Dynamic Networks and Knowledge Discovery

Proceedings of the 2nd Workshop on Dynamic Networks and Knowledge Discovery co-located with ECML PKDD

2014 Nancy, France, September 15, 2014

Rushed Kanawati
Ruggero G. Pensa
Céline Rouveirol

Editors' address:

☐ Rushed Kanawati
LIPN - UMR CNRS 7030

Institut Galilée - Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
E-mail: rushed.kanawati@lipn.univ-paris13.fr

Phone: +33 1 49 40 4079
Fax: +33 1 48 26 07 12

☐ Ruggero G. Pensa
Università di Torino

Dipartimento di Informatica
Corso Svizzera, 185
I-10149 Torino, Italy
E-mail: pensa@di.unito.it

Phone: +39 011 670 6798
Fax: +39 011 75 16 03

☐ Céline Rouveirol
LIPN - UMR CNRS 7030

Institut Galilée - Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
E-mail: celine.rouveirol@lipn.univ-paris13.fr

Phone: ++33 1 49 40 40 79
Fax: +33 1 48 26 07 12

Program Committee

Foreword

Research in modeling, analyzing and mining large-scale networks has attracted an increasing effort in the last few years. Two main reasons, at least, may explain the rapid growth of interest in this field, as attested by the increasing number of scientific publications about this topic:

- On one hand, many datasets studied in various different fields are best described by graphs or linked collection of interrelated objects. Examples cover a wide variety of application fields including: biological system studies, (protein interaction, gene/miRNA regulation, . . .,) the world wide web, bibliographical networks (co-authoring, citation, . . .), P2P networks, semantic networks, and of course the now very popular on-line social networking and microblogging sites (e.g., Facebook, Twitter, Google+), folksonomy-oriented sites (e.g., Foursquare, Delicious, Flickr) and social media platforms (e.g., YouTube, last.fm). Far beyond sharing a networked structure, many of these naturally arising graphs share some non-trivial features (such as power-law node's degree distribution, small separation degree, high clustering coefficient, low density, . . ., etc). This fact has boosted the research in analyzing and mining this class of networks since findings in one field are expected to be easily applied to other analogue fields.

- On the other hand, recent technological advances, in different areas, allow today generating, elaborating and tracking the spatial and temporal evolution of very large scale networks. For example, in systems biology, continuous improvement of technologies has enabled to provide high-throughput and heterogeneous datasets (genomic, proteomic, transcriptomic and metabolomic) allowing to construct huge networks with both rich node and edge meta-data. The possibility of repeating the same experiment at different time points allows to track the evolution of obtained networks, opening the way for understanding the causal relationships between nodes and how these interactions change over time. Purchase data collected on e-commerce sites allow to build very large scale networks connecting customers to products they bought. Again, analyzing and mining such networks would provide new directions for product recommendation computation. On-line social network sites connecting millions of users and publicly available bibliographical databases featuring millions of entries are some examples where a temporal sequence of large-scale networks can be sampled. 10^7 nodes size networks are no more an exception. The spatial evolution of social phenomena is another promising field of research. For instance, investigating how memes diffuse geographically may support the validation or even the discovery of new important sociological hypotheses.

The second edition of our Workshop on Dynamic Networks and Knowledge Discovery has received 15 submissions: 8 were only accepted as long presentations. These are organized into three main sessions:

Application session: This contains two papers. The first one by *Shijaku et.al.* introducing the concept of dynamic embeddedness with an application to the analysis of global pharmaceutical industry interaction network. The second paper is proposed by *Correa and Alves*, in which they provide a functional and visual analytic system for the exploration of enriched metabolic pathways on microbial genetic network.

Large-scale network session: Three papers are included in this session. The first, proposed by *Tabourier et. al.*, tackles the problem of link prediction applying an original rank merging approach. The second paper, by *Grube et. al.*, deals with large-scale network sampling. The last paper, proposed by *Geigl and Helic*, presents a study on alternative approaches of decentralized search, stemming from the very famous papers by Kleinberg and Adamic on the same topic.

Dynamic network session: This session include also three papers. The first one is by *Redmond and Cunningham* in which they propose a method to detect over-represented temporal motif in time-evolving network. the basic idea is to compare the frequency of temporal motif against that of a random temporal network. The second paper, by *Vukadinovic Greetham and Ward*, presents a study of dyadic and multi-actor conversations in twitter. Lastly, *Ben Abdrabbah et al.* present a framework for recommendation computations based on communities detected on time-stamped data.

We would like to thank authors, Program Committee members and all additional reviewers without whom the preparation of this program would not have been possible. Our gratitude also goes to the Computer Science Lab of the Paris-Nord University, the University of Torino and Istituto Nazionale di Alta Matematica that co-supported our workshop through supporting our activities.

R. Kanawati
R. G. Pensa
C. Rouveirol

Contents

Exploring dynamic embeddedness: a network analysis of the global pharmaceutical industry 1991-2012

Elio Shijaku[1], Martin Larraza-Kintana[2] and Ainhoa Urtasun-Alonso[2]

[1] Dept. de Empresa, Universitat Autònoma de Barcelona, Catalonia, Spain
elio.shijaku@uab.cat
[2] Dept. de Gestión de Empresas, Universidad Pública de Navarra, Pamplona, Navarre, Spain
(martin.larraza,ainhoa.urtasun)@unavarra.es

Abstract. We analyze the global pharmaceutical industry network using a unique database that covers strategic transactions (i.e. alliance, financing and acquisition collaborations) for the top 90 global pharmaceutical firms and their ego-network partnerships totaling 4735 members during 1991-2012. The network evolution is traced via a novel method based on the concept of dynamicity that quantifies individual network members (i.e. actors) contribution to the longitudinal period. Specifically, we observe dynamic embeddedness defined for key network centrality measures, and capture the impact of the 2007-2008 global financial crises and the subsequent global and Eurozone recession effects on the strategic transaction flows between the industry's key players as well as their partners. Results suggest the feasibility of dynamicity as a dynamic network indicator as well as the importance of constellation strategic transactions in the study of large network perturbations.

Keywords: longitudinal social network, strategic transaction, dynamicity, dynamic embeddedness

1 Introduction

Organizations are inherently embedded actors of social networks, whose structures evolve dynamically, and as a result of each actor's involvement offer important clues on organizational strategic behavior. Inside a dynamic network, organizations exist as highly mobile entities with their relationships and positional structures continuously changing in time. As such, understanding organizational behavior involves first and foremost capturing organizational dynamics often done by analyzing the longitudinal context where network dynamics is observed. While most literature on longitudinal networks focuses on a more holistic evolution of their structure [2, 4, 11], more recent studies have highlighted the contribution of each actor to the overall network dynamics [1, 6]. This actor-level approach embodied by the concept of dynamicity, relies on the assumption that capturing organization's dynamic behavior in a given network should be based on a combined analysis of both static and dynamic network topologies [12]. Additionally, dynamicity enables researchers to study the effect of specific critical events (i.e. perturbations) that greatly alter the structure of the panel network.

However, quantifying actor involvement and contribution in longitudinal networks, and modeling its behavior against specific perturbations has been limited, with research confined to the effect that organizational crisis has had on organizational communication networks [6, 13]. These few available longitudinal network studies on actor contribution have analyzed network dynamic evolution relying on embeddedness, a well-known concept in social network analysis, long considered a highly strategic resource with important impacts on firm's performance [5, 8]. On this matter, network literature has often relied on centrality-based embeddedness to provide a dynamic image of social network evolution [9, 14].

Furthermore, the majority of research on embeddedness has approached the concept from a dyadic (i.e. a group consisting of only two actors) perspective, bypassing multiple types of firm interaction. Even those studies that focus on the so-called constellation (i.e. interactions between more than two actors) perspective [3, 7], miss out at the relevance of actors engaging in constellation ties of multiple kind, by considering only a single type of collaboration. Additionally, embeddedness' studies have focused heavily on strategic alliance collaborations, a choice well-grounded by the interorganizational collaborations in any given industry, but that often fails to embrace the full picture of strategic interactions' multitude.

We fill these shortcomings by focusing on longitudinal networks generated from strategic transactions, a conceptualization of interorganizational collaborations engaged by a firm with its network partners including strategic alliances, acquisitions and financing collaborations, analyzed under both a dyadic and constellation lens. By doing so, we contribute not only to the literature of alliance collaborations but enhance the currently undernourished network literature on acquisition and financing collaborations as well which play an important role in the dynamics of strategic organizational behavior.

Our study addresses the above gaps by developing and testing a theoretical framework that links the concepts of dynamicity, embeddedness and strategic transactions. By doing so, we uncover the dynamic evolution of the global pharmaceutical industry chosen for its intensive collaboration environment. Given the novel nature of dynamicity as a concept, we attempt to examine some fundamental questions that develop a theory-based understanding of dynamicity and its relationship with embeddedness, as well as analyze the impact that strategic transactions have on such structure: How does dynamicity of centrality measures evolve in a longitudinal network? What is the role of strategic transactions in the evolution of such dynamicity? How does actor´s dynamicity behave in the presence of exogenous events that critically alter the network structure?

Specifically, we expand the actor-level approach on dynamic networks by introducing the concept of dynamic embeddedness, defined as the individual actor's central position variability in a longitudinal network setting compared to its central position variability in a aggregated network. For the purpose of this paper, our focus is exclusively on the dynamicity of structural embeddedness and particularly on the dynamicity of key network centrality measures such as degree, betweenness and closeness. Specifically, we build on a dynamicity model [12] by exploring the critical impact that large exogenous perturbations, such as the 2007-2008 financial crisis, the subsequent 2008-2009 global recession and the more local Eurozone recession of 2011-2013 have on longitudinal networks between top-level actors and their ties in the global pharmaceutical industry.

2 Data and measures

We conduct our analysis on a longitudinal dataset (t = 22 years, 1991-2012) comprising the strategic transactions of 90 leading firms from the pharmaceutical industry in Western Europe, United States, Asia, Africa and Australia. The sample is selected by identifying those firms that appear at least once in the top 50 of the Pharmaceutical Executive Magazine yearly editions for the period 2002-2013. We then use the Pharma and Medtech Business Intelligence database to collect all the strategic transactions that involved the firms in question from 1991 to 2012. During this period, the 90 firms of the sample engaged in alliance, financing and acquisition collaborations with 4645 other firms and institutions creating a total of 12055 strategic transactions.

Due to our selection process, we consider two types of firms, the *core* firms comprised of the top 90 pharmaceuticals and the *periphery* firms including the rest of the actors, with a total population of 4735 firms whose full list is available from the authors. The obtained longitudinal data for both core and periphery firms presents missing actors, since some firms are acquired by others, or simply are not active for any particular year. Our analysis also includes financial data obtained from COMPUSTAT and DATASTREAM databases, supplying missing data when possible using company annual reports. Since the financial data concerns firms from different countries, we convert all currencies to USD with an exchange rate based on the particular year the data is retrieved.

We model each year over the sample period as a separate social network and analyze each network based on a similar approach for the global banking network analysis [10]: (i) the core network, referring to the ties between the top 90 actors; and (ii) the full network comprising all available data from a total of 4735 actors. In our analysis, we consider a weighted undirected tie approach, defined as an N x N "weight" matrix, whose generic entry wij = wji > 0 measures the interaction intensity between any two actors (zero if no link exists between actor i and j). Following this framework and using the software R, we build 22 symmetric 90 x 90 matrices to track the evolution of the core network and 22 symmetric 4735 x 4735 matrices to track the evolution of the full network for the period 1991-2012. Additionally, for dynamicity calculation purposes, we build two matrices which include the aggregated strategic transactions of the entire 22 years period for both types of network.

Network indicators. The network measures of our analysis include three centrality variables (degree, betweenness and closeness centrality) and the dynamicity variable representing the variability of the structural positions of an actor in all short-interval networks compared to its structural position in the aggregated network [12] as shown in equation 1:

$$DDA^{i} = \frac{\sum_{t}^{m} \alpha_{t,t-1} \times \left| OV_{AN} - OV_{t} \right|}{m} \tag{1}$$

where DDA^i is the degree of dynamicity shown by i^{th} actor, OV_{AN} is the observed value (i.e. degree centrality) for the aggregated network, OV_t is the observed value (i.e. degree centrality) for t^{th} yearly network for the i^{th} actor, m is the number of yearly networks considered in the analysis, and $\alpha_{t,t-1}$ is a constant valued according to whether the actor is present or missing in the current and previous short-interval network. The presence of this constant is of crucial important to properly count for actors that disappear from the network due to simple inactivity or possible lack of presence due to network dynamics. The possible combination values that $\alpha_{t,t-1}$ can take are given in Table 1.

Table 1. Possible combination of presence and absence of an actor in two consecutive short-interval networks (Source: Uddin et al. 2013)

Current SIN (Present/Absent)	**Previous SIN** (Present/Absent)	$\alpha_{t,t-1}$
Present	Present	$\alpha_{p,p}$= 1.0
Present	Absent	$\alpha_{p,a}$= 0.5
Absent	Present	$\alpha_{a,p}$= 0.0
Absent	Absent	$\alpha_{a,a}$= 0.0

For the first short-interval network (i.e. $\alpha_{i,0}$ for t = 0) of our analysis, the value of the constant depends on the presence or absence of each actor (i.e. either 0 or 1) at that particular period. The dynamicity model [9] differentiates between two types of dynamicity measures, the dynamicity of an actor represented by equation 1 and the average dynamicity shown by an actor of the t^{th} short-interval network represented by equation 2:

$$DDN^t = \frac{\sum_{t}^{w_t} \alpha_{t,t-1} \times \left| OV_{AN} - OV_t \right|}{w_t} \tag{2}$$

where DDN^t is the average degree of dynamicity shown by an actor of the t^{th} short-interval network meaning the contribution of each actor to the short-interval network´s dynamicity, and w_t is the total number of actors in the t^{th} short-interval (i.e. yearly) network. Therefore, our analytical approach is based on three variables: degree dynamicity, betweenness dynamicity and closeness dynamicity constructed by substituting each obtained centrality value to equations 1 and 2.

Industry indicators. In order to analyze the effect of exogenous critical events such as financial crises and recessions on the global pharmaceutical industry, we construct two main effect variables: (i) global crisis representing the combined effect of the 2007-2008 financial crisis and the global recession of 2008-2009 that followed as a direct consequence, and constructed as a dummy variable that takes the value of 1 for the years 2007-2009 and zero for the rest, and (ii) local crisis representing the exogenous effect of the Eurozone recession during 2011-2013, and constructed as a dummy variable that takes the value of 1 for the years 2011-2012 and zero for the rest.

Control indicators. We use several actor-specific measures such as strategic transaction frequency, R&D intensity, profitability, headquarters (HQ) location and financial leverage age and size. Strategic transaction frequency represents the relative frequency in percentage with which firms engage in strategic transactions. In the analysis, we differentiate between the frequency in percentage of firms engaging in alliance, financing and acquisition collaborations. R&D intensity represents the firm's R&D expenditure scaled by total sales while profitability is measured for each firm by computing the ratio of net income to total assets (ROA). We define financial leverage as the debt-to-total assets ratio including both short- and long-term debt and control for the age of the firms, operationalized as the foundation year minus the year considered in the 2002-2012 panel analysis, and size operationalized as the natural logarithm of company's employees. Finally, since our data consists of multinational firms and knowing that the majority of the top 90 firms are US- or EU-based, we control for headquarters (HQ) location based on two separate dummy variables representing whether firms are U.S. or EU-based.

Model approach. By using a two-step approach to our analysis, first we assess the stability of dynamicity distributions in selected years to capture statistical differences throughout our data using Kolmogorov-Smirnov (henceforth, KS) tests for both core and full networks, second by controlling for firm-specific effects, we investigate the effect that the global crisis (including the 2007-2008 financial crisis and the great 2008-2009 recession), and the local crisis referring to the Eurozone recession, observed for 2011-2012, have on degree, betweenness and closeness dynamicity. For our second step, we run a panel regression model based on random effects (henceforth, RE) with robust estimations based on the model seen below:

$$Y_{it} = \beta_k X_{it} + \alpha_i + u_{it} + \varepsilon_{it}, \qquad i = 1,...,90, \qquad t = 1,...,10,$$

where Y_{it} is firm´s dynamic embeddedness considered as a dependent variable, X_{it} is a vector of firm and industry-specific independent variables including global and local crises, age, size, profitability, financial leverage, R&D intensity, transaction frequency and firm location, α_i is the unknown intercept for each firm, u_{it} is the between-firm error, ε_{it} is the within-firm error, β_k is the coefficient for each k independent variable, i is the number of firms (90 in total) and t is period of time considered (10 years in total or +/- 5 years window before and after the offset of the 2007-2008 financial crisis).

3 Results

We describe the dynamics of the global pharmaceutical industry using four key estimates: (i) tracking dynamic embeddedness evolution based on average dynamicity estimate plots, (ii) monitoring the stability variation of actors' dynamic embeddedness based on KS-tests, (iii) constructing the top five firm rankings based on yearly network average dynamicity estimates, and (iv) understanding the global and local crises causative effect on dynamic embeddedness based on panel regression estimates. Results (i) – (iii) concern the total panel period 1991-2012 while results (iv) concern the panel period 2002-2012.

We track dynamic embeddedness evolution by plotting the cross-sectional averages of dynamic indicators during 1991-2012 as seen in Figure 1. Both panels show that dynamicity values present relative stability before 2007 for degree centrality but vary substantially for betweenness and closeness centrality throughout the study period. Specifically, for the core network, degree and betweenness dynamicity drop respectively 20 percent and 17 percent while closeness dynamicity is almost halved by 40 percent during the global crisis. The more local Eurozone crisis of 2011-2013 (of which we analyze only one year due to sample structure) shows a similar pattern with both networks' dynamicity severely reduced. An exception is closeness centrality, whose dynamicity shows an upward trend for the core network, with signs of a more clustering-oriented tendency.

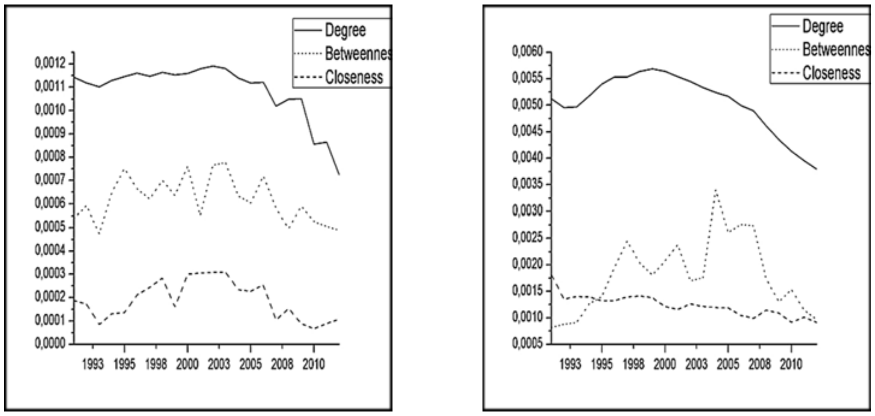


Fig. 1. Core network (left) and full network (right) dynamic embeddedness evolution

We monitor the stability of both core and full networks by comparing the dynamicity distribution in the first year of each decade including last available year's data (1991, 2001 and 2012) with subsequent years in the same decades, a procedure seen in global banking network analysis [10] and whose results are given in Table 2.

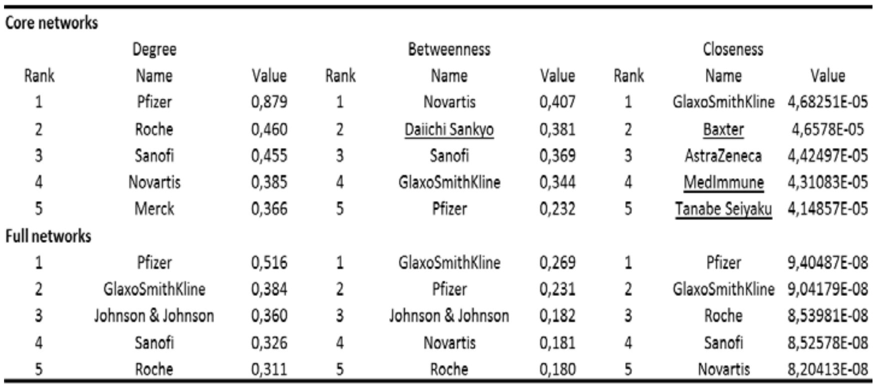
Table 2. Empirical distribution stability for dynamic embeddedness

Core networks				**Full networks**			
	1991	2001	2012		1991	2001	2012
Degree				Degree			
1991-2001	0.00	0.00	1.00	1991-2001	0.54	0.27	1.00
2002-2012	0.36	0.27	0.72	2002-2012	0.54	0.45	0.63
Betweenness				Betweenness			
1991-2001	0.00	0.00	1.00	1991-2001	0.00	0.00	0.18
2002-2012	0.27	0.36	0.54	2002-2012	0.00	0.18	0.00
Closeness				Closeness			
1991-2001	0.54	0.72	0.90	1991-2001	1.00	0.81	1.00
2002-2012	0.63	0.63	0.45	2002-2012	1.00	0.72	0.63

Table 2 shows the proportion of years when the dynamicity distribution is statistically different (at 5 percent level of significance) in each decade compared to 1991, 2001 and 2012. Values of zero mean that the distribution of a particular year compared to a particular decade are statistically close, as is the case for degree and betweenness dynamicity for the years 1991 and 2001 when compared with the 1991-2001 period. This means that in both core and full networks, the firms have kept a similar centrality structure. On the other hand, the distribution for the decade 2002-2012 is statistically different for almost all dynamicity variables in both core and full networks, meaning that the actors' dynamicity has been highly unstable for the second decade. An exception concerns betweenness dynamicity for the full network, whose results show a relatively unaffected actors' brokerage tendency, with only 18 percent of significant distribution change.

Looking at Table 3, we observe that the top five ranking for both degree and betweenness dynamicity includes the biggest pharmaceutical firms (based on their average total sales) which are not underlined, meaning that these firms score high in their centrality position during the core network evolution. Interestingly, closeness dynamicity shows only two big pharmaceuticals in the top five, with a clear tendency of smaller firms reducing their mutual proximities. However, big pharmaceutical firms' hegemony is reinstated in the core network with big pharmaceuticals scoring high in all centrality measures.

Table 3. Firm rankings (1991-2012) for both core and full networks



Core networks								
	Degree			Betweenness			Closeness	
Rank	Name	Value	Rank	Name	Value	Rank	Name	Value
1	Pfizer	0,879	1	Novartis	0,407	1	GlaxoSmithKline	4,68251E-05
2	Roche	0,460	2	Daiichi Sankyo	0,381	2	Baxter	4,6578E-05
3	Sanofi	0,455	3	Sanofi	0,369	3	AstraZeneca	4,42497E-05
4	Novartis	0,385	4	GlaxoSmithKline	0,344	4	MedImmune	4,31083E-05
5	Merck	0,366	5	Pfizer	0,232	5	Tanabe Seiyaku	4,14857E-05
Full networks								
1	Pfizer	0,516	1	GlaxoSmithKline	0,269	1	Pfizer	9,40487E-08
2	GlaxoSmithKline	0,384	2	Pfizer	0,231	2	GlaxoSmithKline	9,04179E-08
3	Johnson & Johnson	0,360	3	Johnson & Johnson	0,182	3	Roche	8,53981E-08
4	Sanofi	0,326	4	Novartis	0,181	4	Sanofi	8,52578E-08
5	Roche	0,311	5	Roche	0,180	5	Novartis	8,20413E-08

Table 4. Dynamic embeddedness during exogenous perturbations: regression estimates

	Dynamic embeddedness		
	Model 1	Model 2	Model 3
Variables	Degree	Betweenness	Closeness
Controls			
Age	-0.000197	-0.000325	-0.0000303
Size	0.00667*	0.0190**	0.000360
HQ location			
US firms	0.0800	-0.0233	0.00944*
EU firms	0.0416	0.00233	0.00626
R&D intensity	-0.0000478***	0.0000134	0.00000624
Profitability	-0.00748	0.0389	-0.0187*
Financial leverage	0.00216	0.0354	-0.0181*
Strategic transaction frequency			
Alliance	0.0196**	0.0131**	0.00523*
Financing	0.00786	0.00295	0.00393
Acquisition	0.0232***	0.0138	0.00251
Main effects			
Global crisis	-0.00222	-0.0154*	-0.0110***
Local crisis	-0.0220**	-0.0236**	-0.0118***
Model statistics			
constant	0.0176	-0.0932	0.0173
R^2 overall	0.0836	0.1269	0.1453
N	751	751	751

Note. Standardized coefficients are reported.
$^* p < .05$ $^{**} p < .01$ $^{***} p < .001$

Looking at the main effects of the regression analysis, we observe the negative effect of the global crisis on dynamicity indicators except degree dynamicity, for which the effect is not significant, meaning that the combined effect of the 2007-2008 crisis and the subsequent global recession of 2008-2009 have not significantly affected the number of strategic transactions originating from each of the core network members. Moreover, we find strong statistical significance for the negative effect that the local Eurozone crisis has had on firms' dynamic embeddedness. Additionally, the type of strategic transaction is found to influence dynamic embeddedness. This effect is understandable considering the relatively high distribution of alliance transactions in the sample (about 75 percent). However, the positive and significant effect of acquisition transactions on degree dynamicity is interesting considering that both acquisition and financing transactions show similar distributions in the sample (about 12.5 percent each). Finally, the observed low R-squared is not necessarily a drawback for the chosen model particularly if we consider that the results present statistically significant predictors and the regressors are used in a panel setting.

4 Discussion and concluding remarks

With respect to the analyses' objective, the results on firm´s dynamic embeddedness suggest that prior to the global crises the global pharmaceutical industry has been relatively stable, with firms' centrality reflecting their market position. Specifically, the top pharmaceutical firms that rank high in terms of sales have a noticeable central position in both core and full networks as observed in the firm rankings. Dynamically speaking, the global pharmaceutical industry has reduced its activity to even lower levels than the beginning of our sampling data, year 1991. While the reduction varies for specific centrality measures, its effect is more prominent after 2007, which coincides with the offset of the 2007-2008 financial crises. The regression results confirm this by showing significant dynamicity reduction during both crises. Furthermore, the regression results indicate that the Eurozone recession has had a far deeper negative effect on global pharmaceutical industry than the global recession.

This study also highlights the importance of acquisition transactions in the expansion of the firms' importance as central hubs. Specifically, the significant effect of acquisitions on degree dynamicity demonstrates the impact that different strategic transactions have on centrality indicators and further reinforces the reasoning behind our choice to study the centrality measures evolution via the dynamicity concept. However, this also raises questions as to why comparable effects of strategic transaction types (i.e. acquisitions and financings) respond differently to centrality-based dynamicity.

Our study's limitations could potentially provide interesting areas of future research. First, we should be careful when generalizing our results about the global pharmaceutical industry, knowing that not all firms in both core and periphery networks are dedicated to pharmaceuticals but come from other adjacent industries such as biotechnology and chemicals. Second, dynamicity measure calculation is based on a novel design which takes into account missing actors during network evolution using a specific constant which should be subject to further research for proper values' assignment.

Finally, the dynamicity measure could be used for other centrality measures (i.e Eigenvector, Bonacich Power) or be included in the analysis of network measures such as actor's structural similarity, structural holes and brokerage elasticity.

Acknowledgments. The authors wish to acknowledge the financial support provided by the Spanish Ministry of Science and Education under projects ECO2010-21393-C04-01 and ECO2013-48496-C4-4-R.

References

1. Braha, D., Bar-Yam, Y.: From Centrality to Temporary Fame: Dynamic Centrality in Complex Networks. Complexity, 12, 2, 59-63 (2006)
2. Brandes, U., Lerner, J., Snijders, T.A.B.: Networks evolving step by step: Statistical analysis of dyadic event data. In: 2009 International Conference Advances in Social Network Analysis and Mining (ASONAM 2009), IEEE Computer Society, pp. 200-205 (2009)
3. Das, T.K., Teng, B-S.: Alliance Constellations: A Social Exchange Perspective, The Academy of Management Review, 27, 3, 445-456 (2002)
4. Gull, K.C., Angadi, A.B., Malagi, K.B.: Framework for Analysis of Dynamic Social Networks Evolution. The International Journal of Engineering and Science, 1, 2, 260-268 (2012)
5. Hoffmann, W.H.: Strategies for managing a portfolio of alliances. Strategic Management Journal, 28, 8, 827-856 (2007)
6. Hossain, L., Murshed, S.T., Uddin, S.: Communication network dynamics during organizational crisis. Journal of Informetrics, 7, 16-35 (2013)
7. Inkpen, A.C., Tsang, E.W.K.: Social Capital, Networks, and Knowledge Transfer. The Academy of Management Review, 30, 1, 146-165 (2005)
8. Karamanos, A.G.: Leveraging micro- and macro-structures of embeddedness in alliance networks for exploratory innovation in biotechnology. R&D Management 42, 1, 71-89 (2012)
9. Lin, Z., Peng, M. W., Yang, H., and Sun, S. L.: How do networks and learning drive MandAs? An institutional comparison between China and the United States. Strategic Management Journal, 30: 1113–1132 (2009)
10. Minoiu, C., Reyes, J.A.: A network analysis of global banking: 1978-2010. Journal of Financial Stability, 9, 168-184 (2013)
11. Snijders, T.A.B.: Statistical models for social networks. Annual Review of Sociology, 37, 129-151 (2011)
12. Uddin, S., Piraveenan, M., Khan, A., Amiri, B.: Conceptual quantification of the dynamicity of longitudinal social networks. SocialCom conference. arXiv:1311.0090v1 (2013)
13. Uddin, S., Chung, K.S.K., Piraveenan, M.: Capturing Actor-level Dynamics of Longitudinal Networks. ASONAM '12 Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), 1006-1011 (2012)
14. Yang, H., Lin, Z(J)., Peng, M.W.: Behind acquisitions of alliance partners: exploratory learning and network embeddedness. Academy of Management Journal, 54, 5, 1069-1080 (2011)

A pipeline for functional and visual analytics of microbial genetic networks

Leandro Corrêa[1], Ronnie Alves[1,2], Fabiana Goés[1], Cristian Chaparro[2] and Lucinéia Thom[3]

[1] PPGCC - Federal University of Pará, Belém, Brazil
`hscleandro@gmail.com`
[2] Vale Institute of Technology, Belém, Brazil
`ronnie.alves@itv.org, cristian.chaparro@itv.org`
[3] PPGC - Federal University of Rio Grande do Sul, Porto Alegre, Brazil
`lucineia@inf.ufrgs.br`

Abstract. Microorganisms abound everywhere. Though we know they play key roles in several ecosystems, too little is known about how these complex communities work. To act as a community they must interact with each other in order to achieve such *community stability* in which proper functions could help to adapt and survive to unbearable conditions. Thus, to effectively understand microbial genetic networks it is necessary to explore them by means of systems biology. An important challenge in systems biology is to determine the structures and mechanisms by which these complex networks control cell processes. In this paper, we present the FUNN-MG pipeline for functional and visual analytics of microbial genetic networks allowing to uncover strong interactions inside microbial communities.

Keywords: systems biology, gene and pathway enrichment analysis, graph representation, graph visualization, metagenomics

1 Introduction

Microorganisms abound in every part of the biosphere including soil, hot springs, on the ocean floor, high in the atmosphere, deep inside rocks within the Earth's crust and in human tissues. They are extremely adaptable to conditions where no one else could be able to survive.

Their adaptability is mainly due to the fact that they live in complex communities. Interactions inside the microbial networks plays essential functions for the maintenance and survival of the community. Unfortunately, too little is known about microbial interactions.

With the recent advent of High-Throughput Sequencing (HTS) technologies, metagenomic [1] sequencing approaches have been applied to investigate characterizations of diverse microbial communities, including target sequencing of the phylogenetic marker gene encoding 16S rRNA and whole-metagenome shotgun

[1] Metagenomics is a discipline that enables the study of the (meta)genomes of uncultured microorganisms [5].

sequencing [1]. Additionally, the rapid development of numerous computational tools and methodologies have been explored for effective interpretation and visualization of taxonomic and metabolic profiling of complex microbial communities. Putting into perspective applications in several domains such as agriculture [2], medicine [3] and biomineralization [4].

Despite the large advance in computational technologies for metagenomics analysis there is still a lack of proper tools to highlight the key interactions in microbial communities, and consequently the genes associated to essential metabolic pathways [5]. This task is usually referred as functional analysis of microbial genetic networks and most of the available pipelines deal with a list of microbial genes rather than interactions. Thus, the genomics highlight the “static” view of the genes available in a metagenome, but the interaction as well as the function that will be performed must be evaluated by an enrichment analysis over a proper database of metabolic pathways such as KEEG.

Metagenomics data analysis poses challenges that could be handled by the utilization of Machine Learning (ML) techniques. In fact, ML has been applied succesfully in several genomics problems. In the context of functional analysis it can provide new ways to explore graphs by using robust statistics, dealing with uncertainty in the data and boosting the search for "hot spots" in large microbial genetic networks.

In this work we propose a computational pipeline to evaluate functional enrichment of microbial genetic networks. A weighted graph is built with its basis on the genes and pathways properly induced from the relative abundance of the metabolic pathways enriched by the associated metagenomic data. In addition, non-supervised ML is applied to enumerate network components (clusters) of microbial genes presenting strong evidence of both interaction and functional enrichment.

The main contribution of the proposed strategy are:

– A functional enrichment analysis which takes into account microbial gene interactions;
– A new visual analytics system to explore interactively the enriched metabolic pathways in microbial genetic networks;
– the FUNN-MG *R* pipeline for the identification of network components (clusters) having strong functional enrichment in microbial communities.

2 Metagenomic pathway-centric network analysis

Metagenomic data analysis is a complex analytical tasks in both biological and computational senses. In sequence-based metagenomics, researchers focus on finding the entire genetic sequence, the pattern of the four different nucleotide bases (A, C, G, and T) in the DNA strands found in a sample. The sequence can then be analyzed in many different ways. For instance, researchers can use the sequence to analyze the genome of the community as a whole, which can offer insights about population ecology, evolution and functioning. In this work, we propose the FUNN-MG *pipeline* (Figure 1) which provides a functional and visual analytic system for the identification and exploration of the *key* functions of a microbial community.

The *pipeline* has four main tasks (the rounded rectangles in Figure 1) that must be executed sequentially: i) identification of the metabolic pathways, ii) evaluation of the enriched pathways, iii) detection of strong components (clusters) and iv) visualization of the microbial gene-pathway network. The first three steps are related to the ML part of the strategy while the remaining step deals with the visual analytics of the graph patterns extracted in the previous steps. Next section we discuss each one of these steps, leaving one particular section to the visualization strategy.

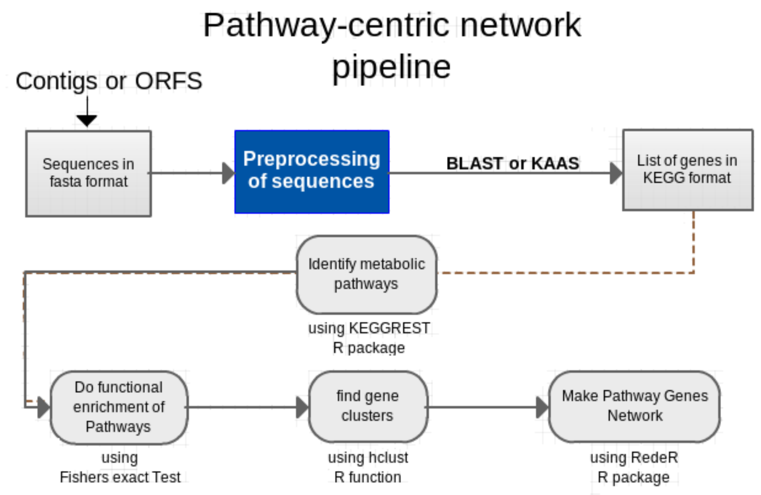

Pathway-centric network pipeline
Contigs or ORFS
Sequences in fasta format
Preprocessing of sequences
BLAST or KAAS
List of genes in KEGG format
Identify metabolic pathways
using KEGGREST R package
Do functional enrichment of Pathways
using Fishers exact Test
find gene clusters
using hclust R function
Make Pathway Genes Network
using RedeR R package

Fig. 1. Metagenomic pathway-centric network pipeline for functional and visual analytics of microbial communities.

3 Materials and Methods

3.1 The metagenomic experimental data

The metagenomic data selected for our experimental study is the Acid Mine Drainage (AMD) biofilm [6], freely available at the site of NCBI [2]. This biofilm sequencing project was designed to explore the distribution and diversity of metabolic pathways in acidophilic biofilms. Acidophilic biofilms are self-sustaining communities that grow in the deep subsurface and receive no significant inputs of fixed carbon or nitrogen from external sources. While some AMD is caused by the oxidization of rocks rich in sulfide minerals, this is a very slow process

[2] http://www.ncbi.nlm.nih.gov/books/NBK6860/

and most AMD is due directly to microbial activity. The AMD metagenome was assembled into 2425 contigs distributed along five main species (see Table 1).

More information regarding the AMD study as well as environmental sequences, metadata and analysis can be obtained at [7].

Species name	Number of contigs
Ferroplasma acidarmanus Type I	412
Ferroplasma sp. Type II	118
Leptospirillum sp. Group II 5-way CG	79
Leptospirillum sp. Group III	959
Thermoplasmatales archaeon Gpl	857

Table 1. The distribution of assembled contigs per species in the AMD metagenome.

3.2 Preprocessing of the metagenomic sequences

We have used the KAAS tool [8] for the identification of 477 microbial genes. This identification was based on the nucleotide percent homology of the groups of orthologous genes [3] found in the KEGG database [9].

The search for microbial genes was carried out in several steps. First, the metagenomic data was split into several groups accordingly to (Table 1), followed by a validation stage of each group within the corresponding species in the KEGG database [7]. KAAS tool was employed sequentially in four steps (Table 2) to obtain the final set of 477 genes:

- Step 1, **finding groups of orthologous genes**: for each specie in the AMD sample we search all its orthologous genes in the KEGG database. For example, the AMD species *Ferroplasma acidarmanus Type I and Type II* are named in KEGG as *Ferroplasma acidarmanus*. So, we use the 530 *contigs* of the associated AMD species as a reference into the KASS tool, retrieving 290 orthologous genes;
- Step 2, **identifying associated species in KEGG**: it basically filters out orthologous genes that are not associated to the reference species. Taking the previous example in Step 1 only 226 genes were kept for the *Ferroplasma acidarmanus* species;
- Step 3, **getting functional annotation in KEGG**: it retrieves the genes associated to pathways in KEGG by using the gene list obtained in Step 2. For instance, 149 genes were retrieved for the *Ferroplasma acidarmanus* specie;
- Step 4, **eliminating duplicated genes**: since pathways are usually associated to one or more genes we deduplicate these genes found in Step 3. So, for the *Ferroplasma acidarmanus* specie we obtained 119 genes.

[3] Orthologous genes are genes in different species that originated by vertical descent from a single gene of the last common ancestor (Homology section on Wikepedia)

All the steps above were executed for all reference species in the AMD sample, taking into account its associated target species in the KEGG database. In (Figure 2) we present this association as well as the distribution of the genes found in the related metagenome.

Id	Species identified	Step 1	Step 2	Step 3	Step 4
fac	*Ferroplasma acidarmanus*	290	226	149	119
lfc	*Leptospirillum ferrooxidans*	450	351	327	217
lfi	*Leptospirillum ferriphilum*	44	33	25	23
tac	*Thermoplasma acidophilum*	26	26	8	8
tar	*Thermoplasmatales archaeon*	412	192	125	107
tvo	*Thermoplasma volcanium*	11	11	3	3
	Genes	1233	839	547	**477**

Table 2. The total number of genes found on each preprocessing step.

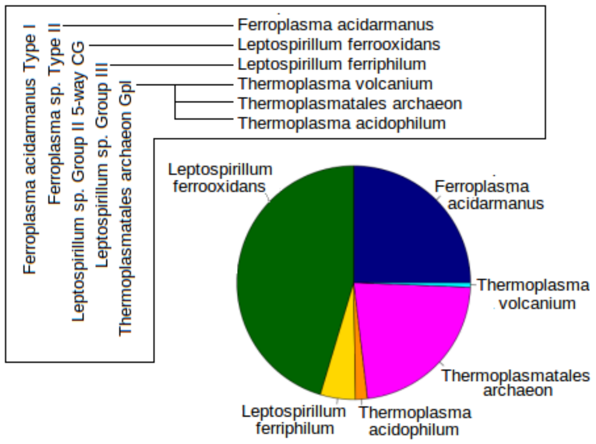


Fig. 2. The dendrogram on the top highlights the association between species in the AMD metagenome and its target species in the KEGG database. In the bottom, a pie chart of the distribution of the 477 genes identified.

3.3 Identifying metabolic pathways

The “*KEGGREST*” *R* package [10] was applied using as reference the list of 477 genes identified, highlighting 95 pathways for the AMD metagenome. Though at this step we cannot assume any strong evidence of functional enrichment regarding to the genes identified.

3.4 Functional enrichment analysis

We devised a functional enrichment strategy based on [11], in which contigency tables are properly set to further apply Fisher's exact test for statistical significance of the enriched metabolic pathways. Fisher's exact test[4] is one of a class of exact tests, so called because the significance of the deviation from a null hypothesis (e.g.: P-value) can be calculated exactly, rather than relying on an approximation that becomes exact in the limit as the sample size grows to infinity, as with many statistical tests.

The main challenge in evaluating the enrichment of a metabolic pathways is the calculation of the probability of finding species covered on each pathway across samples, given that, eventually, only a selected group of species will have an associated pathway. This is also due to the fact that species play distinct roles in the microbial community. As an example, the metabolic pathway *Glutathione metabolism* is annotated for five out of six species identified in the samples (Table 2): *Ferroplasma acidarmanus*, *Leptospirillum ferrooxidans*, *Leptospirillum ferriphilum*, *Thermoplasma acidophilum* e *Thermoplasma volcanium*. So, KEGGREST will only take into account these five species for the enrichment score (Fisher's exact test).

	Gene associated with a pathway	Gene not associated with a pathway	Total gene
Sample	**a** **(6)**	**b** **(364)**	a+b (370)
Population	**c** **(15)**	**d** **(2768)**	c+d (2783)
Total in KEGG	a+c (21)	b+d (3132)	n (3153)

Table 3. The contigency table of the *Glutathione metabolism* pathway which is required for the calculation of the enrichment score.

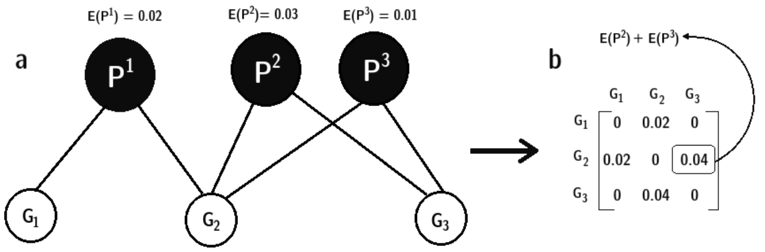
In Table 3 we present the contigency table required to calculate the enrichment of the *Glutathione metabolism* pathway with respect to the microbial genes found in the samples and its corresponding annotations in KEGG. Having this table, we use the *phyper* function in the "*stats*" *R* package for the enrichment score, followed by a test of significance using the *"Firsher's exact test for count data"* *R* package. Finally, we obtained an enrichment score of 0.0077 (p-value = 0.0292) for the the *Glutathionemetabolism* pathway.

After completing the functional analysis for the 95 metabolic pathways, we obtained a list with only 11 enriched pathways (see Table 4) (p-value ≤ 0.05) corresponding to 329 genes. Furthermore, we explore functional modules presenting strong gene interactions by the utilization of a bipartite graph structure $MGP = (G, P, E)$. We called this bipartite graph *Microbial Gene Pathway* (Figure 3. a). MGP vertices are divided into two disjoint sets (G)enes and

[4] http://en.wikipedia.org/wiki/Fishers_exact_test

function	Enrichment	p.value
Purine metabolism	0.033	0.04
Geraniol degradation	6.95e-05	0.01
Cyanoamino acid metabolism	0.008	0.05
Glutathione metabolism	0.007	0.02
Porphyrin and chlorophyll metabolism	0.023	0.03
Metabolic pathways	0.0002	0.0003
Microbial metabolism in diverse environments	0.042	0.05
Carbon metabolism	0.039	0.05
Biosynthesis of amino acids	0.017	0.02
RNA degradation	0.01	0.03
Nucleotide excision repair	0.01	0.03

Table 4. The eleven most significant enriched pathways.



$$\begin{array}{c|ccc} & G_1 & G_2 & G_3 \\ \hline G_1 & 0 & 0.02 & 0 \\ G_2 & 0.02 & 0 & \boxed{0.04} \\ G_3 & 0 & 0.04 & 0 \end{array}$$

Fig. 3. a) The *MGP* bipartite graph with (*G*)enes and (*P*)athways . b) the associated community matrix with the gene-to-gene interaction augmented with the enrichment score.

(*P*)athways, such that every edge (*E*) connects a vertex in (*G*) to one in (*P*). The enrichment score is annotated in the vertice (*P*).

3.5 Finding gene clusters

Several groups of genes interact in microbial communities, and some of these interaction are stronger than others. In addition, these interactions usually correlate to the environment in which they are living. We called these strong gene interactions *community patterns*, and potentially they may play a key role in the stability of the microbial genetic network. We have a hypotheses that any perturbation in such patterns could impact directly in the maintenance of the network. We propose a *structural graph clustering* strategy which takes into account a bipartite graph (*MGP*).

The structural graph clustering uses a community matrix (Figure 3.b) based on the genes and its enriched pathways represented in *MGP*. The community matrix observes three main aspects regarding gene-to-gene interactions:

- The existence of one or more metabolic pathways shared by the genes;
- The amount of metabolic pathways in which genes play;

– The enrichment score associated to each metabolic pathway.

The *MGP* bi-partite graph is an interesting computational structure for both the application of ML techniques and interactive visualization of the microbial genetic network [12]. The *community patterns* are obtained directly through the utilization of a hierarchical clustering (*hclust()* R function) technique over the community matrix. The hierarchical clustering solution (Figure 4) requires an

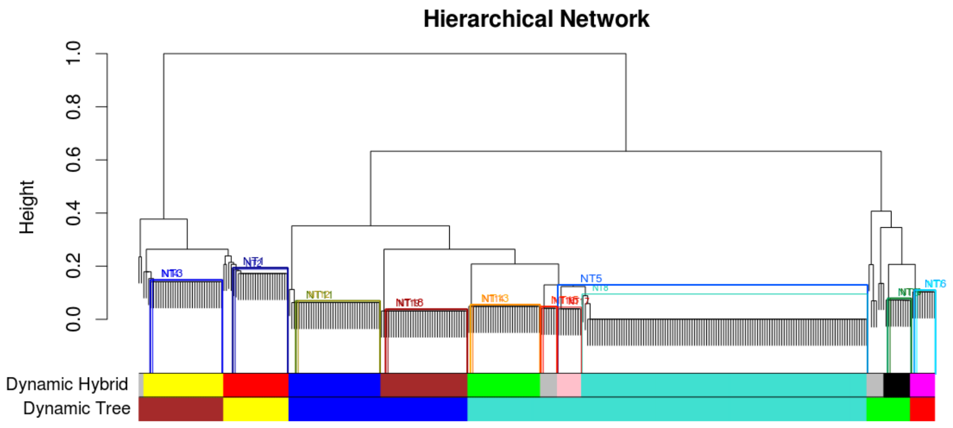


Fig. 4. The hierarchical clustering solution of the community matrix. Two clustering solutions are calculated i) Dynamic Hybrid and ii) Dynamic Tree. Each solution takes into account a distinct cut scheme to form clusters (colored).

euclidean distance matrix that can be built directly through the community matrix. From a biological perspective, the identification of these strong interactions allows for a better understanding of the mechanisms by which these complex networks control cell processes, making it possible to interfere in such processes [13].

The branches of the hierarchical clustering dendrogram correspond to *community patterns* and can be identified using one of a number of available branch cutting methods, for example the constant-height cut or two Dynamic Branch Cut methods. One drawback of hierarchical clustering is that it can be difficult to determine how many (if any) clusters are present in the data set. We employed the *Dynamic Tree Cut R* package to obtain robust clusters [14]. Although the height and shape parameters of the Dynamic Tree Cut method provides improved exibility for branch cutting and module detection, it remains an open research question how to choose optimal cutting parameters or how to estimate the number of clusters in the data set. Two cutting strategies were explored with the *Dynamic Tree Cut*:

– Dynamic tree: the algorithm implements an adaptive, iterative process of cluster decomposition and combination and stops when the number of clusters becomes stable. To avoid over-splitting, very small clusters are joined to their neighboring major clusters;

– Dynamic hybrid: the algorithm can be considered a hybrid of hierarchical clustering and modified Partitioning Around Medoids (PAM), since it involves assigning objects to their closest medoids.

Given that we were looking for compact clusters we decided to use the cutting result obtained with the Dynamic hybrid approach. Thus, 9 clusters and 10 nested subclusters were enumerated. All clusters have the prefix “NT" followed by a sequential number (Table 5). The nested subclusters were calculated with the guide of the *RedeR R* package, and it offered an interesting alternative for the interactive visualization of the microbial genetic networks.

In summary, 308 genes were clustered, corresponding to 96.61% of the enriched pathways related to AMD biofilm. These clusters enclose on average 30 genes, having 6 genes in the most compact cluster and 128 in the largest one. Next, we explore the visual analytic systems over the *MGP* bipartite graph allowing free manipulation of the community patterns as well as the exploration of key hub genes and pathways inside this microbial network.

4 Results and discussion

4.1 Visual analytics system

Given the linked information associated with the concept of microbial communities, it is strongly advised to explore it by graph visualization [15]. The *MGP* bipartite graph fits properly the graph structure required for visualization by the *RedeR R* package. This network visualization system allows several interactive and graph functions such as: zoom, pan, neighborhood highlighting, search, flows, labeling, addition and deletion of graph components.

The structural visualization of the enriched *Microbial Gene Pathway* is presented in Figure 5. The visualization model allows the identification of genes across species and pathways, depicted in distinct colors. It is also possible to explore the degree of connectivity by inspecting the size of the vertices; key players are identified by neighborhood highlighting while clicking on a particular node in the graph network. Such interactive experience allows one to explore resilient aspects of the enriched microbial gene pathway.

The community patterns are explored through the visualization of the graph components associated with the clusters and subclusters (Figure 6). Furthermore, it is also possible to inspect particular spots as well as identify either hub genes, modules or pathways within the network. As an example, the modules are explored as (nested) clusters detected by the proposed pipeline. The Subgroup row in Table 5 identifies these nested clusters. Thus, if one looks to the Group “NT2” we observe a total of 22 genes distributed along the six species (The headers previously described above). NT1 is an example of nested cluster having 1 gene plus 22 genes from NT2, summing up to a total of 23 genes. The symbol “–” shows the there is no nested cluster for that Group.

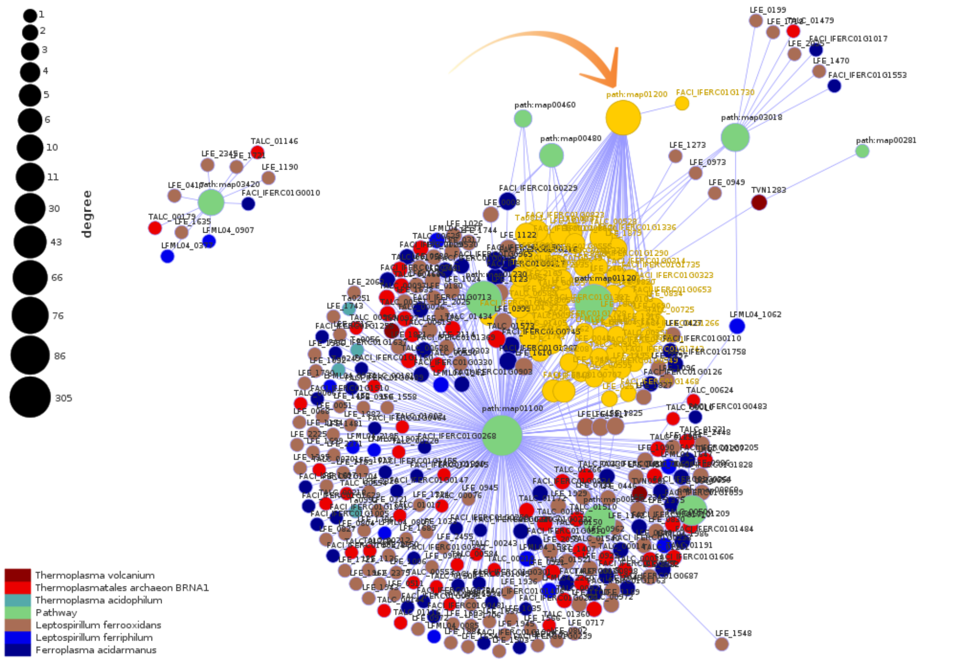


Fig. 5. The enriched Microbial Gene Pathway Network. At the bottom left the legend of the species and associated pathways are represented. Nodes (circles) are related to either species genes or pathways. At the upper left the degree connectivity scale of all nodes. The nodes in highlighting (yellow) are all genes associated to the *Carbon metabolism* pathway (direct orange arrow).

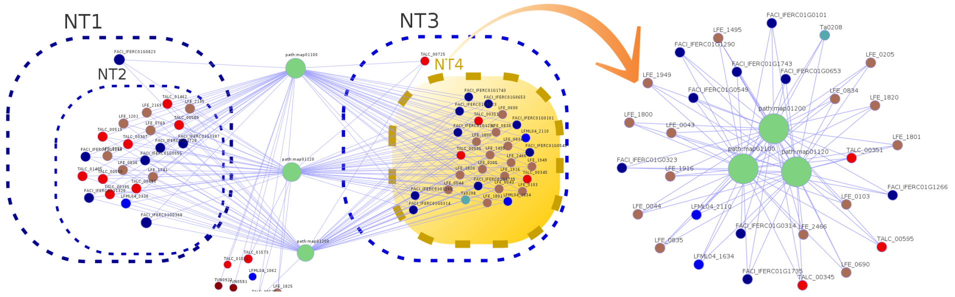


Fig. 6. The representation of the community patterns as clusters (NT1, NT3) and sub-clusters (NT2, NT4). At the right the expanded subnetwork corresponding to elements clustered in NT4.

Group name	NT1	NT2	NT3	NT4	NT5	NT6	NT7	NT8	NT9	NT10
Subgroup	NT2	—	NT4	—	NT17; NT8	NT7	—	—	NT10	—
fac	1	6	0	9	0	0	5	29	0	3
lfc	0	7	0	14	0	1	3	56	1	5
lfi	0	1	0	2	0	0	0	5	0	0
tac	0	0	0	1	0	0	0	4	0	0
tar	0	8	1	3	0	0	0	24	0	0
tvo	0	0	0	0	0	0	0	0	0	1
Total of genes	23	22	30	29	128	9	8	118	10	9

Group name	NT11	NT12	NT13	NT14	NT15	NT16	NT17	NT18	NT19	Total
Subgroup	NT12	—	NT14	—	NT16	—	—	NT19	—	10
fac	0	9	0	7	0	2	1	0	8	80
lfc	0	13	0	12	0	4	5	0	15	136
lfi	0	2	0	2	0	0	2	0	2	16
tac	0	0	0	0	0	0	0	0	0	5
tar	1	10	1	7	1	0	5	1	8	70
tvo	0	0	0	0	0	0	0	0	0	1
Total of genes	35	34	29	28	7	6	10	34	33	—

Table 5. Number of species genes associated to each cluster (Group) and subcluster (Subgroup) calculated by the Dynamic Hybrid cutting strategy.

As an illustration of the visualization, the nested cluster "NT3" having 30 genes is depicted in the middle of Figure 6. As it can be observed the nested cluster "NT3" has 1 gene plus the 29 genes (from "NT4"). The most abundant specie is the "lfc" (colored in brown). Finally, it is presented the eleven enriched pathways (colored in green) connecting all the enumerated nested clusters.

5 Conclusions

The enrichment analysis of microbial genetic networks poses an interesting computational challenge. It is not practical to enumerate all gene-to-gene interaction of a microbial community, so the pathway-centric analysis sound a promising strategy to smooth this combinatorial problem. This strategy has it basis on non-supervised machine learning over a bipartite graph properly built to evaluate the enriched microbial gene pathways.

Interactive visualization of the resulting microbial gene pathway networks allows for the exploration of network metrics enhancing the enrichment analysis. Once all the topological network aspects are understood for a particular metagenome, we envisage the possibility of using such profiles for metagenome comparison as well as classification of unknown microbial genetic network.

Author's contributions

LC and RA performed the analysis and developped the pipeline. RA and CC supervised the study. LC, RA, CC and LT wrote the manuscript.

Acknowledgements

This work is partially supported by the Brazilian National Research Council (CNPq – *Universal calls*) under the BIOFLOWS project [475620/2012-7].

References

1. Hugenholtz, P., Tyson, G.W.: Microbiology: Metagenomics. Nature **455**(7212) (September 2008) 481–483
2. Fierer, N., Leff, J.W., Adams, B.J., Nielsen, U.N., Bates, S.T., Lauber, C.L., Owens, S., Gilbert, J.A., Wall, D.H., Caporaso, J.G.: Cross-biome metagenomic analyses of soil microbial communities and their functional attributes. Proceedings of the National Academy of Sciences **109**(52) (December 2012) 21390–21395
3. Bäckhed, F., Ley, R.E., Sonnenburg, J.L., Peterson, D.A., Gordon, J.I.: Host-Bacterial Mutualism in the Human Intestine. Science **307**(5717) (March 2005) 1915–1920
4. Johnston, C.W., Wyatt, M.A., Li, X., Ibrahim, A., Shuster, J., Southam, G., Magarvey, N.A.: Gold biomineralization by a metallophore from a gold-associated microbe. Nat Chem Biol **advance online publication** (February 2013)
5. Wooley, J.C., Godzik, A., Friedberg, I.: A Primer on Metagenomics. PLoS Comput Biol **6**(2) (February 2010) e1000667+
6. NCBI: Metagenomics: Sequences from the environment [internet]. Sequences from the Environment, Tyson (2013)
7. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., Banfield, J.F.: Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature **428**(6978) (March 2004) 37–43
8. Moriya, Y., Itoh, M., Okuda, S., Yoshizawa, A.C., Kanehisa, M.: KAAS: an automatic genome annotation and pathway reconstruction server. Nucleic acids research **35**(Web Server issue) (July 2007) W182–W185
9. Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., Tanabe, M.: KEGG for integration and interpretation of large-scale molecular data sets. Nucleic acids research **40**(Database issue) (January 2012) D109–D114
10. Tenenbaum, D.: KEGGREST: Client-side REST access to KEGG. R package version 1.0.1.
11. Sreenivasaiah, P.K.K., Rani, S., Cayetano, J., Arul, N., Kim, D.H.o..H.: IPAVS: Integrated Pathway Resources, Analysis and Visualization System. Nucleic acids research **40**(Database issue) (January 2012) D803–D808
12. Goh, K.I., Cusick, M.E., Valle, D., Childs, B., Vidal, M., Barabási, A.L.: The human disease network. Proceedings of the National Academy of Sciences **104**(21) (May 2007) 8685–8690
13. A.L. LEHNINGER, N., D.L: Principios da bioquímica. 5 edn. Volume 1. (2005)
14. Langfelder, P., Zhang, B., Horvath, S.: Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. Bioinformatics **24**(5) (2008) 719–720
15. Herman, I., Melancon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. Visualization and Computer Graphics, IEEE Transactions on **6**(1) (January 2000) 24–43

RankMerging: Learning to rank in large-scale social networks

Lionel Tabourier[1,2], Anne-Sophie Libert[1], and Renaud Lambiotte[1]

[1] naXys, University of Namur, 8 Rempart de la Vierge, 5000 Namur, Belgium
[2] LIP6, University Pierre and Marie Curie, 4 Place Jussieu, 75252 Paris, France

Abstract In this work, we consider the issue of unveiling unknown links in a social network, one of the difficulties of this problem being the small number of unobserved links in comparison of the total number of pairs of nodes. We define a simple supervised learning-to-rank framework, called *RankMerging*, which aims at combining information provided by various unsupervised rankings. As an illustration, we apply the method to the case of a cell phone service provider, which uses the network among its contractors as a learning set to discover links existing among users of its competitors. We show that our method substantially improves the performance of unsupervised metrics of classification. Finally, we discuss how it can be used with additional sources of data, including temporal or semantic information.

1 Introduction

Link prediction is a key field of research for the mining and analysis of large-scale social networks. The reasons lie in its many practical applications: going from recommendation strategies for commercial websites [20] to recovering missing links in incomplete data [31]. Link prediction also has significant implications from a fundamental point of view, as it allows for the identification of the elementary mechanisms behind the creation and decay of links in time-evolving networks [17]. For example, triadic closure, at the core of standard methods of link prediction, is considered as one of the driving forces for the creation of links in social networks [16].

In their seminal formulation, Liben-Nowell and Kleinberg [18] present link prediction as follows: considering a snapshot of a network at time t, the problem is to predict which links will be present at a future time $t' > t$. A standard way to solve this binary classification problem consists in ranking pairs of nodes according to a scalar metric, correlated with the existence of interactions between nodes. Most of these metrics are based on the structural properties of the network of known interactions, either on local, e.g. the number of common neighbors, or on global features, e.g. random walk or hitting time — see [22] for a survey. Other sources of information can be considered, in particular node attributes, such as age or gender [23,2] or geographic location [28]. In certain cases, for example online networks, the users profiles may include rich semantic information as [4].

Interaction attributes, such as frequencies [29], or the time elapsed since the last interaction [27], may also be used as an additional source of information.

However, these ranking features are known to be domain-specific (e.g., [8]). Moreover, as links can play different roles in social networks, they are expected to be surrounded by different types of environments and thus to be best identified by different topological features. For these reasons, schemes based on a single metric are prone to misclassification. A way to circumvent this problem is to combine different metrics for link prediction. Most of the available solutions are unsupervised, such as Borda's method or Markov chain ordering [11]. On the other hand, supervised methods have proven efficient on specific link prediction problems. For instance, the authors of [19] considered degree categories to predict future links in a phonecall network; in both [14] and [8], supervised frameworks are defined to predict links in multimodal networks. Recent works have combined classic tools, such as classification trees, support vector machine, or neural networks, which have been shown to outperform their unsupervised counterparts for predicting links in biological networks and scientific collaboration networks [3,26,8]. However, these methods do not allow the user to set the number of predictions according to his needs, whereas ranking methods are suited to this purpose. Supervised methods built to improve ranking systems have mostly been designed in the context of information retrieval tasks, such as document filtering, spam webpage detection, recommendation or text summarization [21,30]. As such, they primarily aim at high precision on the top-ranked items, and stress the relative ranking of two items [12,5,6].

Here, we propose a simple yet efficient *learning-to-rank* supervised framework specifically designed to uncover links in social networks, by combining rankings obtained from different sources of information. Throughout this article, our Ariadne's thread is the examle of a mobile phone service provider (PSP). Because PSPs only have access to records involving their own clients, they have an incomplete view on the social network as a whole. In several practical applications, however, this missing information is crucial. An important example is churn prediction, that is the detection of clients at risk of leaving to another PSP. This risk is known to depend on the local structure of the social network [7,25]. Therefore, the knowledge of connections between subscribers of other providers is crucial for the design of efficient customer retention campaigns. As we discuss further below, a direct application of our method is the prediction of links on the other side of the frontier accessible to a PSP.

This article is organized as follows. In Section 2, we describe the mobile phone dataset and how it is processed in order to model the situation of a PSP confronted to churn. In Section 3, we briefly present how classic unsupervised learning methods can be applied to the problem under consideration. In Section 4, we develop a supervised machine learning framework, called *RankMerging*, which improves the quality of predictions by aggregating the information from various metrics. We also compare our results to those of standard supervised methods, and show that *RankMerging* is particularly suited to social networks where information is partial and noisy.

2 Dataset

The dataset is a call detail record of approximately $14 \cdot 10^6$ phonecalls of anonymized subscribers of a European PSP during a one month period. Users are described as nodes and an interaction between two users is a directed link. The total number of phone calls between nodes i and j is denoted by the *weight* $w(i, j)$ of this link. In order to filter out calls which are not indicative of a lasting social relationship, we only consider calls on bidirectional links, i.e. links that have been activated in both directions. After this filtering has been applied, interactions between users are considered as undirected. The resulting social network is composed of 1,131,049 nodes, 795,865 links and 10,934,277 calls. From now on, we denote W the activity of a node, that is the sum of the weights of its adjacent links.

A PSP is usually confronted to the following situation: it has full access to the details of phone calls between its subscribers, as well as between one of its subscriber and a subscriber of another PSP. However, connections between subscribers of other PSPs are hidden. In order to simulate this situation from our dataset, we divide our data into two artificial PSPs, A and B. The link to predict are selected in a random way: we split the users of the network into 3 sets (i) A_1 and A_2 nodes are nodes of the first PSP – links among A_2 nodes are links to discover during the learning task, (ii) B nodes belong to the second PSP. Users have been randomly assigned to A_1, A_2 and B according to the proportions 50, 25, 25%. During the learning process, links inside $A_2 \cup B$ are removed. The resulting network ($\mathcal{G}_{Learn}$) contains 848,911 nodes and 597,538 links and we aim at predicting the links among A_2 nodes (49,731 links). During the test phase, the network $\mathcal{G}_{Test}$ contains 1,131,049 nodes and 746,202 links. All the links are thus considered, except for the 49,663 links inside B which we aim at predicting. Notice that according to our simulation, the learning set and the test set are derived from the same distributions, while it could not be the case in situations involving several PSPs. However, this assumption is fair to compare the performances of our method to other prediction techniques.

3 Unsupervised learning

3.1 Prediction evaluation

The quality of a ranking metric is assessed by measuring precision (**Pr**), recall (**Rc**) and **F**-score. Previous works have emphasized the dramatic effect of class imbalance (or skewness) on link prediction problems in social networks, especially in mobile phone networks [19,6]. The fact that the network is sparse and that there are many more pairs of nodes than links makes the prediction and its evaluation tricky, the typical order of magnitude of the classes ratio for a social network made of N nodes being $O(1/N)$ [19]. In the case of unbalanced classes, performance predictors such as the ROC curve are known to be inappropriate – see [10]. In general, the definition of a *good* prediction and thus of an adequate quality estimator depends on the purpose of the link prediction. For example, the goal of an efficient search engine is to provide highly relevant information on a small amount of items. In contrast, a PSP confronted to churn looks for an appropriate trade-off between precision and recall, in order to detect potential churners, without flooding clients with discount offers. For this reason, we aim at improving both precision and recall over a large range so that a PSP would be able to tune the prediction parameters according to its commercial strategy.

3.2 Ranking metrics and results

In this work, we focus on structural features where each pair of nodes is assigned a score based on topological information, and then ranked according to this score. A large number of metrics have been used in the past, see for example [18,22]. The goal of this paper is not to propose elaborate classifiers, but to present a method that takes advantage of how complementary they are. We have therefore chosen classic metrics and generalized them to the case of weighted networks (other generalizations exist in the literature, e.g. [24]), denoted by index w. Their unweighted version is recovered by setting all weights to 1.

Local features. A class of metrics are local (also called *neighborhood rankers*) as they only rank links among nodes which are at most at distance 2. In the following, $\mathcal{N}(i)$ denotes the set of neighbors of node i.

- *Common Neighbors index (CN)*, based on the weighted number of common neighbors shared by nodes i and j:

$$s_{CN_w}(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} w(i,k).w(j,k)$$

- *Jaccard index (Jacc)*:

$$s_{Jacc_w}(i,j) = \frac{\sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} w(i,k) + w(j,k)}{W(i) + W(j)}$$

- *Adamic-Adar index (AA)*:

$$s_{AA_w}(i,j) = \sum_{k \in \mathcal{N}(i) \cap \mathcal{N}(j)} \frac{1}{log(W(k))}$$

Global features. Another class of features are global (or *path rankers*), since they are calculated by using the whole structure of the network, and allow for the ranking of distant pairs of nodes:

- *Katz index (Katz)* [15], computed from the number of paths from node i to node j of length l, i.e. $\nu_{ij}(l)$ according to the following expression (β is a bounded parameter):

$$s_{Katz}(i,j) = \sum_{l=1}^{\infty} \beta^l \nu_{ij}(l)$$

  Note that in the weighted case, the number of paths is computed as if links were multilinks.

- *Random Walk with Restart index (RWR)*, $s_{RWR_w}(i,j)$ is defined as the probability that a random walker starting on node i, going from a node k to a node k' with probability $p.w(k,k')/W(k)$ and returning on i with probability $1-p$, is on j in the steady state of the process.
- *Preferential Attachment index (PA)*, based on the observation that active nodes tend to connect preferentially in social networks [13]:

$$s_{PA_w}(i,j) = W(i).W(j)$$

Both *Katz* and *RWR* are computed using infinite sums, which will be approximated by keeping only the dominating four first terms to reduce the computational cost, meaning that we can only predict links between pairs of nodes at a maximum distance of 4. Moreover, in order to address the problem of class imbalance, known to hinder the performance of PA (e.g. [19]), we have restricted the ranking in this case to pairs of nodes at a maximum distance of 3. Notice that with larger maximum distances, we can increase the maximum recall that can be reached, but at the cost of a very low precision for these predictions.

3.3 Borda's method

The main purpose of this work is to develop a framework to exploit a set of α rankings for link prediction. As we will show in the next section, this problem can be solved in a supervised setting by identifying regions of a α-dimensional space associated to a high performance. Here, we present an unsupervised way to merge ranking features based on social choice theory [11,26]. Borda's method is a *rank-then-combine* method originally proposed to obtain a consensus from a voting system [9]. Each pair is given a score corresponding to the sum of the number of pairs ranked below, that is to say:

$$s_B(i,j) = \sum_{\kappa=1}^{\alpha} |r_\kappa| - r_\kappa(i,j)$$

where $|r_\kappa|$ denotes the number of elements ranked in r_κ. This scoring system may be biased toward global predictors by the fact that local rankings have less elements. To alleviate this problem, unranked pairs in ranking r_κ but ranked in $r_{\kappa'}$ will be considered as ranked in r_κ on an equal footing as any other unranked pair and below all ranked pairs. Borda's method is computationally very cheap, which is a highly desirable property in our case. A comprehensive discussion on this method can be found in [11].

3.4 Results

We plot the results obtained on $\mathcal{G}_{Learn}$ to predict $A_2 - A_2$ links for the above classifiers. For the sake of readability, we only represent a selection of them on Figure 1. The evolution of the **F**-score significantly varies from one classifier to another. For example, it increases sharply for *CN*, and then more slowly until reaching its maximum, while RWR_w rises smoothly before slowly declining. As expected, Borda's aggregation improves the performance of the classification, especially considering the precision on the top-ranked pairs. Given the difficulty of the task, **Pr** is low on average: for instance, when **Rc** is greater than 0.06, **Pr** is lower than 0.3 for all estimators.

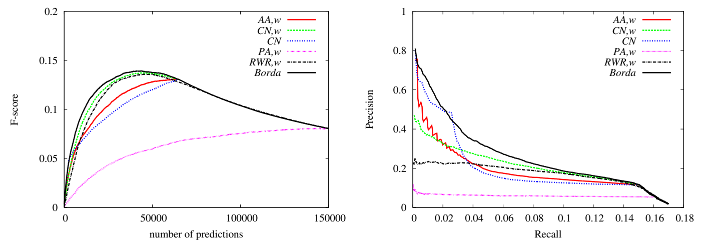


Figure 1. Results on the learning set for various structural classifiers. Left: **F**-score as a function of the number of predictions. Right: **Pr** versus **Rc** curves.

4 *RankMerging* framework

The ranking methods presented in the former section use structural information in complementary ways. In systems such as social networks, where communication patterns of different groups, e.g. family, friends, coworkers etc, are different, one expects that a link detected as likely by using a specific ranking method may not be discovered using another one. This intuition is supported by measuring the correlation between rankings. For example, the rankings produced with s_{CN} and $s_{Jacc,w}$ scores have a 0.052 Spearman correlation coefficient on the test set, which means that some pairs are highly ranked according to one method, but not to the other one. In this section, we design a supervised machine learning framework to aggregate information from various ranking techniques for link prediction in social networks. In a nutshell, it does not demand for a pair to be highly ranked according to all criteria (like with a consensus rule), but at least one. The whole procedure is referred to as *RankMerging*. An implementation and user guide of the algorithm are available on `http://lioneltabourier.fr/program.html`.

4.1 Principle

We first consider the training set to learn the parameters. Let's suppose that we have α different rankings $r_1, ..., r_\alpha$ from the unsupervised learning phase. Each ranked pair is either a true or false positive prediction (tp or fp). We are looking for the combination of rankings that brings the maximum number of tp predictions. Let $\Sigma_i(\rho_i)$ be the number of tp for pairs ranked from position 1 to ρ_i, ρ_i will be named *sliding index* of r_i. Now if we consider simultaneously the α rankings, our goal is to compute the optimal values of the ρ_i for a fixed number of predictions n so that the total number of *tp*, denoted $\mathcal{S}(\rho_1, ..., \rho_\alpha)$, is maximum. It is important to note that a link can only be predicted once: if a pair has been predicted using ranking r_i and then included in the solution associated to $\mathcal{S}$, it cannot be predicted by r_j. It means that we have to consider the links already predicted to guess the future ones, which makes this problem hard to solve exactly. For this reason, the problem is solved heuristically, by using the training algorithm described in Algorithm 1.

The central idea is to find at each step the ranking with the highest number of true predictions in the next g coming steps. For that purpose, we define the *window* $\mathcal{W}_i$ as the set of links predicted according to ranking r_i in the next g steps. Note that links already predicted are not considered in $\mathcal{W}_i$. The number of *tp* in $\mathcal{W}_i$ is its *quality*, denoted $\chi(i)$. The ranking corresponding to the highest quality value is selected (in the case of a tie, we choose randomly), and we add its next top-ranked pair to the output ranking of the learning phase r_M^L. Throughout the process, the sliding indices ρ_i are registered, these values are the essential outputs of the training phase, as they record which ranking contributed to the merged ranking. Then, the indices σ_i which indicate the end of the windows are updated so that $\mathcal{W}_i$ contain exactly g pairs, $\chi(i)$ are updated too, and the process is iterated until r_M^L contains a predefined number of pairs T. To summarize, we are looking for a maximum number of true predictions $\mathcal{S}$ by local search. An important benefit of our learning algorithm is that it needs to go through each ranking only once, so if we have α rankings, it implies a $O(\alpha N)$ temporal complexity. A brute force algorithm demands to consider all possible rankings combinations, meaning that α different possibilities for each prediction must be considered, that is $O(\alpha^N)$ complexity.

Table 1 gives an example of the two first steps of the merging process between two rankings r_A and r_B with $g = 5$. Pairs in the windows are represented with a gray background. Initially, there are 4 *tp* in $\mathcal{W}_A$ and 3 *tp* in $\mathcal{W}_B$, it means that $\chi_A > \chi_B$, so the first link selected is the top-ranked pair available in $\mathcal{W}_A$: $(1, 2)$ (green background). This pair is therefore excluded from the ranking r_B (barred item on red background). At the next step, we have $\chi_A = \chi_B = 4$, the ranking with highest quality is then selected randomly, we suppose here that r_B has been selected so that the next link included in r_M^L is $(5, 18)$. At this step, according to the previously defined notations, $\rho_1 = 1$ and $\Sigma_1(\rho_1) = 0$, while $\rho_2 = 1$ and $\Sigma_2(\rho_2) = 1$.

The test phase of the procedure consists in combining rankings on the test network $\mathcal{G}_{Test}$ according to the ρ_i learned on the training network $\mathcal{G}_{Learn}$. This process does not demand to define sliding windows. The practical implementation is simple: at each step, we look up for the ranking chosen according to the learning process and select the corresponding ranking r_i on the test set. Its highest ranked pair is then added to the merged ranking of the test phase r_M^T if it has not been included previously; if it is already in r_M^T, then we go to the next highest ranked pair of r_i until we have found one which is not in r_M^T. The number of pairs that can be predicted is different in $\mathcal{G}_{Learn}$ and $\mathcal{G}_{Test}$ (resp. n_L and n_T). The implementation should then take into account a scaling factor:

Algorithm 1: *RankMerging* method: training algorithm.

```
inputs  : table of rankings $\mathcal{R}$ ($r_i = \mathcal{R}[i]$); real edge list $E$;
          maximum number of predictions $T$; $g$;
outputs: sliding index table $\rho$; merged ranking $r_M^L$;
// initialization:
begin
    $\mathcal{W}[i] \leftarrow g$ first links in $\mathcal{R}[i]$;              // pairs in window $i$
    $\chi[i] \leftarrow |\mathcal{W}[i] \cap E|$;                             // quality of window $i$
    $\rho[i] \leftarrow 0$;        // sliding index of $r_i$ = start index of window $i$
    $\sigma[i] \leftarrow g$;                                              // end index of window $i$
    $n \leftarrow 0$;                           // counter of the number of predictions
while n ≤ T do
    $i_{max} \leftarrow$ index corresponding to maximum $\chi[i]$;
    $r_M^L[n] \leftarrow \mathcal{R}[i_{max}][\rho[i_{max}]]$;
    n ← n+1;
    $\rho[i_{max}] \leftarrow \rho[i_{max}] + 1$;
    $\forall i$ Update($\mathcal{W}[i], \chi[i], \rho[i], \sigma[i], r_M^L[n]$);
Procedure: Update($\mathcal{W}[i], \chi[i], \rho[i], \sigma[i], r_M^L[n]$):
begin
    if $r_M^L[n] \in \mathcal{W}[i]$ then
        $\mathcal{W}[i] \leftarrow \mathcal{W}[i] \setminus r_M^L[n]$
    while $|\mathcal{W}[i]| \leq g$ do
        $l \leftarrow \mathcal{R}[i][\sigma[i]]$;
        $\sigma[i] \leftarrow \sigma[i] + 1$;
        if $l \notin r_M^L$ then
            $\mathcal{W}[i] \leftarrow \mathcal{W}[i] + l$;
    $\chi[i] \leftarrow |\mathcal{W}[i] \cap E|$;
```

r_A	tp	r_B	tp
(1,2)		(5,18)	x
(1,4)	x	(1,2)	
(5,6)	x	(8,9)	
(6,12)	x	(5,6)	x
(5,18)	x	(7,11)	x
(3,4)		(6,9)	x
(4,9)	x	(1,14)	
(7,11)	x	(2,9)	

$\longrightarrow$

r_A	tp	r_B	tp
(1,2)		(5,18)	x
(1,4)	x	~~(1,2)~~	
(5,6)	x	(8,9)	
(6,12)	x	(5,6)	x
(5,18)	x	(7,11)	x
(3,4)		(6,9)	x
(4,9)	x	(1,14)	
(7,11)	x	(2,9)	

$\longrightarrow$

r_A	tp	r_B	tp
(1,2)		(5,18)	x
(1,4)	x	~~(1,2)~~	
(5,6)	x	(8,9)	
(6,12)	x	(5,6)	x
~~(5,18)~~	x	(7,11)	x
(3,4)		(6,9)	x
(4,9)	x	(1,14)	
(7,11)	x	(2,9)	

Table 1: Two steps illustrating the merging algorithm with rankings r_A and r_B ($g = 5$). Pairs predicted (i.e. $\in r_M^L$) have green backgrounds. Pairs with gray backgrounds are in the windows $\mathcal{W}_A$ and $\mathcal{W}_B$. Barred pairs with red backgrounds have already been predicted and cannot be selected anymore.

if ranking r_i has been selected at step s on the learning set, then r_i should be selected at step $\left\lfloor s \cdot \frac{n_T}{n_L} \right\rfloor$ on the test set.

4.2 Benchmarks for comparison

In order to assess the efficiency of *RankMerging*, we compare its performances to existing techniques. The first one is Borda's method, introduced above. We also consider classic supervised techniques. As we aim at handling large network datasets, we restricted ourselves to computationally efficient ones, namely nearest neighbors (*NN*), classification trees (*CT*) and *AdaBoost* (*AB*). We have used implementations from Python scikit learn toolkit[1]. These techniques are not specifically designed for ranking tasks, so we obtain different points in the precision-recall space by playing with the algorithms parameters, respectively the number of neighbors (*NN*), the minimum size of a leaf (*CT*), and the number of trees (*AB*).

4.3 Protocol and results

Protocol. According to the description in 4.1, ρ_i are computed on $\mathcal{G}_{Learn}$ to discover links among A_2 nodes, and then used to merge rankings on $\mathcal{G}_{Test}$ to discover links between B nodes, applying the scaling factor $n_T/n_L \approx 1.5$ to adapt the ρ_i learnt to the test set and cross-validation is therefore made through a simple hold-out strategy. Determining adequate parameters is not an issue here. The user may indeed aggregate as many rankings as he wants: the merging process is such that the addition of a supplementary ranking is computationally cheap[2], and if a ranking does not bring additional information, it will simply be ignored during the learning process. This property helps *RankMerging* to avoid strong overfitting effects. As of the value of g, our numerical experiments show that the performance of the algorithm is robust over a large range of values (see Table 2), and we extrapolate the best g value on $\mathcal{G}_{Learn}$ for the aggregation on $\mathcal{G}_{Test}$.

Results. We plot on Figure 2 the evolution of the **F**-score and the precision-recall curve obtained with *RankMerging*, for $g = 200$, aggregating the rankings of the following classifiers: AA_w, CN_w, CN, $Jacc_w$, $Katz_w$ ($\beta = 0.01$), PA_w, RWR_w ($p = 0.8$) and Borda's method applied to the seven former ones. We observe that *RankMerging* performs better than Borda, especially for intermediary recall values. This was expected, as *RankMerging* incorporates the information of Borda's aggregation here. In fact, the method has been designed so that *any* unsupervised ranking can be aggregated without any performance loss, so that it should outperforms any unsupervised method taken into account during the learning phase. We measure the area under the **Pr**-**Rc** curves to quantify the performances with a scalar quantity. *RankMerging* increases the area by 8.3% compared to Borda. Concerning the supervised benchmarks, we observe that they perform well, but only for a low number of predictions (comparable to Borda for approximately 1000 to 2000 predictions). Unsurprisingly, *AdaBoost* is

[1] `http://scikit-learn.org/`

[2] Here, the running time is a few seconds on a standard personal computer.

an ensemble method and outperforms *Decision Trees* and *Nearest Neighbors* for an optimal parameter choice, but the performances are of the same order of magnitude, in line with the observations in [1]. As formerly stated, these methods are not designed to cover a wide range of the precision-recall space, and therefore perform very poorly out of their optimal region of use. On the minus side, *RankMerging* has been designed for classification problems with large number of predictions. The window size g implies an averaging effect which causes the method to lack efficiency on the top-ranked items, as can be seen on Fig. 2. As a consequence, it is not suited to problems with low number of predictions, as it is often the case for information retrieval tasks for example.

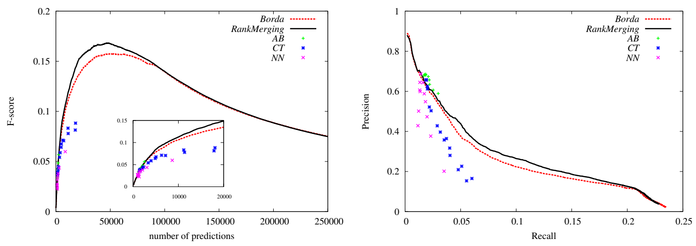


Figure 2. Results of *RankMerging* on the test set ($g = 200$), compared to benchmarks. Left: **F**-score as a function of the number of predictions. Right: **Pr** versus **Rc** curves.

We evaluate the influence of the structural metrics in Table 2. As can be seen, the addition of a ranking does not decrease the quality of the merging process – except for small variations which are considered as statistical fluctuations. A user may therefore aggregate any ranking whatever the source of information is. The dependency on the value of g is also shown in Table 2, and results indicate that the performances are close to the maximum within the interval $[100; 300]$ on both the learning and test sets. This observation suggests the possibility of tuning g in the testing phase from the values of g during the learning process.

AA_w	CN_w	CN	$Jacc_w$	$Katz_w$	PA_w	RWR_w	$Borda$	imp.(%)
x	x	x	x	x	x	x	x	8.3
x	x	x	x	x	x	x		3.2
x	x	x	x	x	x			-0.7
x	x	x	x	x				-1.0
x	x	x	x					-2.0
	x	x	x	x	x	x	x	8.2
		x	x	x	x	x	x	8.1
			x	x	x	x	x	3.7
				x	x	x	x	3.8

g	imp.(%)
10	-0.8
100	5.5
200	5.4
300	5.2
400	5.0
500	4.7
1000	4.0
2000	2.7

g	imp.(%)
10	2.7
100	8.2
200	8.3
300	7.9
400	7.4
500	7.2
1000	6.4
2000	5.6

Table 2: Left: Improvement (in %) to Borda's method of the area under the curve in the precision-recall space, for the aggregation of different rankings. Right: Improvement to Borda's method of the area under the curve in the precision-recall space, for different values of g; left: learning set, right: test set.

5 Conclusion

We presented *RankMerging*, a supervised machine learning framework which combines rankings from any unsupervised classifier to improve the performance of link prediction. This method is straightforward and computationally cheap as its complexity is $O(n.\alpha)$, where α is the number of rankings aggregated and n the number of predictions. It is adapted to prediction in social networks, as n can be tuned according to the users' needs. On the other hand, the precision on top-ranked items is not as high as the results yielded by supervised methods designed for information retrieval. In the case of a PSP, considered in this paper, this parameter would adjust the number of predictions to its commercial strategy.

So far, we have exclusively focused on structural information in order to predict unknown links. However, the framework is generic and any feature providing a ranking for likely pairs of nodes can be incorporated. Additional structural classifiers are an option, but other types of attributes can also be considered, such as the profile of the users (age, hometown etc.), or timings of the interactions. In the latter case, for instance, if i and j are both interacting with k within a short span of time, it is probably an indication of a connection between i and j. From a theoretical perspective, *RankMerging* provides a way to uncover the mechanisms of link creation, by identifying which sources of information play a dominant role in the quality of a prediction. The method could be applied to other types of networks, especially when links are difficult to detect. Applications include network security, for example by detecting the existence of connections between machines of a botnet, and biomedical engineering, for screening combinations of active compounds and experimental environments in the purpose of medicine discovery.

Acknowledgements

We thank E. Viennet, M. Danisch and anonymous reviewers for useful bibliographical indications. This paper presents research results of the Belgian Network DYSCO, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors. We also acknowledge support from FNRS and the European Commission Project *Optimizr*.

References

1. M. Al Hasan *et al.* Link prediction using supervised learning. In SDM'06: Workshop on Link Analysis, Counter-terrorism and Security, 2006.
2. L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In WSDM'11, pages 635–644. ACM, 2011.
3. N. Benchettara *et al.* Supervised machine learning applied to link prediction in bipartite social networks. In ASONAM'10, pages 326–330. IEEE, 2010.
4. C. Bliss *et al.* An evolutionary algorithm approach to link prediction in dynamic social networks. arXiv:1304.6257, 2013.

5. C.J.C. Burges *et al.* Learning to rank using an ensemble of lambda-gradient models. Journal of Machine Learning Research-Proceedings Track, 14:25–35, 2011.
6. P.M. Comar *et al.* Linkboost: A novel cost-sensitive boosting framework for community-level network link prediction. In ICDM'11, pages 131–140. IEEE, 2011.
7. K. Dasgupta *et al.* Social ties and their relevance to churn in mobile telecom networks. In EDBT'08, pages 668–677. ACM, 2008.
8. D. Davis *et al.* Supervised methods for multi-relational link prediction. Social Network Analysis and Mining, 3(2):127–141, 2013.
9. J.C. de Borda. Mémoire sur les élections au scrutin. 1781.
10. C. Drummond and R.C. Holte. Explicitly representing expected cost: An alternative to roc representation. In KDD'00, pages 198–207. ACM, 2000.
11. C. Dwork *et al.* Rank aggregation methods for the web. In WWW'01, pages 613–622. ACM, 2001.
12. Y. Freund *et al.* An efficient boosting algorithm for combining preferences. Journal of machine learning research, 4:933–969, 2003.
13. H. Jeong *et al.* Measuring preferential attachment in evolving networks. EPL, 61(4):567, 2003.
14. H. Kashima *et al.* Link propagation: A fast semi-supervised learning algorithm for link prediction. In SDM'09, volume 9, pages 1099–1110. SIAM, 2009.
15. L. Katz. A new status index derived from sociometric analysis. Psychometrika, 18(1):39–43, 1953.
16. G. Kossinets and D.J. Watts. Empirical analysis of an evolving social network. Science, 311(5757):88–90, 2006.
17. J. Leskovec *et al.* Microscopic evolution of social networks. In KDD'08, pages 462–470. ACM, 2008.
18. D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. JASIST, 58(7):1019–1031, 2007.
19. R.N. Lichtenwalter *et al.* New perspectives and methods in link prediction. In KDD'10, pages 243–252. ACM, 2010.
20. G. Linden *et al.* Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, 7(1):76–80, 2003.
21. Y.T. Liu *et al.* Supervised rank aggregation. In WWW'07. ACM, 2007.
22. L. Lü and T. Zhou. Link prediction in complex networks: A survey. Physica A, 390(6):1150–1170, 2011.
23. Z. Lu *et al.* Supervised link prediction using multiple sources. In ICDM'10, pages 923–928. IEEE, 2010.
24. T. Murata and S. Moriyasu. Link prediction of social networks based on weighted proximity measures. In ICWI, pages 85–88. IEEE, 2007.
25. B. Ngonmang *et al.* Churn prediction in a real online social network using local community analysis. In ASONAM'12, pages 282–288. IEEE, 2012.
26. M. Pujari and R. Kanawati. Supervised rank aggregation approach for link prediction in complex networks. In WWW'12 Companion. ACM, 2012.
27. T. Raeder *et al.* Predictors of short-term decay of cell phone contacts in a large scale communication network. Social Networks, 33(4):245–257, 2011.
28. S. Scellato *et al.* Exploiting place features in link prediction on location-based social networks. In KDD'11, pages 1046–1054. ACM, 2011.
29. T. Tylenda *et al.* Towards time-aware link prediction in evolving social networks. In SNA-KDD'09, page 9. ACM, 2009.
30. F. Wei *et al.* irank: A rank-learn-combine framework for unsupervised ensemble ranking. JASIST, 61(6):1232–1243, 2010.
31. T. Zhou *et al.* Predicting missing links via local information. EPJB, 2009.

Monotone Sampling of Networks[⋆]

Tim Grube[1], Benjamin Schiller[1], and Thorsten Strufe[2]

[1] Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany
lastname@cs.tu-darmstadt.de
[2] Technische Universität Dresden, Nöthnitzer Straße 46, 01187 Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract. Determining the graph-theoretic properties of large real-world networks like social, computer, and biological networks, is a challenging task. Many of those networks are too large to be processed efficiently and some are not even available in their entirety. In order to reduce the size of available data or collect a sample of an existing network, several sampling algorithms were developed. They aim to produce samples whose properties are close to the original network. It is unclear what sample size is sufficient to obtain a sample whose properties can be used to estimate those of the original network. This estimation requires sampling algorithms that produce results that converge smoothly to the original properties since estimations based on unsteady data are unreliable. Consequently, we evaluate the monotonicity of sampled properties while increasing the sample size. We provide a ranking of common sampling algorithms based on their monotonicity of relevant network properties using the results from four nework classes.

1 Introduction

Today's networks are quite large, in many cases too large to understand the network or to compute its properties. We have to reduce the complexity and therefore the size of networks to use the network for analyses and research. We can reduce the size by using graph coarsening or sampling techniques. Graph coarsening and some sampling techniques require the availability of the complete network. This constraint is rarely satisfied. Sampling by exploration allows to gain knowledge about the unavailable network, but it usually distorts properties as the sampling process can be biased. There are two large classes of sampling algorithms, we can sample using a breadth first sampling (BFS) approach, constructing the sample from the local area first, or we can use a random walk (RW) approach, traversing along random paths of nodes and constructing the sample with nodes from deeper areas of the network. The convergence behavior of network properties like the degree distribution depends highly on the underlying network and the used sampling algorithm.

[⋆] This research was funded by the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014 and the Software Campus by the German Federal Ministry of Education and Research (BMBF) under grant no. "01IS12054"

Many work has been done to overcome these biases and many specialized algorithms were developed. These algorithms produce samples, whose properties converge faster to the original networks properties, but as the properties of the original network are typically unknown, it is undecidable whether the quality demands or original properties are met.

Our approach is another way to solve this problem. We are proposing a new metric, which allows to develop an estimator for network properties in future work. This estimator should deliver the properties of the original network if it gets the network properties of the sample, the specification of the sampling algorithm, and the assumed size of the original network. For the development of such an estimator, we need the sampling algorithm to produce a sample with monotone converging network properties. In this paper, we are investigating the convergence monotonicity of the network properties on sampled networks.

The rest of the paper is structured as follows: Section 2 presents the related work about sampling and the evaluation of newly developed sampling algorithms. We present in Section 3 the desired behavior of sampling algorithms. We present the results and the discussion of our work in Section 4 and conclude with a summary and outlook in Section 5.

2 Related Work

The related work lists two typical classes of sampling techniques. The first one is the deletion of nodes or edges. Node deletion techniques use the complete network as basis and are deleting nodes until the network size is reduced to the desired size. Edge deletion uses a similar technique, instead of removing nodes, these algorithms remove edges and perhaps the attached nodes. The complete network has to be available to apply these two techniques. The second technique is sampling by exploration, using this technique, the sampling algorithm traverses from a start node into the network and collect the nodes to the sample. This technique is interesting for at least two reasons: First, it is easy to use by instrumenting crawlers. Second, we do not have this dependency on the availability of the complete network. Sampling by exploring is the common technique to reduce the complexity of networks and gain an excerpt of the whole network.

A lot of research has been done into the direction of developing more sophisticated sampling algorithms in this area. These algorithms are developed to produce samples, whose property values converge faster towards the property values of the original network. The sampling algorithms can be classified into breadth first approaches and random walk ones. Table 1 shows the classification, based on the type of walking, and the abbreviations of the analyzed algorithms.

Breadth First Sampling

The BFS algorithms traverse through the network by focusing on the local neighborhood first. The simplest implementation is a classical BFS which visits all the neighbors of node. Krishnamurthy, Leskovec and Stutzbach [4,7,13] use the BFS in their work. Goodman et al. [2] introduced snowball sampling (SS), which is a variation of the BFS and visits only a specifiable number of new, still unseen neighbors. Lee et al. [6] have evaluated this variation. Similar to SS, respondent-driven sampling (RDS) is developed by Heckarthorn et al. [3] and analyzed by Rasti and Kurant [10,5]. RDS visits a specifiable number of random neighbors, ignoring whether these neighbors are already known. Forest fire (FF), introduced by Leskovec et al. [7], is the last BFS derivate. It collects all neighbors with a certain probability into a walking queue.

Random Walk Sampling

The second class is the one of the random walk algorithms. The simplest one is the random walk sampling (RW). This algorithm traverses through the network by exploring along random neighbors. This sampling algorithm is well studied, e.g. by Stutzbach, Leskovec and Ribeiro [13,7,11]. Intuitively, and mathematically proveable, the RW sampling is biased towards nodes with higher degrees in the network. To overcome this bias, and to collect a representative sample of the network, two methods of correcting the probability for the next node were introduced. Stutzbach et al. [13] called their version random walk with degree correction (RW-DC). Rasti et al. [10] proposed a slightly different correction and named it metropolized hastings random walk (RW-MH). Both approaches depend on the degree of the potential next node on their exploration path.

Stutzbach et al. [13] showed two further variations of the RW: The first one is the random stroll (RS). This variation is moving like the RW but skips intermediate nodes from adding them to the sample. The second algorithm is the combination of RS and RW-DC. Random stroll with degree correction (RS-DC) skips intermediate nodes like the RS and moves with a degree correction like the RW-DC.

Leskovec et al. [7] introduced another variation of the RW. In particular, they introduced a jump probability in the random jump (RJ). This probability allows to move to a farther area of the network to avoid getting caught in a small area of the network.

Ribeiro et al. [11] developed a variation of a random walk with multiple instances. Since a simple parallel execution would suffer from the same problems like the classical RW, they introduced a dependency between the instances. Only one instance, in the default setting the one with the highest node degree on the current position, is allowed to move through the network. The active instance is picked every round again. This sampling algorithm is called frontier sampling (FS).

Another algorithm, similar to the RW, is the depth first sampling (DFS). This algorithm moves to the first neighbor and collects the remaining neighbors in a queue. This queue affects the behavior if the algorithm is getting caught. Even though there are similarities between RW and DFS, the results are quite different due to the impact of the queue in DFS. This algorithm is often used as a kind of baseline, e.g. by Krishnamurthy and Leskovec [4,7].

Table 1. Analyzed sampling algorithms.

Class	**Algorithm**	**Abb.**	**Related Work**
Breadth First Sampling	Breadth First Sampling	BFS	[4,7,13]
	Forest Fire	FF	[7]
	Respondent-driven Sampling	RDS	[3,10,5]
	Snowball Sampling	SS	[2]
Random Walk Sampling	Depth First Sampling	DFS	[4,7]
	Frontier Sampling	FS	[11]
	Random Jump	RJ	[7]
	Metropolized Hastings Random Walk	RW-MH	[10]
	Random Stroll	RS	[13]
	Random Stroll with Degree Correction	RS-DC	[13]
	Random Walk	RW	[13,7,11]
	Random Walk with Degree Correction	RW-RDC	[13]

Network Properties

We analyze the common network properties. The earlier introduced sampling algorithms are partially evaluated with these metrics in their corresponding papers. There are two types of properties: The first one is a single scalar value, for example a floating-point number or an integer. The second type is a distribution, which provides for each possible value a certain probability to find this value in the network. We concentrate our work in this paper on the single scalar values.

The assortativity coefficient (AC) is a measure for the correlation of connected nodes. We use the definition from Newman et al. [9,8]. We analyze the clustering coefficient (CC) in both characteristics, the transitivity which is for example used by Chakrabarti et al. [1] and the average clustering coefficient which is for example used by Krishnamurthy et al. [4]. The degree distribution (DD) allows the derivation of multiple single scalar values. We inspect the average node degree, the average in-degree, the average out-degree and the maximum degree of the networks. We are deriving multiple single scalar values from the shortest path length distribution (SP), too. The characteristic path length is a measure for the expected path length in the network, the diameter is the maximum shortest path length in the network. As the diameter is prone to distortions, we are analyzing the effective diameter of the 90% quantile. This metric computes the maximum shortest path length for the part of 90% connected nodes in the network. We use the definition from Chakrabarti et al. [1] in our implementation. By ignoring the end of the heavy tail, we remove the sensitivity for deformed networks. Table 2 lists the used metrics, submetrics and abbreviations for the metrics.

Table 2. Evaluated network properties, analyzed submetrics.

Metric	**Submetrics**	**Abb.**
Assortativity	Assortativity Coefficient	AC
Clustering	Average Clustering Coefficient Transitivity	CC
Degree	Average Degree (avg) Average In-Degree (avg_{in}) Average Out-Degree (avg_{out}) Maximum Degree (max)	DD
Shortest Path	Characteristic Path Length (cpl) Diameter (diam) Effective Diameter, 90% (effectiveDiam)	SP

3 Monotonicity

We propose a new approach to circumvent the nescience of the original networks properties. Instead of evaluating the sampling algorithms with respect to the speed of convergence, we evaluate the monotonicity of the property convergence along increasing sample sizes. The monotonicity is an important property to support the development of estimators to project the properties of the sampled network to the original network. We rank the well known and commonly used explorative sampling algorithms with respect to the monotonicity properties of their samples.

The looked-for algorithm produces smoothly approximating properties among increasing sampling sizes. We count the direction changes of the property convergence while sampling with increasing sample sizes. Figure 1 shows the aggregrated AC of a webgraph, sampled 100 times with DFS (in blue) and RW (in red). The original networks assortativity coefficient is plotted as a green line. The AC sampled with RW is obviously slower converging than the AC sampled with DFS, even though, the smooth progression allows develop a simpler estimator for the original property values. We are computing the monotonicity by comparing the property

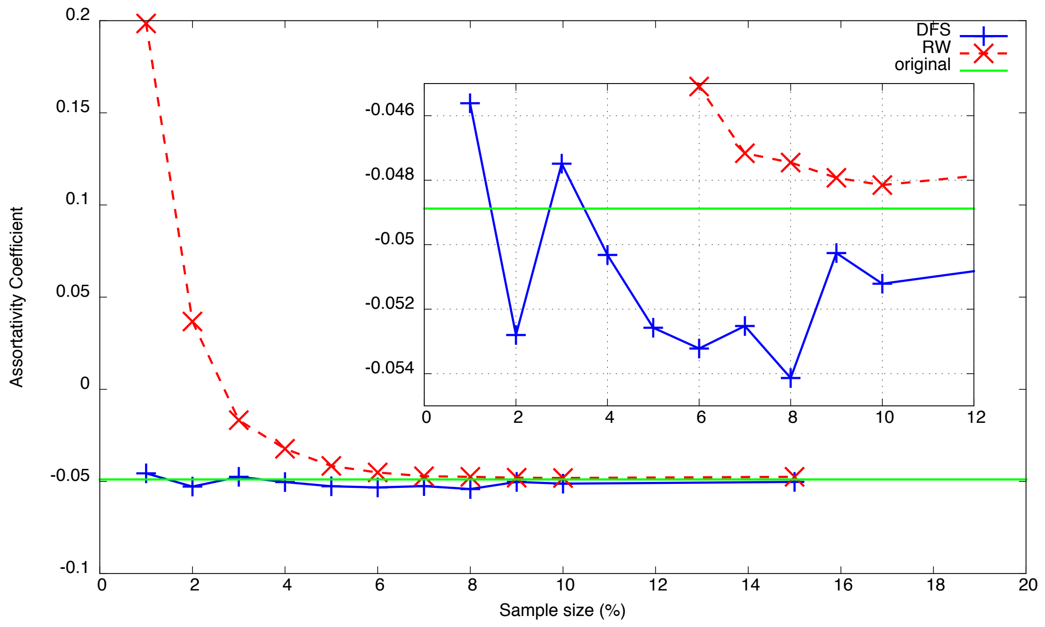


Fig. 1. Monotonicity of the AC on a webgraph, sampled with DFS (blue) and RW (red), original value (green).

values of consecutive samples according to Eq. (1). v_i are the values of a network property in the sample i, the sample $i{+}1$ is the next larger sample, the last v_i yields the property value of the original network.

$$monotonicity_i = \begin{cases} \uparrow_i & : v_i < v_{i+1} \\ =_i & : v_i = v_{i+1} \\ \downarrow_i & : v_i > v_{i+1} \end{cases} \tag{1}$$

Intuitively, a change of monotonicity is defined as a change in the direction of convergence as in Eq. (2) at position i to $i{+}1$. An intermediate equality of successive property values does not cause a change in monotonicity.

$$\uparrow_1 \ \cdots \ \uparrow_i \downarrow_{i+1} \downarrow_{i+2} \ \cdots \tag{2}$$

A *good* sampling algorithm approximates the inspected properties with as few changes in the monotonicity progress as possible. This property definition of a good sampling algorithm is also intuitive as it matches the perception of a good approximation.

4 Evaluation

We used GTNA [12] to implement and compute the sampling process. GTNA allows to integrate the network generation, the application of the sampling algorithms and the computation of the graph-theoretic metrics. We initialized the sampling algorithms with random start nodes, and executed 100 runs for the networks with $\leq 500,000$ nodes and 20 runs for the networks with $> 500,000$ nodes to reduce the impact of randomness in the sampling process. We sampled *1%* - *10%* in *1%* steps and *15%* on all networks, networks with $\leq 500,000$ nodes are also sampled with *20%* and *25%*.

We selected networks based on the related work, which is presented in Section 2. The analyzed networks are listed in Table 3. The networks are available at the SNAP project[3]. We identified four groups of networks based on their network type: social networks in a directed and undirected form, directed p2p networks and a directed webgraph. We calculated the earlier presented network properties (AC,

Table 3. Evaluated networks, classified by their context.

Class	**Type**	**Network**	**Nodes**
social	directed	cit-HepPh	34,546
		cit-HepTh	27,770
		soc-Epinions1	75,879
social	undirected	ca-GrQc	5,242
		com-Youtube	1,134,890
p2p	directed	p2p-Gnutella30	36,682
		p2p-Gnutella31	62,586
webgraph	directed	web-Google	875,713

CC, DD, and SP) to provide a useful evaluation for the property monotonicity of

[3] Stanford Large Network Dataset Collection, available at `http://snap.stanford.edu/data/`

the sampled networks. We show the results in Table 4. The sampling algorithms are ranked according to their monotonicity properties. We ignored the transitivity in this evaluation, since all algorithms perform very good with respect to the monotonicity of the progression of this metrics values. As introduced in Section 3, we are concentrating on single scalar properties, and therefore we are deriving the submetrics as listed in Table 2. These single scalar values are combined as in Eq. (3) to have a single value per property. To avoid an overweighting of one of these submetrics, we are normalizing by the number of submetrics.

$$SP = \frac{(\mathrm{cpl} + \mathrm{diam} + \mathrm{effectiveDiam})}{3}; \quad DD = \frac{(\mathrm{avg} + \mathrm{avg}_{\mathrm{in}} + \mathrm{avg}_{\mathrm{out}} + \max)}{4} \tag{3}$$

We expect to provide a recommendation of sampling algorithms for the complete network class. Therefore, we sum up the single network properties of each network instance in a group to have a cumulated monoticitiy value per network property. To be able to compare these values over the borders of a group, we are normalizing the values by the number of networks within the group. This is formulated in Eq. (4) (*group* represents the network groups, listed in Table 3). The function M returns the value of a certain metric, computed on the given network. M_{group} collects the normalized, added values of the analyzed metrics. M is a placeholder for the computed metrics, listed in Table 2.

$$M_{group} = \frac{1}{|group|} \sum_{nw \in group} M(nw) \ ; \quad M \in \{AC, CC, DD, SP\} \tag{4}$$

To be able to provide a recommendation not only on the basis of a single property but for a monotone sampling of the complete network, we cumulate the values of the single network properties of a group and are normalizing them by the number of metrics. This is shown in Eq. (5) and described by ν.

$$\nu_{group} = \frac{AC_{group} + CC_{group} + DD_{group} + SP_{group}}{4} \tag{5}$$

The Uniform Sampling (US), which is a random node selection algorithm, is used as ground truth. This algorithm is not practicable in real world applications, but as it produces a real random sample it is a good baseline to compare the monotonicity of the analyzed algorithms with. We are providing in Table 4 the by ν sorted ranking of sampling algorithms. Table 4 shows the domination of the random walk algorithms on the directed social network and the webgraph, the BFS algorithms are dominating the undirected social network and the P2P networks. An interesting fact is the stable presence of the US in the upper half of the ranking. Besides the webgraph, the simple algorithms are not far behind the forefront. There is typically either BFS or RW within the best five sampling algorithms. We

Table 4. Summed monotonicity rankings per network group

Rank	ν - directed social		ν - undirected social		ν - p2p		ν - webgraph	
1	RW	0.74	US	1.63	BFS	1.44	RW-DC	0.83
2	RW-DC	0.94	DFS	1.69	US	1.79	RS-DC	0.83
3	RJ	1.01	RDS	1.71	RS-DC	1.83	RJ	0.83
4	RW-MH	1.13	FS	1.91	FF	1.88	RW-MH	0.96
5	FS	1.51	BFS	1.98	RW-MH	1.92	FS	1.42
6	US	1.58	RS-DC	2.08	RS	2.04	RW	1.58
7	RS	1.63	SS	2.10	RJ	2.13	US	1.83
8	RS-DC	2.13	RS	2.43	RW-DC	2.21	RS	1.92
9	DFS	2.25	RJ	2.63	DFS	2.58	FF	2.69
10	SS	2.28	RW-MH	2.71	RDS	2.71	SS	3.27
11	RDS	2.42	RW	3.17	SS	3.17	BFS	3.52
12	FF	2.46	RW-DC	3.25	FS	3.33	RDS	3.69
13	BFS	2.92	FF	3.28	RW	3.96	DFS	3.71

are showing in Figure 2(a) the plotted values of the results for the directed social network. We have built groups for AC, CC, DD, and the SP, the last group is showing the ν-values of the column from Table 4. The AC is completely monotone sampled by the RW and the RW-MH, the other algorithms are including changes in the monotonicity, a high amount of changes is especially visible at the group of BFS algorithms. The plot of the CC values showing similar results, the advantage of the random walk group is even higher, besides the DFS, all algorithms are better than the BFS algorithms. The monotonicity analysis of the DD metric shows similar monotonicity values for all sampling algorithms. The advantage of the RW algorithms is not distinctly present. The SP metric is well preserved by the BFS, the advanced algorithms of the BFS group and the group of RW algorithm produce similar monotonicity values. The advantage of the BFS is intuitive, as the shortest path properties are constantly converging by extending the exploration of the direct neighborhood in rings. The RW algorithms are traversing into the deep of the network and are usually producing longer paths. Beside the SP property, the results for the webgraph are similar. Figure 2(d) shows the plot for the webgraph. The advantage of the BFS on the SP property is not present on the webgraph.

The undirected social network, shown in Figure 2(b), is similarly sampled with all algorithms. The monotonicity of all samples is similar for the combined metrics of the network, DD and SP. The CC has a negative outlier with RW-DC which is not monotonously converging. The AC has two positive outliers BFS and RDS, which produce very monotone AC values.

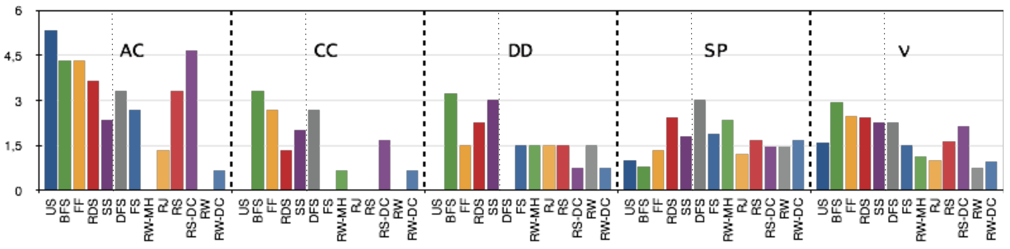
The P2P network, shown in Figure 2(c), is the only network with different monotonicity results, as shown in Table 4. The advantage of the RW algorithms is not present. The AC is dominated by BFS and FF, while the CC is dominated by BFS, RS and RS-DC. The DD is nearly equal monotone for all sampling algorithms, only the BFS has a change in the monotonicity but these values can be neglected with a value of 0.25. The SP property is slightly dominated by RW algorithms. Combined to the ν value, the algorithms perform with similar monotonicity for the complete network. There is no predominant algorithm for this group of P2P networks.

5 Conclusion

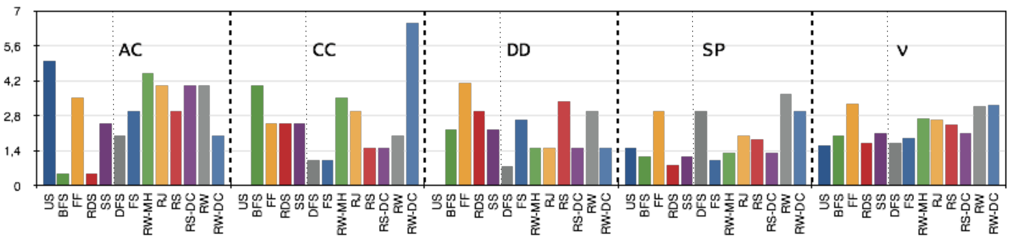
Today's networks are large, often too large to understand and process them directly. The computation of graph-theoretical properties on these large networks is a challenging task. We need to reduce the networks complexity and therefor the size of the network. The main technique for achieving this reduction of complexity is sampling by exploration. These sampling algorithms traverse through the network and collect the sample. Due to the network structure, some algorithms distort the networks properties. Many improved sampling algorithms were introduced to overcome these biased sampling processes. The properties of the sampled network are highly depending on the underlying network and the used sampling algorithm.

As the original networks properties are mostly unknown, we are not able to compare the sample properties with them. Therefore, it is undecidable if or when the quality demands are met. We propose another way to overcome this problem. We evaluate the convergence monotonicity to support the development of an estimator for the common original network properties. The common properties are e.g. the degree distribution, and the shortest path distribution. To be able to provide a useful monotonicity evaluation, we chose networks based on the related work, which analyzes the newly developed sampling algorithms. To evaluate the convergence monotonicity, we sampled multiple times and compute the properties of the samples. The convergence along increasing sample sizes is collected and aggregated. We rank the sampling algorithms with respect to the monotonicity values of their samples.

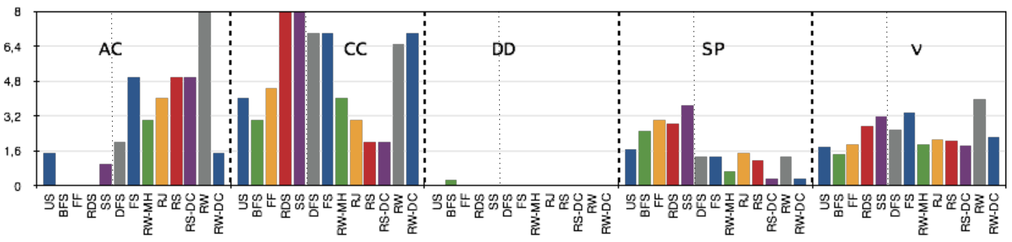
The main results of our evaluation are as follows: the complex algorithms enhancing the simple basic algorithms are not necessarily better in our monotonicity metric. Moreover, the simple algorithms random walk and breadth first sampling are the best algorithms of their group or at least at the forefront of their groups. The random walk algorithms are typically outperforming their breadth first sampling counterparts. The breadth first sampling algorithms are only as good as the random walk algorithms on the P2P and undirected social networks. The shortest path properties are well preserved by the breadth first sampling, they are inferior



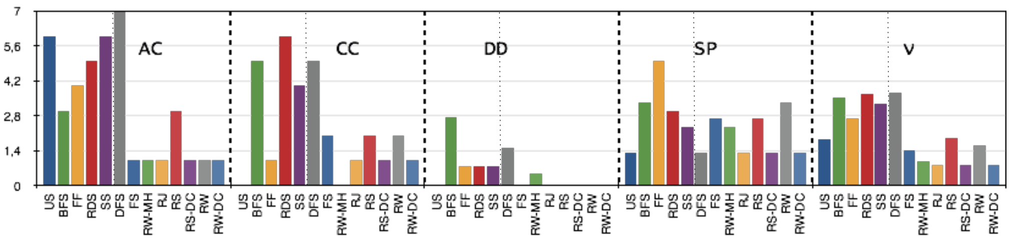
(a)



(b)



(c)



(d)

Fig. 2. Results of the monotonicity of the analyzed sampling algorithms: (a) directed social networks, (b) undirected social networks, (c) p2p networks, (d) webgraphs.

compared to the random walk ones on the webgraph and the P2P networks only. An interesting fact is that the monotonicity of the uniform sampling which is used as ground truth: The uniform sampling is never the most monotone algorithm but is in the better half of the ranking on all networks.

In the future work, we will develop a metric to measure the monotonicity of network properties which are not describable with a single scalar value, but with a distribution. The open question regarding this metric is: What is a monotonely converging distribution and how to measure this monotonicity? After answering this question, we want to develop estimators to assess the properties of the original network by analyzing sampled networks.

References

1. Chakrabarti, D., and Faloutsos, C. Graph Mining : Laws , Generators , and Algorithms. *ACM Computing Surveys 38*, March (2006).
2. Goodman, L. A. Snowball Sampling. *The Annals of Mathematical Statistics 32*, 1 (Mar. 1961), 148–170.
3. Heckathorn, D. D. Respondent-driven sampling: a new approach to the study of hidden populations. *Social problems 44* (1997), 174–199.
4. Krishnamurthy, V., Sun, J., Faloutsos, M., and Tauro, S. Sampling Internet Topologies : How Small Can We Go ? In *International Conference on Internet Computing* (2003).
5. Kurant, M., Markopoulou, A., and Thiran, P. On the bias of BFS (Breadth First Search). In *2010 22nd International Teletraffic Congress (lTC 22)* (Sept. 2010), IEEE, pp. 1–8.
6. Lee, S., Kim, P.-J., and Jeong, H. Statistical properties of sampled networks. *Physical Review E 73*, 1 (Jan. 2006), 016102.
7. Leskovec, J., and Faloutsos, C. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2006), KDD '06, ACM, pp. 631–636.
8. Newman, M. Mixing patterns in networks. *Physical Review E 67*, 2 (Feb. 2003), 026126.
9. Newman, M. J. Assortative Mixing in Networks. *Physical Review Letters 89*, 20 (Oct. 2002), 208701.
10. Rasti, A. H., Torkjazi, M., Rejaie, R., Stutzbach, D., Duffield, N., and Willinger, W. Evaluating Sampling Techniques for Large Dynamic Graphs. Technical Report CIS-TR-08-01, Department of Computer and Information Science, University of Oregon, http://mirage.cs.uoregon.edu/pub/tr08-01.pdf, Sept. 2008.
11. Ribeiro, B., and Towsley, D. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th annual conference on Internet measurement - IMC '10* (New York, New York, USA, Nov. 2010), ACM Press, p. 390.
12. Schiller, B., Bradler, D., Schweizer, I., Mühlhäuser, M., and Strufe, T. GTNA: a framework for the graph-theoretic network analysis. In *Proceedings of the 2010 Spring Simulation Multiconference* (San Diego, CA, USA, 2010), SpringSim '10, Society for Computer Simulation International, pp. 111:1—-111:8.
13. Stutzbach, D., Rejaie, R., Duffield, N., Sen, S., and Willinger, W. Sampling Techniques for Large, Dynamic Graphs. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (Apr. 2006), Ieee, pp. 1–6.

The Role of Homophily and Popularity in Informed Decentralized Search

Florian Geigl and Denis Helic

Knowledge Technologies Institute
Inffeldgasse 13/5. floor, 8010 Graz, Austria
{florian.geigl,dhelic}@tugraz.at
http://kti.tugraz.at/

Abstract. When searching for specific nodes in a network an agent hops from one node to another by traversing network links. If the network is large, the agent typically possesses partial background knowledge or certain intuitions about the network. This background knowledge steers the agent's decisions when selecting the link to traverse next. In previous research two types of background knowledge have been applied to design and evaluate search algorithms: homophily (node similarity) and node popularity (typically represented by the degree of the node). In this paper we present a method for evaluating the relative importance of those two features for an efficient network search. Our method is based on a probabilistic model that represents those two features as a mixture distribution, i.e. as a convex combination of link selection probabilities based on the candidate node popularity and similarity to a given target node in the network. We also demonstrate this method by analyzing four networks, including social as well as information networks. Finally, we analyze strategies for dynamically adapting the mixture distribution during navigation. The goal of our analysis is to shed more light into appropriate configurations of the background knowledge for efficient search in various networks. The preliminary results provide promising insights into the influence of structural features on network search efficiency.

Keywords: Networks, Decentralized Search, Homophily, Search Algorithms

1 Introduction

Nowadays, in many aspects of our daily life we, knowingly or unknowingly, deal with networks. For instance, we participate in large offline and online social networks, we often deal with information networks such as Wikipedia and the Web, or with file sharing peer-to-peer networks. All of these networks are constantly growing and sometimes consist of billions of connected nodes. In many situations we, or some other kind of autonomous agents, have to search for specific nodes in those networks. Apart from being large, these networks also constantly change. For example, friendship connections in a social network are created and removed, or new Web sites are created and connected to old ones. Therefore, search in such large and ever changing networks is a complex and hard task. The task becomes even harder in networks without a global index such as the Google index for the Web. For example, peer-to-peer networks do not rely on the existence of a global index but base their search strategies on the local proximity of a candidate node to a specific target. More recent examples include autonomous swarms of drones building up ad hoc networks. In those systems, information has to be passed from one drone to another, while the network is constantly changing and therefore no central search or routing engine can be built. In such decentralized networks the search performance heavily relies on finding resources within as few hops as possible.

In previous research several decentralized search algorithms have been designed. Among others, these include:

- Kleinberg's algorithm (based on homophily): Informed greedy search works well on small world networks with a clustering exponent of 2. In his experiments the clustering exponent determined the probability of a connection between two nodes as a function of their similarity. The simulation was steered by homophily, i.e. the geographic distance on a network model was used. [4], [5], [6].
- Adamic's algorithm (based on popularity): Power-law graphs can be locally searched by moving to the neighbor node with the highest degree. The costs of the search scale sub-linearly with the graph size. This property makes this algorithm interesting when dealing with large networks [1].
- Jensen's algorithm (based on a fixed combination of homophily and popularity): Using homophily and degree disparity as background information, the authors construct a simple algorithm for decentralized search. In a couple of synthetic and real-world networks they were able to outperform Adamic's as well as Kleinberg's algorithm [9].

All of these algorithms perform remarkably well and are typical heuristic algorithms based on intuitive assumptions on the nature of features needed to efficiently search in networks. In this paper we set out to analyze these assumptions in greater detail, with a final goal of learning more about the influence of those two features (homophily and popularity) on the search performance. Thus, we aim to answer the following research questions:

1. How much information is provided by node homophily and node popularity and how well can this information be used for efficient search in networks?
2. How should we mix node homophily and node popularity to maximize the search performance? Can an adaptive approach to information mixture outperform a static mixture that remains constant throughout navigation.

To answer these questions we simulate a decentralized search approach, informed by a combination of two different local features, which serve as proxies for node popularity and node homophily – degree and cosine similarity to the target node. Additionally, we introduce and analyze adaptive mixing strategies. To evaluate the navigation performance we create a framework, which allows us to examine the impact of different mixtures. Our preliminary results show that in the majority of analyzed networks homophily provides, on average, more important and more useful information for efficient search. However, the optimal results are only achieved in situations where small amount of popularity information is also available to the search agent.

Thus, our work makes the following contributions:

1. *Methodological*: We provide a framework for analyzing various features to inform search in networks. The framework allows to investigate a wide range of feature combinations and to assess their search performance by simulating an agent navigating through a network.
2. *Empirical*: We apply our framework on real networks and learn about the importance of homophily and node popularity as features for informing network search.

Outline: The rest of the paper is organized as follows. We summarize investigations done by other researchers in the field in Section 2. Thereafter we shortly describe the methodology used in the paper in Section 3 followed by the experiments in Section 4 and their results in Section 5. A brief discussion of the results can be found in Section 6. Finally, we wrap up the paper with an outlook for the future work in Section 7.

2 Related Work

The research on decentralized search in networks was initiated by the famous Milgram experiment in the 1960s [10]. In this experiment individuals had to forward a letter to an unknown person – a stockbroker from Boston. The participants were only allowed to send the letter to somebody whom they know by the first name, i.e. to a friend who possibly can reduce the distance or even knows the target person. Remarkably, the average number of hops for the successful letter chains was less then six. Thus, a famous outcome of the experiment was the so-called "Six degrees of separation" phenomenon, which states that people in a large society such as the USA, are connected by friendship chains not longer than six. Later, Leskovec and Horvitz [7] analyzed a messaging network, namely MSN Messenger. Their result were similar as Milgram's – the average length of connection chains between all pairs of the users is only 6.6.

Another interesting result from the Milgram's experiment is the fact, that although they posses only local knowledge of the network, humans are capable of finding those short connection chains even in a large social network. Various researchers put more effort into investigation of this phenomenon and developed so-called decentralized navigation methods.

For example, Kleinberg [4] showed that in small world networks (with small average shortest path and highly connected groups of similar nodes) homophily (node similarity) can be exploited to efficiently find random target nodes. He observed that small world networks having a clustering exponent of 2 are navigable using a homophily feature. In his research the homophily feature was the geographic distance on the lattice of the Watts and Strogatz model as defined in [11].

Later, Adamic [1] showed that networks whose degree distribution follows a power-law distribution can be efficiently navigated with knowledge about node popularity (degrees) only. The algorithm moved in each step to the unvisited neighbor node with the highest degree. In situation where all neighbors where already visited a random neighbor was chosen. Compared to a random walk, which moves in each step to a random node, the algorithm was able to find nodes within just a few hops. Additionally, the average number of hops scales sub-linearly in the number of network nodes.

Simsek and Jensen [9] combined homophily and popularity in their algorithm called expected-value navigation (EVN). For EVN degree and attribute values of neighbors of the current as well as neighbors of the target node were needed. Additionally, they assumed that already visited nodes can be identified as such. In synthetic networks EVN was able to outperform both previously named algorithm.

3 Methodology

In this paper we set out to assess the importance and the role of homophily and popularity for informing decentralized search in networks. We base our approach on simulations of an agent that navigates through a network in search for a given target node. At each simulation step the agent needs to select one node from the list of neighboring candidate nodes. The selection is based on the available information about the candidate nodes and follows different navigation models. In detail, the simulation is based on three elements:

1. The available information, which we represent in the form of probability distributions over two features (homophily and popularity) and a particular mixture distribution of the corresponding probability distributions.
2. A background knowledge model, which defines the mixture distribution and its adaptation during the simulation.
3. A navigation model, which defines the candidate selection strategy.

Next we describe those three elements in more details.

3.1 Features & Mixture

To create probability distribution we use homophily and popularity as basic measures. We assume that the networks are undirected and without multiple links.

Degree: In our paper we use the degree as representation of popularity. Having a network with $\boldsymbol{A}$ as adjacency matrix (with $A_{ij} = 1$ if nodes i and j are connected by a link and $A_{ij} = 0$ otherwise), the degree is defined as follows:

$$k_i = \sum_{j=1}^{n} A_{ij} \quad . \tag{1}$$

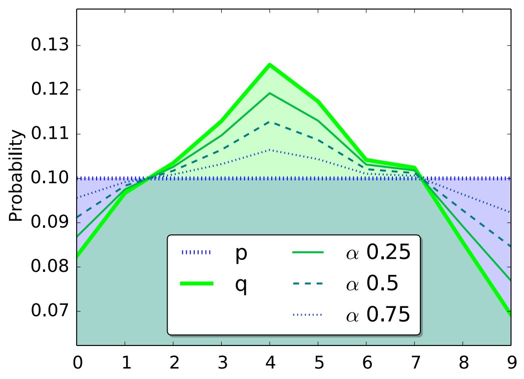


Fig. 1: **Convex combination:** The plot shows two discrete probability distribution p and q. p is uniform distributed, whereas q is non uniform. Varying α (0.25, 0.5, 0.75) one can observe the resulting mixture distributions.

Cosine Similarity: This feature serves as a proxy for homophily. Equation 2 defines the number of common neighbors of nodes i and j. Using this definition we compute the cosine similarity σ_{ij} of two nodes as defined in Equation 3. Cosine similarity results in a non-zero value between two nodes if they share at least one common neighbor.

$$n_{ij} = \sum_k A_{ik} A_{kj} \quad . \tag{2}$$

$$\sigma(i,j) = \frac{n_{ij}}{\sqrt{k_i k_j}} \quad . \tag{3}$$

Mixture Distribution: For the purpose of creating mixtures of these two features we convert them to probability distributions. Therefore, we calculate a probability mass function by dividing each feature value by the sum of all values. Thus, we need only local information for normalization. Precisely for each feature we divide the value of a node by the sum of all values corresponding to the same feature (all neighbors). Hence we get for each feature a probability distribution. For example, the degree normalization we define as following:

$$q_i = \frac{k_i}{\sum_{i=1}^{n} k_i} \quad . \tag{4}$$

Afterwards we apply a convex combination to generate a mixture distribution. Equation 5 defines the steps necessary to calculate the mixtures, where w_l is the weight of a probability distribution l.

$$f_i = \sum_{l=1}^{n} w_l p_i \quad \text{where} \quad \sum_{l=1}^{n} w_l = 1 \quad . \tag{5}$$

We denote the probability distribution of cosine similarity and degree as p and q. Furthermore we set $w_1 = \alpha$ and $w_2 = 1 - \alpha$ since we only have two distributions. This allows us to configure the mixture distribution over α only. Equation 6 defines the mixture distribution of p and q using α.

$$f_i = \alpha p_i + (1 - \alpha) q_i \quad . \tag{6}$$

For instance, Figure 1 visualizes different mixture distributions of p and q. Note that setting α to 0 results in a mixture distribution equal to q. On the opposite if α is set to 1, the resulting mixture distribution equals p.

3.2 Background Knowledge Models

Static Mixture: In our first analysis we use a constant α to create mixture distributions as defined in section 3.1. Recollect α defines the impact of cosine similarity p and degree p onto the mixture distribution. From now on we refer to this model as *static mixture*.

Static Switch: Inspired by human navigation strategies we generate a background knowledge model, which imitate human behavior. Scientists have discovered that humans have two phases when navigating in a network from one random node to another [3], [12].

In the first phase, also known as zoom out phase, humans try to reach a popular and thus highly connected node. As a consequence one characteristic of this phase is that nodes having a higher degree than the current one are chosen. Adamic presented such an algorithm in her work and showed that power-law graphs can be efficiently searched with it [1]. Setting α to 1 for the mixture distribution, allows us to mimic this strategy.

Contrary, the second phase of humans, the zoom in phase, is steered by the similarity of neighbors to the target node. Kleinberg mimicked this behavior and proved that small networks with a clustering exponent of 2 can be searched by doing so [4]. We simulate this behavior using the cosine similarity. By setting α to 0, we generate a mixture distribution equal to p. Thus the agent is only influenced by the similarity.

Recollect that the cosine similarity is always 0 if no neighbor has at least one common neighbor with the target node. Hence p can be a discrete, uniform distribution. This case increases the uncertainty of the mixture distribution if α is greater than 0. One can observe this effect in Figure 1. To tackle this problem, we imitate human behavior again and switch from the zoom out to the zoom in phase at a fixed point. We simulate this by initializing α to 0 and change the value to 1 at a certain point. We name this model *static switch*.

Dynamic Switch: However, the last method lacks of a dynamic transition point, since network size as well as the degree distribution influence the position of the optimal transition point. Thus we initialize the simulation with a certain α and set α to $1 - \alpha$ as soon as p is not uniformly distributed any more. In other words this model switches the weights of p and q in the mixture distribution the moment it reaches a part of the network near to the target node. We call this background knowledge model *dynamic switch*.

3.3 Navigation Models

Greedy Search: The simplest node selection mechanism is a greedy selection. In this strategy the algorithm traverse the network by moving always to the node with the highest probability of all unvisited neighbors. We reference this strategy as *greedy search*.

Stochastic Search: However, the *greedy search* model never selects nodes with a slightly lower probability in the mixture distribution than the maximum. To tackle this we select the next node by drawing randomly from the mixture distribution. We refer to this as *stochastic search*. Consequently this model results in a random walk if the mixture is uniformly distributed. On the other side it selects the same node as *greedy search* if one neighbor has a probability of 1 in the mixture distribution.

Softmax Search: Nevertheless, the randomness of mixture distributions can vary. To either increase or decrease the uncertainty of a distribution we use a softmax function. The softmax function we apply onto our mixture distributions is defined as following:

$$g_i = \frac{e^{\gamma f_i}}{\sum_i e^{\gamma f_i}} \quad . \tag{7}$$

This function increases the randomness of a distribution if γ is smaller than 1, whereas it decreases the uncertainty using values greater than 1. Figure 2 demonstrates the impact of various values for γ. Consequently we have the ability of continuous varying our navigation model. For example, a high γ results in *greedy search*, $\gamma = 1$ in *stochastic search*, and setting γ to zero produces a random walk. We refer to this model as *softmax search* with γ as a parameter.

4 Experiments

4.1 Dataset

For our experiments we use four networks:

Wikipedia for Schools is a subset of Wikipedia articles especially designed for the education of school children. It consists of 4.6 thousand articles (vertices) connected by 119.8 thousand hyper links (edges). The network is directed and belongs to the category of information networks. A power-law distribution provides the best fit for the degree distribution.

Furthermore we use a subset of *Facebook* created out of so called ego-networks where user represents vertices. Additionally, friendship between two users creates an edge in the network between those users. This dataset contains 3.9 thousand users, 88.1 thousand friendships and can be categorized as social network. Its degree distribution follows a log-normal distribution.

As a second social network we test our algorithm on a subset of *Twitter* consisting of 76.2 thousand users and 126.9 thousand edges. Contrary to Facebook, this dataset is directed. This is due the fact that in this social network user A

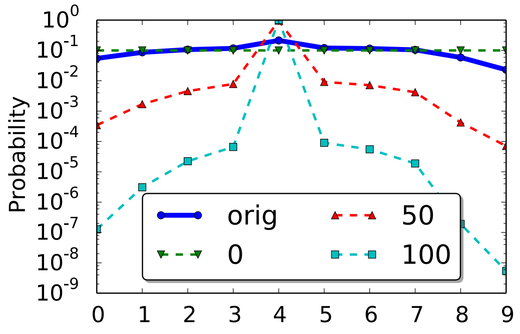


Fig. 2: **Softmax function:** One can see the impact of different values for δ of the softmax function defined in Equation 7. Values smaller than 1 have a flatting effect, whereas values higher than 1 concentrate more probability mass around modes – thus reducing the uncertainty. $\delta = 0$ results in a uniform distribution. Setting $\delta = 1$ does not modify the distribution.

can follow user B without the requirement that B follows A. The best fit for the degree distribution is a log-normal distribution.

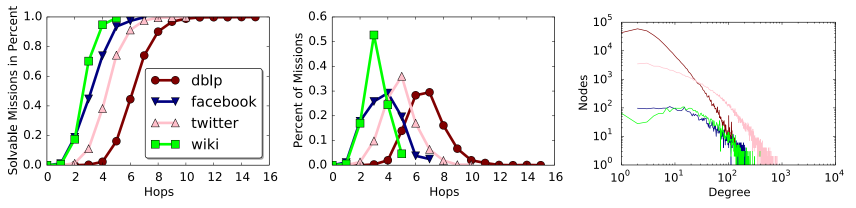
The last network is a co-authorship network, namely *DBLP*, which consists of 317 thousand authors (vertices) and 1 million co-authorship connections (edges). The degree distribution of the network can be fit best with a log-normal distribution.

We calculated the best fit for the degree distribution of these networks with the method proposed by Clauset [2]. The plot containing the degree distributions of all four networks can be found in Figure 3c.

4.2 Experimental Set-up

For each network we generate as many missions as there are nodes in the network. Each mission consists of two randomly chosen nodes reachable from each other. The hop plot and a plot of the shortest path lengths for all missions are shown in Figure 3.

The task of the algorithm is to navigate from one node to the other within as few hops as possible. We consider paths longer than 20 hops or containing revisits of nodes as unsuccessful. Furthermore our framework tries to avoid already visited nodes in each navigation by removing them from the candidate nodes. In our experiments we combine each navigation model with each background knowledge model once. This results in 9 experiments for each network. As evaluation metrics we gauge the success rate and stretch. The success rate is the fraction of successful solved missions, whereas the stretch is the length of the produced paths divided by their corresponding shortest paths lengths. In other words the stretch defines the factor of how much longer produced paths are compared to their shortest path.



(a) **Hop Plot** (b) **Shortest Distances** (c) **Degree Distributions**

Fig. 3: **Properties of Networks & Missions :** Figure 3a shows the hop plots of all missions. Lengths of shortest distances of all missions for each network are plotted in Figure 3b. Notice how long shortest distances of missions in the *DBLP* are compared to *Wiki for Schools*. Figure 3c shows the degree distributions of the networks.

5 Results

Figure 4 shows all results of our experiments. We exclude plots presenting the stretch, since they only emphasize the outcome of the success rate. Rows in Figure 4 are ordered by navigation models, whereas columns refer to the used background knowledge model.

The most interesting result is, that the cosine similarity p seems to be more important for an efficient navigation than the degree information q. Independent of the navigation model and network a fixed value for α near 0.9 seems to be a good choice. As imagined the optimal amount of hops after witch to switch from 0 to 1 for α depends on the network. For instance, the *static switch* model applied onto the *DBLP* - which is a sparse network with a high diameter - benefits from more steps to nodes with higher degrees during the first phase of the navigation. Additionally, for the *DBLP*, this holds for all navigation models. On the other side in *Wikipedia for Schools* the best performance is reached by switching to $\alpha = 0$ before the first hop is made. However, the *dynamic switch* model in combination with a low value for α outperforms all other strategies. This applies to all networks independent of the navigation model. This emphasizes, that in the first steps the uncertainty of mixture distributions with α greater than 0 is increased by the cosine similarity. Additionally, in the zoom in phase the degree information q likely steers in the opposite direction than the cosine similarity q. Consequently the zoom in phase also benefits from the *dynamic switch* model. This is due to the circumstance, that in power-law or log-normal networks a random selected node is likely to have low degree. Thus most missions have a low degree target node.

Anyhow, the difference between *greedy search*, *stochastic search*, and *softmax search* shows how determined the mixture distributions are. Using *softmax search* we can continuously adapted the uncertainty of the mixture distribution. In the last row of Figure 4 the parameter γ of the softmax function is set to 50.

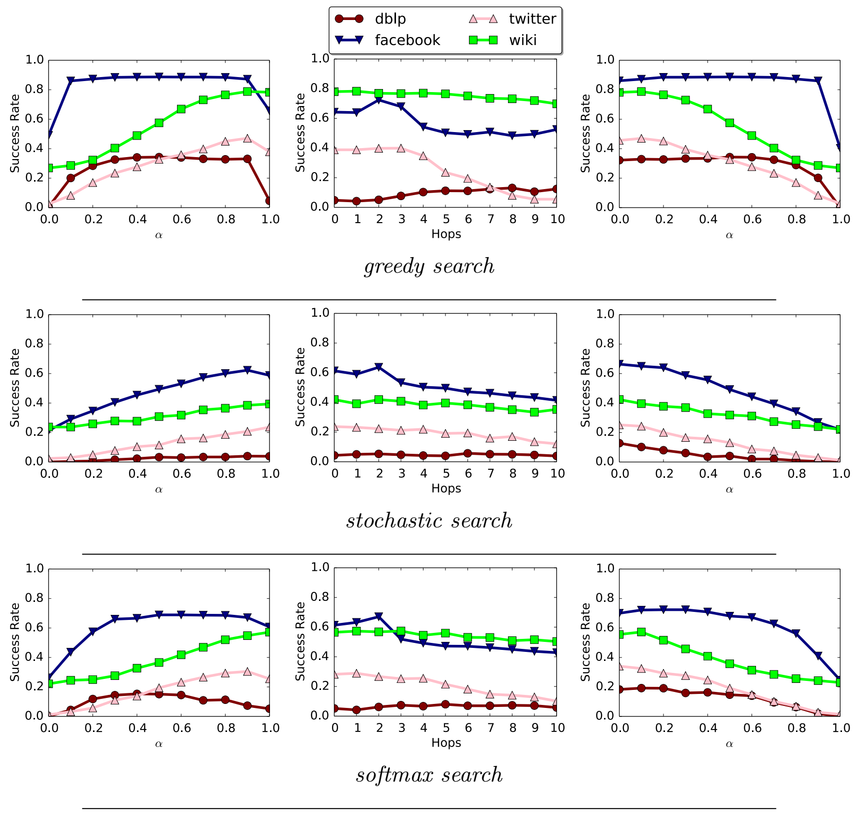


Fig. 4: **Results:** The figure shows the results of our simulations. Each row corresponds to one navigation model. We use the success rate as evaluation metric. The **left** column contains the results produced by the *static mixture* model. In this plot the x-axis defines the α parameter as specified in section 3.1. The **center** column α shows the outcome of the *static switch* model, where the transition point is on the x-axis. In the **right** column results of the *dynamic switch* model can be found. The initial value of α corresponds to the x-axis.

Consequently navigation behavior between *greedy search* and *stochastic search* is obtained.

Additionally to the success rate we gauge the normalized entropy of the mixture distributions. A normalized entropy of 1 is produced by uniform mixture distributions. On the other side, mixture distributions containing a value of 1, induce a normalized entropy of 0. Figure 5 shows the average normalized entropy for value of α between 0 and 1. Moreover on the left side γ is set to 1, whereas on the right a value of 50 is used. Notice the high normalized entropy of the *DBLP* at $\alpha = 1$ on both plots. This may be due the low density of the network

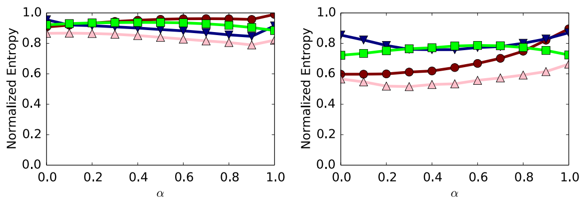


Fig. 5: **Normalized entropy of mixtures:** These plots depict the average normalized entropy of the mixture distributions produced during the simulation for each network. The left plot contains the gauged normalized entropy of unmodified mixture distributions. Contrary, the on the right the softmax function of Equation 7 is applied - using 50 as γ - onto the mixture distributions. Definitely the normalized entropy on the right plot is lower than on the left. This is a consequence of the reduced uncertainty of the mixture distributions through the applied softmax function.

which consequently results in situations where only a small amount of candidate nodes are available. Moreover this nodes are likely to have a high difference in the probability distribution. This is especially the case for cosine similarity.

6 Discussion

In our experiments the cosine similarity plays a more important role (best $\alpha = 0.9$) than the degree information. We believe that this is caused by the properties of the networks we use. All of them are small and have low diameters (3 - 8). Due to these properties the probability of having a neighbor which has a common neighbor with the target node is higher than it would be in larger networks with bigger diameters. Additionally, even if the cosine similarity of all neighbors is zero, the resulting navigation behavior - a random walk - has a high probability of moving to high degree nodes. For example, the famous page-rank algorithm takes advantage of this effect to identify popular nodes in a network [8]. Thus, a uniform mixture distribution favours high degree nodes.

Therefore, we strongly believe that simulations steered by cosine similarity - in power-law or log-normal networks - naturally include a zoom out phase. Nevertheless, additional information of the degree of neighbors may cut down the hops needed for the zoom out phase. Moreover, this intuition is supported by the observation, evident in our experiments where the *dynamic switch* model in combination with a low initial α outperformed all other models.

7 Future Work

One limitation of our research is that the cosine similarity is not a complete local measure of homophily. It includes information about the neighbors of both the current and the target node. Hence we are working on a metric more suitable as a proxy for local homophily. For example, a measure which can only determine if the target is in it's neighborhood or in the neighborhood thereof. This would also allow us to adjust the scope of local knowledge. Furthermore, we plan on applying our framework onto synthetic - especially non power-law/log-normal - networks in future work.

Acknowledgments This research was funded in part by the FWF Austrian Science Fund research projects P24866.

References

1. L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Phys. Rev. E*, 64:046135, Sep 2001.
2. A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.
3. D. Helic. Analyzing user click paths in a wikipedia navigation game. In *MIPRO, 2012 Proceedings of the 35th International Convention*, pages 374–379. IEEE, 2012.
4. J. Kleinberg. Small-world phenomena and the dynamics of information. *Advances in neural information processing systems*, 1:431–438, 2002.
5. J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, Aug 2000.
6. J. M. Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS) 14*, page 2001, Cambridge, MA, USA, 2001. MIT Press.
7. J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th international conference on World Wide Web*, pages 915–924. ACM, 2008.
8. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.
9. O. z. Simsek and D. Jensen. Decentralized search in networks using homophily and degree disparity. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 304–310, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.
10. J. Travers, S. Milgram, J. Travers, and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969.
11. D. J. Watts and S. H. Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.
12. R. West and J. Leskovec. Human wayfinding in information networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 619–628, New York, NY, USA, 2012. ACM.

Identifying Over-represented Temporal Processes in Complex Networks

Ursula Redmond* and Pádraig Cunningham

School of Computer Science and Informatics,
University College Dublin,
Ireland

Abstract. Temporal networks encode interactions between entities as well as the time at which the interactions took place, allowing us to identify systematic processes within the network. We can identify subprocesses or *temporal motifs* that recur frequently across a large network. In this paper, we present a strategy that allows us to identify which of a given set of temporal processes are *over-represented*. This highlights peculiarities of behaviour in the network. Our strategy involves constructing a set of interesting temporal processes, counting their embeddings in the network through subgraph matching, and then comparing this against counts in a temporally random version of the network. The network is randomized by shuffling the time-stamps in the original network. We present an evaluation on data from Prosper.com, a peer-to-peer lending website. Prosper.com was closed for regulatory reasons in 2009 and our evaluation shows interesting differences between the pre- and post-closure networks. In particular, temporal motifs indicating arbitrage are over-represented pre-closure and under-represented afterwards.

1 Introduction

Increasingly, temporal information is included with complex network data sets. Thus, instead of examining a set of static interactions between individuals, a finer-grained understanding of those interactions is now possible. Temporal networks have been used to represent a wide variety of social phenomena, from person-to-person communication to contagious disease spread by physical contact between people. The notion of spreading in a network can be more accurately identified when the times at which interactions took place are recorded.

When analyzing the processes that give a complex network its structure, recurring patterns of interaction often come to light. These frequent patterns are referred to as motifs, and are considered the building blocks of networks [16]. When temporal information is incorporated into the search for motifs, the results can have a clearer interpretation. For example, the initiator of a contagion may be easier to identify, since the first interaction in the contagion would

* This work was supported by Science Foundation Ireland [08/SRC/I1407, SFI/12/RC/2289].

have originated from that individual. Likewise, the potential reach of a piece of information in a communication network may be easier to isolate, given that propagation is time-dependent in the network.

The network we analyze in this paper comes from Prosper.com [20], a peer-to-peer lending platform. Members of the website can register to borrow and lend, and act without a bank as an intermediary. The loans among members are unsecured, so there is a risk that a member to whom you lend may default on their repayments. Temporal information is available with the data, which allows us to examine the structure and temporal dimension of some interesting motifs.

Given the wide range of credit ratings that the members have, and the ability for members to both borrow and lend, the opportunity for arbitrage arises. Members with a good credit rating can borrow money at a low interest rate, and lend the same amount at a high interest rate to members with lower credit ratings, aiming to profit from the difference in rates. The website may also provide an opportunity for members to engage in money laundering. In a simple example, money could be lent to from one member to another, and the borrowing member could default, hence completing the transfer of funds without the regulation of a bank. More complicated examples could also be imagined, involving intermediate members. In both scenarios, the network structure representing the behaviour must be composed of time-respecting paths, in which interactions occur in a non-decreasing temporal order.

The purpose of our current study is to examine the extent to which these time-dependent behaviours occur in the Prosper network to a greater extent than might be expected. To do this, we first count the embeddings of a set of time-respecting network patterns that represent this behaviour, using a subgraph matching algorithm. It is important to note that the patterns are not mined automatically - rather they are specified *a priori* and sought in the network. Then, we repeatedly re-assign the time-stamps on the interactions randomly, counting the embeddings again each time. It turns out that the presence of the time-respecting patterns is highly dependent on the timing of the interactions in the real network. This demonstrates the importance of temporal analysis for understanding behaviour in networks.

The paper is organized as follows. Section 2 presents related work, in the areas of temporal network analysis, subgraph matching and modularity. Section 3 introduces our methods for performing the matching and temporal analysis. Our results are discussed in Section 4. Section 5 concludes the paper and suggests future work.

2 Related Work

In order to asses the frequency of temporal motifs in networks, this paper draws on work from the fields of temporal network analysis and subgraph matching. To asses the significance of certain motifs in a network, we use methods from the area of network modularity.

2.1 Temporal Network Analysis

Given the prevalence of temporal information available with network data, ideas associated with static networks are being revised to take this new aspect into account. A comprehensive review [8] details these concepts. A fundamental concept in this paper is that of a *time-respecting* path, defined as a sequence of contacts which occur at non-decreasing times [10].

In a *reachability graph*, there must be a time-respecting path between nodes i and j for a directed edge to exist between them. Reachability graphs reveal the nodes which are reachable from a single root node [17]. Analysis of the reachability graph within a dating network of high-school students reveals interesting behaviour in relationships [2]. A *time-respecting subgraph* [22] is a generalization of a reachability graph, since it does not require a root node, but insists on reachability along each directed path.

The lifespan of a piece of information in a temporal communication network may be specified by a time window [25], which measures the time between the end of one communication and the beginning of the next. The closer in time the contacts take place, the higher the likelihood that the subject is the same. Similarly, the *relay time* of an interaction captures the time taken for a newly infected individual to spread the infection further via the next interaction they participate in [11]. The spread of information through a temporal network can also be modeled by a cascade. The structure of cascades can reveal spreading and community development [7]. The importance of *time-constrained* cascades is emphasized for understanding contagion [1].

Temporal motifs, as defined by Kovanen *et al.*, are connected subgraphs composed of similar event sequences, where similarity is measured in terms of the topology and temporal ordering of the events [12]. All adjacent events in a temporal motif must occur within time Δt of each other, and the events connected to a node must be consecutive in time. So if a node n in a temporal motif participates in events at times t_0 and t_2, then if an event exists involving n at time t_1, it must also be included in the motif so that the motif is valid. This is distinct from a flow motif, in which directed events that meet head-to-tail must be consecutive in time. Kovanen *et al.* propose an algorithm to find temporal motifs, which do not have the flow requirement. In contrast, the aim of our approach is to efficiently find subgraphs in which interactions occur within a specified time of each other, and in which events meeting head-to-tail are consecutive in time. In subsequent work, Kovanen *et al.* explored temporal motifs in a mobile communication network [13]. By including other attributes of the data, interesting mechanisms were found such as gender-related differences in communication patterns, and a tendency for similar individuals to communicate more often than might be expected.

2.2 Graph and Subgraph Isomorphism

The subgraph isomorphism problem determines whether a given graph contains a subgraph which has the same topological structure as another given graph.

Subgraph isomorphism is an NP-complete problem [6], so the time complexity of brute force matching algorithms increases exponentially with the size of the graphs and query graphs to be matched. This makes the problem prohibitively expensive to solve for large graphs.

Algorithms were developed which restrict the topology of the graphs and hence constrain the complexity. Such methods include the enforcement of planarity [9] or bounded valence [14]. Other approaches depend on deriving associated graphs, and on topological features such as strong regularity [5]. Another type of derived graph used is the canonical form of the graph, as in the Nauty algorithm [15].

Ullmann proposed a backtracking approach to solve the graph and subgraph isomorphism problems [24]. In an extension to the popular algorithm, the search space is pruned based on the degree of nodes in the graphs to be matched. An algorithm by Schmidt *et al.* [23] also employs backtracking, but uses the distance matrix representation of a graph to inspire the pruning steps.

The VF algorithm of Cordella *et al.* presents a depth-first search strategy for graph and subgraph isomorphism [3]. The matching process is described by a state space representation, in which each state of the process is associated with a partial solution. The partial solution includes the elements of the two graphs which match each other so far. The algorithm tries to extend each partial solution based on neighbouring nodes in the query graph and the network graph which maintain the match. The speed of the algorithm compares favourably with Ullmann's popular backtracking approach. An enhanced version, VF2, provides further performance gains [4] by substantially reducing memory requirements.

2.3 Network Motifs

Network motifs are patterns of connected nodes that occur at higher frequencies in real networks than in randomized networks [16]. Detecting network motifs gives insight into the processes that networks encode. Milo *et al.* discovered that classes of networks which performed similar functions had similar network motif profiles. For example, information processing networks from such different application areas as biomolecules within a cell and synaptic connections between the neurons in *Caenorhabditis elegans* were comprised of similar network building blocks.

To make this finding, the authors computed the occurrence frequency of a collection of motifs in a network. The structure of the network was then randomized, although each node in the randomized network maintained the same in- and out-degree as in the original network. The motifs were counted again in the randomized network. This randomization and counting was performed repeatedly, and the mean of the motif occurrences was computed. When the number of embeddings of a given motif is much lower in a randomized network, its frequency in the original network is therefore indicative of the functionality encoded by that network.

In contrast to the work of Milo *et al.*, we aim to unearth significant temporal structures of the network, rather than structural properties in isolation.

To achieve this, we count the time-respecting embeddings of query graphs that we specify. We then randomize the temporal information associated with the network, following a methodology described in Section 3. We then count the time-respecting embeddings in the randomized network. After repeating this step a number of times, we compute the average number of embeddings in the randomized networks. This reveals an interesting set of structures whose prevalence depends on processes encoded in the original version of the network.

3 Methods

This section describes our problem framework. We present our methodology for matching time-respecting subgraphs and identifying their prevalence in randomized versions of real temporal networks.

3.1 The Problem Framework

To find subgraphs embedded in a network which match the query graphs we specify, we must solve the subgraph isomorphism problem in the context of temporal networks. The definition of subgraph isomorphism for static networks may be presented as follows [24]:

Definition 1. *A graph $G2$ is isomorphic to a subgraph of a graph $G1$ if and only if there is a one-to-one correspondence between the node sets of this subgraph and of $G2$ that preserves adjacency.*

Instead of referring to an "edge" between two nodes, we use the term "interaction" to specify a triplet, made up of two nodes and the time of their contact. We define a directed temporal graph as follows:

Definition 2. *A directed temporal graph G consists of a set V of nodes and a set E of three-tuples denoting interactions. An interaction $e_i \in E$ is represented by $e_i = (u_i, v_i, t_i)$, in which u_i is the source node, v_i is the target node and t_i is the initiation time of the interaction.*

In order for a flow of information or a disease contagion to take place in a temporal network, adjacent interactions must be time-respecting.

Definition 3. *Let e_i and e_j be interactions in a directed temporal graph. The interactions are time-respecting if they are adjacent and $0 \leq |t_j - t_i| \leq d$, for some threshold d. If the interactions do not share a source node or a target node, then either $v_i = u_j$ and $t_i \leq t_j$, or $v_j = u_i$ and $t_j \leq t_i$ must be true.*

Time-respecting paths describe a non-decreasing sequence of interactions [19]. A path can be thought of as a mechanism for passing information from a source, along a sequence of intermediaries, to a target. We aim to find subgraphs composed of these paths in temporal networks.

With traditional time-slicing, the specified time window determines the interactions examined, between a minimum and maximum interaction time. However, a time-respecting path has no such bounds in reality. In fact, given the right connectivity and timing of interactions, a path might be initiated when the network is first created, and continue until the latest point in the data set. Under such circumstances, time-slicing can lose a lot of important context and information.

We define a time-respecting subgraph in terms of time-respecting interactions. We seek query graphs that are connected, so we require that the embedded subgraphs are connected.

Definition 4. *A time-respecting subgraph $S = (V', E')$ of a temporal graph $G = (V, E)$ is composed of a set of nodes $V' \subseteq V$, from which any pair of nodes is connected via a set of interactions $E' \subseteq E$ such that the nodes comprising interactions in E' are in V', and every adjacent interaction pair is time-respecting.*

In our implementation, embedded subgraphs are induced. So, given any pair of nodes in an embedded subgraph, all interactions between them are included in the embedding. So, if a potential embedding includes more interactions than specified by the query graph, the embedding will not be returned.

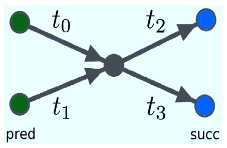


Fig. 1. An example of a time-respecting subgraph. Here, $t_0 \leq t_1 \leq t_2 \leq t_3$. All interactions which are incident to the same node must occur within time d of each other. Thus, we require that $|t_3 - t_0| \leq d$. All incoming edges to a node n must precede all outgoing edges from node n. So, we must have that $t_0 \leq t_2$, $t_0 \leq t_3$, $t_1 \leq t_2$ and $t_1 \leq t_3$.

3.2 The Matching Algorithm

We retain the notation used in the description of the recursive VF2 algorithm by Cordella *et al.* [4]. The matching process is described by a state space representation, in which each state s of the process represents a partial mapping solution. In a state s, a portion of the query graph $G2$ matches a portion of the network graph $G1$. The portion of $G1$ in the mapping is induced. So, given a set of nodes in the mapping, any interactions between them are also present in the mapping.

Given such an intermediate state s, the mapping is extended by first computing candidate node pairs (one node each from $G1$ and $G2$). The candidate node from $G2$ is selected from the set of neighbours of the nodes in $G2$ that are currently in the mapping. This guarantees that the node is connected to the portion of the query graph currently matched. The candidate node from $G1$ is selected in the same way, from the neighbours of the nodes currently matched in the embedding from $G1$, so the embedding will be connected. Once the new nodes are included in the mapping, all interactions between them are also included. The two new, extended portions in the mapping must be graph isomorphic in order to be considered a feasible match. If they are not graph isomorphic, the candidate nodes are discarded as a matching pair, and the process then continues with a new node pair.

If a topological match is confirmed, a semantic match is considered. In our setting, we utilize the dates on which the interactions occur in $G1$. Given an embedding of the subgraph $G2$ in the graph $G1$, we don't require that the dates on each paired interaction match each other, but rather that the partial embedding of $G2$ in $G1$ is time-respecting.. Since we are interested in the actual times at which interactions occurred in the network data, only the semantic feasibility of $G1$ is checked.

The memory requirements of the VF2 algorithm are constrained through the use of data structures which are maintained at each recursion level. We keep track of both topological and temporal information in the same way. A map data structure named *core_1* contains the nodes in the current mapping from $G1$ to $G2$. This provides an efficient way for us to test that a candidate node for inclusion in the mapping will maintain the time-respecting property we require for all of the induced edges.

Before testing the legitimacy of a candidate node *G1_node*, we construct a set of data structures. The list *pred* contains the predecessors of *G1_node* in $G1$ which are also in *core_1*, and thus part of the current mapping. Analogously, *succ* contains the successors of *G1_node* in $G1$ which are also in *core_1*. The lists *pred_dates* and *succ_dates* contain the dates, in increasing order, on which connections between *G1_node* and the relevant predecessor or successor nodes, respectively, were made. The list *dates* combines these dates, sorted in increasing order.

As described in Definition 4, a pairwise comparison of adjacent interactions must ensure that each pair is time-respecting. Accordingly, a candidate node must fulfil these criteria when included in a potential embedding of $G2$ in $G1$.

3.3 Re-assigning Time-stamps

We aim to discover the extent to which the number of embeddings of a query graph in the network is uniquely a property of the temporal aspect of the network. To ascertain this, we repeatedly re-assign the time-stamps on the interactions, and count the number of embeddings again each time. The re-assignment is performed by first stripping all of the time-stamps off the interactions. We then shuffle the order of this time-stamp collection using the shuffle algorithm from Python's built-in *random* module. We then iterate over the entire set of interactions, assigning a time-stamp to each interaction. Thus, the re-assignment is global in scale.

4 Results

To find out whether certain types of time-respecting subgraph are characteristic of real networks, we constructed a temporal network and a set of query graphs in order to perform our experiments. This section details the network data used, the query graphs, our analysis and the results.

4.1 Network Data

The website at Prosper.com [20] provides a forum for prospective borrowers and lenders to connect and exchange funds. Prosper.com allows members to borrow and lend without the presence of a bank. This means that borrowers with low credit-worthiness have a better chance to get loans, since the requirements for being funded are lower. It also gives people a chance to invest smaller amounts, to experiment with lending.

For the purpose of our experiments, we constructed a directed temporal network of lenders and borrowers, connected via loans. An interaction is composed of a source (the lender) and a target (the borrower) and represents the money sent in contribution to a loan request. An interaction also contains the time at which the money was transferred. We set the *d*-value (maximum time allowed between interactions) to 6 days, to reflect the time-scale at which the network operates.

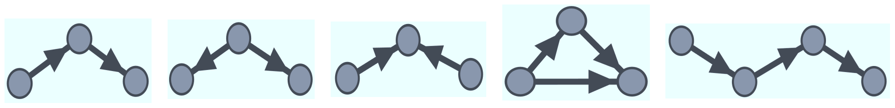
Since the Prosper.com marketplace closed for a period in 2009 due to regulatory issues, we extracted two portions of the network; one before and one after the temporary closure. This allowed us to compare the social behaviours that occurred in the network as a result of different levels of regulation. The details of these networks are listed in Table 1. An important point to note is that the duration of each network is the same, as is the size of each network. So, when the time-stamps are randomly re-assigned, there is the same amount and variation in the time-stamps.

Network	Start Date	End Date	Order	Size
Pre-closure	1^{st} November 2006	31^{st} December 2006	8,690	72,215
Post-closure	1^{st} September 2009	31^{st} October 2009	7,201	77,026

Table 1. The pre-closure network spans the last three months of 2006, while the post-closure network runs from the beginning of September to the end of October in 2009. Both networks have a similar number of interactions, and occur over the same amount of time. This means that shuffling the time-stamps on the interactions ought to have a similar effect in both networks, since the distribution of time-stamps and interactions is comparable.

4.2 Query Graphs

We enumerate some small directed query graphs that have clear interpretations in the context of the Prosper network, illustrated in Fig. 2. These only encode the topological structure of the patterns we are interested in. When we examine their topological embeddings in the network, we also check that the embeddings are time-respecting, so the notion of non-decreasing activations along the paths in the query graphs is maintained.



(a) 2-path (b) 2-out-star (c) 2-in-star (d) feed-forward (e) 3-path

Fig. 2. The query graphs sought in the Prosper network. Each node represents a member of the Prosper marketplace, and each interaction represents a sum of money being transferred via a loan. These queries were chosen since their structure is clear in the context of the network data.

4.3 Analysis

The experiments were performed on a Linux server with a 2 GHz processor, limited to 5GB of physical memory. The algorithms we proposed for performing the subgraph matching and the re-assignment of time-stamps were implemented in the programming language Python [21], using the NetworkX library [18]. The VF2 algorithm is included in this library, and was implemented as part of a project at the Complexity Sciences Center and Physics Department, UC Davis. We extended this implementation to process temporal networks and use temporal information during the matching process. Our implementation can handle directed graphs as well as directed multigraphs (graphs with multiple interactions between nodes).

The results of our experiments are listed in Table 2. In both the pre- and post-closure network, the 2-in-star and 2-out-star queries had the highest number of embeddings. This is likely to be a result of how the Prosper marketplace is used; by members who either exclusively borrow or lend. Borrowers have a high in-degree, since the loans they request are funded from many sources, who all give relatively small amounts. Lenders have a high out-degree, since they need to distribute their lending portfolio over a range of borrowers in order to make a more reliable profit.

When the time-stamps are randomly re-assigned in the pre-closure network, the number of embeddings of the query graphs drops between 20% and 89.1%. This strongly suggests that the actual timing of the interactions was important

Network	Structure	Count	Mean	Std. Dev.	Decrease
Pre-closure	2-path	15,061	11,717.8	166.4	22.2%
	2-out-star	1,083,154	819,451.4	3,826.1	24.3%
	2-in-star	5,209,926	1,083,642.2	2,215.9	79.2%
	feed-forward	1,130	123.0	12.5	89.1%
	3-path	1,584	1,267.5	181.5	20.0%
Post-closure	2-path	6,237	6,411.4	179.8	-2.8%
	2-out-star	1,064,034	786,908.3	2,784.4	26.0%
	2-in-star	17,900,180	3,806,272.3	9,061.4	78.7%
	feed-forward	1,516	217.0	20.7	85.7%
	3-path	825	847.0	219.7	-2.7%

Table 2. Comparing the number of embeddings found in the pre- and post-closure networks. The count lists the number of embeddings with the original time-stamps in place. The mean shows the average over 100 separate counts of the number of embeddings after randomly re-assigning the time-stamps every time. In the pre-closure network, embedding counts dropped after shuffling the time-stamps. The same was true in the post-closure network, except for the path query graphs, indicating that their presence in the network is not necessarily a property of the original network.

for the processes to take place. The greatest drop in the number of embeddings occurs with the queries containing a higher in-degree. This makes sense, since a borrower needs to get funds from multiple lenders at around the same time for a loan to go ahead. If the time-stamps are shuffled, this condition may not be met. This demonstrates the effectiveness of our strategy; real social behaviour in the network is shown to be dependent on interaction timing.

The most interesting results relate to the path queries in the post-closure network. An intermediate node in a path may represent an arbitrageur. An arbitrageur aims to profit from the difference in interest rates between the loan taken on and the loans given to borrowers. The timing of this sequence of loans is important for the arbitrage to be successful. The different results for the pre- and post-closure network indicate the influence of greater regulation within the marketplace. Specifically, the number of embeddings of path queries does not decrease when the time-stamps are re-assigned. Thus, the existence of the time-respecting paths in the original network does not reveal a process that is unique to the network. This is consistent with the fact that stronger regulation may have discouraged arbitrage. In the case of money laundering, the existence of intermediate individuals is also a possibility. So, this result also indicates that if attempts at money laundering occurred in the pre-closure network, it was discouraged by greater regulation.

5 Conclusions and Future Work

The primary aim of this work is to evaluate the importance of temporal information in a network for identifying the processes that underly the network topology. Specifically, we examined the network from Prosper.com to see if the existence of some suspicious patterns was dependent on the time at which the interactions which made up the pattern took place. To do this, we specified some query graphs to search for in the network and counted their time-respecting embeddings. Then, we randomly re-assigned the time-stamps and counted the embeddings again. We did this latter step 100 times, and took the average of the counts. Almost all the counts dropped in comparison with the actual network.

Since Prosper.com closed due to regulatory issues in 2009, we compared a portion of the pre- and post-closure network to see if there was a change in behaviour. The query graphs associated with arbitrage or potentially money laundering behaviour were prevalent in the pre-closure network, but not so in the post-closure network. This was revealed by the fact that, after temporal randomization, the number of embeddings dropped in the pre-closure network, but did not change in the post-closure network. This is likely to be an effect of increased regulation on the lending platform.

The time at which interactions take place is a key component of network formation, and can help to explain many types of emergent social behaviour. In future, we aim to apply these methods to other networks which contain temporal information, especially networks which operate at a finer temporal grain. This will help to validate the performance of our algorithm, and may give an insight into which processes play a significant role in the networks in question. Given our prior knowledge of the Prosper network, our validation of what constitutes an interesting pattern is intuitive. In future, a method for automatically extracting over-represented patterns would overcome this dependency and potentially yield unforeseen network behaviour.

References

1. Baños, R.A., Borge-Holthoefer, J., Moreno, Y.: The role of hidden influentials in the diffusion of online information cascades. arXiv preprint arXiv:1303.4629 (2013)
2. Bearman, P.S., Moody, J., Stovel, K.: Chains of affection: The structure of adolescent romantic and sexual networks. American Journal of Sociology 110(1), 44–91 (2004)
3. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: Performance evaluation of the VF graph matching algorithm. In: Image Analysis and Processing, 1999. Proc. Intl Conf. on. pp. 1172–1177. IEEE (1999)
4. Cordella,L. P., Foggia,P., Sansone, C., Vento, M.: An Improved Algorithm for Matching Large Graphs. 3^{rd} IAPR-TC15 Workshop Graph-Based Representations in Pattern Recognition pp. 149–159 (2001)
5. Corneil, D.G., Gotlieb, C.C.: An efficient algorithm for graph isomorphism. Journal of the ACM (JACM) 17(1), 51–64 (1970)
6. Gary, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of np-completeness (1979)
7. Ghosh, R., Lerman, K.: A framework for quantitative analysis of cascades on networks. In: Proc. of the fourth ACM Intl. conf. on Web search and data mining. pp. 665–674. ACM (2011)
8. Holme, P., Saramäki, J.: Temporal networks. Physics reports 519(3), 97–125 (2012)

9. Hopcroft, J.E., Wong, J.K.: Linear time algorithm for isomorphism of planar graphs (preliminary report). In: Proc. of the sixth annual ACM symposium on Theory of computing. pp. 172–184. ACM (1974)
10. Kempe, D., Kleinberg, J., Kumar, A.: Connectivity and inference problems for temporal networks. In: Proc. of the thirty-second annual ACM symposium on Theory of computing. pp. 504–513. ACM (2000)
11. Kivelä, M., Pan, R.K., Kaski, K., Kertész, J., Saramäki, J., Karsai, M.: Multiscale analysis of spreading in a large communication network. Journal of Statistical Mechanics: Theory and Experiment 2012(03), P03005 (2012)
12. Kovanen, L., Karsai, M., Kaski, K., Kertész, J., Saramäki, J.: Temporal motifs in time-dependent networks. Journal of Statistical Mechanics: Theory and Experiment 2011(11), P11005 (2011)
13. Kovanen, L., Kaski, K., Kertész, J., Saramäki, J.: Temporal motifs reveal homophily, gender-specific patterns and group talk in mobile communication networks. Proc. of the National Academy of Sciences 110(45), 18070–18075 (2013)
14. Luks, E.M.: Isomorphism of graphs of bounded valence can be tested in polynomial time. Journal of Computer and System Sciences 25(1), 42–65 (1982)
15. McKay, B.D.: Practical graph isomorphism. Department of Computer Science, Vanderbilt University (1981)
16. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298(5594), 824–827 (2002)
17. Moody, J.: The importance of relationship timing for diffusion. Social Forces 81(1), 25–56 (2002)
18. NetworkX Developers: NetworkX. networkx.github.io (2013)
19. Pan, R.K., Saramäki, J.: Path lengths, correlations, and centrality in temporal networks. Physical Review E 84(1), 016105 (2011)
20. Prosper Marketplace Inc.: Personal Loans and Online Investing – Peer to Peer Lending – Prosper. http://www.prosper.com/ (2013)
21. Python Software Foundation: Python. www.python.org (2012)
22. Redmond, U., Cunningham, P.: A temporal network analysis reveals the unprofitability of arbitrage in the prosper marketplace. Expert Systems with Applications (2012)
23. Schmidt, D.C., Druffel, L.E.: A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. Journal of the ACM (JACM) 23(3), 433–445 (1976)
24. Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM (JACM) 23(1), 31–42 (1976)
25. Zhao, Q., Tian, Y., He, Q., Oliver, N., Jin, R., Lee, W.C.: Communication motifs: a tool to characterize social communications. In: Proc. of the 19th ACM Intl conf. on Information and knowledge management. pp. 1645–1648. ACM (2010)

Conversations on Twitter: Structure, Pace, Balance

Danica Vukadinović Greetham[1] and Jonathan A. Ward[2]

[1] Centre for the Mathematics of Human Behaviour
Department of Mathematics and Statistics
University of Reading, UK
d.v.greetham@reading.ac.uk

[2] Department of Applied Mathematics
University of Leeds, UK
j.a.ward@leeds.ac.uk

Abstract. Twitter is both a micro-blogging service and a platform for public conversation. Direct conversation is facilitated in Twitter through the use of @'s (mentions) and replies. While the conversational element of Twitter is of particular interest to the marketing sector, relatively few data-mining studies have focused on this area. We analyse conversations associated with reciprocated mentions that take place in a data-set consisting of approximately 4 million tweets collected over a period of 28 days that contain at least one mention. We ignore tweet content and instead use the mention network structure and its dynamical properties to identify and characterise Twitter conversations between pairs of users and within larger groups. We consider conversational balance, meaning the fraction of content contributed by each party. The goal of this work is to draw out some of the mechanisms driving conversation in Twitter, with the potential aim of developing conversational models.

Keywords: Twitter mentions networks, conversations models, maximal cliques

1 Introduction

The rapid uptake of online social media, combined with consumer behavioural changes around television and news broadcasting, has instigated a sea change in attitudes within the advertising and marketing sectors. A frequently encountered adage is that "everything is about conversation and not about broadcasting" [10,6]. By facilitating public addressability through the @ sign (so called 'mentions') and enabling private messages, Twitter has confirmed their intention to function as a communication channel as well as a broadcasting tool. Access to large quantities of data produced by Twitter users has resulted in a surge of interest from the academic community [20], who have largely focused on Twitter's information flow and retweet behaviour, and hence implicitly the underlying network of 'followers' (e.g. [22,21]). While broadcasting short messages, or micro-blogging, remains an important component of Twitter use, to our knowledge comparatively little work has addressed the mining of (public) conversations on a large scale [3,19,14]. Consequently, we focus in this paper on analysing the network of communication patterns resulting from mentions in Twitter.

Although it may not always be clear, even from message content, what intention a user had in mind when posting—information seeking or information sharing, broadcasting or conversation—we have tried to specifically extract conversations by focusing our data-analysis on reciprocated tweets. Moreover, we have completely ignored the content of conversations and concentrated on structural and dynamic properties of the underlying mentions network. Our main objective was to mine actionable insights that could inform our knowledge of conversational mechanisms and the frequency/timings of tweets. Our hope is that empirical observations and quantifiable insights from this analysis could inform a simple, data driven model of the timing and structure of Twitter conversations. One possible application would be for automated recommendations of conversation trends, as discussed in [3,1].

A large number of registered Twitter accounts are operated by automated software scripts, known as *bots* [18]. While such accounts are encouraged for the purpose of developing applications and services, bots whose functions violate Twitter policy (e.g. spammers) are common. The analysis of conversational patterns and the development of associated models have potential application for those trying to develop algorithms that can identify nuisance bots. Furthermore, the identification of groups of Twitter users who, through conversational behaviour, are particularly influential on a specific topic would be particularly attractive in the marketing sector. Thus, understanding conversational structure could impact the design and implementation of social media campaigns and potentially provide a quantitative comparison between Twitter discourse and other channels of communication, such as face-to-face, telephone, SMS, forums or email. In addition, curating and recommending conversational trends, for both Twitter and more generally in online social media, is crucial for social networking sites as it is one of the main characteristics of user experience. We believe that a better understanding of the structure, dynamics and balance of multi-user conversation is key to improving such automated curation systems. Ultimately, we hope that studying Twitter conversation can ultimately improve user experience.

In Section 2, we give an account of previous work in this space. Our results of pairwise and multiple conversations and the Twitter dataset we used are presented in Section 3. Finally, in Section 4 we summarise and describe possible directions of future work.

2 Previous work

The phenomenal uptake of Twitter over the last few years has resulted in a rapidly growing interest in mining Twitter data and particularly sentiment analysis of tweets. A recent study analyzing a large amount of Twitter and Facebook data [12] found correlations between friendship/follower relations and positive/negative moods of Twitter users. Diurnal and seasonal mood rhythms that are common across different cultures have also been identified in cross-cultural Twitter data [5], shedding light on the dynamics of positive and negative affect.

A study of conversations within a sample of 8.5k tweets collected over an hour long period [9] found that the @ sign appeared in about 30% of the collected sample, its function was mostly for addressing (as intended) and it was relatively well reciprocated—around 30% of messages containing an @ were reciprocated within an hour. The majority of these conversations were short, coherent exchanges between two people, but longer exchanges did occur, sometimes consisting of up to 10 people. They found that

> *"…Tweets with @ signs are more focused on an addressee, more likely to provide information for others, and more likely to exhort others to do something—in short, their content is more interactive. "*

Twitter conversations also contain both momentarily salient or 'peaky' topics, signified by increased word-use frequency of specific terms, as well as more 'persistent conversations', in which less salient terms recur over longer periods [14]. In addition, words that relate to negative emotions are less persistent [22].

In [3], several algorithms for recommending conversations based on the lengths, topic and 'tie-strength'[3] of conversations were compared. Their results showed that the different uses of Twitter (social vs. informational) had a big influence on the algorithm's performance — recommendations based on tie strength were preferred by social users, whilst those based on topic were preferred by informational users. Related work considered automated curation of online conversations to present discussion threads of interest to users in e.g. Facebook and Google+. [1]. Key to this was the prediction of conversation length around a topic and re-entry of interlocutors. In another work concerning Twitter conversation [13], a relatively large corpus and content (topic) analysis of 1.3 million tweets was used to develop an unsupervised model of dialogue from open-topic data.

In our work we completely ignore content, instead focusing on timing, structure and balance of conversation between pairs of individuals as well as multi-user conversations. Our contribution is an attempt to map the structure of Twitter exchanges over a relatively large dataset, while offering some new methods to mine conversation data and improve statistical models of dialogue.

3 Analysis

3.1 Data

The Twitter data-set investigated in this paper was collected on our behalf by Datasift, a certified Twitter partner, allowing us to access the full Twitter

[3] Tie-strength is an increasing function of the number of exchanged messages between two people and the number of messages exchanged between them and their mutual friends.

firehose rather than being rate-limited by the API. The data-set consists of all UK based[4] Twitter users that sent tweets with at least one mention between 8 Dec 2011 and 4 Jan 2012 (28 days in total). In the remainder of the paper, use of the word 'tweet' will specifically mean tweets containing at least one mention. Mentions are messages that include an @ followed by a username. Thus if person a puts "@b", it designates that a is addressing the tweet to b specifically. Mentions are not private messages and can be read by anyone who searches for them. A tweet can be addressed to several users simultaneously using @ repetitively. Any Twitter user can mention any other Twitter user, they don't have to be related in any way. Since conversational characteristics are influenced by many factors, including language, culture, community membership etc., one has to keep in mind the natural limitations of the results of our analysis.

We preprocessed the data, removing empty mentions and self-addressing[5] and created a directed multigraph, or mentions network, containing $3,614,705$ timestamped arcs (individual mentions) from a total of $819,081$ distinct usernames, or nodes. Of these distinct usernames, $732,043$ were "receivers", i.e. to whom a message was addressed, and $137,184$ were "tweeters", i.e. people who tweeted a message with a mention. There were approximately 50k nodes that appeared both as tweeters and receivers. Note that our graph is a multigraph, meaning that multiple arcs are allowed between pairs of nodes, each having a direction and timestamp.

3.2 Conversations

An important feature of both face-to-face conversation [16,15] and computer-mediated communication [8], is the process of turn-taking. Thus in sequences of mentions between pairs of users, say a and b, we might expect that sequences like $ABABAB$ would be more common than say $AAABBB$, where we use A to denote that party a mentions party b and likewise B to denote that party b mentions party a.

To establish if this is the case, we assume the null hypotheses that contributions are independent events with probability P_A that party a contributes to a conversation and thus probability $P_B = 1 - P_A$ that party b contributes. For a given interaction sequence of length N between parties a and b, we are interested in the number of occurrences of B following A and vice-versa. We call these *transitions*, thus the sequence $ABAABBA$ of length $N = 7$, has 4 transitions. Note that we focus on reciprocated interactions, meaning that each party makes at least one contribution and consequently that there is by default at least one transition in all interactions that we consider. We call the remaining transitions the *excess transitions*. For any sequence of length N, the maximum possible number of excess transitions is clearly $N - 2$. Under the null hypotheses, excess

[4] All Twitter users appearing in our data-set had selected the UK as their location.

[5] Self-mentioning was surprisingly common in the data-set: 12,680 different users created a total of 44,319 self-mentions, with the maximum being 5,586 from an automated service that advertises itself at the end of each tweet.

transitions occur with probability $P_T = 2P_A(1 - P_A)$. Since we assume that transitions are independent, the probability distribution of a given number of excess transitions is binomial, and thus the expected number is $E_T = (N-2)P_T$ with variance $V_T = (N-2)P_T(1-P_T)$.

To test the null hypothesis, we consider all reciprocated pairwise interaction sequences in our Twitter data-set. For each sequence having n_X contributions from party $X \in \{A, B\}$, we assume that the probability of party a contributing is simply $n_A/(n_A + n_B)$. This does not yield any problematic probabilities (i.e. 0 or 1) since both parties always make at least one contribution.

Each sequence may have a different number of interactions and a different transition probability, but assuming that the pairwise interactions are independent, the expectation and variance of the ensemble is simply equal to the sum of the interaction expectations and variances respectively. Doing this, we find that the expected number of transitions is 85,390 with a standard deviation of 226.3, but we observe 88,758 transitions in practice, more than 15 standard deviations above the expected value. We take this as strong evidence that we can reject the null hypothesis and thus infer that the data contains a significant level of turn-taking and hence conversation.

Each sequence of pairwise interactions may constitute a number of different conversations, but ascertaining when one conversation ends and another begins may be an extremely difficult task, especially when the goal is to apply an automated processes to a large data-set. Instead of using a time-intensive lexical analysis, we investigate whether we can detect conversations by applying a simple threshold rule to the time gap between responses, where we assume that a time gap that is larger than the threshold indicates the start of a new conversation.

This method requires that we can identify a suitable threshold. To achieve this, we divide each sequence of pairwise interactions up according to a given threshold, then define distinct conversations to be reciprocated sub-sequences, i.e. sequences containing a contribution from both parties. Thus the number of sub-sequences n_{I} is always larger than the number of distinct conversations n_{C}. In Fig. 1(a) and (b) we plot the mean number of sub-sequences and the mean number of distinct conversations respectively over a range of threshold values. The number of distinct conversations n_{C} has a peak value at approximately 9hrs. This peak is expected, since we only count reciprocated interactions as distinct conversations. Thus small threshold values, which split an interaction sequence up into a large number of short sub-sequences (see Fig. 1(a)), result in relatively few distinct conversations because many of the sub-sequences feature contributions from only one party. High threshold values also result in a small number of conversations, but this is simply because they do not split the sequence up into many sub-sequences. Thus the maximum at 9hrs is a natural choice of threshold and corresponds to one's intuition that conversations may reflect diurnal patterns.

The mean and median number of tweets during conversations were 13.09 and 4 respectively, but the distribution was heavy tailed (see Fig. 2).

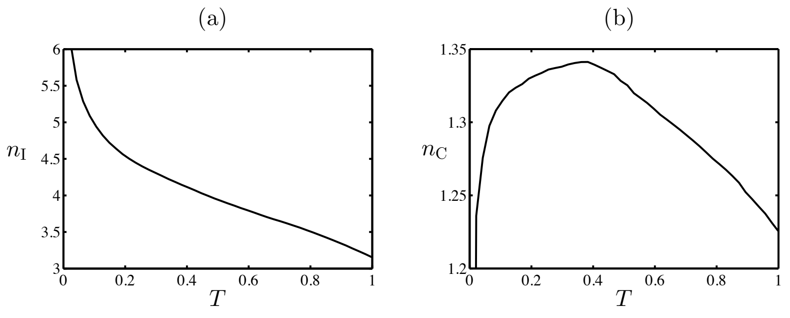


Fig. 1: Panel (a): Mean number of subsequences for a range of threshold values. Panel(b): Mean number of distinct conversations for a range of threshold values. Note that T, time threshold in hours, is normalised on the x-axis.

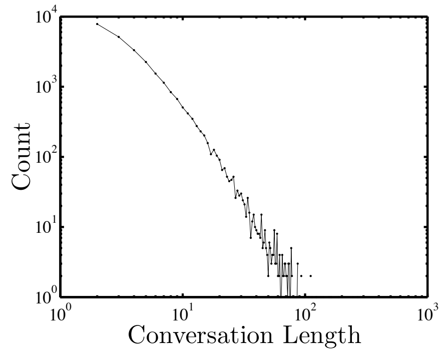


Fig. 2: Distribution of conversation length.

We now consider whether the number of contributions from each party are similar, or 'balanced' within pairwise interactions and conversations. For a given sequence of tweets, there are two ways to compute balance, we can either consider the ratio of means $b = \langle \max(n_A, n_B) \rangle / \langle \min(n_A, n_B) \rangle$ or the mean of ratios $\beta = \langle \max(n_A, n_B) / \min(n_A, n_B) \rangle$. We will use the subscripts 'I' and 'C' to denote whether these have been calculated for interactions or conversations respectively. Since we only consider reciprocated interactions, both quantities are well-defined and we would generally expect $b < \beta$. For the total number of interactions between pairs, we find that $b_{\mathrm{I}} = 2.424$ and $\beta_{\mathrm{I}} = 3.457$. Thus on average, one party contributes around 3 times as much as the other. For the sub-set of conversations, we find that $b_{\mathrm{C}} = 1.148$ and $\beta_{\mathrm{C}} = 1.425$. These are much closer to 1, and hence more what we would expect from typical, balanced conversations. The distribution of conversation contribution ratios is plotted in Fig. 3(a), which illustrates that conversations are most likely to be balanced, but some extremely unbalanced conversations do occur. In Fig. 3(b), for each

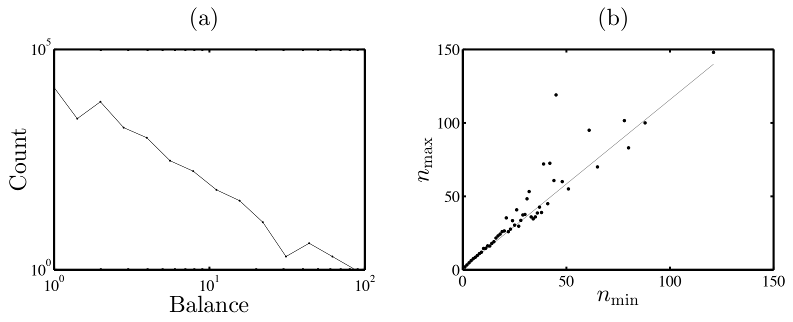


Fig. 3: Panel (a): Distribution of conversation balance. Panel (b): Mean maximum conversation contribution as a function of minimum contribution.

minimum conversation contribution $n_{\min} = 1, 2, 3, \ldots$, we compute the mean of the maximum contribution $n_{\max}$. There is a roughly linear trend (the grey line is $n_{\max} = 1.148n_{\min} + 1$), which further illustrates conversational balance.

3.3 Multi-user conversations

By allowing multiple @ signs in one message, a Twitter user could send a tweet to several recipients simultaneously, facilitating multi-user conversations or *multicasting*. Note that because of the 140 character limit there is a physical limit on how many users each message can be multicast to.

In this part of analysis, our aim is to

– Identify multi-users exchanges;
– Determine how many users typically engage in them;
– Identify their time-frame, pace and how balanced they are.

In addition, are all users equally involved, or do some dominate the discussion? Are the same people at the heart of different multi-user conversations? What are the enablers and inhibitors of conversation flowing in the sense of pauses between consecutive contributions?

3.4 Identification of multi-users conversations

The reciprocated mentions data represents a directed multi-graph G (where an edge from A to B implies at least one edge from B to A), thus multi-user exchanges correspond to strongly-connected[6] subgraphs of G with $k > 2$ participants. We ran a non-recursive version of Tarjan's algorithm [17,11], as

[6] A directed graph is called strongly-connected if there is a path from each vertex in the graph to every other vertex. This means that for two vertices a and b there is a path in both directions, i.e. from a to b and also from b to a. Strongly-connected components of a graph are maximal subgraphs that are strongly-connected.

implemented in NetworkX [7], to get a list of the strongly-connected components of G. Pairwise conversations were discussed in Section 3.2, so we excluded all strongly-connected components of size 2 from the present analysis. Each strongly-connected component of at least three vertices was then transformed into an *undirected* multi-graph and we ran the NetworkX implementation of the modified Bron's algorithm [2] to find all maximal cliques[7]. We then disregarded all cliques of size two. We found in total 2190 cliques of size 3, 4, 5 and 6. The total number of users in these cliques was 3275 which is around 20% of users who reciprocated mentions.

In order to take the time elapsed between consecutive messages into account, we use the same threshold method explained in subsection 3.2, this time demanding for an exchange to be a"conversation" that there is a contribution from all parties and got relatively similar results (see Fig 4). The number of exchanges which had contribution from all parties was at peak around 9 and 11 hours. We took a threshold of 9 hours which gave us 334 multiuser conversations of sizes 3, 4, and 5 (see Fig 5a).

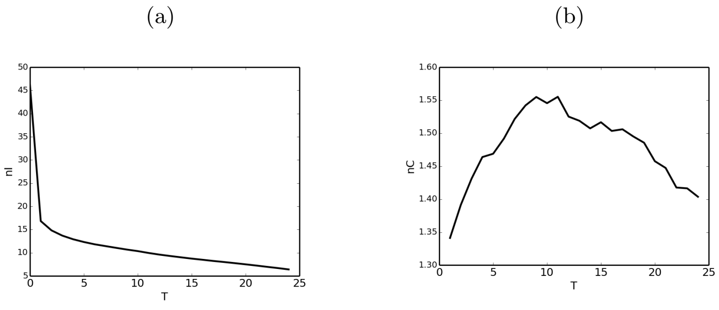


Fig. 4: Panel (a): Mean number of subsequences for a range of threshold values. Panel (b): Mean number of distinct conversations for a range of threshold values (threshold T in hours).

Most users (out of 646) in our dataset were involved in just one multi-user conversation, but a small number were involved in multiple conversations. The users' involvement in multi-user conversation is illustrated in Fig 5b.

When examining the time-frame of multi-user exchanges, we found that the correlation coefficient between the total number of exchanges between clique members and the average difference between consecutive exchanges was -0.244 (see Fig 6a). This was not surprising, since we would expect lively conversations (with lots of exchanged messages) to have a relatively fast pace, in contrast to a casual exchange of messages with longer differences inside our chosen 9 hour time-window. The same picture is obtained from looking at the median time differences between consecutive messages across different clique sizes (see Fig

[7] Maximal cliques are the largest complete subgraphs containing a given node.

(a) (b)

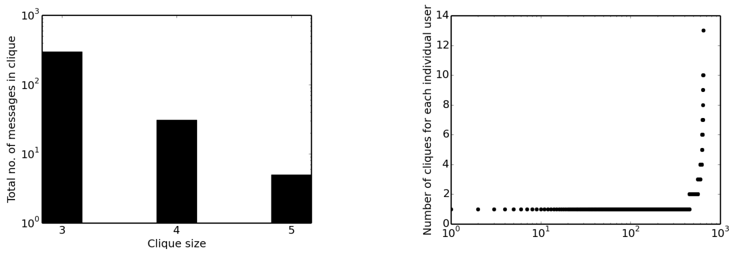


Fig. 5: Panel (a): A size of cliques versus a number of instances (log y-axis). Panel (b):Number of cliques individual users were involved in (log x axis).

6b). We also investigated how balanced multi-user exchanges were, although

(a) (b)

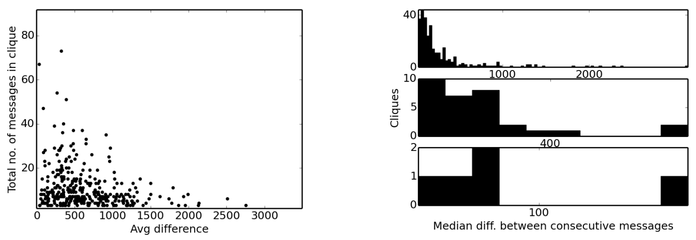


Fig. 6: Panel (a): Average difference in seconds between two consecutive messages in clique versus total number of exchanges. Panel (b): Histogram of medians of differences in seconds between two consecutive messages for cliques of size 3, top, size 4, middle and size 5 bottom.

this situation is more complicated than in the pairwise case.

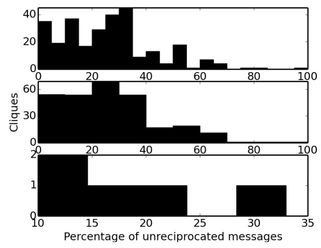
Firstly, we looked at the difference between the number of tweets received and sent by individual clique members. For each node, we computed the difference of their in-degree and out-degree. We summed up the positive values[8] and to normalise, we divided by the total number of exchanged messages. In this way, we obtained a percentage of ‘unreciprocated’ messages, where reciprocity is not

[8] Clearly the number of sent and received messages within a group are equal, thus summing the differences between in- and out-degree over individual members in the group is by definition equal to zero.

toward a sender but toward a whole group. We show the histograms for the different sizes of cliques in Fig. 7a. Across all clique sizes and in most of the multi-user conversations around 30% messages were unreciprocated. In a small number of conversations of 3 or 4 users a larger percentage were unreciprocated, i.e. they were dominated by certain members, but also a large number of cliques were very balanced (with unreciprocated messages at $0 - 10\%$), meaning every individual received and sent a similar number of tweets.

Finally, we looked at so-called 'floor-gaining' [4], i.e. how much input each user had over the course of a group exchange[9]. We compared the out-degree of each user within a clique, (remember that each clique is a directed multigraph) with the mean number of edges $r = |n_{\mathrm{E}}|/|n_{\mathrm{V}}|$, where n_{E} is the total number of edges within the clique and n_{V} is the total number of vertices within the clique. In a 'round robin' group conversation, with balanced turn taking, each user would send out r messages, i.e. be responsible for an equal percentage $p = 100r/e$ of the total number n_{E} of exchanged messages. For each clique size, we looked at how many users' representations were greater than or equal to p, i.e. those users who 'dominate' the conversation. On Fig 7b below, we present the histogram for a number of dominant users in the cliques of size 3, 4 and 5. This shows that in

(a)



(b)

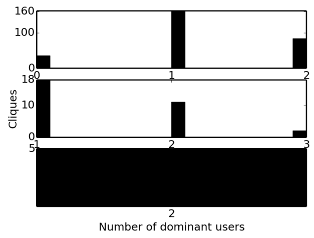


Fig. 7: Panel(a): The percentage of 'unreciprocated' messages for cliques of size 3, top, size 4, middle and size 5 bottom. Panel (b): A number of dominant users in cliques of size 3, top, size 4, middle and size 5 bottom.

most of the cliques of size 3 and 4, one user was responsible for the majority of communication, whilst in cliques of size five, 2 users were dominant. However in about 13% of all cliques of size 3 no users dominated, confirming that Twitter is used for multi-user conversations and not just pairwise conversations.

[9] We argue that the action of tweeting in multiuser exchanges can be regarded as floor-gaining, since tweets with mentions can in principal be read by a wider audience than the group conversing.

4 Conclusions

We looked at conversations in Twitter, based on the underlying structure and timings in approximately 4 million UK tweets with mentions over a period of 28 days. We structured the data as a multigraph to make use of graph algorithms. We proposed a simple method of identifying conversations between pairs of users, based on a time-threshold on the time-to-next tweet, and found evidence that a threshold of 9hrs gives a good indication of distinct conversations. We observed that the conversations detected using this method appeared to be balanced, meaning that each party involved contributed approximately equally to the conversation. This was not the case within more general interactions, in which one agent typically contributed around three times as much as the other.

Although finding cliques in graphs is computationally demanding, because of the sparsity of interactions patterns within the data-set, extracting multi-user exchanges was feasible and relatively fast. We were able to find all cliques within the graph and, using the threshold method, identify conversations for up to a maximum of 5 users. Most of those exchanges were fast-paced. We also found that the number of messages in multi-user exchanges was reciprocal to the average time difference between them. When looking at the balance of multi-user conversations, we found that most exchanges are dominated by just one or two users, with some evidence of well-balanced group exchanges in between 3 users. Regarding the number of received and sent messages by each individual in a group, we found that some were dominated by one or two users, but also some were well balanced.

Further work needs to be done using content information to explore how topics flow through multi-user exchange and if there is any relationship between time-differences between messages and topic. We hope that the insights gained from our analysis could help to develop an understanding of the mechanisms and dynamics of Twitter conversations, with potential scope for generating models of micro-blogging behaviour.

Acknowledgment

This work is partially funded by the RCUK Digital Economy programme via EPSRC grant EP/G065802/1 'The Horizon Hub' and EPSRC MOLTEN EP/I016031/1. We would like to thank Datasift for the provision of the data analysed, and to Colin Singleton and Bruno Gonçalves for very useful feedback and comments.

References

1. L. Backstrom, J. Kleinberg, L. Lee, and C. Danescu-Niculescu-Mizil. Characterizing and curating conversation threads: expansion, focus, volume, re-entry. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 13–22, New York, NY, USA, 2013. ACM.

2. C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, Sept. 1973.
3. J. Chen, R. Nairn, and E. Chi. Speak little and well: recommending conversations in online social streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 217–226, New York, NY, USA, 2011. ACM.
4. C. Edelsky. Who's got the floor? *Language in Society*, 10:383–421, 1981.
5. S. Golder and M. W. Macy. Diurnal and seasonal mood tracks work sleep and daylength across diverse cultures. *Science*, 333:1878–1881, 2011.
6. D. Graham. Twitter: value-added conversation is more important than broadcasts, rts. memeburn, August 2012.
7. A. Hagberg, D. Schult, and P. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference (SciPy2008), Passadena, CA USA*, pages 11–15, August 2008.
8. S. Herring. Computer-mediated discourse. In D. Tannen, D. Schiffrin, and H. Hamilton, editors, *Handbook of Discourse Analysis*. Oxford: Blackwell, 2001.
9. C. Honeycutt and S. C. Herring. Beyond microblogging: Conversation and collaboration via twitter. In *Proceedings of the Forty-Second Hawai'i International Conference on System Sciences, Los Alamitos*, 2009.
10. G. Leboff. Marketing is a conversation? oh yeah ...about what? The Marketing Donut, October 2011.
11. E. Nuutila and E. Soisalon-Soinen. On finding the strongly connected components in a directed graph. *Information Processing Letters*, 49(1):9–14, 1994.
12. D. Quercia, L. Capra, and J. Crowcroft. The social world of twitter: Topics, geography, and emotions. In *The 6th international AAAI Conference on weblogs and social media, Dublin*, 2012.
13. A. Ritter, C. Cherry, and B. Dolan. Unsupervised modeling of twitter conversations. In *HLT-NAACL 2010*, 2010.
14. D. A. Shamma, L. Kennedy, and E. F. Churchill. Peaks and persistence: modeling the shape of microblog conversations. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 355–358, New York, NY, USA, 2011. ACM.
15. J. Sidnell. *Conversation Analysis: An Introduction*. Blackwell, 2010.
16. D. Starkey. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23(2):283–292, 1972.
17. R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.
18. K. Thomas, C. Grier, V. Paxson, and D. Song. Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 243–258. ACM, 2011.
19. C. Wang, M. Ye, and B. A. Huberman. From user comments to on-line conversations. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 244–252, New York, NY, USA, 2012. ACM.
20. S. Williams, M. Terras, and Warwick. What people study when they study twitter: Classifying twitter related academic papers. *Journal of Documentation*, 69, 2013.
21. S. Wu, J. Hofman, W. Mason, and D. Watts. Who says what to whom on twitter. In *WWW, 2011*, 2011.
22. S. Wu, C. Tan, J. Kleinberg, and M. Macy. Does bad news go away faster? In *5th International AAAI Conference on Weblogs and Social Media, 2011*, 2011.

Collaborative Filtering based on Dynamic Community Detection

Sabrine Ben Abdrabbah, Raouia Ayachi, and Nahla Ben Amor

LARODEC, Université de Tunis, ISG Tunis,
2000, Bardo, Tunisia
abidrabbah.sabrine@gmail.com,raouia.ayachi@gmail.com,nahla.benamor@gmx.fr

Abstract. *With the increase of time-stamped data, the task of recommender systems becomes not only to fulfill users interests but also to model the dynamic behavior of their tastes. This paper proposes a novel architecture, called Dynamic Community-based Collaborative filtering (D2CF), that combines both recommendation and dynamic community detection techniques in order to exploit the temporal aspect of the community structure in real-world networks and to enhance the existing community-based recommendation. The efficiency of the proposed D2CF is dealt with a comparative study with a recommendation system based on static community detection and item-based collaborative filtering. Experimental results show a considerable improvement of D2CF recommendation accuracy, whilst it addresses both of scalability and sparsity problems.*

Keywords: Recommendation systems, Collaborative filtering, Dynamic community detection, Time varying graphs

1 Introduction

Several types of recommenders have been proposed and can be categorized into three major categories [16], namely content-based filtering, collaborative filtering, and hybrid approaches.

In this work, we focus on collaborative filtering which predicts users interests/preferences from those of remaining users sharing similar tastes. Collaborative filtering is considered as one of the most used techniques due to its efficiency and its high accuracy . Typically, the recommendation process for this technique starts when users express their preferences by rating items. The system analyzes these ratings to determine the exact preferences of the user, then, matches the active user's preferences and the preferences collection to discover the category of users having similar taste with the active user. Finally, the system recommends a set of items for the active user according to the preferences of their similar users.

From another side, the community detection presents a growing interest for many researchers, especially in web applications. A panoply of community detection algorithms exist in literature and most of them focus on static community detection, but recently the dynamic aspect of networks has sparked a new line of research. Static community detection algorithms have been explored in collaborative filtering, based on the idea that community structure will enhance the performance of recommendations [1, 3–6]. Nevertheless, these ones are not able to deal with the dynamic aspect of real-world networks.

The purpose of this paper is to model the dynamic behavior of users in recommender systems. We assume that users behaviors are learned from users' ratings data and more specifically by looking firstly at the items which have been rated by each user and then finding how to link items to each other over time in order to build the dynamic network of items. Our major contribution consists in presenting a novel approach named Dynamic Community-based Collaborative Filtering (denoted D2CF for short) capturing dynamic communities of items which present the evolution of users interests and preferences over time to over recommendations more suitable for real-world networks.

This paper is organized as follows: Section 2 presents the basic concepts of recommendation and community detection. Related work is provided in Section 3. Our proposed architecture D2CF is presented in Section 4. Section 5 is dedicated to the experimental study.

2 Basic concepts for recommendation and community detection

This section gives a brief overview on both recommendation systems and community detection.

2.1 Recommendation

Recommender systems have been proposed to address the information overload problem by filtering the relevant data and suggesting items of potential interest to users. Formally, in a typical recommendation system, there is a set of users U and a set of items I. The task of recommender system is to predict user's preferences P for each item in I. The output is a list L containing items with the highest preference values. Content-based filtering, collaborative filtering and hybrid approaches are three major categories of recommendation methods:

- *Content-based approach* selects items based on their content along user's profile. Its principle is to recommend items similar to the ones that the user has preferred in the past.
- *Collaborative filtering approach* infers user's preferences from remaining users having similar tastes. Methods pertaining to this approach can be divided into user-based and item-based methods.
- *Hybrid approach* combines content-based and collaborative filtering methods, in other words it takes into account both the users and the items properties.

In this work, we will focus on the most widely used recommendation method, namely Item-based Collaborative Filtering [19]. In such a case, the user's preference is predicted on an item using the average ratings of similar items by the same user as expressed in Equation (1) where S represents the most similar items to the item i, $s(i, j)$ denotes the similarity degree between items i and j and $r_{u,j}$ corresponds to the rating of user u on item j.

$$P_{u,j} = \frac{\sum_{j \in S} s(i,j) \quad r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{1}$$

The similarity value can be calculated in many ways. Common methods are *Cosine similarity* and *Pearson Correlation* [19].

Due to the nature of the data used in collaborative filtering, this approach suffers, as the case of all methods, from one or more weaknesses such as the cold start problem when a new user starts with an empty profile, the sparsity problem occurring when available data are insufficient for identifying similar users, and the scalability problem when there is an excessive information of users and items. To overcome these problems, several recommendation methods have been implemented using different techniques. We cite in particular, clustering techniques [13], Bayesian techniques [12] and community detection techniques [1–4].

2.2 Community detection

Community detection techniques aim to find subgroups where the amount of interactions inside the group is more than the interaction outside it, and this can help to understand the collective behavior of users.

The communities identification process depends on the nature of networks, either static or dynamic. Static networks are basically constructed by aggregating all observed interactions over a period of time and representing it as a single graph. Dynamic networks, also called time varying graphs can be either a set of independent snapshots taken at different time steps [7, 8] or a temporal network that represents sequences of structural modifications over time [9]. In what follows, we present both static and dynamic community detection.

Static community detection: A panoply of community detection algorithms exists in literature. The first idea using static networks was proposed by Girvan and Newman [15]. It is based on a modularity function representing a stopping criterion, aiming to obtain the optimum partitioning of communities. In the same context, Guillaume et al. [11] have proposed *Louvain* algorithm to detect communities using the greedy optimization principle that attempts to optimize the gain of modularity. Rosvall and Bergstrom [17] have presented *Infomap*, considered as a solution to the simplest problem of static and non-overlapping community detection. The mentioned algorithms are not able to detect overlapping communities where a node can belong to more than one community in the same time. To ensure this basic property, Palla et al. [14] have proposed the *Clique-Percolation Method (CPM)* to extract communities based on finding all possible k-cliques in the graph. This method requires the size of the cliques in input.

Dynamic community detection: Several researchers explored the dynamic aspect of networks to identify communities structure and their development over time. Hopcroft et al. [7] have proposed the first work on dynamic community detection which consists in decomposing the dynamic network into a set of snapshots where each snapshot corresponds to a single point of time. The authors applied an agglomerative hierarchical method to detect communities in each snapshot and then they matched these extracted ones in order to track their evolution over time. Palla et al. [8] have used the (CPM) method of static community detection to extract communities from different snapshots. Then, they tried to look for a matching link between them to detect their structural changes over time. Methods applying static algorithms on snapshots cannot cover the real evolution of communities structures over time because it seems harder for these methods to recognize the same community from two different time steps of network.

To overcome this problem, new studies have exploited another representation of data that takes into account all temporal changes of the network in the same graph. We cite, in particular, the intrinsic Longitudinal Community Detection *(iLCD)* algorithm proposed by Cazabet et al. [9]. The algorithm uses a longitudinal detection of communities in the whole network presented in form of a succession of structural changes. Its basic idea was inspired from multi-agent systems. In the same context, Nguyen et al. [10] have proposed *AFOCS* algorithm to detect overlapping communities in a dynamic network N composed of the input network structure N_0 and a set of network topology changes $\{N_1, N_2, ..., N_n\}$.

3 Related works

There has recently been much research on merging community detection and recommender systems in order to provide more personalized recommendations related to users belonging to the same community. In fact, community-based recommendation is a two-step approach. The first step consists in identifying groups in which users should share similar properties and the second step uses the community into which the target user pertains to recommend new items.

Using static community detection algorithms, Kamahara et al. [2] have proposed a community-based approach for recommender systems which can reveal unexpected user's interests based on a clustering model and an hybrid recommendation approach. In the same context, Qin et al. [3] have applied CPM method on the Youtube Recommendation Network of reviewers to detect communities of videos. These latters are used to provide the target user by a local recommendations which consists in recommending videos pertaining to the same community of the video watched by him. This approach aims to propose a more diverse list of items for target user. Another aim behind incorporating community detection to recommendation is to provide a solution to the cold start problem, and this idea was proposed by Sahebi et al. [4] while applying *Principal Modularity Maximisation* method to extract communities from different dimensions of social networks. Based on these latent communities of users, the recommender system is able to propose relevant recommendations for new users.

Qiang et al. [1] have defined a new method of personalized recommendation based on multi-label Propagation algorithm for static community detection. The idea consists in using the overlapping community structures to recommend items using collaborative filtering. More recently, Zhao et al [5] proposed the *Community-based Matrix Factorization (CB-MF)* method based on communities extracted using *Latent Dirichlet Allocation method (LDA)* on twitter social networks. In [6], authors focused on community-based recommendation of both individuals and groups. They used the Louvain community detection method on the social network of movies building from the Internet Movie Database (IMDb) in order to provide personalized recommendations based on the constructed communities.

These methods only deal with static networks, derived from aggregating data over all time, or taken at a particular time. The accumulation of an important mass of data in the same time and in the same graph can lead to illegible graphs, not able to deal with the dynamic aspect of real-world networks. To take into account the evolution of users behaviors over time using a kind of community-based dynamic recommendation, a first attempt was established by Lin et al. [16]. The main idea consists in providing a dynamic user modeling method to make recommendations by taking into consideration the dynamic users' patterns and the users' communities. This approach is limited since it uses a manual method to identify communities, which is not efficient especially when we deal with strongly evolving and large networks. More recently, Abrouk et al. [20] proposed to use the fuzzy k-means clustering from time to time to dynamically detect the users' interests over time. Then, they exploited these formed communities to determine user's preference for new items with regard to the updated users' ratings. In [21], author proposed an article recommender system to recommend documents for users based on the same members of communities which are identified according to their interests while browsing the web. The detection of users' interests is repeated continuously in subsequent time intervals in order to deal with the dynamic aspect of new portals. In both the previously presented methods, applying clustering techniques for time to time cannot cover the real evolution of community structure over time. In fact, several structural changes may occur and get lost without being detected. Besides, the temporal complexity of these methods increases in large networks.

Aiming to benefit from the whole advantages of the dynamic community detection process as part of recommender systems, we propose a global architecture allowing to ensure this combination as detailed below.

4 Proposed architecture

The proposed architecture, called *Dynamic Community-based Collaborative Filtering*, denoted by D2CF, is based on three main steps as shown in Figure 1.

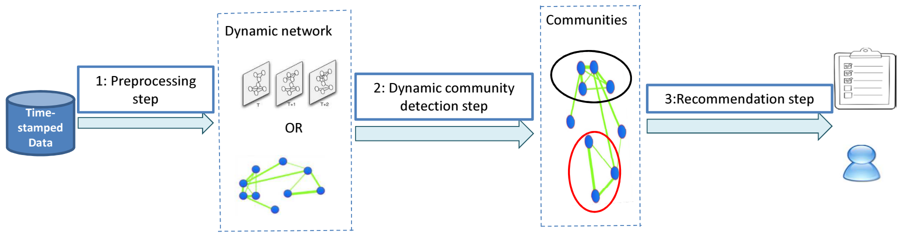


Fig. 1. D2CF architecture

4.1 The pre-processing step

This step consists in building a dynamic network in which the evolution of the users' interests (i.e. the set of items be looked at or ranked) over time is represented. We assume that if a user does not rate a given item then this latter is not yet watched by him. It is important to note that, in this work, the nodes are the items and not the users and hence the communities are groups of items. The interactions between nodes are modeled using the co-ratings relationship. Indeed, two nodes interact with each other if at least one user gives the same rating to both of them in the same time. With the aim to prepare the list of network changes over time, we have adapted the method of temporal network building [18] (i.e. An edge is established between two nodes if these ones have interacted with each others at least N times over a period of P days). The values of N and P control the set of edges adding and removal in the network and depend not only on the network topology but also on the information that we want to extract in order to create semantic links between the network nodes (e.g. messages interactions seem to be more intense than calls interactions). If over a period of P days, there are fewer than N interactions between two nodes, the edge will be automatically removed. These changes (edges removal and edges creations) can be represented either in the same graph as temporal network or in different graphs as a sequence of snapshots where each one corresponds to network changes over a specific time period (i.e. one hour, one day, ect.). The resulting dynamic network of items is considered as a generic model that represents the evolution of users interests over time.

4.2 Dynamic community detection step

Once the dynamic network (temporal network or a sequence of snapshots) is constructed in the pre-processing step, we are now able to use it as input to dynamic community detection method. Based on the network interactions, a community can be defined as a set of items that are extremely related to each other physically (strongly connected) and semantically (a learning pattern of items that tends to have the same interests of several users over a period of time). The advantage here is that a community is not restricted to item-related topics but it contains various topics (e.g. Horror, Comedy, Romance, ect.). The novelty in this work is to apply a dynamic community detection algorithm that takes into account the evolution of the network behavior in order to obtain a more appropriate community structure. To ensure this step, we choose to use the *iLCD* algorithm [9]. This choice is justified by the fact that *iLCD* detects communities in both static and dynamic networks depending on disposal data. Moreover, this algorithm takes into account the evolution of the network which enables to identify the dynamical communities more accurately. Such community detection can be more powerful because this analysis matches with the reality of networks. The algorithm deals with both large evolving networks and the overlap of communities. Extracted communities are atomic in the sense that there are not other relevant communities inside it. Finally, all operations in *iLCD* are made at a local level (i.e. a community can interact with only the nodes that are linked to and having at least one node in common), which can allow to ensure that the complexity will not grow exponentially with the size of the network. The life cycle of a community may be described via three phases:

- A new community is born when a new clique is formed in the graph.
- A modification of an internal community can lead to merge two communities having at least one node in common or to split the main community into two new communities.
- A community dies when there is no disposal nodes in its structure.

4.3 Recommendation step

In this step, the learned patterns (communities) will be exploited to help the recommender system to predict the users' future interests based on certain categories given by these communities. Firstly, a target item should be identified for each user. The target item in this case is the item in which the active user is more interested (The item with the highest rating value). The items that belong to the communities of target item and that are not rated by the active user are selected as candidate items. The list of top k recommended items for the active user contains the k candidate items that have the highest predicted preferences. Our objective is to compute the preference of the user u on the candidate item i based on the items that belong to the communities of i. By doing this the recommendation is restricted on the communities to which the candidate item pertains. To this end, we propose an adaptation of the traditional item-based Collaborative filtering method (See Equation (1)). In fact, instead of computing user's preferences taking into consideration all items present on the whole network to discover the most similar items to the candidate item, we only rely on the items that pertain to candidate item's communities extracted from the dynamic network as shown in Figure 2.

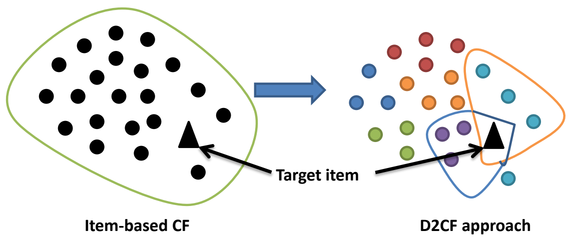


Fig. 2. The principle of user's preference computation taking into account the community structure in the network

Thus, we can formally define the preference prediction as follows:

$$P_{u,i} = \frac{\sum_{j \in C} s(i,j)\ r_{u,j}}{\sum_{j \in C} |s(i,j)|} \tag{2}$$

where C is the set of items pertaining to the community of i, $r_{u,j}$ is the rating given by the active user u to the item j and $s(i,j)$ is the similarity degree between items i and j.

We propose to compute the similarity $s(i,j)$ using the *Pearson correlation similarity* measure [19].

Finally, the recommendation list contains the candidate items i having the highest preference values $P_{u,i}$.

In the case where the user is new (no ratings history), the target item of this user is learned by browsing item in the recommender system. We select then the candidate items that belong to the communities of the target item. The recommendation list contains the candidate items which are ranked according to their similarities relative to target item.

5 Experimental study

To evaluate the effectiveness of D2CF, we propose to use the movieLens dataset available through the movieLens website (http://movieLens.umn.edu). This dataset contains in total 100.000 ratings collected by 943 users on 1682 movies, from 19-09-1997 to 22-04-1998. The score of rating is ranged from 1 to 5. Each user has rated at least 20 movies. The ratings information are timestamped. We suppose that each movie is represented by a node and a community is defined as a set of nodes. If a user rates a movie, this means that he is interested in watching it. MoviLens data are represented as a sequence of temporal events in the following way:

- user U_1 rates movie I_1 with 5 at T_1,
- user U_2 rates movie I_1 with 3 at T_1,

– user U_2 rates movie I_5 with 5 at T_2, etc.

To experimentally determine the impact of the training size on the quality of the recommended movies, we propose to test three scenarios:

– *Set_1* : For each user, we randomly select 90% of his ratings as instances in the training set and the remaining ones will be used in the testing set.
– *Set_2* : For each user, we randomly select 40% of his ratings as instances in the training set and 10% will be used in the testing set.
– *Set_3* : For each user, we randomly select 20% of his ratings as instances in the training set and 10% will be used in the testing set.

The data selection should take into account the evolution over time, so that, instances of the testing set should be chosen after those of the training set. We run 10-fold cross-validation on data. The precision is defined to evaluate the validity of a given recommendation list and it is formulated to detect the average of the true recommendations relative to the total number of the proposed recommendations. While the recall metric is defined as the ratio of the number of recommended objects collected by users appearing in the test set to the total number of the objects actually collected by these users.

The implementation of the whole system needs several parameters. Our choices regarding the three steps of D2CF can be summarized as follows:

1. *Pre-processing step*: Our idea is to extract movies interactions in such a way that we know what a movie has been assessed with another one with the same score by the same producer in the same time. In fact, if an interaction between two movies occurred more than N times over a period of P days, an edge is established between them. We define the values of P and N such a way that we conserve more links between nodes. After performing several tests on the movieLens data, we set P and N respectively to 200 and 30 for *Set_1*, 200 and 20 for *Set_2* and 200 and 5 for *Set_3*. This means that, in *Set_1*, an edge is established between two nodes, if these ones have interacted at least 30 times over a period of 200 days and this edge is removed if less than 30 interactions have occurred between them over a period of 200 days after the edge creation date. The movies that are not very visible in the users' ratings behaviors are considered as outliers. The outliers are the nodes that are disconnected of the core of the network due to their low interactions with other movies (i.e. there are less than N users who give the same rating for both of movies).
2. *Dynamic community detection step*: In this stage, we are able to apply any state of art dynamic community detection algorithm. In this experiment we choose to use *iLCD* algorithm to extract communities from the temporal network built above. Since the quality of resulting communities depends on the threshold (i.e. is a parameter using as input in the community detection step to determine the belonging or not of new added nodes in a community), we choose the value 0.5 to obtain by the end of the process, overlapping, small and dense communities.

3. *Recommendation step*: Using detected communities, we are now able to generate the top k recommendation list of movies to the active user. This step requires both the user's ID to look for his target item and the community structure as input parameters to select the candidate items that may interest the active user. Then, the predicted preference of each candidate item is computed using *Pearson correlation-based similarity* measure. The items which have the top k preference predictions are recommended to the active user.

In order to evaluate both of the effectiveness and efficiency of our proposed D2CF approach we compare the performance of this one with the following methods:

– The Static Community-based Collaborative Filtering (denoted S2CF for short), which is a static version of our proposed architecture. We keep the same parameters used for the dynamic network without taking into account the temporal dimension. This is possible since, as mentioned before, the *iLCD* algorithm allows both static and dynamic community detection.
– The traditional item-based Collaborative Filtering with *Pearson correlation-based similarity* measure available from Apache mahout library in Java.

The obtained results are summarized in Table 1. We can notice that our D2CF method outperforms traditional recommendation methods: item-based collaborative filtering and Collaborative filtering based on static community detection. In fact in Set_3, our approach is able even with small set of users' data to provide users with a wealthy and varied recommendation list based on the communities of users' interests. The recall and precision values for both item-based and static community-based collaborative filtering decrease as we increase the training set size but our approach combining recommendation and dynamic community detection still provide better recommendation quality as shown in Figure 5. In fact, D2CF gives its best results (i.e. D2CF proposes for one user an average of 6 good items out of every 10 recommended items while S2CD offers an average of two good items out of every 10 recommended items and finally item-based collaborative filtering gives an average of 0,15 good items per user). These results show that our approach is the best in the case of scalable data, which is explained by the fact that the dynamic network learned by more users' data better performs the prediction of users' preferences for unseen items.

Table 1. Precision and Recall values for Set_1, Set_2 and Set_3

	Set_1		Set_2		Set_3	
Approach	Precision	Recall	Precision	Recall	Precision	Recall
D2CF	**0.603**	**0.687**	**0.3**	**0.49**	**0.223**	**0.34**
S2CF	0.2	0.26	0.21	0.195	0.197	0.221
Item-based CF	0.015	0.02	0.084	0.091	0.18	0.2

D2CF approach presents a significantly improvement on recommendation on both small and large sets. We can say that this approach addresses both

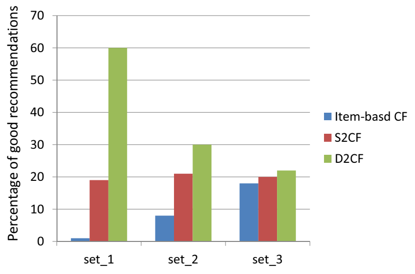


Fig. 3. Impact of the dataset size on the recommendation quality

scalability and sparse problems and it is able to handle the real-world networks by providing a dynamic recommendation based on dynamic communities.

6 Conclusion

In this paper, we propose a Dynamic Community-based Collaborative Filtering approach that combines recommendation and dynamic community detection. This approach is able to deal with real-world networks as it takes into account the evolutionary aspect of the users' interests over time. The experimental results show that our proposed D2CF outperforms both of item-based collaborative filtering and collaborative filtering based on static communities. As a future work, we will explore the similarity computation process of users pertaining to the same community in the recommendation context.

References

1. Qiang, H., Yan, G.: A method of personalized recommendation based on multi-label propagation for overlapping community detection. In: the 3rd International Conference on System Science Engineering Design and Manufacturing Informatization, 1, pp. 360–364. October (2012)
2. Kamahara, J., Asakawa, T., Shimojo, S., Miyahada, H.: A commynity-based recommendation system to reveal unexpected interests. In: Multimedia Modelling Conference, pp. 433–438. January (2005)
3. Qin, S., Menezes, R., Silaghi, M.: A recommender system for youtube based on its network of reviewers. In: the IEEE International Conference on Social Computing, (2010)
4. Sahebi, S., Cohen, W.: Community-based recommendations: a solution to the cold start problem. In RSWEB'11, (2011)
5. G. Zhao, G., Lee, M. L., Hsu, W., Chen, W., Hu, H.: Community-based user recommendation in uni-directional social networks. IN: the 22th ACM international

conference on Conference on information knowledge management, pp. 189–198. October 2013.
6. Fatemi, M., Tokarchuk, L.: A community based social recommender system for individuals groups. In: the 2013 International Conference on Social Computing (SocialCom'13), pp. 351–356. Sept (2013)
7. Hopcroft, J., Khan, O., Kulis, B., Selman, B.: Tracking evolving communities in large linked networks. In: the national academy of sciences of the United States of America, 1, pp. 5249–5253. (2004)
8. Palla, G., Barabasi, A., Vicsek, T.: Quantifying social group evolution. In: Nature, 446, pp. 664–667. April (2007)
9. Cazabet, R., Amblard, F.: Simulate to detect: a multi-agent system for community detection. In: The 2011 ACM International Conference on Web Intelligence and Intelligent Agent Technology(WI-IAT), 2, pp. 402–408. August 2011.
10. Nguyen, N., Dinh, T., Tokala, S., Thai, M.T.: Overlapping communities in dynamic networks: Their detection and mobile applications. In: the 17th annual international conference on Mobile computing and networking (MobiCom'11), pp. 85–96. Sept (2011)
11. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in larges networks. Journal of Statical Mechanics : Theory and Experiment. (2008)
12. Condliff, M. K., Lewis, D. D., Madigan, D.: Bayesian mixed-effects models for recommender systems. In: ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation, (1999)
13. Sarwar, B. M., Karypis, G., Konstan, J., Riedl, J.: Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. (2002)
14. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. In: Nature, 435(7043), pp. 814–818. (2005)
15. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Phisical review E, 69(2). (2004)
16. Song, X., Lin, C., Tseng, B., Sun, M.: Modeling evolutionary behaviors for community-based dynamic recommendation. In: the 2006 SIAM International Conference on Data Mining, (2006)
17. Rosvall, M., Bergstrom, C.: Maps of random walks on complex networks reveal community structure. In: the National Academy of sciences, 105(4), pp. 1118–1123. (2008)
18. Cazabet, R.: Dynamic Community detection on temporal networks. PhD thesis, Université Toulouse 3 Paul Sabatier. (2013)
19. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: the 10th international conference on world wide web, www'01, pp. 285–295. (2010)
20. Abrouk, L., Gross-Amblard, D., Cullot, N.: Community detection in the collaborative web. International Journal of Managing Information Technology. pp. 1–9.(2010)
21. Hönsch, M.: Detecting user communities based on latent and dynamic interest on a news portals. In: the 7th student research conference in informatics and information technologies, 3, pp. 47-50. ACM(2011)