Understanding the User Interactions on GitHub: A Social Network Perspective

Erzheng Fu[§], Yingqiu Zhuang[§], Jianxi Zhang, Jiayun Zhang, Yang Chen
School of Computer Science, Fudan University, China
{ezfu18, zhuangyq18, jianxizhang18, jiayunzhang15, chenyang}@fudan.edu.cn

***Abstract*—As one of the biggest online developer communities, GitHub supports interactions between millions of developers around the world. This paper focuses on detecting and analyzing the interactions on GitHub from several perspectives. First, from a global viewpoint, we build an interaction graph based on *GitHub Events* to investigate the general structure and attributes of the GitHub interaction network. Second, from the perspective of important users on GitHub, we pay particular attention to those who bridge social circles by spanning across communities. Concretely, we apply the structural hole theory to identify and explore the structural hole spanners in this interaction network. Last but not least, we compare structural hole spanners with ordinary users in several aspects and excavate distinctions between them. To our best knowledge, this is the first study that applies structural hole theory to the interaction graph of a leading social network. It not only provides a unique standpoint on user interactions, but also gives a comprehensive inspection of structural hole spanners.**

***Index Terms*—GitHub, interaction graph, structural hole spanner, online social network.**

I. Introduction

Thanks to the prosperity of online social networks (OSNs) [1], such as Twitter and Facebook, tremendous user interactive data is being generated, used and stored from time to time. There are already a series of studies on user interactions in general-purpose OSNs including Facebook and Twitter [2] [3]. These OSNs offer classic functions such as friendship creation, information spreading, and life-logging. However, less attention has been paid to function-oriented OSNs, for example, GitHub, which is known as one of the largest developer collaboration platforms and source code hosts in the world. GitHub also serves as a social network for developers. Being a function-oriented OSN, user interactions on GitHub are more purposeful. Meanwhile, as a professional platform, most of the interactions on GitHub are development-centric, and the users do not interact as frequently as in general-purpose OSNs. Hence, we take the first step by constructing an interaction graph of over 6 million GitHub users around the world. Analysis of this graph suggests that user interaction on GitHub is quite sparse and unilateral, revealing the uniqueness of its structure.

Having built and studied the overall features of the interaction graph, we want to focus more on one particular type of important users. To better illustrate the relationship between communities, *Burt* put forward the structure hole theory that communities are separated by holes inside the social structure [4], and individuals who occupy those holes are known as *structural hole spanners (SH spanners)*. Those users play a critical role in information diffusion across communities and they can benefit from their pivotal positions. Hence, we are motivated to look into GitHub's inter-community interactions by employing the structural hole theory and shed light on this specific set of users. We manage to apply two representative SH spanner detection algorithms, *Constraint* [4] and *HIS* [5], to the interaction graph and find out most possible SH spanners. We find that these SH spanners generally have larger degrees, higher betweenness centrality values and more connected communities. These findings reveal their significance in the structure of the whole interaction graph.

To look into SH spanners in more detail and examine the differences between SH spanners and ordinary users, we crawl users' profile data provided by GitHub API to refer to multiple attributes. Lots of salient characteristics have been uncovered related to SH spanners. For instance, SH spanners have much more social connections than the average, and they pay more attention to the completion of their profiles. Generally, they conduct interactive actions more frequently on GitHub. These findings disclose the properties of SH spanners based on user behavior and personal information.

Through the whole process of our work, we explore from macroscopical to microscopical perspectives on GitHub interaction network, and attain more knowledge of graph structures, communities and SH spanners. Our discovery provides an informative and comprehensive prospect of user interactions on GitHub.

II. Related Work

OSNs have become very popular in recent decades. Representative studies have been conducted on various social media, including Twitter [3], Quora [6], Foursquare [7] and a great many. As a representative software development-centric OSN, GitHub also catches the attention of researchers [8] [9] [10].

Wilson et al. [11] proposed the idea of interaction graph. They demonstrated that this interaction graph could better model the interactions between Facebook users than traditional social graphs. However, no one has yet studied a large-scale GitHub interaction graph based on user events. The most relevant touch was [12], which only inquired into *Push Event*, an event type that generally engages only one user and thus can not cover various types of interactions on GitHub.

[§]Co-First Author

Community detection is important for us to apply the structural hole theory to this graph. State-of-the-art community detection algorithms can be divided into two groups. One is based on modularity [13], represented by the *Louvain* algorithm [14] which we apply in our paper. The other is based on label propagation, represented by Label Propagation Algorithm [15]. The *Louvain* algorithm is based on multi-level optimization of modularity, which is the deviation of the number of connected edges between the given community and a random node distribution [16]. It is often considered a benchmark for community closeness. The algorithm uses greedy method and calculates modularity increment by using the number of connected edges within the community minus the number of connected edges between random graph and the community. We choose the *Louvain* algorithm in our paper due to its representativeness and scalability.

The concept of SH Spanner is being reviewed so frequently that many SH spanner detection algorithms have been raised: *HIS* and *MaxD* [5], *WeakTie-local* and *WeakTie-Bi* [17], HAM [18], *Greedy* and *AP_Greedy* [19]. Machine learning-based methods are introduced as well [20]. But none of them have been applied to the interaction graph of a leading OSN before.

Meanwhile, structural hole theory has been applied to many realistic problems, especially in enterprise and online social network settings. Not exhaustively listed, *Rodan* [21] analyzed the underlying mechanisms and found innovation plays the key mediating role in that relationship. *Gao et al.* [22] disclosed how social network information affected the choices of seeking collaborators based on the structural hole theory.

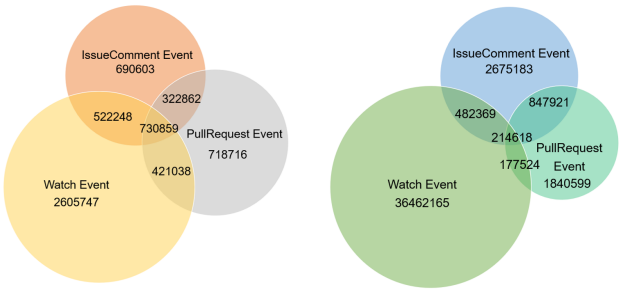
III. Interaction Graph

In this section, we introduce how we explore GitHub by constructing an interaction graph based on three of the most dominant events on GitHub. We also examine the basic properties of this graph and thoroughly analyze the graph-related metrics of this developer community.

It is worth explaining again the significance of our carrying out user interaction study on GitHub. As previously mentioned, the user group on GitHub is relatively more development-centric. Moreover, GitHub provides diverse event types of user interactions, creating a rich context for interactions. Besides the well-known functions like "comment", "star" (corresponding to "like" on general-purpose OSNs), "watch" ("follow"), there are as many as 40+ event types on GitHub. Our study has found a good direction to investigate and settle the complexity of user interactions.

A. Graph Construction

GitHub is a working platform providing a group of functional features. Unlike traditional OSNs, which make more efforts on friendship creation, information spreading or life-logging, GitHub is more a collaborative platform centered on software development. To obtain a comprehensive overview of interactions in the GitHub community, we adopt the interaction graph model [11]. We consider a *GitHub Event* (i.e. one user action) between a GitHub user and a repository of another user as a valid user interaction. In detail, we select three kinds of the most dominant interactive events on GitHub, which are *Watch Events*, *IssueComment Events*, and *PullRequest Events*, to represent the main user interactive behaviors. We implement it as a directed graph $G = (V,\ E)$ using data from Jan. 1st, 2019 to Dec. 31st, 2019, gathered from GH Archive[1] (a well-known project to record the public GitHub timeline). A node v in V represents a GitHub user and an edge (v_a, v_b) in E represents that there was an event of one of those three types triggered by v_a and targeted at a repository belonging to v_b. All isolated nodes are removed from the graph.



(a) Node (b) Edge

Fig. 1: (a) shows the number of nodes involved in the three event types. (b) shows the number of edges constructed by the three event types. Both illustrate that Watch Event has overwhelming advantages in quantity, and the three parts do not overlap much.

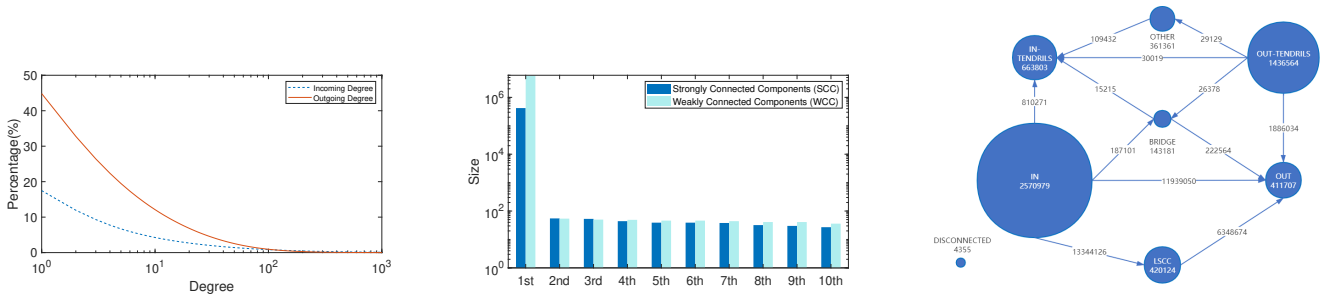
Eventually, we acquire a graph G of 6,012,074 nodes and 42,700,112 edges. Although we only select events within a one-year window due to feasibility, it has already covered about 15% of GitHub users, which is a significant portion of the GitHub population.

B. Graph Analysis

1) Composition: Graph G consists of nodes and edges from three types of events, so we first take a look at the numbers of nodes and edges related to each event type. From Fig. 1 we can easily observe the proportion of each event type separately and how these three event types overlap with each other. Conclusively, *Watch Events* have the highest intensity, but the quantity of its intersections with the other two event types is relatively small. Meanwhile, *IssueComment Events* and *PullRequest Events* intersect with each other more than with *Watch Events*, showing a stronger relationship between these two types. The low overlapping rate shows that every event type has its own significance on GitHub. It endorses our approach of taking multiple event types into account to build the user interaction network inclusively.

2) Degree: A node's degree is the number of edges it connects with. Since G is a directed graph, we consider incoming and outgoing edges separately. Fig. 2(a) shows the

[1]https://www.gharchive.org/



(a) CCDF of Degree (b) Sizes of Top 10 Components (c) Macrosturcture

Fig. 2: (a) illustrates that the average of outgoing degrees is larger than the average of incoming degrees, and most users have incoming or outgoing degrees less than or equal to 1. (b) shows that except for the LSCC and the LWCC, all other components contain less than 100 users. (c) is the macrostructure of the GitHub interaction graph, where the IN and the OUT-TENDRILS components together take over more than half of all nodes.

complementary cumulative distribution function (CCDF) of incoming/outgoing degree of all nodes in graph G. Overall, the sum of the average incoming and outgoing degrees is 7.10, which is relatively small compared with that of other OSNs like Twitter [3]. We also notice that over 80% of nodes have incoming degrees less than or equal to 1. For outgoing degrees, this number is about 60%. Additionally, the largest incoming degree is 256,190, while the largest outgoing degree is 763,941.

It is worth noting that many users on GitHub, especially those with larger degrees, are unilaterally connected. That is to say, they do not have both incoming and outgoing edges. More particularly, 62.51% of all nodes in this graph only have outgoing edges, 17.93% only have incoming edges, and the remaining 20% have both.

We also check this feature by measuring the balance metric $B(u)$, the incoming degree divided by the outgoing degree of a node u. We can see only 8.60% of nodes falling in the balancing interval $(B(u) \in [0.5, 2])$, unlike Weibo and Twitter networks, which have 57.4% and 49% of balanced nodes, respectively [23].

3) Reciprocity: Reciprocity is a quantitative parameter to measure the probability of mutual linking in a graph. It is defined as the number of pairs of symmetric edges (for example, edges (v_a, v_b) and (v_b, v_a) are interpreted as symmetric) divided by the aggregate of edges. Overall, graph G has reciprocity of 0.006989. We can once again observe unilateral connections featured in the GitHub community.

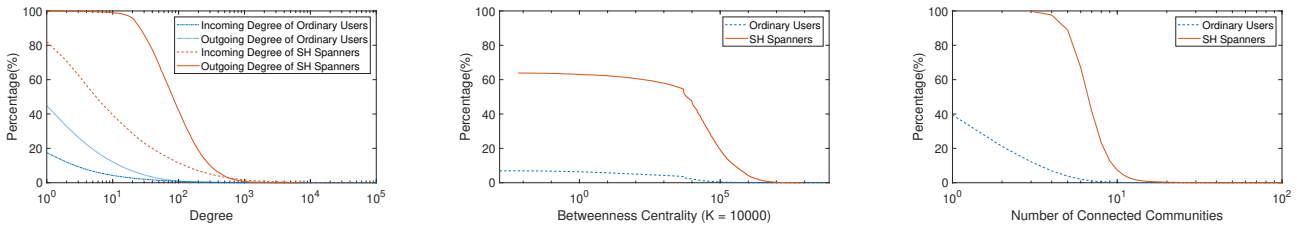
4) Clustering Coefficient: The clustering coefficient describes at which level nodes tend to cluster. We calculate the clustering coefficient of each node separately. As expected, over 90% of nodes have a clustering coefficient of 0, again reflecting the rationality of their low balance and reciprocity values, and showing that a large proportion of users on GitHub unintentionally avoid getting involved in large-scale activities and only maintain necessary professional interactions with certain users. This may result from why they use GitHub, which we mentioned before.

5) Connected Component: A connected component in graph theory is the spanning subgraph of a set of nodes that can reach each other through any path in the subgraph. In a directed graph, a maximal component (cannot add another node to satisfy the condition) is called a strongly connected component (SCC). If we render all edges in the graph undirected, such maximal component is known as a weakly connected component (WCC). Nodes in one component have high connectivity. That is, in a social network, information is easier to go over every part of this component.

There are 1,492,661 SCCs and 53,771 WCCs in graph G. The largest WCC (LWCC) covers 96.41% of nodes, demonstrating that most nodes can be associated with some known connections. But the largest SCC (LSCC) only covers 6.99% of nodes, in part because of the low balance, reciprocity and clustering coefficient values. Fig. 2(b) shows the sizes of the top 10 SCCs and WCCs. Apart from the LSCC and the LWCC, most connected components are rather small, where users have small social intercourse circles. We believe it is because many GitHub users only adopt this platform for professional software development-related activities.

6) Macrostructure: Adapting the method proposed by *Broder et al.* [24] and further improved by *Gabielkov et al.* [25], we depict the macrostructure of graph G, showing in Fig. 2(c). The general idea of macrostructure is to divide the graph into 8 components through the agency of the LSCC to trace the information propagation throughout the whole graph. Unlike the macrostructure of Twitter [25], or Foursquare [7], where the LSCC contains more than half of all nodes, the LSCC in the GitHub interaction graph is substantially smaller. The biggest component here is the IN, a group of users more likely with few followers and many followees according to [25], which is similar for the OUT-TENDRILS. In the case of graph G, these two parts are mainly composed of nodes with only outgoing edges. These results are consistent with what we have discovered in previous subsections.

7) Community: A community in a graph is a group of nodes densely connected internally. Studying the community



(a) CCDF of Degree (b) CCDF of Betweenness Centrality (c) CCDF of Connected Communities

Fig. 3: The blue and red bundles of curves represent different CCDFs of ordinary users and SH spanners found by HIS respectively. (a) illustrates that SH spanners have larger degree numbers, and outweigh outgoing degrees. (b) shows that SH spanners have considerably larger betweenness centrality. (c) illustrates that most users connect to less than 10 communities, and SH spanners have greater connectivity in general.

structure is important to fathom network structure. We apply the *Louvain* Algorithm [14] to group nodes into different communities and set the number of the level parameter to 6. It turns out that there are about 100,000 communities in total, and about 90% of nodes are concentrated in the top 10 communities, whereas more than 90% of communities have less than 20 users. The modularity value Q is about 0.467, showing a significant community structure [16]. Therefore, we will pay more attention to users who bridge different communities in the next section.

C. Summary

In this section, we build an interaction graph based on the three of the most significant event types on GitHub to model user interactions and apply multiple methods to understand key features of the graph, which turns out unique as an OSN. First, most users in this graph do not interact as actively as other OSNs. For instance, the Twitter network has an average degree of 35 [3], about 5 times as many as graph G. Second, most users tend to participate in the graph unilaterally. Even in similar directed OSNs like Quora, more than 50% of users have bilateral edges [6]. However, only 20% of users in graph G have both. Consequently, it is not well balanced, and its reciprocity is also far smaller. Flickr, on the contrary, has much higher reciprocity of 84% [26]. Considering that we adapt interaction graph to create relations and take different event types into account to engage more interactions, the reciprocity of 0.70% is peculiar.

We also study the macrostructure of this graph, digging up the uneven distribution of users in the graph structure. Moreover, we sort out the inner communities in graph G. The whole graph is composed of several prime parts covering a considerable number of users and a great number of sporadic parts attached to the edges of the graph.

IV. SH Spanner Detection

In Section III, we have studied the overall properties of the interaction graph built to manifest interactions on GitHub and made its distinctive features clear from different angles. Besides getting an integrated cognition of the graph, we are also interested in the difference between users. Therefore, we would like to figure out who plays the primary role in transmitting information across communities.

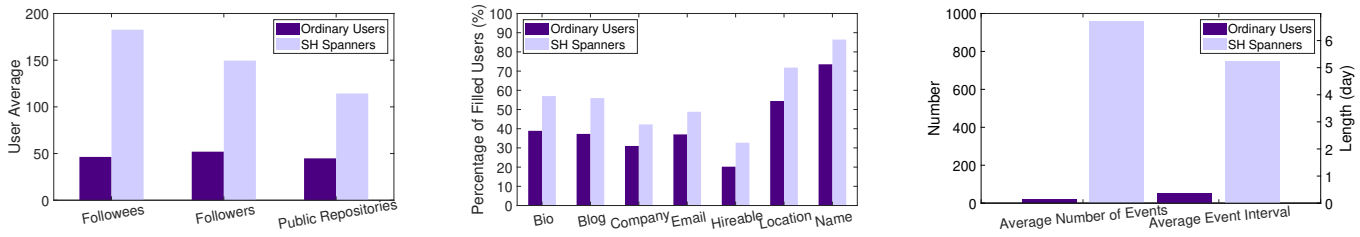
Here we introduce the concept of the structural hole. Structural hole theory was first proposed by *Burt* [4]. It declares that groups in a relationship network are separated by holes inside the social structure. This theory has shown a wide range of applications, especially in enterprise and OSN settings, which represent common scenes of social networks from either offline or online perspectives. It reveals the relationship between structural holes and individual managerial performance [27].

Structure hole spanners (SH spanners) are a class of users who span structural holes, bridging between different communities, thus mainly in control of information diffusion hubs and other key resources in a network. Detecting SH spanners is a fundamental problem of applying structural hole theory in OSNs. We adopt two established methods, *Constraint* [4] and *HIS* [5], to find the top 10,000 users who are most likely to be SH spanners in our GitHub interaction network. In particular, *Constraint* chooses the top-k nodes with the lowest constraint values as SH spanners, while *HIS* selects users who are connected to more opinion leaders.

We manage to get the lists of the top 10,000 SH spanners using these two algorithms and conduct further studies on how these SH spanners differ from ordinary users in the network. Due to the page limit, we only focus on the SH spanners identified by the *HIS* algorithm, since the SH spanners uncovered by the *Constraint* metric would demonstrate a similar trend.

1) Degree: As the CCDF of degree is already given in the previous section, we separate the data into an SH spanner and an ordinary user group, shown in Fig. 3(a). It indicates that SH spanners hold absolute advantages over ordinary users in respect of both incoming degrees and outgoing degrees. As for *HIS*, a degree of more than 200 is the threshold for being an SH spanner. *HIS* also seems to prefer selecting users with more outgoing interactions. All SH spanners found by *HIS* have outgoing degrees larger than 0, demanding that each of them has actively triggered at least one interactive event.

2) Betweenness Centrality: Betweenness centrality (BC) of a node is defined as the number of shortest paths passing the node divided by the sum of all shortest paths between each pair of nodes in the graph. It is a measurement for how many



(a) Number of Followees, Followers and Repositories (b) Profile Field Filling Rate (c) Event Quantity and Frequency

Fig. 4: (a) makes it clear that SH spanners have more followers, followees and public repositories than ordinary users. (b) illustrates that SH spanners generally have more completed profile attributes. (c) shows that SH spanners work harder than ordinary users on GitHub. They trigger events more in quantity and more frequently.

critical paths a node occupies, and in the same way, to what extent a node controls information distribution in OSNs. It is not very feasible to calculate the exact BC in graph G due to its scale, so we utilize an alternative estimate method proposed by *Brandes and Pich* [28], with parameter K set to 10,000.

Fig. 3(b) is the CCDF of BC of ordinary users and SH spanners. SH spanners hold remarkably higher BC values, confirming that they are more focal in information distribution.

3) Connected Communities: In Section III-B, we discovered the communities in graph G using the *Louvain* algorithm and assigned every node to one of those communities. As SH spanners should be the joint of these communities, we are curious about how many communities an SH spanner or an ordinary user directly connects to. We draw Fig. 3(c) to visualize the results.

Ordinary users are not apt to span a lot of communities: more than a half just stay in their own communities and never reach out, and few span more than 10 communities. Different for SH spanners, each of them connects with at least 3 communities, serving as bridges. *HIS* has also found some outstanding SH spanners who span more than 100 communities, acting as "hubs" of various communities.

The results above indicate that an SH spanner is more active in arching over communities. They participate in a lot more of events, owning the spotlights, thus carrying more significance on bounding communities and accelerating information spreading. The results confirm the validity of the structure hole theory in this interaction network.

With all these findings, we are curious about whether there is a significant difference in user behavior between these SH spanners and ordinary users. We will answer this question in the next section.

V. SH spanner Feature Analysis

In this section, we systematically examine the distinction between SH spanners and ordinary users. To achieve this, we should obtain as much information as possible from users. We have crawled 1,510,480 users' profile data directly from GitHub using three rented Vultr servers. Besides basic entries like *bio*, *email*, and *company*, we also crawled each user's public repositories, follower list, followee list, starred list, etc. The crawling lasted from Mar. 2020 to May 2020 due to lots of hierarchical URLs. We used parallel crawling to speed up and avoid restrictions from GitHub. To contrast SH spanners with ordinary users from different angles, we uniformly sampled 100,000, the same number of SH spanners that *HIS* lists, ordinary users into the comparing set.

1) Social Connectivity: In terms of user information, the numbers of followers and followees are still important metrics to consider. The first two pairs of bars in Fig. 4(a) distinctively show that SH spanners typically have more social connections. Having more followers and followees denotes that SH spanners in the interaction graph are more likely to gain social connectivity. This does not go beyond our guts: if a user interacts with users from more communities, they will also follow more users, and gain more followers. We could also see from Fig. 4(a) that they engage with more public repositories.

2) Profile Attributes: As shown in Fig. 4(b), for almost every entry, the proportion of SH spanners who fill in them is noticeably higher than that of ordinary users. Especially for the *hireable* entry, SH spanners' proportion is about twice as much as ordinary users'. It evinces the fact that SH spanners generally attach more information to their profiles. A complete profile can show the corresponding user's profession and dedication, providing rich information for meaningful interactions and communications. So there is no wonder why SH spanners seem to be more interested in filling their profiles.

3) Events: Since the GH Archive dataset also includes the exact timestamp of each event, we also enumerate the number and frequency of events triggered by each user of three types that we use to build the interaction graph. Fig. 4(c) has shown the average number of events and the average time interval between two events of both SH spanners and ordinary users. Unsurprisingly, SH spanners tend to trigger more events every week, and the average time interval between is also shorter. The difference here is significant, which again shows their dedication and contribution to the platform.

4) Word Frequency: On GitHub, users can write a brief message called *bio* on their profile. The content in the description is often representative of the user's life status, fields of interest, walks of life or other personal stuff. This attribute

(a) SH spanners (b) Ordinary users

Fig. 5: Word Cloud of the "bio" entry on user profiles

provides more information to know about the users. We find the most used words in the *bio* field and draw the word cloud to display the highest-frequency words. As shown in Fig. 5, the two word clouds of SH spanners and ordinary users are similar in many ways. For instance, their leading words all include "Developer", "Software" and "Engineer". But SH spanners and ordinary users still turn out to have quite different frequency of occurrence of some words such as "University" and "interested".

In short, SH spanners' features express not only in the structure of the interaction graph, but also in their external characteristics, which exist in their profiles and activities.

VI. Conclusion and Future Work

In this paper, we present a comprehensive analysis of GitHub user interaction graph based on the data of three kinds of *GitHub Events* in a one-year window. In conclusion, the graph has some remarkable features: sparse, unilateral connected and unevenly distributed. To investigate important users, we focus on SH spanners in the interaction graph, outcropping them and comparing them with ordinary users in multiple facets, and confirming their importance in user interactions on GitHub.

Comparing SH spanners found by different algorithms can be one of our future work, because they may also have differences in user interactions. Moreover, applying traditional SH spanners detection algorithms on a whole graph is time-consuming. Since most OSNs are dynamic, it is very difficult to execute entire graph-based algorithms from time to time. Therefore, based on our findings, we will explore how to identify SH spanners only by their behavioral features, instead of relying on the graph structure in our future work.

Acknowledgement

This work has been sponsored by National Natural Science Foundation of China (No. 62072115, No. 61602122, No. 71731004). Yang Chen is the corresponding author.

References

[1] L. Jin, Y. Chen, T. Wang, P. Hui, and A. V. Vasilakos, "Understanding User Behavior in Online Social Networks: A Survey," *Communications Magazine, IEEE*, vol. 51, no. 9, pp. 144–150, 2013.

[2] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proc. of ACM WOSN*, 2009.

[3] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?," in *Proc. of WWW*, 2010.

[4] R. S. Burt, *Structural holes: The social structure of competition*. Harvard university press, 1992.

[5] T. Lou and J. Tang, "Mining structural hole spanners through information diffusion in social networks," in *Proc. of WWW*, 2013.

[6] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao, "Wisdom in the social crowd: an analysis of quora," in *Proc. of WWW*, 2013.

[7] Y. Chen, J. Hu, Y. Xiao, X. Li, and P. Hui, "Understanding the User Behavior of Foursquare: A Data-Driven Study on a Global Scale," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1019–1032, 2020.

[8] Y. Hu, J. Zhang, X. Bai, S. Yu, and Z. Yang, "Influence analysis of Github repositories," *SpringerPlus*, vol. 5, no. 1, p. 1268, 2016.

[9] M. Menichinelli, "A data-driven approach for understanding Open Design. Mapping social interactions in collaborative processes on GitHub," *The Design Journal*, vol. 20, no. sup1, pp. S3643–S3658, 2017.

[10] J. Zhang, Y. Chen, Q. Gong, X. Wang, A. Y. Ding, Y. Xiao, and P. Hui, "Understanding the Working Time of Developers in IT Companies in China and the United States," *IEEE Software*, vol. 38, no. 2, pp. 96–106, 2021.

[11] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proc. of ACM EuroSys*, 2009.

[12] A. Lima, L. Rossi, and M. Musolesi, "Coding Together at Scale: GitHub as a Collaborative Social Network," in *Proc. of ICWSM*, pp. 295–304, 2014.

[13] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, no. 2, p. 026132, 2005.

[14] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, p. 10008, 2008.

[15] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[16] M. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[17] C. Song, W. Hsu, and M. L. Lee, "Mining brokers in dynamic social networks," in *Proc. of CIKM*, 2015.

[18] L. He, C.-T. Lu, J. Ma, J. Cao, L. Shen, and P. S. Yu, "Joint community and structural hole spanner detection via harmonic modularity," in *Proc. of KDD*, 2016.

[19] W. Xu, M. Rezvani, W. Liang, J. X. Yu, and C. Liu, "Efficient Algorithms for the Identification of Top-k Structural Hole Spanners in Large Social Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1017–1030, 2017.

[20] Q. Gong, J. Zhang, X. Wang, and Y. Chen, "Identifying Structural Hole Spanners in Online Social Networks Using Machine Learning," in *Proc. of ACM SIGCOMM Posters and Demos*, 2019.

[21] S. Rodan, "Structural holes and managerial performance: Identifying the underlying mechanisms," *Social Networks*, vol. 32, no. 3, pp. 168–179, 2010.

[22] G. Gao, P. Hinds, and C. Zhao, "Closure vs. Structural Holes: How Social Network Information and Culture Affect Choice of Collaborators," in *Proc. of CSCW*, 2013.

[23] T. Wang, Y. Chen, Y. Wang, B. Wang, G. Wang, X. Li, H. Zheng, and B. Y. Zhao, "The power of comments: fostering social interactions in microblog networks," *Frontiers of Computer Science*, vol. 10, no. 5, pp. 889–907, 2016.

[24] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, no. 1-6, pp. 309–320, 2000.

[25] M. Gabielkov, A. Rao, and A. Legout, "Studying social networks at scale: macroscopic anatomy of the twitter social graph," in *ACM SIGMETRICS*, 2014.

[26] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *Proc. of WWW*, 2009.

[27] E. Choi and K. C. Lee, "Relationship between social network structure dynamics and innovation: Micro-level analyses of virtual cross-functional teams in a multinational B2B firm," *Comput. Hum. Behav.*, vol. 65, pp. 151–162, 2016.

[28] U. Brandes and C. Pich, "Centrality estimation in large networks," *International Journal of Bifurcation and Chaos*, vol. 17, no. 07, pp. 2303–2318, 2007.