3.7%	15.3%	3.6%	7.9%
Greece		2008	10.2%	12.5%	23.1%	26.7%
Ireland			9.3%	10.7%	19.9%	36.2%
Italy		2008	4.1%	5.8%	6.0%	12.1%
Luxembourg	2008	2008	.	.	.	.
Netherlands		2008	.	.	.	.
Portugal		2008	5.0%	2.9%	10.8%	16.8%
Spain		2008	10.1%	17.3%	23.2%	48.5%
Sweden			3.5%	3.7%	4.4%	6.2%
Switzerland	2008	2008	.	.	.	.
United Kingdom	2007	2007	5.5%	26.4%	19.6%	44.1%
United States			5.7%	2.4%	24.4%	26.1%

Tab. 6.7: Classification algorithm and probit model in-sample performance: The table summarizes the in-sample performance of the classification algorithm and probit models, focusing on the 2007-2008 crises. The algorithm and probits are run, respectively, on the full sample (1978-2010) and core country subsample (2003-2010). Columns 1-2 report the year in which the algorithm predicts a crisis. Columns 3-6 report predicted crisis probabilities from the benchmark and augmented probits; missing values refer to countries for which data on at least one macroeconomic variable is missing.

	(1)	(2)	(3)	(4)	(5)	(6)
	Core		Full		Periphery	
k-year ahead prediction	Precision	Recall	Precision	Recall	Precision	Recall
k=0	0.12	0.50	0.02	0.15	0.01	0.17
k=1	0.26	0.64	0.03	0.23	0.02	0.25
k=2	0.37	0.71	0.06	0.42	0.03	0.33
k=3	0.44	0.71	0.09	0.46	0.05	0.42
k=4	0.49	0.71	0.10	0.50	0.07	0.58
k=5	0.53	0.71	0.13	0.58	0.08	0.58

Tab. 6.8: Classification algorithm out-of-sample performance: The table summarizes the out-of-sample performance of the classification algorithm in predicting the onset of crises for the full sample, core, and periphery countries. The algorithm is run on the 2003-2010 period. The measures of performance are precision (correctly predicted crises/total predicted crises) and recall (correctly predicted crises/total actual crises).

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
	Algorithm			Probit			
				Benchmark	Augmented	Benchmark	Augmented
	2008	2007	2006	2008		2007	
Belgium	xx	x	x	0.7%	1.1%	0.1%	0.3%
Denmark	xx	x		0.3%	0.4%	0.1%	0.1%
Germany		x		0.8%	1.4%	0.1%	0.3%
Greece				4.9%	4.2%	3.0%	1.9%
Ireland	xx	x	x	5.8%	10.1%	0.1%	0.4%
Italy				0.9%	1.5%	0.2%	0.4%
Luxembourg				.	.	.	.
Netherlands		x	x	.	.	.	.
Portugal			x	1.3%	1.6%	0.3%	0.7%
Spain	xx	x	x	5.6%	12.1%	1.3%	2.0%
Sweden	xx	x	x	0.9%	1.1%	0.1%	0.2%
Switzerland	xx	x		.	.	.	.
United Kingdom				10.9%	20.0%	0.9%	2.3%
United States		xx		12.9%	12.0%	1.4%	2.4%

Tab. 6.9: Classification algorithm and probit model out-of-sample performance: The table summarizes the *out-of-sample* performance of the classification algorithm and probit models, focusing on the 2007-2008 crises. In columns 1-3, "x" marks a predicted crisis in the year given by the column heading. "xx" marks crisis predictions that are accurate. Columns 4-7 report crisis probabilities predicted for the years 2007 and 2008 by the benchmark and augmented probits. Both the algorithm and the probits are run on the core country subsample on a rolling basis, i.e., over 1978-2006 for 2007 prediction, and over 1978-2007 for 2008 prediction.

Fig. 6.6: Classification algorithm - Examples of rules. The table reports examples of sub-rules generated by the classification algorithm when run on the subsample of core countries over 2003- 2010

Network indicator	Type of variable	Lag structure	Lower Bound	Upper Bound
ANND (in-in)	growth rate	t-3	0.01589	0.02977
ANND (in-out)	growth rate	t-5	0.04853	0.06859
ANND (out-in)	growth rate	t-1	0.00744	0.01215
ANND (out-out)	growth rate	t-2	0.01424	0.01991
ANNS (in-out)	growth rate	t-5	0.09538	0.10602
BCC Fagiolo 2007 ("cycle")	level	t-3	0.18191	0.19096
BCC Lopez-Fernandez 2004	growth rate	t-5	-0.28473	-0.22316
HHI	level	t-5	0.00583	0.00607
HHI	level	t-4	0.00574	0.00597
HHI	level	t-1	0.00570	0.00581
HHI	level	t-2	0.00571	0.00585
In-degree	growth rate	t-5	0.02041	0.03529
In-degree	level	t-5	167.00	174.00
In-degree	level	t-4	170.00	177.00
In-strength	level	t-5	63.51	66.46
In-strength	level	t-4	64.05	68.03
In-strength	level	t-3	65.10	68.58
In-strength	level	t-2	66.33690	68.57583
AN-in	growth rate	t-3	-0.00229	0.00408
AN-in	level	t-5	0.91924	0.96267
Out-degree	level	t-3	154.00	171.00
WCC Fagiolo 2007 ("cycle")	level	t-3	0.06724	0.06957
WCC Fagiolo 2007 ("middleman")	level	t-3	0.06822	0.07180
WCC Fagiolo 2007 ("out")	level	t-3	0.07107	0.07164

Appendix Ap

Ap.1 Diffusion Centrality

Ap.1.1 Diffusion Models

Below we provide details on the diffusion models used in the experimental evaluation.

The Flickr model consists of the following rule

$$p(v) : \mu_{v',v} \times \mu_p \times \mu_q \times dp_F \leftarrow e(v',v) : \mu_{v',v} \wedge p(v') : \mu_p \wedge q(v') : \mu_q$$

saying that if vertex v' has properties q and p then it can diffuse property p to its neighbor v. The value dp_F is a constant representing the probability that the vertex v will receive property p.

The Jackson-Yariv model is a diffusion model stating that a vertex will receive (adopt) a property p according to the cumulative effect of its neighbors and the ratio of the benefit to the cost of the vertex for adopting p. Suppose that v_i is an agent having a default behavior that can be changed in the new behavior p, and that v_i has specific cost c_i and benefit b_i for adopting the behavior p. Then, the Jackson-Yariv model can be expressed by the rule

$$p(v_i) : \tfrac{b_i}{c_i} \times r(\textstyle\sum_j E_j) \times \frac{\sum_j w_j}{\sum_j E_j} \times w_q \times dp_{_{JY}} \leftarrow$$
$$\textstyle\bigwedge_{v_j|(v_j,v_i)\in E} \big(e(v_j,v_i) : E_j \wedge p(v_j) : w_j\big) \wedge q(v_i) : w_q$$

where

1. $r(\sum_j E_j)$ is a function describing how the number of neighbors of v_i affects the benefits to v_i for adopting behavior p,

2. $\frac{\sum_j w_j}{\sum_j E_j}$ is the fraction of the neighbors of v_i having property p, and

3. $dp_{_{JY}}$ is a constant representing the probability that the vertex v will adopt the property p.

In our experiments we set the function $r(\sum_j E_j)$ to be a logarithmic function normalized by the logarithm of the maximum in-degree d_{max}^{in} of the network, and having values within the interval $[0.1, 2]$, i.e., $r(\sum_j E_j) = 1.9 \times \frac{\ln(\sum_j E_j)}{\ln(d_{max}^{in})} + 0.1$. When the annotation μ of an atom $p(v)$, with $v \in V$, becomes greater than 1, then we set $\mu = 1$. Moreover, observe that the vertex v_i can adopt property p only if it also has property q.

For the STEAM data, we set $\frac{b_i}{c_i} = 1$ for all vertices, while for the Non-Game Social Network Data we randomly assigned $\frac{b_i}{c_i}$ to the vertices according to a normal distribution with $0.5 \le \frac{b_i}{c_i} \le 1.5$.

The SIR model is a classic disease model which labels each vertex with *susceptible* if it has not had the disease but can receive it from one of its neighbors, *infectious* if it has caught the disease and t units of time have not expired, and *recovered* when the vertex can no longer catch or transmit the disease. The diffusion rules are following

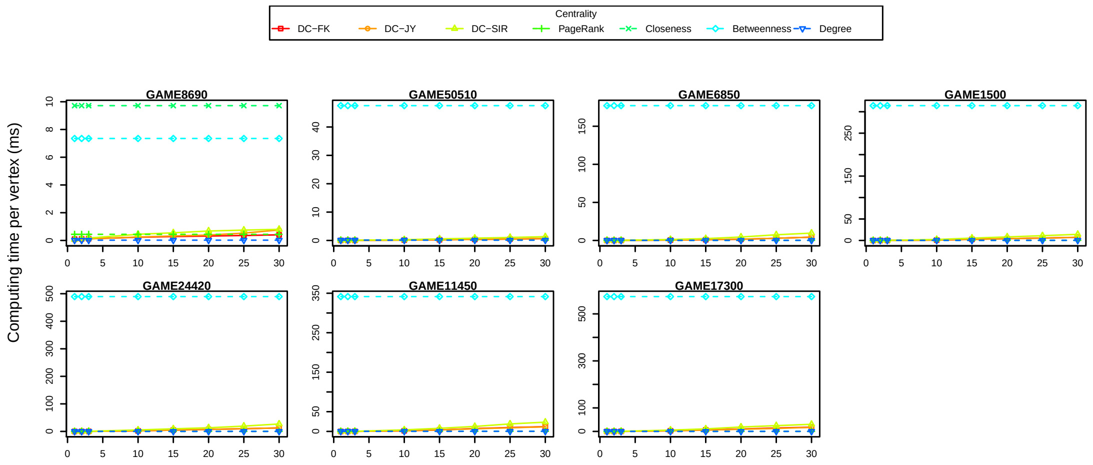
$$\begin{aligned} p(v) :&(1 - R) \times \mu^e_{v',v} \times \mu^p_{v'} \times (1 - R') \times \mu^q_v \times dp_{_{SIR}} \leftarrow \\ &\quad r_t(v) : R \wedge e(v', v) : \mu^e_{v',v} \wedge p(v') : \mu^p_{v'} \wedge r_t(v') : R' \wedge q(v) : \mu^q_v \end{aligned}$$

$$\begin{aligned} r_i(v) :&\mu^r_v \leftarrow r_{i-1}(v) : \mu^r_v, \; i \in [2, t] \\ r_1(v) :&\mu^p_v \leftarrow p(v) : \mu^p_v \end{aligned}$$

Here, only the vertices having property q can be susceptible and the diffusion property p represents that a vertex is infected. Properties r_i $(i \neq t)$ express that the vertex is in the infectious state at time $t - 1$ and r_t means that the vertex is recovered. In our experiments, we set $t = 2$. The constant $dp_{_{SIR}}$ is the probability that the vertex v will be infected.

Ap.1.2 STEAM Data Runtime

(a) STEAM Data: Runtime of HyperDC algorithm for Exact Diffusion Centrality computation with Flickr, Jackson-Yariv, and SIR diffusion models compared with Classical Centrality Measures Averaged Per Vertex when $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$.



(b) STEAM Data: Runtime of HyperDC algorithm for Exact Diffusion Centrality computation with Flickr, Jackson-Yariv, and SIR diffusion models compared with Classical Centrality Measures Averaged Per Vertex when $\delta_q \in \{1\%, 2\%, 3\%, 4\%, 5\%\}$. Detailed view of part of part (a).

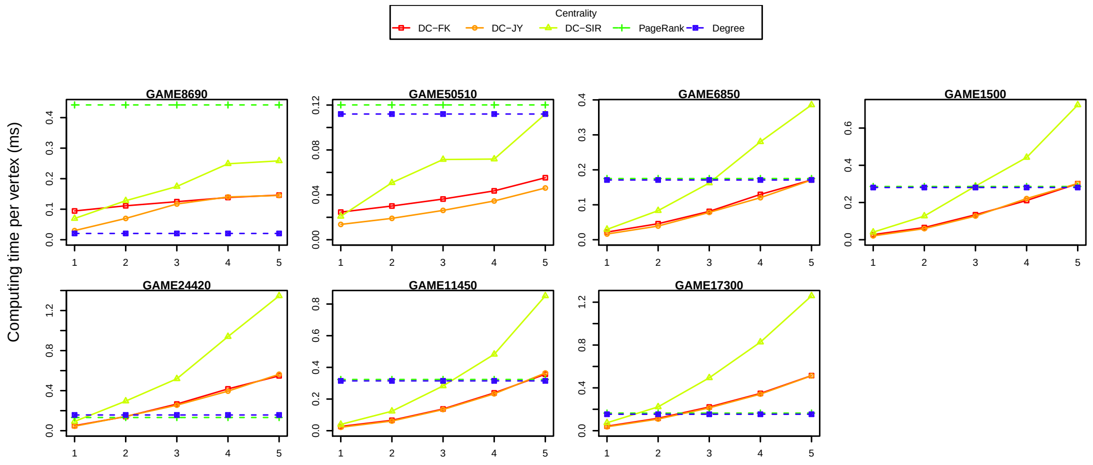


Fig. Ap.1: STEAM Data Runtime

Fig. Ap.2: STEAM Data: p-values of T-test for runtimes of each centrality measure and HyperDC algorithm. When p-value is less than or equal to 0.5, the cell color is green.

T-Test, **Betweenness** and Diffusion centrality runtimes in STEAM networks										
Model	p-value when % of vertices having condition property q is									
	1%	2%	3%	4%	5%	10%	15%	20%	25%	30%
Flickr	0.014	0.014	0.014	0.015	0.015	0.015	0.015	0.016	0.016	0.017
Jackson-Yariv	0.014	0.015	0.016	0.027	0.033	0.043	0.047	0.058	0.065	0.069
SIR	0.014	0.014	0.015	0.015	0.015	0.015	0.016	0.017	0.018	0.019

T-Test, **PageRank** and Diffusion centrality runtimes in STEAM networks										
Model	p-value when % of vertices having condition property q is									
	1%	2%	3%	4%	5%	10%	15%	20%	25%	30%
Flickr	0.004	0.082	0.890	0.333	0.121	0.030	0.021	0.019	0.019	0.017
Jackson-Yariv	0.775	0.328	0.329	0.337	0.331	0.320	0.302	0.288	0.250	0.226
SIR	0.021	0.923	0.129	0.041	0.022	0.013	0.011	0.014	0.012	0.012

T-Test, **Degree** and Diffusion centrality runtimes in STEAM networks										
Model	p-value when % of vertices having condition property q is									
	1%	2%	3%	4%	5%	10%	15%	20%	25%	30%
Flickr	0.010	0.140	0.977	0.284	0.105	0.029	0.021	0.019	0.019	0.017
Jackson-Yariv	0.857	0.319	0.328	0.337	0.331	0.320	0.302	0.288	0.250	0.226
SIR	0.044	0.802	0.112	0.037	0.020	0.013	0.011	0.014	0.012	0.012

Ap.1.3 Spread for Non-Game Social Network Data

Tab. Ap.1: Non-Game Social Network Data, Case 1: Average ratio of the $spread$ generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different δ_q.

		Average ratio of the DC spread to the best spread of other centrality measures for different δ_q.									
Model	Network	1%	2%	3%	4%	5%	10%	15%	20%	25%	30%
FlickrModel	Douban-dataset	56.2	14.3	31.6	18.0	8.1	4.9	3.7	2.7	2.1	1.9
	BlogCatalog-dataset	3.9	2.9	2.6	1.6	1.8	2.0	1.2	1.2	1.2	1.1
	email-Enron	5.3	3.0	2.6	3.5	3.0	1.8	1.7	1.5	1.5	1.4
	email-EuAll	240.1	6.1	4.4	4.3	6.3	3.3	3.1	2.6	2.4	2.2
	soc-Epinions	36.7	11.4	6.5	4.2	3.8	3.9	2.0	2.0	1.9	1.7
	wiki-Vote	9.2	13.5	2.0	4.1	2.7	1.7	1.5	1.3	1.2	1.3
	Average	58.6	8.5	8.3	5.9	4.3	2.9	2.2	1.9	1.7	1.6
JYModel	Douban-dataset	4.8	3.8	2.9	2.6	2.3	1.8	1.5	1.3	1.1	1.1
	BlogCatalog-dataset	1.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	1.3	1.2	1.1	1.1	1.1	1.1	1.1	1.1	1.0	1.0
	email-EuAll	1.8	1.7	1.6	1.5	1.5	1.5	1.4	1.4	1.4	1.4
	soc-Epinions	1.7	1.5	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3
	wiki-Vote	1.7	1.4	1.3	1.3	1.2	1.2	1.2	1.2	1.2	1.1
	Average	2.0	1.8	1.6	1.5	1.4	1.3	1.2	1.2	1.2	1.2
SIRModel	Douban-dataset	2.3	1.7	1.4	1.4	1.3	1.1	1.0	1.0	1.0	1.0
	BlogCatalog-dataset	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-EuAll	1.6	1.6	1.5	1.4	1.4	1.4	1.3	1.3	1.3	1.3
	soc-Epinions	1.3	1.2	1.2	1.2	1.2	1.1	1.1	1.1	1.2	1.1
	wiki-Vote	1.0	1.0	1.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Average	1.4	1.2	1.2	1.2	1.1	1.1	1.1	1.1	1.1	1.1

Tab. Ap.2: Non-Game Social Network Data, Case 1: Average ratio of the $spread$ generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different k.

		Average ratio of the DC spread to the best spread of other centrality measures for different k.									
Model	Network	10	20	30	40	50	60	70	80	90	100
FlickrModel	Douban-dataset	20.8	25.4	17.2	17.6	14.7	10.4	9.2	9.4	9.4	9.4
	BlogCatalog-dataset	3.2	2.4	2.0	1.9	1.8	1.7	1.6	1.6	1.6	1.6
	email-Enron	3.5	3.1	2.8	2.7	2.6	2.2	2.2	2.1	2.0	2.0
	email-EuAll	73.0	82.9	86.9	6.0	5.7	5.4	3.9	3.8	3.8	3.3
	soc-Epinions	11.7	10.8	9.0	8.5	8.6	5.8	5.5	4.8	4.8	4.7
	wiki-Vote	5.2	5.9	4.7	4.8	4.6	3.8	2.5	2.5	2.4	2.3
	Average	19.6	21.7	20.4	6.9	6.3	4.9	4.2	4.0	4.0	3.9
JYModel	Douban-dataset	2.5	2.4	2.3	2.3	2.3	2.3	2.3	2.3	2.2	2.2
	BlogCatalog-dataset	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
	email-EuAll	2.0	1.8	1.6	1.5	1.4	1.4	1.4	1.4	1.4	1.3
	soc-Epinions	1.6	1.4	1.4	1.4	1.4	1.4	1.3	1.3	1.3	1.3
	wiki-Vote	1.1	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
	Average	1.6	1.5	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
SIRModel	Douban-dataset	1.2	1.2	1.3	1.3	1.3	1.3	1.4	1.4	1.4	1.4
	BlogCatalog-dataset	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.9
	email-EuAll	1.8	1.7	1.5	1.4	1.3	1.3	1.3	1.3	1.2	1.2
	soc-Epinions	1.4	1.3	1.2	1.2	1.2	1.1	1.1	1.1	1.1	1.1
	wiki-Vote	1.0	1.1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Average	1.2	1.2	1.2	1.1	1.1	1.1	1.1	1.1	1.1	1.1

Tab. Ap.3: Non-Game Social Network Data, Case 2: Average ratio of the $spread$ generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different δ_p values.

		Average ratio of the DC spread to the best spread of other centrality measures for different δ_p.					
Model	Network	0%	0.1%	0.2%	0.3%	0.4%	0.5%
FlickrModel	Douban-dataset	124.0	145.3	123.9	127.6	129.7	14.7
	BlogCatalog-dataset	383.3	392.1	6.7	5.0	2.0	7.6
	email-Enron	198.7	245.3	224.5	226.1	3.0	2.2
	email-EuAll	181.1	236.1	194.1	205.1	251.8	7.2
	soc-Epinions	266.1	227.3	222.3	232.4	208.0	8.5
	wiki-Vote	106.9	119.5	112.6	106.0	5.3	8.3
	Average	210.0	227.6	147.3	150.4	100.0	8.1
JYModel	Douban-dataset	4.4	5.1	4.7	5.2	5.4	2.7
	BlogCatalog-dataset	3.4	3.4	3.4	1.9	1.1	1.0
	email-Enron	5.4	5.6	5.7	5.8	1.9	1.1
	email-EuAll	3.5	3.9	4.4	4.1	3.7	1.6
	soc-Epinions	3.3	3.2	3.2	3.2	3.4	1.4
	wiki-Vote	1.7	1.8	1.7	1.8	1.7	1.3
	Average	3.6	3.8	3.8	3.7	2.9	1.5
SIRModel	Douban-dataset	9.4	21.0	10.2	12.6	14.6	1.6
	BlogCatalog-dataset	26.5	26.7	24.4	24.5	1.5	1.0
	email-Enron	26.5	26.5	28.6	29.8	27.2	1.0
	email-EuAll	22.5	31.0	38.8	34.0	25.5	1.5
	soc-Epinions	25.3	23.8	24.0	24.2	25.0	1.3
	wiki-Vote	6.3	6.2	6.4	6.4	6.8	1.0
	Average	19.4	22.5	22.1	21.9	16.8	1.2

Tab. Ap.4: Non-Game Social Network Data, Case 2: Average ratio of the $spread$ generated by diffusion centrality to the best spread generated by any of the classical centrality measures for different k values.

		Average ratio of the DC spread to the best spread of other centrality measures for different k.									
Model	Network	10	20	30	40	50	60	70	80	90	100
FlickrModel	Douban-dataset	963.2	23.8	20.9	16.9	16.8	16.6	13.4	12.3	12.3	12.1
	BlogCatalog-dataset	1298.9	8.9	4.2	3.3	2.6	2.2	2.1	2.0	1.8	1.7
	email-Enron	1466.8	6.6	4.2	3.9	3.7	3.2	3.0	2.9	2.7	2.7
	email-EuAll	1743.6	9.3	8.8	5.9	5.4	4.5	4.2	3.8	3.5	3.2
	soc-Epinions	1895.9	8.7	6.2	5.3	5.0	4.3	4.0	4.0	3.9	3.8
	wiki-Vote	724.8	9.9	6.7	4.8	3.9	3.3	3.0	2.8	2.7	2.5
	Average	1348.9	11.2	8.5	6.7	6.2	5.7	4.9	4.6	4.5	4.3
JYModel	Douban-dataset	18.9	3.2	3.1	3.1	3.0	2.9	2.9	2.9	2.8	2.8
	BlogCatalog-dataset	14.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	32.5	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
	email-EuAll	21.2	2.1	1.9	1.7	1.6	1.5	1.4	1.4	1.4	1.4
	soc-Epinions	16.8	1.6	1.5	1.4	1.4	1.4	1.4	1.3	1.4	1.4
	wiki-Vote	4.9	1.1	1.3	1.4	1.3	1.3	1.3	1.3	1.4	1.3
	Average	18.1	1.7	1.6	1.6	1.6	1.6	1.5	1.5	1.5	1.5
SIRModel	Douban-dataset	101.7	1.4	1.4	1.5	1.5	1.6	1.6	1.7	1.7	1.7
	BlogCatalog-dataset	165.4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-Enron	223.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	email-EuAll	242.4	1.9	1.8	1.5	1.5	1.4	1.3	1.3	1.3	1.3
	soc-Epinions	195.3	1.4	1.3	1.2	1.2	1.2	1.2	1.1	1.1	1.1
	wiki-Vote	45.9	1.0	1.1	1.1	1.0	1.0	1.0	1.0	1.0	1.0
	Average	162.4	1.3	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2

Ap.1.4 Table of Notations

Notations for social networks

Symbol	Explanation	Section
VP	Set of vertex predicate symbols, also called properties	2.1
EP	Set of edge predicate symbols	2.1
$\mathcal{S}$	Social Network (SN), which is a tuple $(V, E, \mathsf{VL}, \omega)$	2.1
V	Set of vertices	2.1
E	Set of (labeled) edges; $E \subseteq V \times V \times \mathsf{EP}$	2.1
$\mathsf{VL}(v)$	Set of properties of vertex v; $\mathsf{VL}(v) \subseteq \mathsf{VP}$	2.1
$\omega(e)$	Weight of edge e	2.1

Notations for generalized annotated programs

Symbol	Explanation	Section
$A : \mu$	Annotated atom, where A is an atom and μ is an annotation term	2.1
Π	Generalized annotated program (GAP)	2.1
$\Pi_{\mathcal{S}}$	GAP representing SN $\mathcal{S}$	2.1
$grd(r)$	Set of ground instances of rule r	2.1
$grd(\Pi)$	Set of all ground instances of all rules in GAP Π	2.1
$\mathbf{T}_\Pi$	Operator mapping an interpretation of GAP Π to another interpretation of Π	2.1
$\mathsf{lfp}(\Pi)$	Least fixed point of $\mathbf{T}_\Pi$	2.1

Notations for diffusion centrality

Symbol	Explanation	Section
$\mathsf{dc}(v)$	Diffusion centrality of vertex v	2.1
p	Diffusive property	2.1
$\mathcal{S} \oplus p(v)$	Insertion of $p(v)$ into SN $\mathcal{S}$	2.1
$\mathcal{S} \ominus p(v)$	Removal of $p(v)$ from SN $\mathcal{S}$	2.1
DCP	Problem of finding the DC of all vertices	2.1
kDCP	Problem of finding a set of k vertices having the highest DC	2.1
HyperLFP	Algorithm to compute the least fixed point of a GAP	2.1
HyperDC	Algorithm to solve DCP	2.1
CBAF	Algorithm to approximately solve kDCP	3.2

Notations for optimizations

Symbol	Explanation	Section
$dep(\Pi)$	Dependency graph of GAP Π	2.1
$R_{\Pi,p}$	Set of predicate symbols that can reach property p in $dep(\Pi)$	2.1
$M_{\Pi,p}$	Set of predicate symbols that are mutually recursive with property p in $dep(\Pi)$	2.1
$I_{\Pi,p}$	p-interfered predicate set	2.1
Π_p	$\{r \in \Pi \mid$ the head predicate symbol of r belongs to $R_{\Pi,p}\}$	2.1
Π_p^*	$\{r \in \Pi_p \mid$ every predicate symbol q in the body of r is s.t. $q \notin I_{\Pi,p}\}$	2.1

Notations for the HyperLFP and HyperDC algorithms

Symbol	Explanation	Section
h	Hyperedge, i.e., a pair $\langle S, t\rangle$ where S is the source set of vertices and t is the target vertex	2.1
H	Set of hyperedges	2.1
$S(h)$	Source set of hyperedge h	2.1
$t(h)$	Target vertex of hyperedge h	2.1
$\mathcal{H}(\mathcal{S}, \Pi, p)$	Diffusion hypergraph for SN $\mathcal{S}$, GAP Π, and property p	2.1

Notations for the CBAF algorithm

Symbol	Explanation	Section
$\mathcal{S}_C$	Coarsened network	3.2
T_C	Top-k vertices in $\mathcal{S}_C$ having highest DC	3.2
SIM	Vertex similarity function in network coarsening	3.2
$mergeVP$	Vertex properties merging function in network coarsening	3.2
$weight$	Edge weight function in network coarsening	3.2
π	Mapping from vertices of $\mathcal{S}$ to vertices of $\mathcal{S}_C$	3.2
$\theta \in (0, 1]$	Contraction factor in network coarsening	3.2
$\rho \in (0, 1]$	Neighbors threshold in network coarsening	3.2
mvw	Merged vertex weight function for $\mathcal{S}_C$	3.2
d_C	Neighbors distance for extending T_C	3.2
d_I	Neighbors distance for extending $\pi^{-1}(T_C)$	3.2

Notations for the experimental evaluation

Symbol	Explanation	Section
δ_p	$\%$ of vertices having property p	2.1
δ_q	$\%$ of vertices having property q	2.1
$seeds$	Top-k vertices to which the diffusive property p is assigned	2.1
$ratio_{time}$	Ratio of the runtime of CBAF to the runtime of HyperDC	2.1
$ratio_{spread}$	Ratio of the spread of CBAF to the spread of HyperDC	2.1

Tab. Ap.5: Notations

Ap.2 Systematic Banking Crises prediction

Ap.2.1 Network Indicator Definitions

This appendix provides formal definitions for the network indicators employed in the analysis. Consider a weighted directed network $G = (V, E, w)$ where:

- V is a finite set of nodes (countries)
- $E \subseteq V \times V$ is a finite set of directed edges
- $w : E \to [0, 1]$ assigns a weight to each edge.

Edge weights represent log-transformed real cross-border banking exposures (scaled by the log-product GDP of the two nodes in the baseline network; and unscaled in the alternative network). In what follows, adjacent nodes of a node v are given by $N_v = N_v^{in} \cup N_v^{out}$ where $N_v^{in} = \{v' | w(v', v) > 0\}$ and $N_v^{out} = \{v' | w(v, v') > 0\}$ and $b_{v,v'} = 1$ if $w(v, v') > 0$ and 0 otherwise. We compute the following network indicators:

1. In-degree and Out-degree. The in-degree of a node v is denoted d_v^{in} and represents the total number of a node's creditors. The out-degree of a node v is denoted d_v^{out} and represents the total number of a node's debtors.

$$d_v^{in} = \sum_{v' \in V} b_{v',v}$$

$$d_v^{out} = \sum_{v' \in V} b_{v,v'}$$

2. In-strength and Out-strength. The in-strength of a node v is denoted $str^{in}(v)$ and refers to the total weight of in-coming edges (a node's liabilities). The out-strength of a node v is denoted $str^{out}(v)$ and represents the total weight of out-going edges (a node's assets or

exposures).

$$str^{in}(v) = \sum_{(v',v)\in E} w(v',v)$$

$$str^{out}(v) = \sum_{(v,v')\in E} w(v,v')$$

3. A^{in} and A^{out}. These are measures of in-strength and out-strength that are normalized by the total strength of each nodes' neighbors.

$$A^{in}(v) = \frac{str^{in}(v)}{\sum_{(v',v)\in E} str^{in}(v')}$$

$$A^{out}(v) = \frac{str^{out}(v)}{\sum_{(v,v')\in E} str^{in}(v')}$$

4. AN^{in} and AN^{out}. These indicators denote the average A^{in} and A^{out} values normalized across all nodes in the GBN.

$$AN^{in}(v) = \frac{\sum_{(v',v)\in E} A^{in}(v')}{\sum_{v''\in V} A^{in}(v'')}$$

$$AN^{out}(v) = \frac{\sum_{(v,v')\in E} A^{out}(v')}{\sum_{v''\in V} A^{out}(v'')}$$

5. Average nearest node degree (ANND). The ANND denotes the average in-degree (or out-degree) of neighbor nodes connected toward (or from) a node v.

$$ANND^{in,in}(v) = \frac{\sum_{(v',v)\in E} d_{v'}^{in}}{d_v^{in}}$$

$$ANND^{out,in}(v) = \frac{\sum_{(v',v)\in E} d_{v'}^{out}}{d_v^{in}}$$

$$ANND^{in,out}(v) = \frac{\sum_{(v,v')\in E} d_{v'}^{in}}{d_v^{out}}$$

$$ANND^{out,out}(v) = \frac{\sum_{(v,v')\in E} d_{v'}^{out}}{d_v^{out}}$$

6. Average nearest node strength (ANNS). The ANNS denotes the average in-strength (or out-strength) of neighbor nodes connected to (or from) a node v.

$$ANNS^{in,in}(v) = \frac{\sum_{(v',v)\in E} str^{in}(v')}{d_v^{in}}$$

$$ANNS^{out,in}(v) = \frac{\sum_{(v',v)\in E} str^{out}(v')}{d_v^{in}}$$

$$ANNS^{in,out}(v) = \frac{\sum_{(v,v')\in E} str^{in}(v')}{d_v^{out}}$$

$$ANNS^{out,out}(v) = \frac{\sum_{(v,v')\in E} str^{out}(v')}{d_v^{out}}$$

7. Binary and weighted local clustering coefficients. We use two types of clustering coefficients for various types of triangle-type relationships respectively introduced in [148] and [78].

$$BCC^1(v) = \frac{\sum_{v',v''\in N_v} b_{v',v''}}{|N_v| \times (|N_v| - 1)}$$

$$WCC^1(v) = \frac{\sum_{v',v''\in N_v} w(v',v'')}{|N_v| \times (|N_v| - 1)}$$

as defined in [148], and:

$$BCC^2_{Cycle}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{b_{v,v'}b_{v',v''}b_{v'',v}}}{d_v^{in} \times d_v^{out} - d_v^{\leftrightarrow}}$$

$$BCC^2_{Middleman}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{b_{v,v'}b_{v'',v'}b_{v'',v}}}{d_v^{in} \times d_v^{out} - d_v^{\leftrightarrow}}$$

$$BCC^2_{In}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{b_{v',v}b_{v',v''}b_{v'',v}}}{d_v^{in} \times (d_v^{in} - 1)}$$

$$BCC^2_{Out}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{b_{v,v'}b_{v',v''}b_{v,v''}}}{d_v^{out} \times (d_v^{out} - 1)}$$

$$WCC^2_{Cycle}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{w(v,v')w(v',v'')w(v'',v)}}{d_v^{in} \times d_v^{out} - d_v^{\leftrightarrow}}$$

$$WCC^2_{Middleman}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{w(v,v')w(v'',v')w(v'',v)}}{d_v^{in} \times d_v^{out} - d_v^{\leftrightarrow}}$$

$$WCC^2_{In}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{w(v',v)w(v',v'')w(v'',v)}}{d_v^{in} \times (d_v^{in} - 1)}$$

$$WCC^2_{Out}(v) = \frac{\sum_{v',v''\in N_v} \sqrt[3]{w(v,v')w(v',v'')w(v,v'')}}{d_v^{out} \times (d_v^{out} - 1)}$$

where $d_v^{\leftrightarrow} = |N_v^{in} \cap N_v^{out}|$ as defined in [78]. See online appendix for a graphical representation of the triangle taxonomies.

8. Herfindahl-Hirschmann index (HHI). This is the standard measure of competition or market power concentration from the international organization literature and is not strictly a network indicator. The HHI helps gauge the degree of diversification (or concentration) of a node's cross-border banking portfolios.

$$HHI(v) = \sum_{v'\in N_v^{in}} (\frac{\sum_{(v',v)\in E} w(v',v)}{\sum_{(v'',v)\in E} w(v'',v)})^2$$

Group of indicators	List of indicators
1) Degree and strength	
1-period lagged levels:	d^{in}, d^{out}
1st PC: 90.8 percent	str^{in}, str^{out}
	A^{in}, A^{out}
1-period lagged growth rates:	AN^{in}, AN^{out}
1st PC: 43.5percent	
2nd PC: 32.0 percent	
2) Clustering	
1-period lagged levels:	BCC^{1}, [148]
1st PC: 88.3 percent	WCC^{1}, [148]
2nd PC: 7.0 percent	BCC^{2}_{Cycle}, [78]
1-period lagged growth rates:	$BCC^{2}_{Middleman}$, [78]
1st PC: 69.0 percent	BCC^{2}_{In}, [78]
2nd PC: 14.1 percent	BCC^{2}_{Out}, [78]
	WCC^{2}_{Cycle}, [78]
	$WCC^{2}_{Middleman}$, [78]
	WCC^{2}_{In}, [78]
	WCC^{2}_{Out}, [78]
3) Neighbor connectedness	
1-period lagged levels:	$ANND^{in,in}$
1st PC: 89.7 percent	$ANND^{out,in}$
	$ANND^{in,out}$
1-period lagged growth rates:	$ANND^{out,out}$
1st PC: 58.1 percent	$ANNS^{in,in}$
2nd PC: 36.8 percent	$ANNS^{out,in}$
	$ANNS^{in,out}$
	$ANNS^{out,out}$

Ap.2.2 Principal Component Analysis (PCA)

We group 26 network indicators into three groups – degree and strength, clustering, and neighbor connectedness – and extract the first and second principal components (1st PC and 2nd PC) through PCA. All indicators are computed on the baseline GBN. Each group include the following indicators:

Bibliography

[1] Abdul Abiad. Early warning systems: A survey and regime switching approach. page IMF Working Paper No. 32., 2003.

[2] Daron Acemoglu, Asuman Ozdaglar, and Alireza Tahbaz-Salehi. Systemic risk and stability in financial networks. pages MIT, Department of Economics, mimeo, 2013.

[3] Sibel Adali, Xiaohui Lu, and Malik Magdon-Ismail. Attentive betweenness centrality (ABC): considering options and bandwidth when measuring criticality. In *SocialCom-PASSAT*, pages 358–367, 2012.

[4] Sibel Adali, Xiaohui Lu, and Malik Magdon-Ismail. Local, community and global centrality methods for analyzing networks. *Social Network Analysis and Mining*, 4(1), 2014.

[5] Eytan Adar and Lada A. Adamic. Tracking information epidemics in blogspace., 2005.

[6] Rakesh Agrawal, Tomasz Imieliski, and Arun Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):ACM, 1996.

[7] Lucia Alessi and Carsten Detken. Quasi real time early warning indicators for costly asset price boom/bust cycles: A role for global liquidity. *European Journal of Political Economy*, 27:520–533, 2011.

[8] Fraklin Allen, Ana Babus, and Elena Carletti. Asset commonality, debt maturity and systemic risk. *Journal of Financial Economics*, 104(3):519–534, 2012.

[9] Franklin Allen and Douglas Gale. Financial contagion. *Journal of Political Economy*, 108(1):1–33, 2000.

[10] Roy M. Anderson and Robert M. May. Population biology of infectious diseases: Part I. *Nature*, 280(5721):361, 1979.

[11] Roy M. Anderson and Robert M. May. *Infectious Diseases of Humans*. Oxford University Press, 1991.

[12] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51):21544–21549, 2009.

[13] Paul Asquith, Rebecca Oman, and Christopher Safaya. Short sales and trade classification algorithms. *Journal of Financial Markets*, 13(1):157–173, 2010.

[14] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

[15] Ricardo A. Baeza-Yates and Emilio Davis. Web page ranking using link attributes. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW (Alternate Track Papers & Posters)*, pages 328–329. ACM, 2004.

[16] Bahman Bahmani, Kaushik Chakrabarti, and Dong Xin. Fast personalized pagerank on mapreduce. In Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis, editors, *SIGMOD Conference*, pages 973–984. ACM, 2011.

[17] Norman Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London, 1975.

[18] Emanuele Baldacci, Iva Petrova, Nazim Belhocine, Gabriela Dobrescu, and Samah Mazraani. Assessing fiscal stress. page IMF Working Paper No. 100., 2011.

[19] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Topic-aware social influence propagation models. *Knowl. Inf. Syst.*, 37(3):555–584, 2013.

[20] Jeff Baumes, Mark Goldberg, Malik Magdon-Ismail, and William Wallace. Discovering hidden groups in communication networks. In Hsinchun Chen, Reagan Moore, Daniel Zeng, and John Leavitt, editors, *Intelligence and Security Informatics*, volume 3073, pages 378–389. Springer Berlin / Heidelberg, 2004.

[21] Murray A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965.

[22] Roi Becker, Yifat Chernihov, Yuval Shavitt, and Noa Zilberman. An analysis of the steam community network evolution. In *IEEE 27th Convention of Electrical & Electronics Engineers in Israel (IEEEI)*, pages 1–5, 2012.

[23] Andrew Berg, Eduardo Borensztein, and Catherine Pattillo. Assessing early warning systems: How have they worked in practice? *IMF Staff Papers*, 52(3):462–502, 2005.

[24] Andrew Berg and Cathering Pattillo. Are currency crises predictable? a test. *IMF Staff Papers*, 46(2):107–138, 1999.

[25] Mark Berg, Otfried Cheong, Marc Kreveld, and Mark Overmars. *Computational Geometry*, chapter 5 – Orthogonal Range Searching, pages 95–120. Springer Berlin Heidelberg, 2008.

[26] Pavel Berkhin. Survey: A survey on pagerank computing. *Internet Mathematics*, 2(1):73–120, 2005.

[27] Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change in informational cascades. *Journal of Political Economy*, 100(5):992–1026, October 1992.

[28] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *ACM Conference on Computer and Communications Security*, pages 833–844, 2012.

[29] Monica Billio, Mila Getmansky, Andrew W. Lo, and Loriana Pelizzon. Ecnometric measures of connectedness and systemic risk in the finance and insurance sectors. *Journal of Financial Economics*, 104:535–559, 2012.

[30] BIS. Guide to the international financial statistics. pages Monetary and Economic Department BIS Paper No. 49 (Basel, Switzerland: Bank for International Settlements)., 2009.

[31] BIS. Guidelines to the international locational financial statistics. pages Monetary and Economic Department (Basel, Switzerland: Bank for International Settlements)., 2011.

[32] Sven Blank and Claudia Buch. The euro and cross-border banking: Evidence from bilateral data. *Comparative Economic Studies*, 49(3):389–410, 2007.

[33] Sven Blank and Claudia Buch. International bank portfolios: short and long-run responses to macroeconomic conditions. *Review of International Economics*, 18(2):289–306, 2010.

[34] John C. Bluedorn, Jorg Decressin, and Marco E. Terrones. Do asset price drops foreshadow recessions? page IMF Working Paper No. 13/203, 2013.

[35] Paolo Boldi. Totalrank: ranking without damping. In Allan Ellis and Tatsuya Hagino, editors, *WWW (Special interest tracks and posters)*, pages 898–899. ACM, 2005.

[36] Philip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[37] Ulrik Brandes. A graph-theoretic perspective on centrality. *Social Networks*, 30(2):136–145, 2008.

[38] Linda Briesemeister, Patric Lincoln, and Philip Porras. Epidemic profiles and defense of scale-free networks. *WORM 2003*, Oct. 27 2003.

[39] Sergey Brin and Larry Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[40] Matthias Broecheler, Andrea Pugliese, and V. S. Subrahmanian. DOGMA: A disk-oriented graph matching algorithm for RDF databases. In *ISWC*, 2009.

[41] Matthias Broecheler, Paulo Shakarian, and V. S. Subrahmanian. A scalable framework for modeling competitive diffusion in social networks. In *SocialCom/PASSAT*, 2010.

[42] Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, Antonino Nocera, and Domenico Ursino. Measuring betweenness centrality in social internetworking scenarios. In *OTM Workshops*, pages 666–673, 2013.

[43] Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Supporting information spread in a social internetworking scenario. In *NFMCP*, pages 200–214, 2012.

[44] Matthieu Bussiere. In defense of early warning signals. page Banque de France Working Paper No. 420, 2013.

[45] Julian Caballero. Banking crises and financial integration. page IDB Working Paper No. 364, 2012.

[46] Julian Caballero, Christopher Candelaria, and Galina Hale. Bank relationships and the depth of the current economic crisis. pages FRBSF Economic Letter 2009–38, December 14, 2009.

[47] Çigdem Aslay, Nicola Barbieri, Francesco Bonchi, and Ricardo A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.

[48] Eugenio Cerutti, Stijn Claessens, and Patrick McGuire. *Systemic risk in global banking: What available data can tell us and what more data are needed?*, chapter NBER Systemic Risk and Macro Modeling. Cambridge, MA: University of Chicago Press, 2012.

[49] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In *In Proceedings of the 18th International World Wide Web Conference (WWW'09)*, Madrid, Spain, April 2009.

[50] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM TISSEC*, 10(4), 2008.

[51] Soumen Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *WWW*, pages 571–580. ACM, 2007.

[52] Duen Horng (Polo) Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. Polonium : Tera-scale graph mining for malware detection. In *SDM*, Mesa, AZ, April 2011.

[53] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[54] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.

[55] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.

[56] Yi-Cheng Chen, Wen-Chih Peng, and Suh-Yin Lee. Efficient algorithms for influence maximization in social networks. *Knowl. Inf. Syst.*, 33(3):577–601, 2012.

[57] Jiefeng Cheng, Jeffrey Xu Yu, Bolin Ding, Philip S. Yu, and Haixun Wang. Fast graph pattern matching. In *ICDE Conf.*, pages 913–922. IEEE, 2008.

[58] Chun-Ying Chiang, Liang-Hao Huang, Bo-Jr Li, Jiaojiao Wu, and Hong-Gwa Yeh. Some results on the target set selection problem. *J. Comb. Optim.*, 25(4):702–715, 2013.

[59] Matteo Chinazzi, Giorgio Fagiolo, Javier A. Reyes, and Stefano Schiavo. Post-mortem examination of the international financial network. *Journal of Economic Dynamics and Control*, 37:1692–1713, 2013.

[60] Krishna Prasad Chitrapura and Srinivas R. Kashyap. Node ranking in labeled directed graphs. In David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans, editors, *CIKM*, pages 597–606. ACM, 2004.

[61] Michael Chui. Leading indicators of balance-of-payments crises: A partial review. page Bank of England Working Paper No. 171, 2002.

[62] Mario A. Cleves. Comparing areas under receiver operating characteristic curves from two or more probit or logit models. *The Stata Journal*, 2(3):301–313, 2002.

[63] F.C. Coelho, O.G. Cruz, and C.T. Codeco. Epigrass: A tool to study disease spread in complex networks. *Source Code for Biology and Medicin*, 3(3):361, 2008.

[64] Rama Cont, Amal Moussa, and Edson Bastos e Santos. *Network structure and systemic risk in banking systems*, chapter Handbook of Systemic Risk. New York: Cambridge University Press, 2012.

[65] Chad Creighton and Samir Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19(1):79–86, 2003.

[66] Era Dabla-Norris and Yasemim Bal Gunduz. Exogenous shocks and growth crises in low-income countries: A vulnerabilty index. *World Development*, 59:360–378, 2014.

[67] Asli Demirguc-Kunt and Enrica Detragiache. The determinants of banking crises in developing and developed countries. *IMF Staff Papers*, 45, 1998.

[68] Asli Demirguc-Kunt and Enrica Detragiache. Cross-country empirical studies of systemic banking distress: A survey. *National Institute Economic Review*, 192, 2005.

[69] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.

[70] Raffaele Di Natale, Alfredo Ferro, Rosalba Giugno, Misael Mongiovì, Alfredo Pulvirenti, and Dennis Shasha. SING: Subgraph search in non-homogeneous graphs. *BMC Bioinformatics*, 11:96, 2010.

[71] Shlomi Dolev, Yuval Elovici, and Rami Puzis. Routing betweenness centrality. *Journal of the ACM*, 57(4), 2010.

[72] Pedro Domingos and Matthew Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[73] Hali J. Edison. Do indicators of financial crises work? an evaluation of an early warning system. page Board of Governors of the Federal Reserve System International Finance Discussion Paper No. 675, 2000.

[74] Barry Eichengreen, Andrew K. Rose, and Charles Wyplosz. Exchange market mayhem: The antecedents and aftermath of speculative attacks. *Economic Policy*, 10(2):251312, 1995.

[75] Michael Elkin and David Peleg. Approximating k-spanner problems for k¿ 2. *Theoretical Computer Science*, 337(1):249–277, 2005.

[76] Katrina Ellis, Roni Michaely, and Maureen O'Hara. The accuracy of trade classification rules: evidence from nasdaq. *Journal of Financial Markets*, 35(04):529–551, 2000.

[77] Stephen Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, May 2004.

[78] Giorgio Fagiolo. Clustering in complex directed networks. *Physical Review E*, 76(2):026107, 2007.

[79] Giorgio Fagiolo, Stefano Schiavo, and Javier A. Reyes. The evolution of the world trade web: A weighted-network analysis. *Journal of Evolutionary Economics*, 20(4):479–514, 2010.

[80] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[81] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards scaling fully

personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[82] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[83] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215239, 1979.

[84] Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 71–80. ACM, 2011.

[85] Prasana Gai, Andrew Haldane, and Sujit Kapadia. Complexity, concentration and contagion. *Journal of Monetary Economics*, 58:453–470, 2010.

[86] Tumer Kapan Galina Hale and Camelia Minoiu. Crisis transmission in the global banking network. pages Paper presented at the Workshop on the Economics of Cross–Border Banking, Paris (December 13–14), 2013.

[87] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology in spread of epidemics. *IEEE INFOCOM*, 2005.

[88] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. The effect of network topology on the spread of epidemics. In *IEEE INFOCOM*, Los Alamitos, CA, 2005. IEEE Computer Society Press.

[89] Christos Gkantsidis, Thomas Karagiannis, and Milan Vojnovic. Planet scale software updates. In *SIGCOMM*, pages 423–434, 2006.

[90] Kwang-Il Goh, Eulsik Oh, Hawoong Jeong, Byungnam Kahng, and Doochul Kim. Classification of scale-free networks. *Proceedings of the National Academy of Sciences*, 99(20):12583–12588, 2002.

[91] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 2001.

[92] Pierre-Olivier Gourinchas and Maurice Obstfeld. Stories of the twentieth century for the twenty-first. *American Economic Journal: Macroeconomics*, 4(1):226–265, 2012.

[93] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. A data-based approach to social influence maximization. *PVLDB*, 5(1):73–84, 2011.

[94] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. Learning influence probabilities in social networks. WSDM '10, 2010.

[95] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. CELF++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, pages 47–48, 2011.

[96] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, pages 211–220, 2011.

[97] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. *ICDM*, 2011.

[98] Mark Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[99] Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

[100] Alvaro Graves, Sibel Adali, and Jim Hendler. A method to rank nodes in an RDF graph. In *ISWC*, 2008.

[101] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04*, 2004.

[102] Adrien Guille, Hakim Hacid, Cécile Favre, and Djamel A. Zighed. Information diffusion in online social networks: a survey. *SIGMOD Record*, 42(2):17–28, 2013.

[103] R. Guimerà, L. Danon, A. Dìaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68:065103, Dec 2003.

[104] Alexander Gutfraind. Optimizing topological cascade resilience based on the structure of terrorist networks. *PLoS ONE*, 5(11), 2010.

[105] Galina Hale. Bank relationships, business cycles, and financial crises. *Journal of International Economics*, 88(2):312–325, 2012.

[106] A. Harth and S. Decker. Optimized index structures for querying RDF from the Web. In *Proceedings of the 3rd Latin American Web Congress*, pages 71–80, 2005.

[107] Masazumi Hattori and Yuko Suda. *Developments in a cross-border bank exposure network*, volume 29, chapter Research on Global Financial Stability: The Use of BIS International Financial Statistics, pages 16–31. Committee on the Global Financial System Publications, 2007.

[108] Taher H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[109] Yukio Hayashi, Masato Minoura, and Jun Matsukubo. Recoverable prevalence in growing scale-free networks and the effective immunization. *arXiv:cond-mat/0305549 v2*, Aug. 6 2003.

[110] H. W. Hethcote. Qualitative analyses of communicable disease models. *Mathematical Biosciences*, 28(3-4):335–356, 1976.

[111] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42, 2000.

[112] Kiyotaka Ide, Ryota Zamami, and Akira Namatame. Diffusion centrality in interconnected networks. *Procedia Computer Science*, 24:227 – 238, 2013.

[113] IMF. The imf-fsb early warning exercise–design and methodological toolkit. pages IMF Occasional Paper (Washington, DC: International Monetary Fund). Available on http://www.imf.org/external/np/pp/eng/2010/090110.pdf, 2010.

[114] Mohammad Tanvir Irfan and Luis E. Ortiz. On influence, stable behavior, and the most influential individuals in networks: A game-theoretic approach. *Artificial Intelligence*, 215:79–119, 2014.

[115] M. Jackson and L. Yariv. Diffusion on social networks. *Economie Publique*, 16:69–82, 2005.

[116] Riko Jacob, Dirk Koschtzki, Katharina Lehmann, Leon Peeters, and Dagmar Tenfelde-Podehl. Algorithms for centrality indices. In Ulrik Brandes and Thomas Erlebach, editors, *Network Analysis*, volume 3418, pages 62–82. Springer Berlin / Heidelberg, 2005.

[117] Qingye Jiang, Guojie Song, Gao Cong, Yu Wang, Wenjun Si, and Kunqing Xie. Simulated annealing based influence maximization in social networks. In *AAAI*, 2011.

[118] Oscar Jorda, Moritz Schularick, and Alan M. Taylor. Sovereigns versus banks: Credit, crises, and consequences. page NBER Working Paper No. 19506, 2013.

[119] Paul A. Dreyer Jr. and Fred S. Roberts. Irreversible k-threshold processes: Graph-

theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

[120] Sebnem Kalemli-Ozcan, Elias Papaioannou, and Jose-Luis Peydro. Financial regulation, financial globalization and the synchronization of economic activity. *Journal of Finance*, page forthcoming, 2013.

[121] Graciela Kaminsky, Saul Lizondo, and Carmen M. Reinhart. Leading indicators of currency crises. *IMF Staff Papers*, 45(1):1–48, 1998.

[122] Graciela Kaminsky and Carmen Reinhart. The twin crises: The causes of banking and balance-of-payments crises. *American Economic Review*, 89(3):473–500, 1999.

[123] Chanhyun Kang, Cristian Molinaro, Sarit Kraus, Yuval Shavitt, and V. S. Subrahmanian. Diffusion centrality in social networks. In *ASONAM*, pages 558–564, 2012.

[124] Murat Karabatak and M. Cevdet Ince. An expert system for detection of breast cancer based on association rules and neural network. *Expert Systems with Applications*, 36(2):3465–3469, 2009.

[125] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. *The University of Minnesota*, 2, 1995.

[126] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.

[127] D. Kempe, J.M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 137–146. ACM, 2003.

[128] David Kempe, Jon M. Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, pages 1127–1138, 2005.

[129] M. Kendall. A new measure of rank correlation. *Biometrika*, 30, 1938.

[130] J. O. Kephart and S. R. White. Measuring and modeling computer virus prevalence. *IEEE Computer Society Symposium on Research in Security and Privacy*, 1993.

[131] Michael Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Log. Program.*, 12(3&4):335–367, 1992.

[132] Myunghwan Kim and Jure Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58, 2011.

[133] Masahiro Kimura and Kazumi Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.

[134] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. Efficient estimation of influence functions for SIS model on social networks. In *IJCAI*, pages 2046–2051, 2009.

[135] Masahiro Kimura, Kazumi Saito, and Ryohei Nakano. Extracting influential nodes for information diffusion on a social network. In *AAAI*, pages 1371–1376, 2007.

[136] David Krackhardt. The strength of strong ties: The importance of philos in organizations. In N. Nohria and Robert G. Eccles, editors, *Networks and organizations: Structure, Form, and Action*, pages 216–239. Harvard Business School Press, 1992.

[137] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM Press.

[138] M. Lad, X. Zhao, B. Zhang, D. Massey, and L. Zhang. Analysis of BGP Update Burst During Slammer Attack. In *The 5th International Workshop on Distributed Computing*, Dec 2005.

[139] Luc Laeven and Fabian Valencia. Systemic banking crises: An update. page IMF Working Paper No. 12/163, 2012.

[140] Luc Laeven and Fabian Valencia. Systemic banking crises database. *IMF Economic Review*, 61(2):225–270, 2013.

[141] Amy Nicole Langville and Carl Dean Meyer. Survey: Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2003.

[142] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC*, pages 1059–1068, 2010.

[143] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 420–429, 2007.

[144] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[145] Jun Li, Zhen Wu, and Eric Purpus. CAM04-5: Toward Understanding the Behavior of BGP During Large-Scale Power Outages. *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–5, Nov. 2006.

[146] Chen Liao and Shiyan Hu. Polynomial time approximation schemes for minimum disk cover problems. *Journal of Combinatorial Optimization*.

[147] Roy Lindelauf, Peter Borm, and Herbert Hamers. The influence of secrecy on the communication structure of covert networks. *Social Networks*, 31(2):126–137, 2009.

[148] L. Lopez-Fernandez, G. Robles, and J.G. Barahona. Applying social network analysis to the information in cvs repositories. *26th International Conference on Software Engineering - W17S Workshop*, 2004.

[149] Hongjun Lu, Jiawei Han, and Ling Feng. Stock movement prediction and n-dimensional inter-transaction association rules. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, page 12, 1998.

[150] William G. Madow and Lillian H. Madow. On the theory of systematic sampling, i. *The Annals of Mathematical Statistics*, 15(1):1–24, 1944.

[151] Michael Mathioudakis, Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Antti Ukkonen. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 529–537. ACM, 2011.

[152] Yasuko Matsubara, Yasushi Sakurai, B. Aditya Prakash, Lei Li, and Christos Faloutsos. Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 6–14, 2012.

[153] Yasuko Matsubara, Yasushi Sakurai, W. G. Van-Panhuis, and Christos Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *KDD*, pages 105–114, 2014.

[154] A G McKendrick. Applications of mathematics to medical problems. In *Proceedings of Edin. Math. Society*, volume 44, pages 98–130, 1925.

[155] Yasir Mehmood, Nicola Barbieri, Francesco Bonchi, and Antti Ukkonen. Csi: Community-level social influence analysis. In *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*. 2013.

[156] Camelia Minoiu, Chanhyun Kang, VS Subrahmanian, and Anamaria Berea. Does financial connectedness predict crises? In *Quantitative Finance Journal, volume 15, issue 4*, pages 607–624, 2015.

[157] Camelia Minoiu and Javier A. Reyes. A network analysis of global banking: 1978-2010. *Journal of Financial Stability*, 9:168–184, 2013.

[158] Alan Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In Constantine Dovrolis and Matthew Roughan, editors, *Internet Measurement Conf.*, pages 29–42. ACM, 2007.

[159] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *Security & Privacy, IEEE*, 1(4):33–39, 2003.

[160] David Moore, Colleen Shannon, and Kimberly C. Claffy. Code-red: a case study on the spread and victims of an internet worm. In *Internet Measurement Workshop*, pages 273–284, 2002.

[161] Carlo Morselli, Cynthia Gigure, and Katia Petit. The efficiency/security trade-off in criminal networks. *Social Networks*, 29(1):143 – 153, 2007.

[162] Danil Nemirovsky and Konstantin Avrachenkov. Weighted pagerank: Cluster-related

weights. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-277. National Institute of Standards and Technology (NIST), 2008.

[163] Juhani Nieminen. On the centrality in a graph. *Scandinavian Journal of Psychology*, 15(1):332336, 1974.

[164] Elizabeth R. Odders-White. On the occurrence and consequences of inaccurate trade classification. *Journal of Financial Markets*, 3(3):259–286, 2000.

[165] Michael Ovelgonne, Chanhyun Kang, Anshul Sawant, and VS Subrahmanian. Covertness centrality in networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 863–870, 2012.

[166] Aditya Pal and Scott Counts. Identifying topical authorities in microblogs. In *WSDM*, pages 45–54, 2011.

[167] Evangelos E. Papalexakis, Tudor Dumitras, Duen Horng Chau, B. Aditya Prakash, and Christos Faloutsos. Spatio-temporal mining of software adoption & penetration. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013.

[168] R. Pastor-Santorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters 86*, 14, 2001.

[169] B. Aditya Prakash, Deepayan Chakrabarti, Michalis Faloutsos, Nicholas Valler, and Christos Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*, 2011.

[170] B. Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. Spotting culprits in epidemics: How many and which ones? In *ICDM*, 2012.

[171] Manish Purohit, B Aditya Prakash, Chanhyun Kang, Yao Zhang, and VS Subrahmanian. Fast influence-based coarsening for large networks. In *20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014.

[172] Rami Puzis, Polina Zilberman, Yuval Elovici, Shlomi Dolev, and Ulrik Brandes. Heuristics for speeding up betweenness centrality computation. In *SocialCom-PASSAT*, pages 302–311, 2012.

[173] Carmen M. Reinhart and Vincent R. Reinhart. Capital flow bonanzas: An encompassing view of the past and present. page NBER Working Paper No. 14321, 2009.

[174] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[175] Dani Rodrik. The real exchange rate and economic growth. *Brookings Papers on Economic Activity*, 39(2):365–439, 2008.

[176] Everett M. Rogers. *Diffusion of Innovations, 5th Edition*. Free Press, August 2003.

[177] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966.

[178] Kazumi Saito, Masahiro Kimura, Kouzou Ohara, and Hiroshi Motoda. Efficient discovery of influential nodes for SIS models in social networks. *Knowl. Inf. Syst.*, 30(3):613–635, 2012.

[179] Thomas C. Schelling. *Micromotives and macrobehavior*. W.W. Norton and Co., 1978.

[180] Stefano Schiavo, Javier A. Reyes, and Giorgio Fagiolo. International trade and financial integration: A weighted network analysis. *Quantitative Finance*, 10:389–399, 2010.

[181] Moritz Schularick and Alan M. Taylor. Credit booms gone bust: Monetary policy, leverage cycles, and financial crises, 1870-2008. *American Economic Review*, 102(2):1029–1061, 2012.

[182] F. Schweitzer, Giorgio Fagiolo, D. Sornette, F. Vega-Redondo, A. Vespignani, and D. R. White. Economicnetworks:thenewchallenges. *Science*, 325:422, 2009.

[183] P. Shakarian, V.S. Subrahmanian, and M.L. Sapino. Using generalized annotated programs to solve social network optimization problems. In *Technical Communications of the 26th International Conference on Logic Programming, ICLP 2010*, volume 7 of *LIPIcs*, pages 182–191. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[184] Paulo Shakarian, Matthias Broecheler, V. S. Subrahmanian, and Cristian Molinaro. Using generalized annotated programs to solve social network diffusion optimization problems. *ACM Trans. Comput. Log.*, 14(2):10, 2013.

[185] Michael Sintek and Malte Kiesel. RDFBroker: A signature-based high-performance RDF store. In *ESWC*, pages 363–377, 2006.

[186] Stuart Staniford, David Moore, Vern Paxson, and Nicholas Weaver. The top speed of flash worms. In *WORM*, pages 33–42, 2004.

[187] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, Berkeley, CA, USA, 2002. USENIX Association.

[188] Joseph Stiglitz. Risk and global economic architecture: Why full financial integration may be undesirable. page NBER Working Paper No. 15718, 2010.

[189] V.S. Subrahmanian and J. Ernst. Method and system for optimal data diagnosis. pages US Patent Nr. 7474987 dated January 6, 2009, 2009.

[190] V.S. Subrahmanian, A. Mannes, A. Sliva, J. Shakarian, and J. Dickerson. Computational analysis of terrorist groups: Lashkar-e-taiba. *Information Systems and Applications*, XIV:Springer, 2013.

[191] Piotr L. Szczepanski, Tomasz P. Michalak, and Michael Wooldridge. A centrality measure for networks with community structure based on a generalization of the Owen value. In *ECAI*, pages 867–872, 2014.

[192] Benjamin M. Tabak, Marcelo Takami, Jadson M. C. Rocha, Daniel O. Cajuiero, and Sergio R. S. Souza. Directed clustering coefficient as a measure of systemic risk in complex banking networks. *Physica A: Statistical Mechanics and its Applications*, 394(C):211–216, 2014.

[193] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *KDD*, pages 807–816, 2009.

[194] Yannis Theoharis, Vassilis Christophides, and Gregory Karvounarakis. Benchmarking database representations of rdf/s stores. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 685–701. Springer, 2005.

[195] H. Tong, B. Aditya Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *ACM CIKM*, 2012.

[196] Hanghang Tong, B. Aditya Prakash, Charalampos E. Tsourakakis, Tina Eliassi-Rad, Chris-

tos Faloutsos, and Duen Horng Chau. On the vulnerability of large graphs. In *ICDM*, 2010.

[197] Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. Annotated rdf. *ACM Trans. Comput. Logic*, 11(2):10:1–10:41, January 2010.

[198] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Min. Knowl. Discov.*, 25(3):545–576, 2012.

[199] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, and L. Zhang. Observation and Analysis of BGP Behavior under Stress. In *IMW*, 2002.

[200] D. Watts and J. Peretti. Viral marketing for the real world. *Harvard Business Review*, 2007.

[201] Nicholas Weaver and Dan Ellis. Reflections on Witty: Analyzing the attacker. *;login: The USENIX Magazine*, 29(3):34–37, June 2004.

[202] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*, pages 261–270, 2010.

[203] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF storage and retrieval in Jena2. *SWDB Conf.*, 3:7–8, 2003.

[204] Janet L. Yellen. Interconnectedness and systemic risk: Lessons from the financial crisis, 2013. Remarks at the AEA/AFA Joint Luncheon, San Diego, CA, January 4.

[205] Shijie Zhang, Shirong Li, and Jiong Yang. GADDI: distance index based subgraph matching in biological networks. In *EDBT Conf.*, pages 192–203, 2009.

[206] Shijie Zhang, Shirong Li, and Jiong Yang. SUMMA: subgraph matching in massive graphs. In *CIKM Conf.*, pages 1285–1288, 2010.

[207] Ke Zhu, Ying Zhang, Xuemin Lin, Gaoping Zhu, and Wei Wang 0011. NOVA: A novel and efficient framework for finding subgraph isomorphism mappings in large graphs. In *DASFAA Conf.*, pages 140–154, 2010.

[208] Lei Zou, Lei Chen, and M. Tamer Özsu. Distancejoin: Pattern match query in a large graph database. *VLDB Conf.*, 2(1):886–897, 2009.