

# Diffusion Centrality in Social Networks

Chanhyun Kang\*, Cristian Molinaro†, Sarit Kraus‡, Yuval Shavitt§ and V.S. Subrahmanian\*

\* Department of Computer Science, University of Maryland, USA

Email: {chanhyun,vs}@cs.umd.edu

† Department of Electronics, Computer and System Sciences, Università della Calabria, Italy

Email: cmolinaro@deis.unical.it

‡ Department of Computer Science, Bar-Ilan University, Israel

Email: sarit@cs.biu.ac.il

§ School of Electrical Engineering, Tel Aviv University, Israel

Email: shavitt@eng.tau.ac.il

**Abstract**—Though centrality of vertices in social networks has been extensively studied, all past efforts assume that centrality of a vertex solely depends on the structural properties of graphs. However, with the emergence of online “semantic” social networks where vertices have properties (e.g. gender, age, and other demographic data) and edges are labeled with relationships (e.g. friend, follows) and weights (measuring the strength of a relationship), it is essential that we take semantics into account when measuring centrality. Moreover, the centrality of a vertex should be tied to a diffusive property in the network - a Twitter vertex may have high centrality w.r.t. jazz, but low centrality w.r.t. Republican politics. In this paper, we propose a new notion of diffusion centrality (DC) in which semantic aspects of the graph, as well as a diffusion model of how a diffusive property  $p$  is spreading, are used to characterize the centrality of vertices. We present a hypergraph based algorithm to compute DC and report on a prototype implementation and experiments showing how we can compute DCs (using real YouTube data) on social networks in a reasonable amount of time. We compare DC with classical centrality measures like degree, closeness, betweenness, eigenvector and stress centrality and show that in all cases, DC produces higher quality results. DC is also often faster to compute than both betweenness, closeness and stress centrality, but slower than degree and eigenvector centrality.

## I. 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* [1], [2], *betweenness centrality* [3], [4], *stress centrality* [5], *closeness centrality* [6], [7], *eigenvector centrality* [8] only take the structure of the network into account - they do not take properties of the vertices or properties or weights of edges into account when computing centrality. As a consequence, any “semantics” embedded in the network is ignored. This can cause serious problems as shown in the following toy example.

*Example 1 (HIV)*: Figure 1 shows 4 people  $a, b, c, d$ , where  $b$  has HIV. Solid edges denote sexual relationships, while dashed edges denote friend relationships (Figure 1 shows undirected edges as sexual and friend relationships are symmetric). Edge weights denote the intensity of these relationships. The table below shows the centrality of all vertices according to the most common centrality measures in the literature.

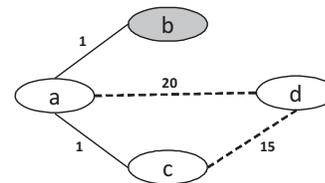


Fig. 1: A small HIV social network. Shaded vertices denote people with HIV.

Centrality Measure	a	b	c	d
Degree	1	0.33	0.66	0.66
Betweenness	2	0	0	0
Stress	2	0	0	0
Closeness	0.33	0.2	0.25	0.25
Eigenvector	0.375	0.125	0.25	0.25

Intuitively, the “central” person in this network (from the point of view of minimizing spread of HIV) is  $b$ , because he is the only person with HIV. However,  $b$  has the lowest centrality according to all five centrality measures above.

Another problem with existing centrality measures is that they *ignore how properties (e.g. HIV) diffuse through the SN*, solely focusing on the structure of the network. We can readily think of networks (e.g. Twitter) where person A has 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 the structural properties of the graph and by a vertex’s ability to diffuse a given property. *This paper proposes the novel notion of diffusion centrality that takes an SN, a diffusive property  $p$ , and a previously learned diffusion model  $\mathcal{D}$  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 [9]. The contributions of the paper are as follows: (i) We formally define *diffusion centrality* and show how it captures the intuitions of Example 1. (ii) We propose a “hypergraph fixed point algorithm” and use it to develop the HyperDC algorithm. (iii) We report on an

experimental evaluation comparing diffusion centrality with classical centrality measures in terms of both running time and “quality” of the central vertices determined by the different centrality measures. Experimental results show that diffusion centrality determines higher quality central vertices, is often faster to compute than betweenness, closeness and stress centrality, but slower than degree and eigenvector centrality.

## II. PRELIMINARIES

We formally define social networks (SNs) and a language to express diffusion models.<sup>1</sup> We assume the existence of a set  $\mathbf{VP}$  of unary predicate symbols (to capture properties of vertices in a social network), called *vertex predicate symbols* (also referred to as *properties*), and a set  $\mathbf{EP}$  of ternary 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 4-tuple  $\mathcal{S} = (V, E, \mathbf{VL}, \omega)$  where:

- 1)  $V$  is a finite set whose elements are called *vertices*;
- 2)  $E \subseteq V \times V \times \mathbf{EP}$  is a finite set of (*labeled*) *edges*;
- 3)  $\mathbf{VL} : V \rightarrow 2^{\mathbf{VP}}$  assigns a set of properties to each vertex;
- 4)  $\omega : E \rightarrow \mathbb{R}$  assigns a weight to each edge.

*Example 2:* Consider the SN of Example 1. Here,  $\mathbf{VP} = \{\text{hiv}\}$  and  $\mathbf{EP} = \{\text{sp}, \text{fr}\}$ , where sp and fr stand for sexual and friend relationships, respectively. The SN is defined as:

- $V = \{a, b, c, d\}$ .
- $E = \{\langle a, b, \text{sp} \rangle, \langle b, a, \text{sp} \rangle, \langle a, c, \text{sp} \rangle, \langle c, a, \text{sp} \rangle, \langle a, d, \text{fr} \rangle, \langle d, a, \text{fr} \rangle, \langle c, d, \text{fr} \rangle, \langle d, c, \text{fr} \rangle\}$ .
- $\mathbf{VL}(b) = \{\text{hiv}\}; \mathbf{VL}(a) = \mathbf{VL}(c) = \mathbf{VL}(d) = \emptyset$ .
- $\omega(\langle a, b, \text{sp} \rangle) = \omega(\langle b, a, \text{sp} \rangle) = \omega(\langle a, c, \text{sp} \rangle) = \omega(\langle c, a, \text{sp} \rangle) = 1; \omega(\langle a, d, \text{fr} \rangle) = \omega(\langle d, a, \text{fr} \rangle) = 20; \omega(\langle c, d, \text{fr} \rangle) = \omega(\langle d, c, \text{fr} \rangle) = 15$ .

Diffusion models specify how vertex properties “propagate”. Diffusion models fall into three categories: *tipping models* in which a vertex adopts a behavior when a sufficiently large percentage of its neighbors adopt the behavior [10], [11], *cascading models* in which diffusions cascade across the network (cascade models have been developed for product adoptions [12], the SIR model of disease spread [13], marking photos as favorites in Flickr [14]), and *homophilic models* in which vertices adopt behaviors on the basis of their intrinsic properties but not on the basis of network structure [15].

Consider an SN  $\mathcal{S} = (V, E, \mathbf{VL}, \omega)$ . We assume the existence of a set  $\mathcal{V}$  of variables ranging over vertices and a set  $\mathcal{W}$  of variables ranging over real numbers. If  $p \in \mathbf{VP}$  and  $X \in V \cup \mathcal{V}$ , then  $p(X)$  is a *vertex atom*. If  $e \in \mathbf{EP}$ ,  $X_1, X_2 \in V \cup \mathcal{V}$ , and  $W \in \mathcal{W} \cup \mathbb{R}$ , then  $e(X_1, X_2, W)$  is an *edge atom*. If  $W_1, W_2 \in \mathcal{W} \cup \mathbb{R}$  and  $op \in \{=, \neq, \leq, <, \geq, >\}$ , then  $W_1 op W_2$  is a *comparison atom*. An atom is *ground* iff no variable appears in it.

A *diffusion rule*  $r$  for a property  $p \in \mathbf{VL}$  is an expression:

$$\mathbb{P}(p(X) \mid A_1 \wedge \dots \wedge A_n) = c$$

<sup>1</sup>Other syntaxes can also be used to express diffusion models. The syntax is not claimed as a major contribution of this paper.

where  $p(X)$  is a vertex atom,  $A_1, \dots, A_n$  are (vertex or edge or comparison) atoms, and  $c \in [0, 1]$  is a real number.  $body(r)$  denotes the set  $\{A_1, \dots, A_n\}$ . Intuitively, this diffusion rule states that the confidence of  $X$  having property  $p$ , given that  $A_1 \wedge \dots \wedge A_n$  holds is  $c$ .  $r$  is *ground* iff there are no variables occurring in it.  $grd(r)$  denotes the set of *ground instances* of diffusion rule  $r$ , i.e., the set of all ground rules obtained from  $r$  by replacing every occurrence of a variable in  $\mathcal{V}$  with a vertex and every occurrence of a variable in  $\mathcal{W}$  with a real number, with multiple occurrences of the same variable being replaced in the same way.

A *diffusion model*  $\mathcal{D}$  w.r.t. property  $p$  is a finite set of diffusion rules for  $p$ .  $grd(\mathcal{D})$  denotes the set of all ground instances of diffusion rules in  $\mathcal{D}$ , i.e.  $grd(\mathcal{D}) = \bigcup_{r \in \mathcal{D}} grd(r)$ .

*Example 3:* A simple diffusion model  $\mathcal{D}_{hiv}$  for hiv may contain the following rules:

$$\begin{aligned} \mathbb{P}(\text{hiv}(X) \mid \text{sp}(X, Y, W) \wedge W > 0 \wedge \text{hiv}(Y)) &= 0.9. \\ \mathbb{P}(\text{hiv}(X) \mid \text{fr}(X, Y, W) \wedge \text{sp}(Y, Z, W') \wedge \\ &W > 10 \wedge W' > 0 \wedge X \neq Z \wedge \text{hiv}(Z)) = 0.4. \\ \mathbb{P}(\text{hiv}(X) \mid \text{sp}(X, Y, W) \wedge \text{sp}(Y, Z, W') \wedge \\ &W > 0 \wedge W' > 0 \wedge X \neq Z \wedge \text{hiv}(Z)) = 0.6. \end{aligned}$$

The first rule says that the confidence of a vertex having HIV is 90% if one of its sexual partners has HIV. The second rule says that the confidence of a vertex having HIV, given that it has a good friend (with weight over 10) who is a sexual partner of a vertex with HIV is 40%. The last rule can be similarly read.

*Example 4:* Suppose the SN in Figure 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:

$$\mathbb{P}(\text{adopter}(Y) \mid \text{adopter}(X) \wedge \text{male}(X) \wedge \text{fr}(X, Y, W) \wedge W > 0) = 0.6$$

The vertex who 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 1 that  $d$  is not the most relevant vertex. 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).

In addition to cascade models shown in the HIV example, our syntax can express homophilic models and tipping models. The following two rules express homophilic diffusion:

$$\begin{aligned} \mathbb{P}(\text{adopter}(X) \mid \text{male}(X) \wedge \text{young}(X)) &= 0.7. \\ \mathbb{P}(\text{adopter}(X) \mid \text{female}(X) \wedge \text{old}(X)) &= 0.3. \end{aligned}$$

The following is a tipping model saying that if two or more adopters have instant messaging with  $X$ , then  $X$  is an adopter with confidence 0.7.

$$\mathbb{P}(\text{adopter}(X) \mid \text{IM}(X, Y, W) \wedge W > 0 \wedge \text{IM}(X, Y', W') \wedge W' > 0 \wedge \text{adopter}(Y) \wedge \text{adopter}(Y') \wedge Y \neq Y') = 0.7.$$

### III. DIFFUSION CENTRALITY

Diffusion centrality tries to measure how well a vertex  $v$  can diffuse a property  $p$  (e.g. the hiv property), given the semantics and structure of an SN  $\mathcal{S}$  and a diffusion model  $\mathcal{D}$  for property  $p$ . In order to define this formally, we need a number of intermediate definitions.

*Definition 2 (Labeling):* Suppose  $\mathcal{S} = (V, E, \text{VL}, \omega)$  is an SN and  $p$  is a vertex predicate. A  $p$ -labeling of  $\mathcal{S}$  is a mapping  $\ell : V \rightarrow [0, 1]$ .  $\ell$  is *compatible* with  $\mathcal{S}$  iff for each  $v \in V$  such that  $p \in \text{VL}(v)$ , it is the case that  $\ell(v) = 1$ .

A  $p$ -labeling  $\ell$  states the confidence that a given vertex has property  $p$ . We define an ordering  $\preceq$  on  $p$ -labelings as:  $\ell_1 \preceq \ell_2$  iff for each vertex  $v \in V$ ,  $\ell_1(v) \leq \ell_2(v)$ .  $\ell_\perp$  denotes the  $p$ -labeling defined as follows: for each vertex  $v \in V$ , if  $p \in \text{VL}(v)$ , then  $\ell_\perp(v) = 1$ , otherwise  $\ell_\perp(v) = 0$ . Clearly,  $\ell_\perp$  is compatible with  $\mathcal{S}$  and intuitively captures the initial distribution of property  $p$  in  $\mathcal{S}$  – no diffusion model is considered by  $\ell_\perp$ . To capture the effect of a diffusion model, we need to find a labeling that is compatible with both the SN and the diffusion model. To do this, we show how a diffusion model and a network “propagate” a property from one vertex to another using a mapping that transforms labelings. We start by defining enabled rules.

*Definition 3 (Enabled Rule):* Let  $\mathcal{S} = (V, E, \text{VL}, \omega)$  be an SN and  $\mathcal{D}$  a diffusion model for a property  $p$ . Let  $r \in \text{grad}(\mathcal{D})$  be the ground diffusion rule:

$$\mathbb{P}(p(v) \mid A_1 \wedge \dots \wedge A_n) = c$$

where  $v$  is a vertex and each  $A_i$  is a ground atom.  $r$  is *enabled* (w.r.t.  $\mathcal{S}$ ) iff:

- for each vertex atom  $q(v') \in \text{body}(r)$  such that  $q \neq p$ , it is the case that  $q \in \text{VL}(v')$ ;
- for each edge atom  $e(v_1, v_2, w) \in \text{body}(r)$ , it is the case that  $\langle v_1, v_2, e \rangle \in E$  and  $\omega(\langle v_1, v_2, e \rangle) = w$ ; and
- each comparison atom in  $\text{body}(r)$  is true (over the reals).

*Example 5:* Consider the diffusion model of Example 3.

$$\mathbb{P}(\text{hiv}(d) \mid \text{fr}(d, a, 20) \wedge \text{sp}(a, c, 1) \wedge 20 > 10 \wedge 1 > 0 \wedge d \neq c \wedge \text{hiv}(c)) = 0.4.$$

is a ground instance of the second diffusion rule of  $\mathcal{D}_{\text{hiv}}$ . This rule is enabled w.r.t. the SN of Figure 1.<sup>2</sup>

*Definition 4 (Labeling Transformation):* Let  $\mathcal{S} = (V, E, \text{VL}, \omega)$  be an SN and  $\mathcal{D}$  a diffusion model for a property  $p$ . We associate with  $\mathcal{S}$  and  $\mathcal{D}$ , a mapping  $\text{T}_{\mathcal{S}, \mathcal{D}}$  that maps  $p$ -labelings to  $p$ -labelings.

$$\text{T}_{\mathcal{S}, \mathcal{D}}(\ell)(v) =$$

$$\max \left( \begin{array}{l} \{\ell(v)\} \cup \\ \{c \times \prod_{p(v') \in \text{body}(r)} \ell(v') \mid \exists r \in \text{grad}(\mathcal{D}) \text{ s.t.} \\ \quad r \text{ is enabled and of the form} \\ \quad \mathbb{P}(p(v) \mid A_1 \wedge \dots \wedge A_n) = c\} \end{array} \right)$$

<sup>2</sup>Notice that the atom  $\text{hiv}(c)$  does not play any role in determining whether the rule is enabled or not.

We define the *iterations* of the  $\text{T}_{\mathcal{S}, \mathcal{D}}$  operator as follows:  $\text{T}_{\mathcal{S}, \mathcal{D}} \uparrow 0 = \ell_\perp$ ;  $\text{T}_{\mathcal{S}, \mathcal{D}} \uparrow (k+1) = \text{T}_{\mathcal{S}, \mathcal{D}}(\text{T}_{\mathcal{S}, \mathcal{D}} \uparrow k)$ .

*Proposition 1:* The operator  $\text{T}_{\mathcal{S}, \mathcal{D}}$  is monotonic (w.r.t.  $\preceq$ ) and has a least fixed point, denoted  $\text{lfp}(\text{T}_{\mathcal{S}, \mathcal{D}})$ .

*Example 6:* The least fixed point of the HIV diffusion model assigns 0.9 to  $\text{hiv}(a)$ , 1 to  $\text{hiv}(b)$ , 0.81 to  $\text{hiv}(c)$  and 0.4 to  $\text{hiv}(d)$ . We will show how to compute it later.

Given an SN  $\mathcal{S} = (V, E, \text{VL}, \omega)$ , 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  $(V, E, \text{VL}', \omega)$  where  $\text{VL}'$  is exactly like  $\text{VL}$  except that  $\text{VL}'(v) = \text{VL}(v) \cup \{p\}$ . 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  $(V, E, \text{VL}'', \omega)$  which is just like  $\mathcal{S}$  except that  $\text{VL}''(v) = \text{VL}(v) - \{p\}$ . We now define diffusion centrality.

*Definition 5 (Diffusion Centrality):* Let  $\mathcal{S} = (V, E, \text{VL}, \omega)$  be an SN and  $\mathcal{D}$  a diffusion model for a property  $p$ . The *diffusion centrality* of a vertex  $v \in V$  w.r.t.  $\mathcal{D}$  is defined as follows:

$$\text{dc}(v) = \frac{\sum_{v' \in V - \{v\}} \text{lfp}(\text{T}_{\mathcal{S} \oplus p(v), \mathcal{D}})(v')}{\sum_{v'' \in V - \{v\}} \text{lfp}(\text{T}_{\mathcal{S} \ominus p(v), \mathcal{D}})(v'')}$$

Intuitively, this definition says that to compute the diffusion centrality of vertex  $v$ , we follow two steps:

- 1) 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.
- 2) We then 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 of these two numbers captures the “impact” that would occur in terms of diffusion of property  $p$  if vertex  $v$  had property  $p$ .<sup>3</sup> We illustrate via the HIV example.

*Example 7:* The only vertex in our HIV example with property  $\text{hiv}$  is  $b$ . Therefore it is easy to compute the values of the negative summand of Definition 5 from the values in Example 6. They turn out to be 2.21 for  $a$ , 2.3 for  $c$ , and 2.71 for  $d$ . As  $b$  is the only vertex with HIV, the negative summand for  $b$  is 0 because if we assume  $b$  does not have HIV, then nobody in the network has it and no one gets infected.

As far as the positive summand is concerned, consider  $b$ . Assuming  $b$  has HIV makes no difference because  $b$  already has HIV according to  $\mathcal{S}$ , i.e.  $\mathcal{S} = \mathcal{S} \oplus \text{hiv}(b)$ . Hence, we can use the values in Example 6 to compute the positive summand of  $\text{dc}(b)$ , viz. 2.11.

<sup>3</sup>Considering just the first summation of Definition 5 is wrong - here’s why. Suppose we have an SN and a vertex  $v$  s.t. the first summation of  $\text{dc}(v)$  is 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  $p$ ).

Suppose we now assume  $a$  has HIV.  $b$  has HIV with confidence 1 because  $\mathcal{S}$  says so;  $c$  has HIV with 0.9 confidence via the first diffusion rule;  $d$  has HIV with 0.4 confidence because the second rule applies. Thus, the value of the first summand for  $a$  is 2.3. The following table summarizes the values we obtain for the positive and negative summands of Definition 5 when we compute diffusion centrality of all vertices.

	a	b	c	d
Positive Summand	2.3	2.11	2.3	2.71
Negative Summand	2.21	0	2.3	2.71
Diffusion Centrality	0.09	2.11	0	0

$b$  has the highest centrality w.r.t.hiv – note that classical centrality measures (see Introduction) do not capture this because  $b$  is not a “central” vertex from a purely topological perspective.<sup>4</sup>

*Example 8:* If we return to the cell phone case (Example 4), we see that the DC of  $d$  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 differently important.

*Definition 6 (Diffusion Centrality Problem):* Let  $\mathcal{S} = (V, E, \text{VL}, \omega)$  be an SN and  $\mathcal{D}$  a diffusion model for a property  $p$ . Given  $V' \subseteq V$  and a threshold  $\tau \in [0, |V| - 1]$ , find all pairs  $\langle v, \text{dc}(v) \rangle$  s.t.  $v \in V'$  and  $\text{dc}(v) \geq \tau$ .

Note that if  $V' = V$  and  $\tau = 0$ , then we are asking for the diffusion centrality of every vertex.

#### IV. ALGORITHMS

In this section, we develop the HyperDC algorithm to solve the diffusion centrality problem. The algorithm uses an efficient hypergraph-based algorithm (HyperLFP) as an essential part to compute the least fixed point, we therefore first present HyperLFP.

##### A. The HyperLFP Algorithm

As HyperLFP uses hypergraphs, we first define hypergraphs.

*Definition 7:* A weighted directed hypergraph is a triple  $\langle V, H, W \rangle$  where: (i)  $V$  is a finite set of vertices. (ii)  $H$  is a finite set of (directed) hyperedges. A hyperedge is a pair  $\langle S, t \rangle$  where  $S$  is a (possibly empty) subset of  $V$ , called *source set*, and  $t$  is a vertex in  $V$ , 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. (iii)  $W : H \rightarrow [0, 1]$  is a function assigning a weight to each hyperedge.

We now define a diffusion hypergraph that captures how property  $p$  diffuses through SN  $\mathcal{S}$  according to diffusion model  $\mathcal{D}$ . The hypergraph does not depend on the particular initial  $p$ -labeling of  $\mathcal{S}$ , but depends only on  $\mathcal{D}$  and the structure of  $\mathcal{S}$  in terms of edges and vertex properties other than  $p$ . *Therefore, the diffusion hypergraph has to be computed only once and*

<sup>4</sup>Note that if we add another person to the SN, say  $b1$ , that is identical to  $b$ , i.e., has one sexual partner  $a$  and has HIV, then the diffusion centrality of  $b$  will become 0. Moreover, if  $b$  does not have HIV according to the original network, i.e., no one in the network has HIV, then  $a$  will have the highest diffusion centrality and  $b$  and  $c$  will have the same diffusion centrality.

can be used with different  $p$ -labelings of  $\mathcal{S}$ .<sup>5</sup> In addition, the hypergraph allows us to eliminate (ground) diffusion rules that are not enabled.

*Definition 8 (Diffusion Hypergraph):* Let  $\mathcal{S} = (V, E, \text{VL}, \omega)$  be an SN and  $\mathcal{D}$  a diffusion model for a property  $p$ . The hyperedge associated with a ground rule  $r \in \text{grd}(\mathcal{D})$  of the form  $\mathbb{P}(p(v) | A_1 \wedge \dots \wedge A_n) = c$  is defined as  $\langle \{p(v') \mid p(v') \in \text{body}(r)\}, p(v) \rangle$  and is denoted by  $\text{hedge}(r)$ . The *diffusion hypergraph* for  $\mathcal{S}$  and  $\mathcal{D}$  is a weighted directed hypergraph  $\mathcal{H}(\mathcal{S}, \mathcal{D}) = \langle N, H, W \rangle$  where: (i)  $N = V$ , (ii)  $H = \{\text{hedge}(r) \mid r \in \text{grd}(\mathcal{D}) \text{ and } r \text{ is enabled}\}$ , and (iii) for each  $h \in H$

$$W(h) = \max\{c \mid \mathbb{P}(p(v) \mid A_1 \wedge \dots \wedge A_n) = c \text{ is an enabled ground diffusion rule of } \mathcal{D} \text{ and its associated hyperedge is } h\}$$

The basic idea of the HyperLFP algorithm is that hyperedges that propagate a value greater than zero are kept in a data structure *Heap*, where each hyperedge is associated with the value it propagates to its target vertex; hyperedges in *Heap* with higher values are visited first; when a value is propagated by a hyperedge to its target vertex, say  $v$ , only the hyperedges that can be possibly affected by this are inspected and possibly added to *Heap* – these are the hyperedges having  $v$  in the source set. HyperLFP ensures that when value  $w$  is assigned to a vertex  $v$ , then  $w$  is the final value for  $v$  in the least fixed point; hence the hyperedge that propagated  $w$  as well as any other hyperedge having  $v$  as target vertex do not need to be considered anymore to see if a new higher value can be assigned to  $v$ .

---

##### Algorithm 1 HyperLFP

---

**Input:** A social network  $\mathcal{S} = (V, E, \text{VL}, \omega)$   
A diffusion model  $\mathcal{D}$  for a property  $p$   
The diffusion hypergraph  $\mathcal{H}(\mathcal{S}, \mathcal{D}) = \langle N, H, W \rangle$

**Output:**  $\text{lfp}(\mathcal{T}_{\mathcal{S}, \mathcal{D}})$

- 1:  $p[1..|V|], u[1..|V|], c[1..|V|]$
- 2:  $p[v] = 1$  for all  $v \in V$  s.t.  $p \in \text{VL}(v)$
- 3:  $p[v] = 0$  for all  $v \in V$  s.t.  $p \notin \text{VL}(v)$
- 4:  $u[v] = \{h \mid h \in H \wedge v \in S(h)\}$  for all  $v \in V$
- 5:  $c[v] = 0$  for all  $v \in V$
- 6:  $\text{Heap} = \emptyset$
- 7: **for each**  $h \in H$  **do**
- 8:   **if**  $\bigwedge_{v \in S(h)} p[v] == 1 \wedge W(h) > c[t(h)]$  **then**
- 9:      $c[t(h)] = W(h)$
- 10:     Add  $\langle h, W(h) \rangle$  to *Heap*
- 11: **while**  $\text{Heap} \neq \emptyset$  **do**
- 12:    $\langle \langle S, t \rangle, w \rangle = \text{deleteMax}(\text{Heap})$
- 13:   **if**  $p[t] == 0$  **then**
- 14:      $p[t] = w$
- 15:     **for each**  $h \in u[t]$  **do**
- 16:        $w' = W(h) \times \prod_{v \in S(h)} p[v]$
- 17:       **if**  $p[t(h)] == 0 \wedge w' > c[t(h)]$  **then**
- 18:          $c[t(h)] = w'$
- 19:         Add  $\langle h, w' \rangle$  to *Heap*
- 20: **return**  $p$

---

The current  $p$ -labeling is stored in array  $p$ , initially set to  $\ell_{\perp}$  (lines 2–3). For each vertex  $v \in V$ ,  $u[v]$  keeps track of the hyperedges having  $v$  in their source set, whereas  $c[v]$  keeps track of the highest value a hyperedge already added to *Heap*

<sup>5</sup>This allows us to save time in computing diffusion centrality which requires computing the least fixed point for different initial  $p$ -labelings.

propagates to  $v$ . The **for each** loop on lines 7–10 adds a hyperedge  $h$  together with its weight  $W(h)$  to *Heap* if every vertex  $v$  in the source set has  $p[v] == 1$  (i.e.,  $h$  propagates a value greater than 0 to its target vertex) and there is no hyperedge already in *Heap* that propagates a greater value to  $t(h)$ . At each iteration of the **while** loop on lines 11–19, a pair  $\langle\langle S, t \rangle, w \rangle$  with maximum  $w$  is retrieved from *Heap*. If  $p[t]$  has not been set to a value greater than 0, then it is set to  $w$ , otherwise another hyperedge is retrieved from *Heap*. Indeed, when the algorithm assigns a value to  $p[t]$ , then this is the value of  $t$  in the least fixed point (another property of the algorithm is that higher values are assigned first). If  $w$  is assigned to  $p[t]$ , then hyperedges that can be affected by this are inspected (**for each** loop on lines 15–19). Specifically, only the hyperedges having  $t$  in the source set are inspected. For each of them, if the current labeling assigns 0 to the target vertex, the hyperedge propagates a value greater than 0, and there has not been any hyperedge added to *Heap* propagating a higher value, then the hyperedge is added to *Heap* along with the value it propagates.

*Theorem 1:* HyperLFP correctly computes  $\text{lfp}(\mathbb{T}_{\mathcal{S}, \mathcal{D}})$ .

*Proposition 2:* The worst-case time complexity of Algorithm HyperLFP is  $O(|N| + |H| \cdot (\log|H| + u_{max} \cdot S_{max}))$ , where  $u_{max} = \max_{v \in V} \{|\{h \mid h \in H \wedge v \in S(h)\}|\}$  and  $S_{max} = \max_{h \in H} \{|S(h)|\}$ .

### B. The HyperDC Algorithm

When computing the diffusion centrality of a vertex  $v$ , we note that  $\mathcal{S}$ ,  $\mathcal{S} \oplus p(v)$ , and  $\mathcal{S} \ominus p(v)$  differ only on whether vertex  $v$  has property  $p$  or not. The following simple proposition says how this can be leveraged.

*Proposition 3:* Consider a SN  $\mathcal{S} = (V, E, \text{VL}, \omega)$ , a diffusion model  $\mathcal{D}$  for a property  $p$ , and a vertex  $v \in V$ . Let  $F = \text{lfp}(\mathbb{T}_{\mathcal{S}, \mathcal{D}})$ . (i) If  $p \in \text{VL}(v)$ , then  $dc(v) = \sum_{v' \in V - \{v\}} F(v') - \sum_{v'' \in V - \{v\}} \text{lfp}(\mathbb{T}_{\mathcal{S} \ominus p(v), \mathcal{D}})(v'')$   
(ii) If  $p \notin \text{VL}(v)$ , then  $dc(v) = \sum_{v' \in V - \{v\}} \text{lfp}(\mathbb{T}_{\mathcal{S} \oplus p(v), \mathcal{D}})(v') - \sum_{v'' \in V - \{v\}} F(v'')$

When computing  $dc(v)$ , the definition requires us to compute the difference of two summations. The above result says that if  $F$  has been computed once (in advance), then we only need to compute one summation, thus reducing the expected running time significantly because each summation requires a least fixed point computation. The first (resp. second) case says that if vertex  $v$  originally already had (resp. did not have) the diffusive property, then the first (resp. second) summation is already captured in  $F$ , and so we only need to compute the second (resp. first) one.

Step 1 of Algorithm 2 leverages Proposition 3, computing  $\text{lfp}(\mathbb{T}_{\mathcal{S}, \mathcal{D}})$  just once and reusing it intelligently so that only one fixed point computation needs to be done for each  $dc(v')$  computation. Step 1 also leverages the HyperLFP algorithm's efficiency. HyperDC also prunes intelligently: when computing DC for a vertex  $v'$ , every time a new value is assigned to a vertex during the least fixed point computation, we check if the DC of  $v'$  is going to be below the threshold for sure (lines 22 and 25); if so, the DC computation for  $v'$  is aborted.

---

### Algorithm 2 HyperDC

---

**Input:** A social network  $\mathcal{S} = (V, E, \text{VL}, \omega)$   
A diffusion model  $\mathcal{D}$  for a property  $p$   
The diffusion hypergraph  $\mathcal{H}(\mathcal{S}, \mathcal{D}) = \langle N, H, W \rangle$   
 $V' \subseteq V$  and a threshold  $\tau \in [0, |V| - 1]$   
**Output:**  $\{\langle v', dc(v') \rangle \mid v' \in V' \wedge dc(v') \geq \tau\}$   
1:  $p_S = \text{HyperLFP}(\mathcal{S}, \mathcal{D}, \mathcal{H}(\mathcal{S}, \mathcal{D}))$   
2:  $d_S = \sum_{v \in V} p_S[v]$   
3:  $Result = \emptyset$   
4: **for each**  $v' \in V'$  **do**  
5:    $p[1..|V|], u[1..|V|], c[1..|V|]$   
6:    $p[v] = 1$  for all  $v \in V$  s.t.  $p \in \text{VL}(v)$   
7:    $p[v] = 0$  for all  $v \in V$  s.t.  $p \notin \text{VL}(v)$   
8:   **if**  $p \in \text{VL}(v')$  **then**  $p[v'] = 0$   
9:   **else**  $p[v'] = 1$   
10:    $u[v] = \{h \mid h \in H \wedge v \in S(h)\}$  for all  $v \in V$   
11:    $c[v] = 0$  for all  $v \in V$   
12:    $P = \sum_{v \in V} p[v]$ ;  $N = |\{v \mid v \in V \wedge p \in \text{VL}(v)\}|$ ;  $Heap = \emptyset$   
13:   **for each**  $h \in H$  **do**  
14:     **if**  $\bigwedge_{v \in S(h)} p[v] == 1 \wedge W(h) > c[t(h)]$  **then**  
15:        $c[t(h)] = W(h)$   
16:       Add  $(h, W(h))$  to *Heap*  
17:     **while** *Heap*  $\neq \emptyset$  **do**  
18:        $\langle\langle S, t \rangle, w \rangle = \text{deleteMax}(Heap)$   
19:       **if**  $p[t] == 0$  **then**  
20:          $p[t] = w$ ;  $P = P + w$ ;  $N = N + 1$   
21:         **if**  $p \in \text{VL}(v')$  **then**  
22:           **if**  $(d_S - p_S[v']) - (P - 1) < \tau$  **then**  
23:             **go to line 4**  
24:         **else**  
25:           **if**  $(P - 1 + (|V| - N) \times w) - (d_S - p_S[v']) < \tau$  **then**  
26:             **go to line 4**  
27:           **for each**  $h \in u[t]$  **do**  
28:              $w' = W(h) \times \prod_{v \in S(h)} p[v]$   
29:             **if**  $p[t(h)] == 0 \wedge w' > c[t(h)]$  **then**  
30:                $c[t(h)] = w'$   
31:               Add  $(h, w')$  to *Heap*  
32:           **if**  $p \in \text{VL}(v')$  **then**  $dc(v') = (d_S - p_S[v']) - (P - p[v'])$   
33:           **else**  $dc(v') = (P - p[v']) - (d_S - p_S[v'])$   
34:           **if**  $dc(v') \geq \tau$  **then** Add  $(v', dc(v'))$  to *Result*  
35:     **return** *Result*

---

To achieve this, we maintain a variable  $P$ , which keeps the sum of the  $p[v]$ 's, and  $N$ , which is the number of vertices  $v$  s.t.  $p[v] > 0$ . In the **while** loop on lines 17–31, if a new value is assigned to  $p[t]$  (line 20), then the pruning condition is checked (lines 21–26):

**Case 1:** If  $p \in \text{VL}(v')$ , then we check if  $(d_S - p_S[v']) - (P - 1) < \tau$ . In this case, the positive summand of the definition of DC has already been computed and what is being computed is the negative summand. The positive summand is equal to  $d_S - p_S[v']$ , so this quantity is fixed, whereas the negative summand is at least  $P - 1$  and can only grow. Hence, if the condition above is true, we can stop the computation of diffusion centrality for  $v'$  because we already know that it is not going to be greater than or equal to  $\tau$ .

**Case 2:** If  $p \notin \text{VL}(v')$ , then we check if  $(P - 1 + (|V| - N) \times w) - (d_S - p_S[v']) < \tau$ . In this case the negative summand of the definition of diffusion centrality has already been computed and what is being computed is the positive summand. The negative summand is equal to  $d_S - p_S[v']$ , so this quantity is fixed. We compute an upper bound for the positive summand as  $(P - 1 + (|V| - N) \times w)$ . Specifically,  $P - 1$  accounts for the vertices for which a value w.r.t. the least fixed point has been determined (1 is deducted to ignore the value of  $v'$  itself) – notice that this is correct because if a value is assigned to a vertex, then the value is the final one in the least fixed point.

$(|V|-N) \times w$  gives an upper bound for the value the remaining vertices can cumulatively have in the least fixed point:  $|V|-N$  is the number of vertices with a value still equal to 0 according to the current labeling, and  $w$  is the maximum value that can be assigned to them — this follows from the fact that values are assigned in a non-increasing order.

*Theorem 2:* HyperDC correctly solves the DC problem.

*Proposition 4:* The worst-case time complexity of Algorithm HyperDC is  $O(|V| \cdot (|N| + |H| \cdot (\log|H| + u_{max} \cdot S_{max})))$ , where  $u_{max} = \max_{v \in V} \{|\{h \mid h \in H \wedge v \in S(h)\}|\}$  and  $S_{max} = \max_{h \in H} \{|S(h)|\}$ .

## V. EXPERIMENTAL RESULTS

We compared diffusion centrality with classical centrality measures in terms of running time and overall spread induced by the most central vertices.

We implemented the HyperLFP and HyperDC algorithms in around 5000 lines of Java code. All algorithms were run on an Intel Xeon @ 2.40 GHz, 24 GB RAM.

We considered YouTube data by randomly choosing connected subgraphs of 20K to 100K vertices in steps of 20K. The data consisted of user-ids and friend relations between people (edges). The diffusive property was membership in specific YouTube groups. We used the diffusion models of [14] to describe “favoriting” of Flickr photos for our experiments with different values of their probability of transmission parameter. Intuitively, the diffusion model says that “If  $X$  is a member of group  $g$  and  $X$  is a friend of  $Y$  and  $X$  has a property  $q$ , then  $Y$  is a member of  $g$  with confidence  $\rho_1$ .”

### A. Running time.

The following table reports the average time (in milliseconds) to compute centrality for one single vertex according to different centrality measures as the size of the SN increases.

	Number of vertices in the SN				
	20K	40K	60K	80K	100K
<b>Degree</b>	0.03	0.07	0.11	0.14	0.31
<b>Eigenvector</b>	0.42	0.43	0.60	0.68	0.91
<b>Diffusion</b>	8.71	17.77	29.36	44.20	53.08
<b>Betweenness</b>	88.80	231.57	424.52	581.56	781.60
<b>Closeness</b>	1,652.35	6,436.18	17,275.48	—	—
<b>Stress</b>	1,684.43	6,512.61	—	—	—

For DC, we considered three runs where different randomly chosen vertices were given property  $q$  – in every run,  $q$  was given to 2.5% of the vertices (in the following we also show running times with higher percentages). For each run we computed the average time to compute DC for one vertex and then computed the average of this time over the three runs.

For every centrality measure, the running time increases as the size of SN grows. Computing DC is slower than computing degree and eigenvector centrality, but faster than computing betweenness, closeness and stress centrality. This is not surprising as degree centrality is trivial to compute - and it is also well known that eigenvector centrality (used in PageRank) is also very fast. More precisely, the worst case time complexity to compute betweenness (resp. closeness, degree, eigenvector) centrality is  $O(mn)$  (resp.  $O(mn + n^2 \log n)$ ,  $O(m)$ ,  $O(n)$ ), where  $n$  is the number of vertices and  $m$  is the number of edges of the SN. Running times to compute closeness (resp.

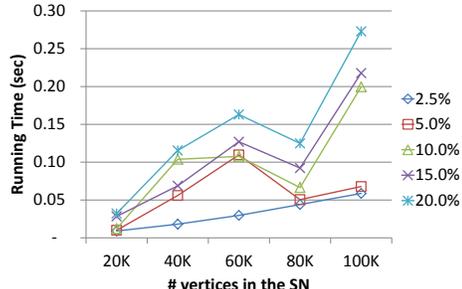


Fig. 2: Avg. time to compute DC for one vertex.

stress) centrality for SNs with 80K and 100K (resp. 60K, 80K and 100K) vertices are not reported: in those cases computing centrality for all vertices of the SN took more than 12 days.

Figure 2 shows the average time to compute DC for one vertex as the size of the SN is varied from 20K to 100K vertices and the percentage of vertices in the SN with property  $q$  varies from 2.5% to 20% – each running time is the average of 5 runs with different random choices of  $g$ ,  $\rho_1$ , and the distribution of  $q$  across the vertices.<sup>6</sup> We note that as the percentage of vertices in the graph with property  $q$  increases, HyperDC takes longer. The reason for this is that when this percentage is small, it restricts the number of ground diffusion rules that are enabled, thus reducing the time for least fixed point computation. It is worth noting that, even when 20% of the vertices in the original SN have property  $q$ , the running time for DC is still lower than the time to compute betweenness, closeness and stress centrality.

### B. Quality

In order to assess the “quality” of the centrality measures we found the 10, 20, 30, 40, 50 most central vertices according to each centrality measure (we also considered randomly chosen vertices), assigned the diffusive property to them (recall that the diffusive property is membership in groups), and computed the overall spread of the diffusive property.

Figure 3a (resp. 3b, 3c, 3d, 3e) shows the overall spread as the number of most central vertices having the diffusive property varies from 10 to 50 – the SN has 20K (resp. 40K, 60K, 80K, 100K) vertices and 2.5% of the vertices were given property  $q$ . The values reported on the X-axis are the overall spread obtained by assigning the diffusive property to the most central vertices divided by the spread in the original social network (each value is the average over three different runs).

Obviously, for any centrality measure and SN size, the spread increases as the number of vertices having the diffusive property increases. Diffusion centrality always gives a much higher spread than any other centrality measure. Figure 3a shows that, in the SN with 20K vertices, betweenness and degree centrality yield pretty much the same spread, eigenvector centrality gives a similar spread except when the top 10 and 50 vertices have the diffusive property, stress centrality

<sup>6</sup>Varying the percentage of vertices having  $q$  does not affect the runtime of the other centrality measures as they ignore semantics aspects of the SN.

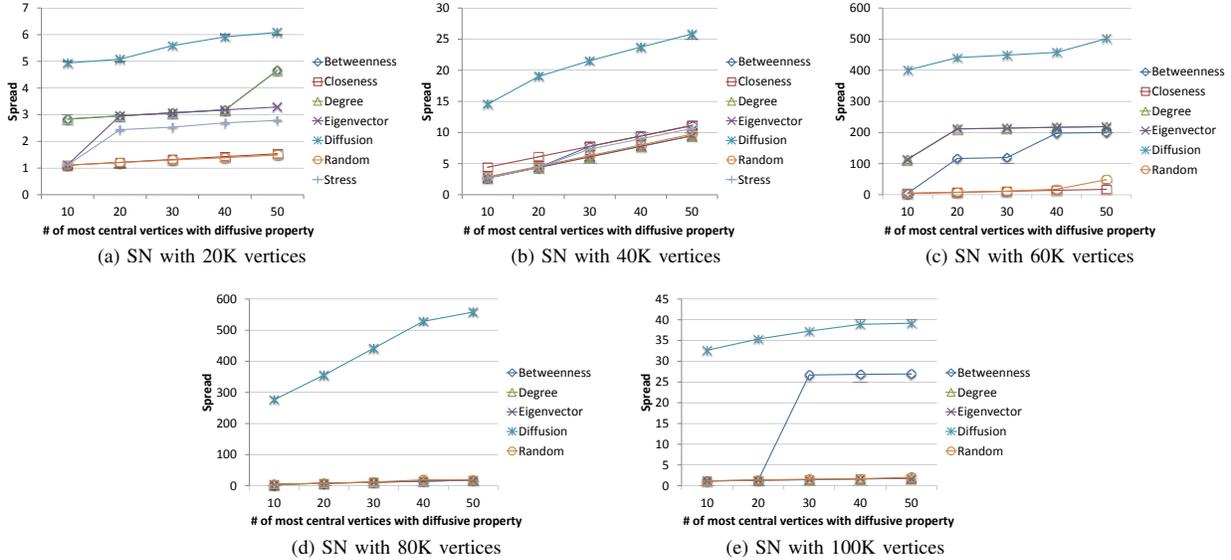


Fig. 3: Diffusive property spread induced by the most central vertices.

yields a lower spread, and the lowest spread is given by closeness centrality and randomly chosen vertices. Diffusion centrality has the highest spread. With SNs having 40K and 80K vertices diffusion centrality yields again the highest spread (this is notable with 80K vertices), the spread of the other centrality measures are close each other (see Figures 3b and 3d). Figure 3c shows that diffusion centrality is again better than any other centrality measure. A lower spread is given by degree and eigenvector centrality, then an even lower spread is given by betweenness centrality; randomly chosen vertices and vertices with highest closeness centrality yield the lowest spread. Figure 3e shows that diffusion centrality is once again better than other centrality measures, a lower spread is given by betweenness centrality, the lowest spread is given by the remaining centrality measures.<sup>7</sup>

## VI. CONCLUSION

Though many different vertex centrality measures have been proposed, they consider only the topology of the network for the purpose of determining vertex centralities. In this paper, we have proposed the novel notion of *diffusion centrality*. Diffusion centrality takes into account the fact that social networks today have both structural and semantical aspects. Moreover, we propose, for the first time, a framework where centrality of a vertex depends on a property of interest and the vertex's ability to diffuse that property. We presented the HyperDC algorithm to compute diffusion centrality and experimentally assessed it using real-world data from YouTube showing that HyperDC works well in practice. In addition, we compared diffusion centrality with classical centrality measures like degree, closeness, betweenness, stress, and eigenvector centrality and shown that diffusion centrality produces higher quality

<sup>7</sup>Stress (resp. closeness) centrality is not reported for SNs with 60K-100K (resp. 80K-100K) vertices because their running times were very high.

results. Diffusion centrality is also often faster to compute than betweenness, closeness and stress centrality, but slower than degree and eigenvector centrality.

**Acknowledgements.** Some of the authors of this paper were funded in part by ARO grants W911NF0910206 and 4021UMUSA0525 ONR grant N000140910685.

## REFERENCES

- [1] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, p. 215239, 1979.
- [2] J. Nieminen, "On the centrality in a graph," *Scandinavian Journal of Psychology*, vol. 15, no. 1, p. 332336, 1974.
- [3] U. Brandes, "A graph-theoretic perspective on centrality," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.
- [4] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.
- [5] A. Shimbel, "Structural parameters of communication networks," *Bulletin of Mathematical Biology*, vol. 15, pp. 501–507, 1953.
- [6] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, pp. 581–603, 1966.
- [7] M. A. Beauchamp, "An improved index of centrality," *Behavioral Science*, vol. 10, no. 2, pp. 161–163, 1965.
- [8] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972.
- [9] M. Broecheler, P. Shakarian, and V. S. Subrahmanian, "A scalable framework for modeling competitive diffusion in social networks," in *SocialCom/PASSAT*, 2010.
- [10] T. Schelling, *Micromotives and macrobehavior*, T. Schelling, Ed. W.W. Norton and Co., 1978.
- [11] M. Granovetter, "Threshold models of collective behavior," *American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.
- [12] M. Jackson and L. Yariv, "Diffusion on social networks," in *Economie Publique*, vol. 16, no. 1, 2005, pp. 69–82.
- [13] R. M. Anderson and R. M. May, "Population biology of infectious diseases: Part I," *Nature*, vol. 280, no. 5721, p. 361, 1979.
- [14] M. Cha, A. Mislove, and P. K. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *WWW*, 2009, pp. 721–730.
- [15] S. Aral, L. Muchnik, and A. Sundararajan, "Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 51, pp. 21 544–21 549, 2009.