M-BiRank: Co-ranking developers and projects using multiple developer-project interactions in open source software community

Dengcheng Yan (yandengcheng@gmail.com)
Anhui University https://orcid.org/0000-0003-1417-5269
Bin Qi
Anhui University
Yiwen Zhang
Anhui University
Zhen Shao
Anhui University

Research

Keywords: M-BiRank, Ranking, Multiplex bipartite network, Social collaborative coding, GitHub

Posted Date: September 18th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-34629/v3

Version of Record: A version of this preprint was published on October 27th, 2020. See the published version at https://doi.org/10.1186/s13638-020-01820-3.

RESEARCH

M-BiRank: Co-ranking developers and projects using multiple developer-project interactions in open source software community

Dengcheng Yan[1,2]
, Bin Qi[1]
, Yiwen Zhang[1*]
and Zhen Shao[1]

*Correspondence:
zhangyiwen@ahu.edu.cn
[1]School of Computer Science and Technology, Anhui University, Jiulong Road No. 111, 230601, Hefei, China
Full list of author information is available at the end of the article

Abstract

Social collaborative coding is a popular trend in software development and such platforms as GitHub provides rich social and technical functionalities for developers to collaborate on open source projects through multiple interactions. Developers often follow popular developers and projects for learning, technical selection and collaboration. Thus identifying popular developers and projects is very meaningful. In this paper, we propose a multiplex bipartite network ranking model, M-BiRank, to co-rank developers and projects using multiple developer-project interactions. Firstly, multiple developer-project interactions such as commit, issue and watch is extracted and a multiplex developer-project bipartite network is constructed. Secondly, a random layer is selected from this multiplex bipartite network and initial ranking scores are calculated for developers and projects using BiRank. Finally, initial ranking scores diffuse to other layers and mutual reinforcement is taken into consideration to iteratively calculate ranking scores of developers and projects in different layers. Experiments on real world GitHub dataset show that M-BiRank outperforms degree centrality, traditional single layer ranking methods as well as multiplex ranking method.

Keywords: M-BiRank; Ranking; Multiplex bipartite network; Social collaborative coding; GitHub

1 Introduction

Open source software community is now a main driven force of innovations and plenty of software developers collaborate on millions of open source software projects, among which are many popular software projects that drive the innovations of different fields [1, 2]. For example, deep learning frameworks such as TensorFlow, PyTorch and MXNet contributed by famous companies simplify the building of deep leaning models, which to some extent speed up the innovations in the field of artificial intelligence in both academia and industry [3, 4].

In open source software community, developers from different areas usually take the social collaborative coding paradigm and participate in different portions of a common project using the social and technical functionalities provided the community [5, 6]. Taking GitHub as an example, developers from different areas and with different technical backgrounds can collaborate on a project by committing codes,

commenting on issues, star/fork a project for improving technical skills or technical selections, and follow other professional developers for keeping pace with new trends. Much like the role of opinion leaders in social networks, influential developers and projects drive the technical trends and the prosperity of open source community. Thus, identifying influential developers and projects will be of great significance. Existing work on influence analysis for open source software community mainly focus on applying traditional unipartite single layer graph ranking methods [7, 8], including PageRank [9] and HITS [10], although many new graph ranking methods for more complex network structures, such as bipartite network [11, 12] and multiplex network [13, 14], have been proposed. On the other hand, existing graph ranking methods haven't merged bipartite network and multiplex network as a single network model, which is necessary for our case to model multiple interactions between developers and projects. Potential applications of influence analysis of open source software community would include service recommendations [15, 16, 17, 18, 19] and risk assessment [20].

In this paper, we focus on modeling multiple interactions between developers and projects as a multiplex bipartite network and propose a new ranking method based on it in an iterative and mutually enhanced way. The main contributions of this research are many folds:

- We propose multiplex bipartite network model to represent multiple interactions between developers and projects.
- We propose a new ranking model called M-BiRank on multiplex bipartite network which takes into account the mutual reinforcement between different types of nodes as well as different layers.
- We apply the proposed model to real world GitHub dataset, showing that our model outperforms baseline ranking models.

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to related works on ranking models from the perspective of network and its applications in software engineering. In Section 3, details about the proposed M-BiRank model are illustrated. Then the experiment results and discussions are given in Section 4. Finally, we briefly summarize our work and explain future directions in Section 5.

2 Related work

Identifying influential nodes in social networks has been a hot topic for decades. Existing works mainly focus on either structural properties or diffusion dynamics. Plenty of structure-based metrics and random walk-based methods have been proposed.

Structure-based metrics usually base itself on some intuition for centrality from either local or global views. Degree is the most common local structure-based centrality metrics. Based on degree, Chen et al. [21] proposed a semi-local centrality metric, considering both the nearest and the next nearest neighbors. Chen et al. [22] further considered the negative impact of local clustering on information diffusion in networks and proposed ClusterRank. In addition to extending degree, several local structure-based centrality metrics are originated from H-index, which is originally used to measure the citation impact of a scholar or a journal [23]. Zhao et al. [24]

first extended H-index concept to networks and defined the h-Degree metrics for weighted networks. Liu et al. [25] combined the H-index of both node's itself and its neighbors and proposed a local h-index centrality. Lü et al. [26] revealed the relation among degree, H-index and coreness, and introduced a family of H-indices.

Local structure-based centrality metrics benefit from low computational complexity at the cost of reducing effectiveness. While global structure-based centrality metrics can better identify influential nodes from a global view of the whole network. Earlier researches in sociology introduced several global structure-based centrality metrics, including closeness centrality [27], betweenness centrality [28] and eigenvector centrality [29]. Recently researches introduced eigenvector centrality to more complex network structures. Wang et al. [30, 31] extended eigenvector centrality to multilayer networks under a framework of tensor decomposition.

Random walk-based methods apply resource diffusion dynamic process in networks and measure node's influence according the final resource the node obtains at stationary state of the dynamic process. Typical random walk-based ranking methods include PageRank [9] and HITS [10]. To solve the problem of dangling nodes of PageRank, Lü et al. [32] added a ground node to the original network, making the original network connected and proposed the parameter-free LeaderRank method. Halu et al. [13] extended PageRank to multiplex network and proposed Multiplex PageRank, which included four kinds of intra-layer enhancement mechanism [33]. To address the ranking problem on bipartite networks, He et al. [11] proposed the BiRank method. Instead of modeling pairwise interactions, higher-order network model have recently been proposed and applied to ranking nodes with group interactions. Treating scientific collaboration as a group interaction, Liang et al. [34] modeled it as a hypergraph and proposed HHGBiRank. From another view of higher-order structure of networks, that is motif, Zhao et al. [35] proposed Motif-based PageRank.

In addition to identifying influential nodes for general purpose social networks, needs have also emerged for open source software community, a special kind of social network. Xuan et al. [8] constructed several social networks based on the communications between developers in Apache and applied degree, PageRank and HITS for developer ranking. Hu et al. [7] studied the problem of influence identification of developers in GitHub and proposed a Following-Star-Fork-Activity based approach. Joblin [5] employed several activity counts, centrality metrics and network structural properties to distinguish core and peripheral developers.

3 Method

In this section, we will present a *M*ultiplex *Bi*partite *Rank*ing method, called *M-BiRank*, for co-ranking developers and projects in open source software community. As is shown in Figure 1, the proposed M-BiRank consists of three parts and incorporates two basic assumptions that address the issues in Section 1. We will start by giving the definition of multiplex bipartite network and introducing the notations. Then, thorough explanations and mathematical formulations are given for the two basic assumptions, that is, mutual reinforcement between different types of entities and between different layers of network. Finally, we will introduce the overall algorithm and time complexity analysis of it.

3.1 Multiplex bipartite network

Definition 1 Multiplex bipartite network. A multiplex bipartite network is a set of M bipartite networks $G^A = (U \cup P, E^A)$ where $A = 1, ..., M$. U and P represent two different kinds of nodes and all layers have the same sets of nodes U and P, while the set of edges E^A depends on the layer A. Specially, there is no edge between same kinds of nodes. A multiplex bipartite network is formally defined as $G = (G^1, ..., G^M)$. Each network G^A is described by the bipartite weight matrix W^A ($\in \mathbb{R}^{|U|\times|P|}$) with elements w_{ij}^A, where $w_{ij}^A > 0$ if there is a link with weight w_{ij}^A between nodes u_i and p_j in layer A, and $w_{ij}^A = 0$ otherwise.

In this paper, we model multiple interactions between developers and projects as a developer-project multiplex bipartite network. The notations we will use throughout the article are summarized in Table 1.

3.2 Mutual reinforcement between developers and projects

Most ranking methods in networks adopt the intuition as PageRank and HITS that an influential node should be linked by many other influential nodes, which is also applicable in the case of developer-project bipartite network. For example, an elite developer usually participates in popular projects. In open source software community, it is quite a practice to estimate the influence of a developer by how popular the projects she/he participates in are and how much she/he contributes to these popular projects. And a project with influential developers or organizations as major contributors always attracts large number of attention. Taking TensorFlow as an example, it got thousands of stars quickly upon its first release in GitHub because it is supported by Google.

This intuition forms our first assumption that a developer (project) should be ranked high if it is connected to high-ranked projects (developers) in a certain layer A, which can be formulized as follows:

$$p_j^A = \sum_{i=1}^{|U|} w_{ij}^A u_i^A \tag{1}$$

$$u_i^A = \sum_{j=1}^{|P|} w_{ij}^A p_j^A \tag{2}$$

In order to employ a prior belief on nodes' importance and provide better ranking results, we also adopt query vector and symmetric normalization as BiRank. The prior belief on nodes' importance and rankings from network structure are balanced with two parameters γ and λ. The final formulation of the mutual reinforcement between developers and projects is as following:

$$p_j^A = \gamma \sum_{i=1}^{|U|} \frac{w_{ij}^A}{\sqrt{d_i^A}\sqrt{d_j^A}} u_i^A + (1-\gamma) p_j^0 \tag{3}$$

$$u_i^A = \lambda \sum_{j=1}^{|P|} \frac{w_{ij}^A}{\sqrt{d_i^A}\sqrt{d_j^A}} p_j^A + (1-\lambda) u_i^0 \tag{4}$$

3.3 Mutual reinforcement between different layers

Besides considering the mutual reinforcement between developers and projects in each single layer, we also take into account the mutual reinforcement between different layers. From our experience as open source software developers, we could firmly assume different interactions between developers and projects reflect different aspects of influence and only a composition of all the aspects could reflect a comprehensive influence of developers and projects. For example, committing code to a project indicates a developer's coding skill and commenting issues of a project may show a developer's design skill or bug-fixing skill. The influence of a developer should be measured by summarization of both coding and design skills.

To implement mutual reinforcement between different layers, we choose to incorporate the ranking scores of developers and projects from the first layer as an enhancement of the query vectors of the second layer. The mathematical formulations are given in Equations (5) and (6).

$$p_j^B = \gamma \sum_{i=1}^{|U|} \frac{w_{ij}^B}{\sqrt{d_i^B}\sqrt{d_j^B}} u_i^B + (1-\gamma) p_j^A \tag{5}$$

$$u_i^B = \lambda \sum_{j=1}^{|P|} \frac{w_{ij}^B}{\sqrt{d_i^B}\sqrt{d_j^B}} p_j^B + (1-\lambda) u_i^A \tag{6}$$

To be more clearer, we transform Equations (5) and (6) to their equivalent matrix form in Equations (7) and (8)

$$p^B = \gamma (S^B)^T u^B + (1-\gamma) p^A \tag{7}$$

$$u^B = \lambda (S^B) p^B + (1-\lambda) u^A \tag{8}$$

where $S^B = (D_u^B)^{-\frac{1}{2}} W^B (D_p^B)^{-\frac{1}{2}}$ is the symmetric normalization of weight matrix W^B.

3.4 Overall algorithm

By combing both the mutual reinforcement between developers and projects in each single layer and that between different layers, we finally propose the M-BiRank method to co-rank developers and projects in open source software community and the overall algorithm is shown in Algorithm 1.

3.5 Time complexity analysis

The overall time complexity of M-BiRank is a summarization of each layer's time complexity. For each layer, according to Equations (7) and (8), the time cost mainly depends on the multiplication of $(S^B)^T u^B$ and $(S^B) p^B$, showing a time complexity of $O(|U| \cdot |P|)$. However, most real world networks are usually very sparse and only non-zero elements (which correspond to existing edges) should be stored and computed regarding to matrix multiplication of $(S^B)^T u^B$ and $(S^B) p^B$. Thus, the

Algorithm 1 M-BiRank Algorithm.

Input:
Weight matrix W^A and W^B, query vectors u^0,p^0 and hyper-parameters γ, λ;
Output:
Ranking vectors u,p;
Symmetrically normalize $W^A : S^A = (D_u^A)^{-\frac{1}{2}} W^A (D_p^A)^{-\frac{1}{2}}$;
Randomly initialize p^A and u^A;
while Stopping criteria is not met **do**
  $p^A \leftarrow \gamma(S^A)^T u^A + (1-\gamma)p^0$;
  $u^A \leftarrow \lambda S^A p^A + (1-\lambda)u^0$;
end while
Symmetrically normalize $W^B : S^B = (D_u^B)^{-\frac{1}{2}} W^B (D_p^B)^{-\frac{1}{2}}$;
Initialize p^B and u^B using p^A and u^A, respectively;
while Stopping criteria is not met **do**
  $p^B \leftarrow \gamma(S^B)^T u^B + (1-\gamma)p^A$;
  $u^B \leftarrow \lambda S^B p^B + (1-\lambda)u^A$;
end while
$p \leftarrow p^B$, $u \leftarrow u^B$;
return u and p.

time complexity of layer A is $O(c_A|E_A|)$ and the overall time complexity of M-BiRank is $O(\sum_{A=1}^{M} c_A|E_A|)$, where c_A denotes the number of iterations in layer A and $|E_A|$ denotes the number of edges in layer A.

4 Experiment

In this section, the performance of M-BiRank model is evaluated against the GHTorrent dataset [36].

4.1 Datasets

GHTorrent [36] monitors the GitHub public event time line and retrieves information of developers, projects and interaction details between them from these events [37]. We choose the GHTorrent dataset as of November 1st, 2018 and extract the relationships between developers and projects, both of which mainly belongs to PHP community. The steps of data preprocess include: (1) Choose issues and commits which belong to PHP projects; (2) Keep developers and projects which exist in both issues and commits; (3) Construct multiplex developer-project bipartite network, using a function of issue/commit number as edge weight, that is, $w_{ij}^{issue} = log(\#issues) + 0.3$ or $w_{ij}^{commit} = log(\#commits) + 0.3$. The detail information about the dataset used in our experiment is shown in Table 2:

4.2 Evaluation metrics

In order to evaluate and compare the performance of M-BiRank and baseline methods, both correlation analysis and SIR model are adopted.

Correlation analysis mainly focuses on comparing predictions against the ground truth and Pearson Correlation Coefficient (PCC) [38] is chosen. In our experiment, the number of watch of projects and the number of followers of developers are set as the ground truth for the rankings of projects and developers, respectively. PCC reflects the correlation degree of two variables through the linear correlation between vectors, which is defined as follows:

$$PCC(x,y) = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{9}$$

where n represents the number of elements, x_i and y_i represent the ith element of sample x and y, respectively, and the value range of PCC is $[-1, 1]$.

However, the ground truth in correlation analysis is some kinds of degree in networks, which is a rough metric in evaluating the influence of developers or projects. To rank more precisely, dynamic models are needed for simulating the influence diffusion process [39]. SIR model [40] is a classical epidemic model and is often used to evaluate the ability of information spreading of a node in social networks. Generally, a influential user with a higher ranking score will spread his/her opinions to more developers. The transmission process of SIR model are shown in Figure 2, where S (Susceptible), I (Infected) and R (Removed) denote the susceptible, infected and recovered nodes. At the initial step of the transmission process, several infected nodes are set, and then the transmission is iteratively repeated until no new nodes are infected [41]. At each step, infected nodes infect its susceptible neighbors with the probability α, and infected nodes recovery to removed status with the probability β. So SIR model is suitable for evaluating the ability of information spreading of a node. By applying nodes with highest ranking scores of different ranking methods as the initial infected nodes and comparing the final number of affected nodes (both Infected and Removed nodes), the effectiveness of different ranking methods can be compared.

4.3 Baseline methods

We compare M-BiRank with several baseline methods.

Degree [42]: The degrees of developers and projects in different layers of multiplex developer-project bipartite network are calculated and **averaged**.

PageRank [9]: PageRank ranks nodes by iteratively propagating scores on the network and is usually suitable for single layer monopartite network. In this experiment, we apply it to multiplex developer-project bipartite network with two different setup. **PageRank-Avg** ignores types of nodes and applies PageRank algorithm directly to different layers of the multiplex developer-project bipartite network. The final ranking score of a node is the average of different layers. **PageRank-Add** merges different layers of multiplex developer-project bipartite network into a single layer of developer-project bipartite network and uses the average edge weights of different layers as edge weights of this single layer bipartite network. Then we apply PageRank algorithm to this single layer of developer-project bipartite network ignoring types of nodes. Finally, bot **PageRank-Avg** and **PageRank-Add** rank developers and projects separately according their final ranking scores. The hyper parameter is set to 0.85.

BiRank [11]: BiRank is a propagation-based ranking method on bipartite networks and adopts a normalization strategy in the iterative process. **BiRank-Avg** applies BiRank algorithm to different layers of multiplex developer-project bipartite network separately and averages the ranking scores in different layers as the final ranking scores. **BiRank-Add** firstly merges different layers of multiplex developer-project bipartite network into a single layer of developer-project bipartite network with the average of the edge weights in different layers as edge weights. Both of the hyper parameters are set to 0.85.

Multiplex PageRank [13]: Multiplex PageRank considers the impact of the centrality of a node in one layer on that in another layer and introduces nodes'

centrality of the preceding layer to current layer in four ways. In this experiment, we choose the Additive Multiplex PageRank and have two different setup, that is, **MPR-Commit** uses the commit layer as the first layer and **MPR-Issue** uses the issue layer as the first layer. The hyper parameter is set to 0.85.

M-BiRank: M-BiRank is the method we proposed for ranking nodes in multiplex bipartite network. As the setup in **Multiplex PageRank**, **M-BiRank-Commit** uses the commit layer as the first layer and **M-BiRank-Issue** uses the issue layer as the first layer. The hyper parameters γ and λ are set to 0.85. Each element of the query vector u^0 (p^0) for the corresponding node(developer/project) is set to the sum of its all edges' weights over the total sum of all edges' weights of the whole developer-project bipartite network of the first layer.

4.4 Results

We compare the experimental results of M-BiRank with baseline methods by both correlation analysis and SIR modeling.The hyper parameters for M-BiRank γ and λ are both set to 0.85.

4.4.1 Correlation analysis

In correlation analysis, the follower number of developers and the watch number of projects are set as the ground truth for ranking developers and projects, respectively. Pearson Correlation Coefficient(PCC) is calculated between the ranking results from M-BiRank and baseline methods and ground truth rankings. The results are shown in Table 3.

From the results of correlation analysis, we have the following observations:

(1) M-BiRank model we proposed outperforms all the baseline methods for both developer ranking and project ranking. This indicates that it is necessary to model multiple interactions between developers and projects as a multiplex bipartite network, which not only considers mutual enhancement between developers and projects but also takes into account mutual enhancement between different interactions. This highly agrees with real world practice. For example, a project with elite developers participating in is usually a popular project and a developer participating in popular projects is often an elite developer. Developers have different ways to take part in certain projects such as committing code or solving issues, and different ways are tightly coupled.

(2) Comparing the different settings of M-BiRank itself, M-BiRank-Commit performs better than M-BiRank-Issue in most cases, which means it is better to take the commit layer of the multiplex developer-project bipartite network as the initial layer for M-BiRank model. This also agrees with real world practice. Issue is a helper function in social collaborative coding which provides a discussion board for software developers about bugs and designs. While commit is a main function during software development for developers, thus the commit layer is more important. So M-BiRank-Commit performs better in identifying more influential developers and projects. In Section 4.4.2, we only compare M-BiRank-Commit with benchmark methods.

4.4.2 SIR simulation

In this section, to evaluate the information spreading ability of top-100 developers ranked by different methods, SIR model is adopted on commit layer of the developer-project multiplex bipartite network. M-BiRank is compared against each baseline methods separately. For each comparison, the initial infected nodes (developers) for SIR model is the top-100 developers ranked by each method excluding those ranked top-100 by both methods. During the SIR process, an infected node infects each of its neighbors with probability $\alpha = 0.005$ simultaneously and recoveries to Removed state with probability $\beta = 0.006$. For each SIR simulation, we run 300 iterations at most and repeat 10 times to average the value of each step. The results are shown in Figure 3 - Figure 7 and several significant observations are found:

(1) The result of comparison between different settings of M-BiRank itself in Figure 3 indicates M-BiRank-Commit performs better, which is in perfect accordance with the result found in correlation analysis in Section 4.4.1. Thus only M-BiRank-Commit is compared against baseline methods in the rest part of this section.

(2) M-BiRank outperforms all the baseline methods in identifying influential developers, which means nodes' types and mutual reinforcement among different interactions play important roles and multiplex bipartite network can model multiple interactions between two different types of nodes more precisely. Specially, the performance difference between M-BiRank and BiRank is larger than that between M-BiRank and Multiplex PageRank(MPR), from which we can conclude that considering mutual reinforcement among different interactions is of more importance than distinguishing nodes' types.

(3) The number of final infected projects is more than that of developers in both M-BiRank and all the baseline methods. According to researches on epidemics on networks, information spreads faster and broader in networks with shorter average path length. From Table 2, we can see the average degree of projects is larger than that of developers.

4.5 Case study

In addition to correlation analysis and SIR simulation, we further do a detailed case study to show the effectiveness of our model in identifying influential developers and projects. Top-20 developers and projects ranked by our model M-BiRank are listed in Table 4 and Table 5, respectively, followed by their ranks in baseline methods.

Table 4 indicates baseline methods and M-BiRank ranks the first six developers similarly, while some influential PHP developers ranked in top 20 by M-BiRank are not identified or ranked with lower scores by baseline methods. For example, Fabien Potencier (GitHub ID: fabpot) and Taylor Otwell (GitHub ID: taylorotwell), the most active contributors of the two most popular PHP framworks, Symfony and Laravel, are not identified as influential developers by some of the baseline methods. From GitHub as of June 1st, 2020, Symfony and Laravel have 23.3k and 59.4k stars, respectively, and Fabien Potencier and Taylor Otwell have 10.4k and 18.6k followers, respectively. Taylor Otwell has more followers than Fabien Potencier and Laravel is more popular than Symfony, but Fabien Potencier is ranked higher than Taylor Otwell because Laravel is based on some popular components of Symfony. Thus we can conclude that Fabien Potencier is more influential than Taylor Otwell.

As for projects, from Table 5 we can see both M-BiRank and baseline methods rank popular PHP frameworks with higher scores. But some important PHP components identified by M-BiRank are not identified as influential projects or ranked with lower scores by baseline methods. For example, *illuminate/database*, a popular ORM library, is ranked with high score by M-BiRank but is ranked with lower score by BiRank-Add and PageRank-Add, and is never identified as influential projects by BiRank-Avg, PageRank-Avg and MPR-Commit. As we know, in modern web development ORM is quite critical because it is responsible for accessing database.

4.6 Experimental settings discussion

In the experiment, several key settings will affect the performance of M-BiRank and a brief discussion about these settings is showed as follows.

First, we will study the impact of edge weight in the ranking process. Both unweighted and weighted developer-project multiplex bipartite network are constructed and for weighted case the interaction times are summed as edge weight. Then correlation analysis on top-k developers and projects are applied and the results are shown in Figure 8, from which it can be concluded that weighted developer-project multiplex bipartite network performs better than unweighted case and edge weight plays an important role in identifying influential developers and projects.

Then, experimental settings for SIR simulation are discussed. It can be seen from Figure 9 that the more initial infected nodes are set, the more final infected nodes. It is also obvious that the same number of top-k projects being set as initial infected nodes will result in more final infected nodes.

Finally, the hyper parameters γ and λ of our proposed M-BiRank model are analyzed. For simplicity, we consider the condition that γ and λ are equal. It can be concluded from Figure 10 that both prior belief of developers' (projects') importance and rankings from network structure play roles in the final rankings of developers (projects) and their contributions to final rankings are approximately equal.

5 Conclusions

In this work, we study the problem of identifying influential developers and projects in open source software community. We model multiple interactions between developers and projects as a multiplex bipartite network and propose an iterative refinement ranking method M-BiRank by incorporating the mutual reinforcement between developers and projects as well as between multiple developer-project interactions. The proposed M-BiRank is evaluated against four baseline methods on real world GitHub dataset. Extensive experimental analysis and case study show M-BiRank significantly outperforms baseline methods in both correlation analysis and SIR simulation.

The general idea behind the proposed M-BiRank is modeling multiple kinds of entities and interactions in open source software community into a single network and incorporating mutual reinforcement between different kinds of entities as well as between different types of interactions when ranking. As we know, there are other entities such as blogs and organizations in addition to developers and projects in open source software community and plenty of interactions between them such as user-user following and project-project dependency. In future work, more entities

and interactions could be introduced and modeled as a heterogeneous information network and mutual reinforcement in ranking would be generalized using meta-path.

Abbreviations
M-BiRank: Multiplex Bipartite Ranking;
PCC: Pearson Correlation Coefficient;
MPR: Multiplex PageRank

Acknowledgements
None.

Author's contributions
Dengcheng Yan and Bin Qi designed the model. Bin Qi conducted the experiment and prepared the figures. Dengcheng Yan wrote the entire article. All authors read and approved the final manuscript.

Funding
This work is supported by the National Natural Science Foundation of China (Grant No. 61872002), the University Natural Science Research Project of Anhui Province (Grant No. KJ2019A0037), the University Collaborative Innovation Program of Anhui Province (Grant No. GXXT-2019-013) and the Doctoral Scientific Research Foundation of Anhui University (Grant No. Y040418194).

Availability of data and materials
The data sets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer Science and Technology, Anhui University, Jiulong Road No. 111, 230601, Hefei, China.
[2]Institutes of Physical Science and Information Technology, Anhui University, Jiulong Road No. 111, 230601, Hefei, China.

References
1. Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., Shen, X.S.: Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. IEEE Trans. on Cloud Comput. (2019). doi:10.1109/TCC.2019.2923692
2. Zhang, Y., Pan, J., Qi, L., He, Q.: Privacy-preserving quality prediction for edge-based iot services. Future Gener. Comput. Syst. **114**, 336–348 (2021). doi:10.1016/j.future.2020.08.014
3. Chen, Y., Zhang, N., Zhang, Y., Chen, X.: Dynamic computation offloading in edge computing for internet of things. IEEE Internet Things J. **6**(3), 4242–4251 (2019). doi:10.1109/JIOT.2018.2875715
4. Chen, Y., Zhang, Y., Wu, Y., Qi, L., Chen, X., Shen, X.: Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply. IEEE Internet Things J. (2020). doi:10.1109/JIOT.2020.2992522
5. Joblin, M., Apel, S., Hunsen, C., Mauerer, W.: Classifying developers into core and peripheral: An empirical study on count and network metrics. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), pp. 164–174 (2017). doi:10.1109/ICSE.2017.23. https://doi.org/10.1109/ICSE.2017.23
6. Yan, D.-C., Wei, Z.-W., Han, X.-P., Wang, B.-H.: Empirical analysis on the human dynamics of blogging behavior on github. Physica A **465**, 775–781 (2017). doi:10.1016/j.physa.2016.08.054
7. Hu, Y., Wang, S., Ren, Y., Choo, K.-K.R.: User influence analysis for github developer social networks. Expert Syst. Appl. **108**, 108–118 (2018). doi:10.1016/j.eswa.2018.05.002
8. Xuan, Q., Fu, C., Yu, L.: Ranking developer candidates by social links. Adv. Complex Syst. **17**, 1550005 (2014). doi:10.1142/S0219525915500058
9. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab (1999)
10. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999). doi:10.1145/324133.324140
11. He, X., Gao, M., Kan, M.-Y., Wang, D.: Birank: Towards ranking on bipartite graphs. IEEE Trans. Knowl. Data Eng. **29**(1), 57–71 (2016). doi:10.1109/TKDE.2016.2611584
12. Liu, H., Kou, H., Yan, C., Qi, L.: Keywords-driven and popularity-aware paper recommendation based on undirected paper citation graph. Complexity (2020). doi:10.1155/2020/2085638
13. Halu, A., Mondragón, R.J., Panzarasa, P., Bianconi, G.: Multiplex pagerank. PloS One **8**(10), 78293 (2013). doi:10.1371/journal.pone.0078293
14. Iacovacci, J., Rahmede, C., Arenas, A., Bianconi, G.: Functional multiplex pagerank. EPL **116**(2), 28004 (2016). doi:10.1209/0295-5075/116/28004
15. Zhang, Y., Zhou, Y., Wang, F., Sun, Z., He, Q.: Service recommendation based on quotient space granularity analysis and covering algorithm on spark. Knowledge-Based Syst. **147**, 25–35 (2018). doi:10.1016/j.knosys.2018.02.014
16. Zhang, Y., Cui, G., Deng, S., Chen, F., Wang, Y., He, Q.: Efficient query of quality correlation for service composition. IEEE Trans. Serv. Comput. (2018). doi:10.1109/TSC.2018.2830773

17. Zhang, Y., Wang, K., He, Q., Chen, F., Deng, S., Zheng, Z., Yang, Y.: Covering-based web service quality prediction via neighborhood-aware matrix factorization. IEEE Trans. Serv. Comput. (2019). doi:10.1109/TSC.2019.2891517
18. Zhang, Y., Yin, C., Wu, Q., He, Q., Zhu, H.: Location-aware deep collaborative filtering for service recommendation. IEEE Trans. Syst., Man, Cybern. Syst. (2019). doi:10.1109/TSMC.2019.2931723
19. Qi, L., Wang, X., Xu, X., Dou, W., Li, S.: Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing. IEEE Transactions on Network Science and Engineering (2020). doi:10.1109/TNSE.2020.2969489
20. Zhou, C., Li, A., Hou, A., Zhiwang, Z., Zhang, Z., Dai, P., Wang, F.: Modeling methodology for early warning of chronic heart failure based on real medical big data. Expert Syst. Appl. **151**, 113361 (2020). doi:10.1016/j.eswa.2020.113361
21. Chen, D., Lü, L., Shang, M.-S., Zhang, Y.-C., Zhou, T.: Identifying influential nodes in complex networks. Physica A **391**(4), 1777–1787 (2012). doi:10.1016/j.physa.2011.09.017
22. Chen, D.-B., Gao, H., Lü, L., Zhou, T.: Identifying influential nodes in large-scale directed networks: the role of clustering. PLoS One **8**(10) (2013). doi:10.1371/journal.pone.0077455
23. Liu, H., Kou, H., Yan, C., Qi, L.: Link prediction in paper citation network to construct paper correlation graph. EURASIP J. Wireless Com. Network **2019**(1) (2019). doi:10.1186/s13638-019-1561-7
24. Zhao, S.X., Rousseau, R., Fred, Y.Y.: h-degree as a basic measure in weighted networks. J. Informetr. **5**(4), 668–677 (2011). doi:10.1016/j.joi.2011.06.005
25. Liu, Q., Zhu, Y.-X., Jia, Y., Deng, L., Zhou, B., Zhu, J.-X., Zou, P.: Leveraging local h-index to identify and rank influential spreaders in networks. Physica A **512**, 379–391 (2018). doi:10.1016/j.physa.2018.08.053
26. Lü, L., Zhou, T., Zhang, Q.-M., Stanley, H.E.: The h-index of a network node and its relation to degree and coreness. Nat. Commun. **7**, 10168 (2016). doi:10.1038/ncomms10168
27. Sabidussi, G.: The centrality index of a graph. Psychometrika **31**(4), 581–603 (1966). doi:10.1007/BF02289527
28. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35–41 (1977). doi:10.2307/3033543
29. Stephenson, K., Zelen, M.: Rethinking centrality: Methods and examples. Soc. Networks **11**(1), 1–37 (1989). doi:10.1016/0378-8733(89)90016-6
30. Wang, D., Wang, H., Zou, X.: Identifying key nodes in multilayer networks based on tensor decomposition. Chaos **27**(6), 063108 (2017). doi:10.1063/1.4985185
31. Wang, D., Zou, X.: A new centrality measure of nodes in multilayer networks under the framework of tensor computation. Appl. Math. Model. **54**, 46–63 (2018). doi:10.1016/j.apm.2017.07.012
32. Lü, L., Zhang, Y.-C., Yeung, C.H., Zhou, T.: Leaders in social networks, the delicious case. PloS One **6**(6) (2011). doi:10.1371/journal.pone.0021202
33. Zhong, W., Yin, X., Zhang, X., Li, S., Dou, W., Wang, R., Qi, L.: Multi-dimensional quality-driven service recommendation with privacy-preservation in mobile edge environment. Comput. Commun. **157**, 116–123 (2020). doi:10.1016/j.comcom.2020.04.018
34. Liang, R., Jiang, X.: Scientific ranking over heterogeneous academic hypernetwork. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 20–26 (2016)
35. Zhao, H., Xu, X., Song, Y., Lee, D.L., Chen, Z., Gao, H.: Ranking users in social networks with motif-based pagerank. IEEE Trans. Knowl. Data Eng. (2019). doi:10.1109/TKDE.2019.2953264
36. Gousios, G.: The ghtorrent dataset and tool suite. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 233–236 (2013). doi:10.5555/2487085.2487132. https://dl.acm.org/doi/10.5555/2487085.2487132
37. Xu, X., Fu, S., Qi, L., Zhang, X., Liu, Q., He, Q., Li, S.: An iot-oriented data placement method with privacy preservation in cloud environment. J. Netw. Comput. Appl. **124**, 148–157 (2018). doi:10.1016/j.jnca.2018.09.006
38. Hauke, J., Kossowski, T.: Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. Quaestiones geographicae **30**(2), 87–93 (2011). doi:10.2478/v10117-011-0021-1
39. Li, J., Cai, T., Deng, K., Wang, X., Sellis, T., Xia, F.: Community-diversified influence maximization in social networks. Inf. Syst. **92**, 101522 (2020). doi:10.1016/j.is.2020.101522
40. Yang, R., Wang, B.-H., Ren, J., Bai, W.-J., Shi, Z.-W., Wang, W.-X., Zhou, T.: Epidemic spreading on heterogeneous networks with identical infectivity. Phys. Lett. A **364**(3-4), 189–193 (2007). doi:10.1016/j.physleta.2006.12.021
41. Xu, X., Mo, R., Dai, F., Lin, W., Wan, S., Dou, W.: Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. IEEE Trans. Ind. Inform. **16**(9), 6172–6181 (2019). doi:10.1109/TII.2019.2959258
42. Bonacich, P.: Factoring and weighting approaches to status scores and clique identification. J. Math. Sociol. **2**(1), 113–120 (1972). doi:10.1080/0022250X.1972.9989806
43. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. Rev. Mod. Phys. **87**(3), 925 (2015). doi:10.1103/RevModPhys.87.925

Figures

Figure 1: **M-BiRank Framework.** M-BiRank contains three iterative steps.

Tables

Figure 2: **SIR model.** (Source: Figure is adapted from Pastor-Satorras et al. [43])

Figure 3: **Performance comparison between M-BiRank-Commit and M-BiRank-Issue.** The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

Figure 4: **Performance comparison between M-BiRank and Degree.** The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

Figure 5: **Performance comparison between M-BiRank and PageRank.** The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

Figure 6: **Performance comparison between M-BiRank and BiRank.** The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

Figure 7: **Performance comparison between M-BiRank and MPR.** The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

Figure 8: **Impact of edge weight.** The horizontal axis K indicates the number of initial infected nodes in SIR modeling. The vertical axis N_I indicates the number of final inftected nodes (including Infected and Removed nodes).

Figure 9: **Impact of the number of initial infected nodes on final infected nodes.**

Figure 10: **Hyper parameters analysis.**

Table 1: Notations and explanations.

Notation	Explanation
u_i^A	ranking score of developer i in layer A
p_j^A	ranking score of project j in layer A
W^A	weight matrix of layer A, its element w_{ij}^A represents the edge weight between developer i and project j in layer A
γ, λ	hyper parameters
u_i^0, p_j^0	query vectors for developer i and project j
θ	threshold for SIR simulation
$\|U^A\|$, $\|P^A\|$	numbers of developers and projects in layer A
d_i^A, d_j^A	degrees of developer i and project j in layer A
S^A	symmetric normalization of weight matrix W^A

Table 2: The statistics of the dataset.

Data	Count
developers	147105
projects	126415
developer-project commit relations	205478
developer-project issue relations	441558
developer-project watch relations	68566
developer-developer follow relations	10701

Table 3: Correlation analysis.

Method	Top-20		Top-50		Top-100	
	Project	Developer	Project	Developer	Project	Developer
Degree	0.441	0.090	0.383	0.173	0.393	0.219
BiRank-Avg	0.402	-0.014	0.442	0.091	0.462	0.016
BiRank-Add	0.398	-0.022	0.422	-0.062	0.487	0.030
PageRank-Avg	0.430	-0.116	0.450	-0.027	0.498	-0.035
PageRank-Add	0.357	-0.111	0.419	-0.067	0.475	-0.004
MPR-Issue	0.350	0.113	0.437	0.160	0.458	0.202
MPR-Commit	0.464	-0.066	0.459	-0.013	0.516	0.017
M-BiRank-Issue	0.374	0.057	0.423	0.068	0.447	**0.273**
M-BiRank-Commit	**0.468**	**0.152**	**0.459**	**0.261**	**0.541**	0.143

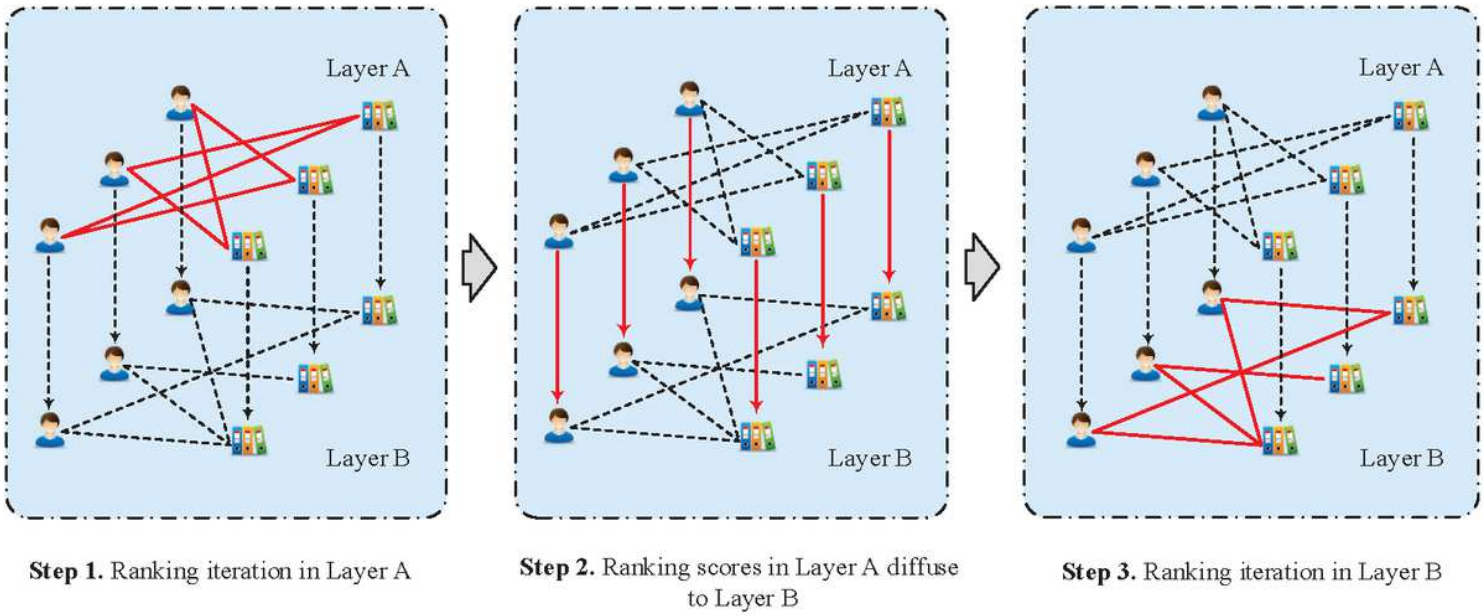
Table 4: Top-20 developers.

Developer	Ranking							
	M-BiRank	Degree	BiRank-Avg	BiRank-Add	PageRank-Avg	PageRank-Add	MPR-Issue	MPR-Commit
scrutinizer-auto-fixer	1	1	10	2	5	2	1	67
GrahamCampbell	2	3	2	3	2	3	2	7
danielbachhuber	3	7	3	6	4	6	9	3
weierophinney	4	2	13	4	12	4	4	42
stof	5	12	5	7	6	7	6	4
freekmurze	6	5	19	5	21	5	5	83
Nyholm	7	8	9	11	11	12	10	15
kartik-v	8	6	35	20	43	20	7	95
Ocramius	9	7	12	12	13	10	11	19
crynobone	10	16	8	9	10	9	12	11
fabpot	11	20	81	34	55	23	8	-
clue	12	18	7	13	8	13	17	8
sagikazarmark	13	23	6	10	7	11	15	5
lsmith77	14	37	16	17	16	15	16	14
dbu	15	9	24	29	20	18	20	17
cordoval	16	19	4	8	3	8	19	2
pippinsplugins	17	14	22	26	24	29	26	25
taylorotwell	18	43	-	30	66	25	13	-
dantleech	19	27	17	23	17	21	21	16
splitbrain	20	13	21	14	26	14	29	47

Table 5: Top-20 projects.

Project	Ranking							
	M-BiRank	Degree	BiRank-Avg	BiRank-Add	PageRank-Avg	PageRank-Add	MPR-Issue	MPR-Commit
symfony/symfony	1	1	2	2	2	2	1	2
laravel/framework	2	3	1	1	1	1	2	1
magento/magento2	2	7	4	4	4	4	3	4
yiisoft/yii2	4	2	3	3	3	3	4	3
owncloud/core	5	12	5	5	5	5	5	6
joomla/joomla-cms	6	5	6	7	7	7	6	7
cakephp/cakephp	7	8	8	9	9	9	7	9
Automattic/jetpack	8	60	13	16	18	19	11	19
symfony/symfony-docs	9	4	14	21	19	20	9	20
EllisLab/CodeIgniter	10	6	10	11	11	11	10	12
pyrocms/pyrocms	11	12	12	12	13	12	14	14
composer/composer	12	22	7	6	6	6	8	5
woocommerce/woocommerce	13	18	9	8	8	8	12	8
phalcon/cphalcon	14	17	11	10	10	10	13	10
illuminate/database	15	16	-	53	-	55	15	-
sonata-project/SonataAdminBundle	16	32	18	20	15	18	17	16
Sylius/Sylius	17	23	16	22	16	22	16	15
yiisoft/yii	18	19	20	18	21	17	18	21
PrestaShop/PrestaShop	19	14	31	32	41	34	24	54
laravel/laravel	20	35	19	19	22	21	22	18

Figures



Step 1. Ranking iteration in Layer A

Step 2. Ranking scores in Layer A diffuse to Layer B

Step 3. Ranking iteration in Layer B

Figure 1

M-BiRank Framework. M-BiRank contains three iterative steps.

Figure 2

SIR model. (Source: Figure is adapted from Pastor-Satorras et al. [43])

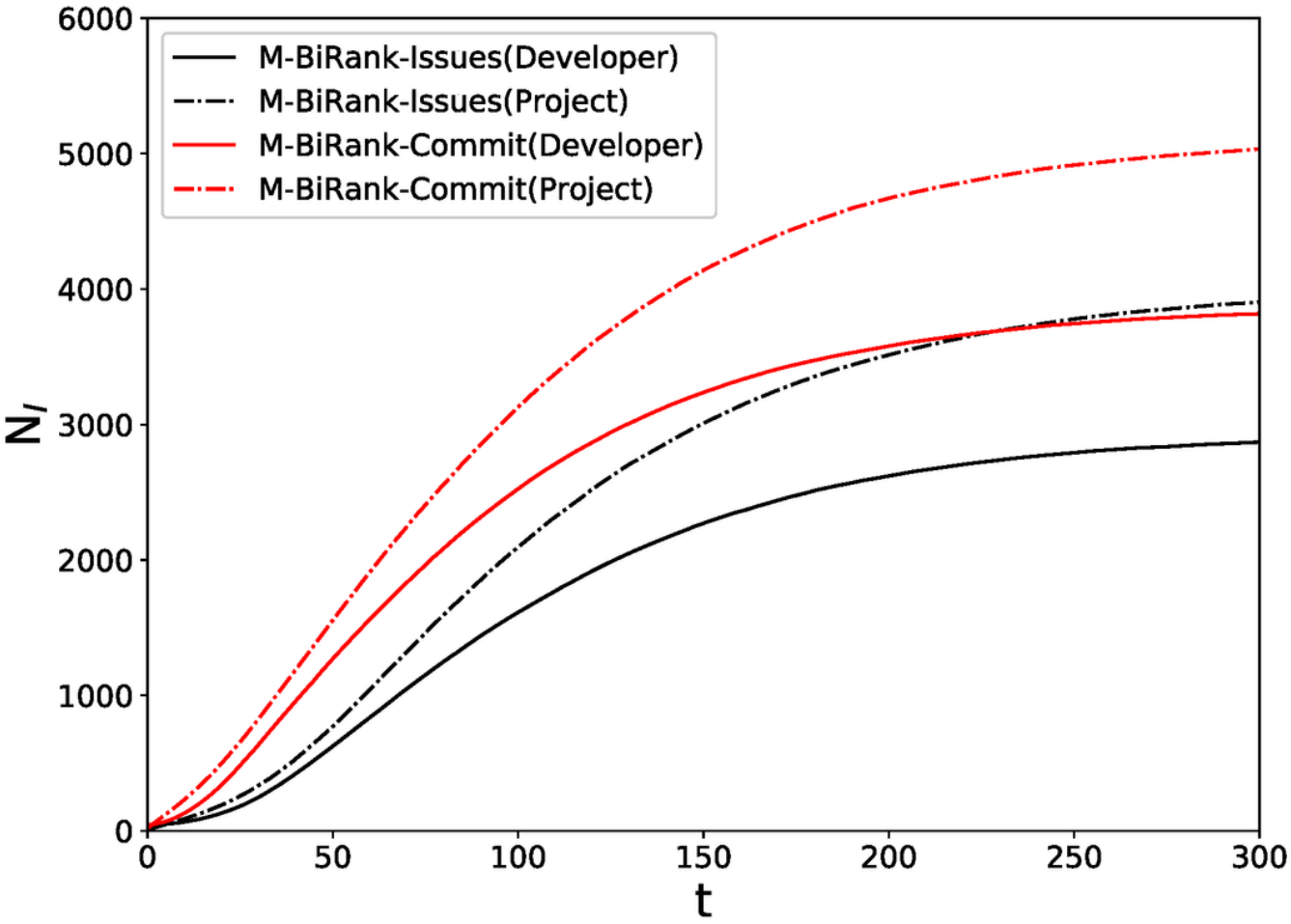


Figure 3

Performance comparison between M-BiRank-Commit and MBiRank-Issue. The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

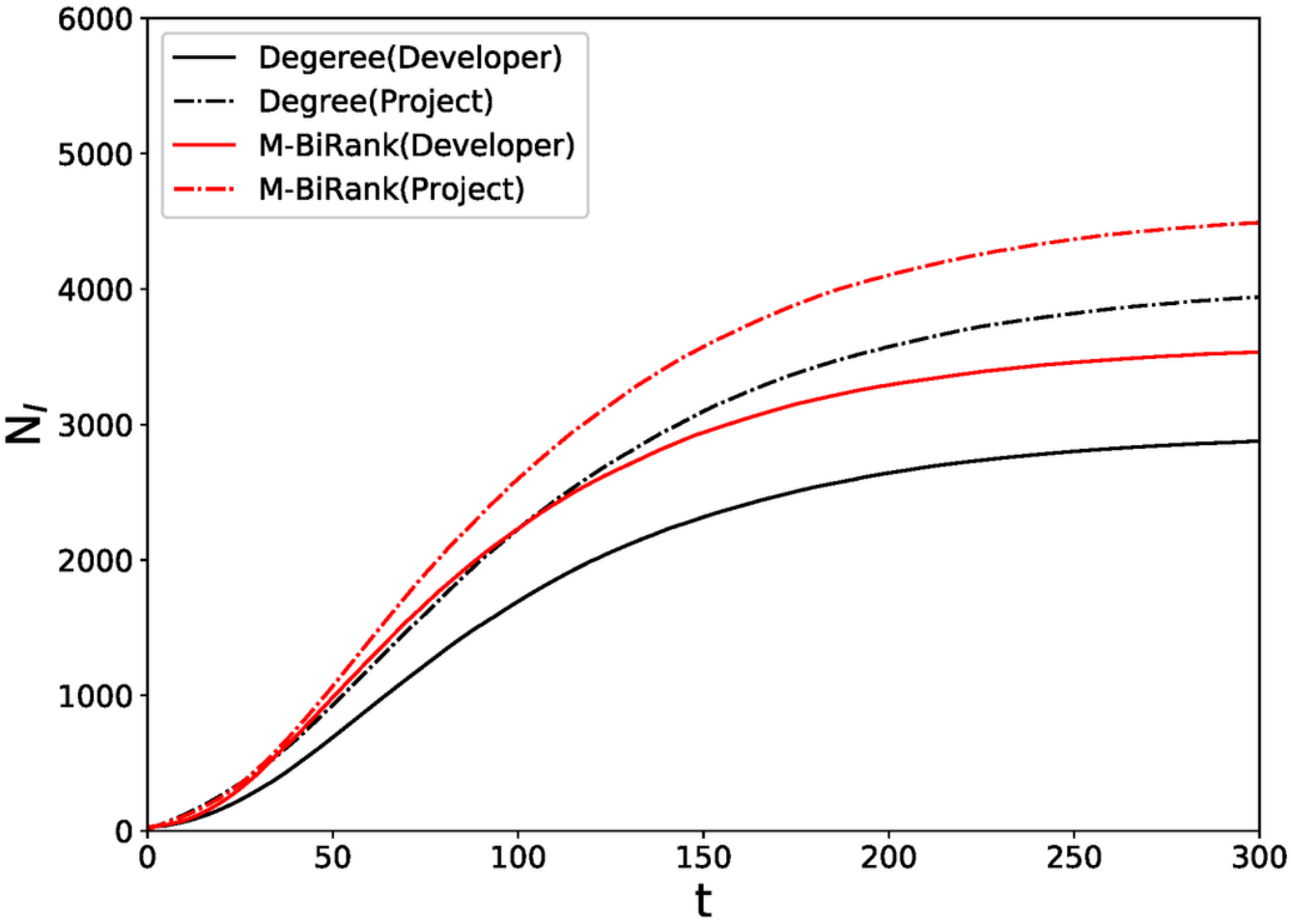


Figure 4

Performance comparison between M-BiRank and Degree. The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

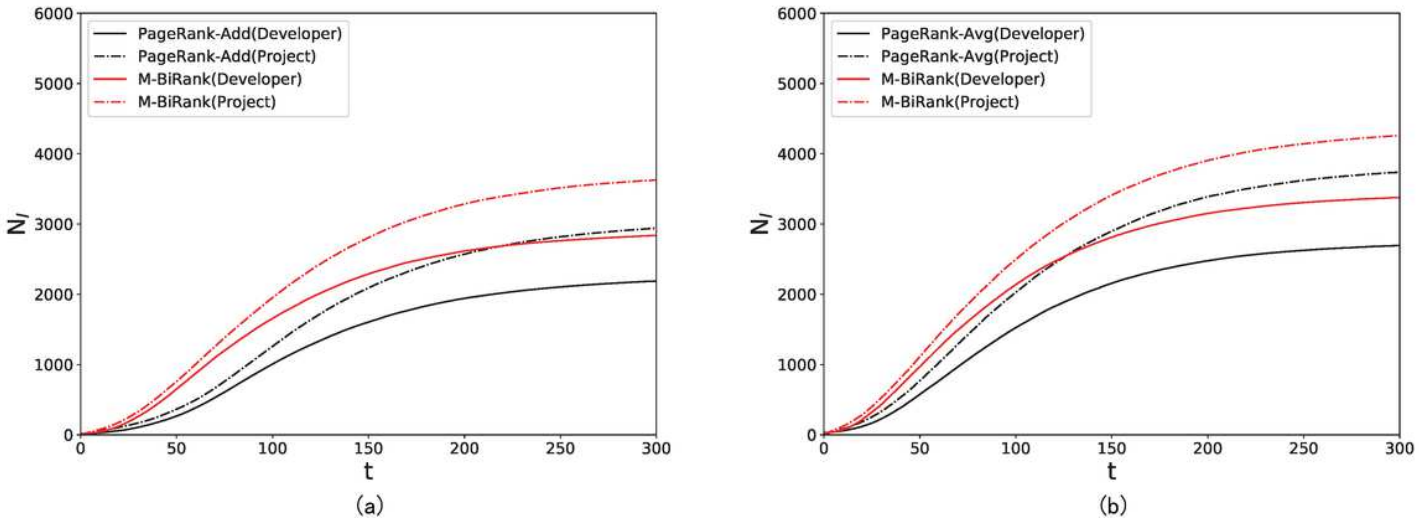


Figure 5

Performance comparison between M-BiRank and PageRank. The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

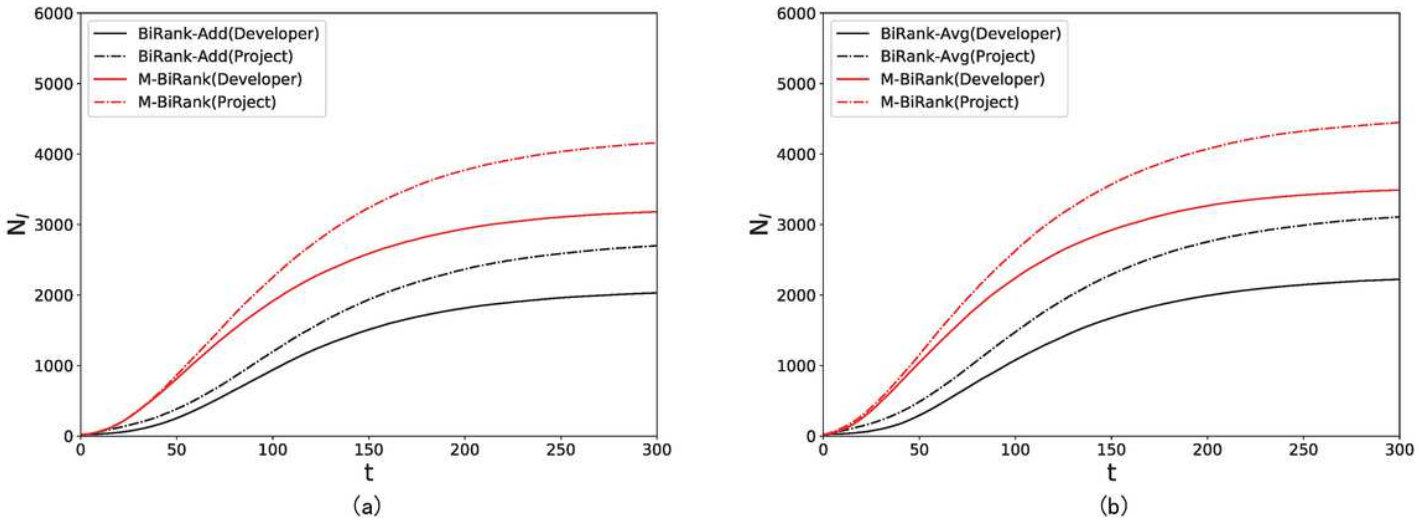


Figure 6

Performance comparison between M-BiRank and BiRank. The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

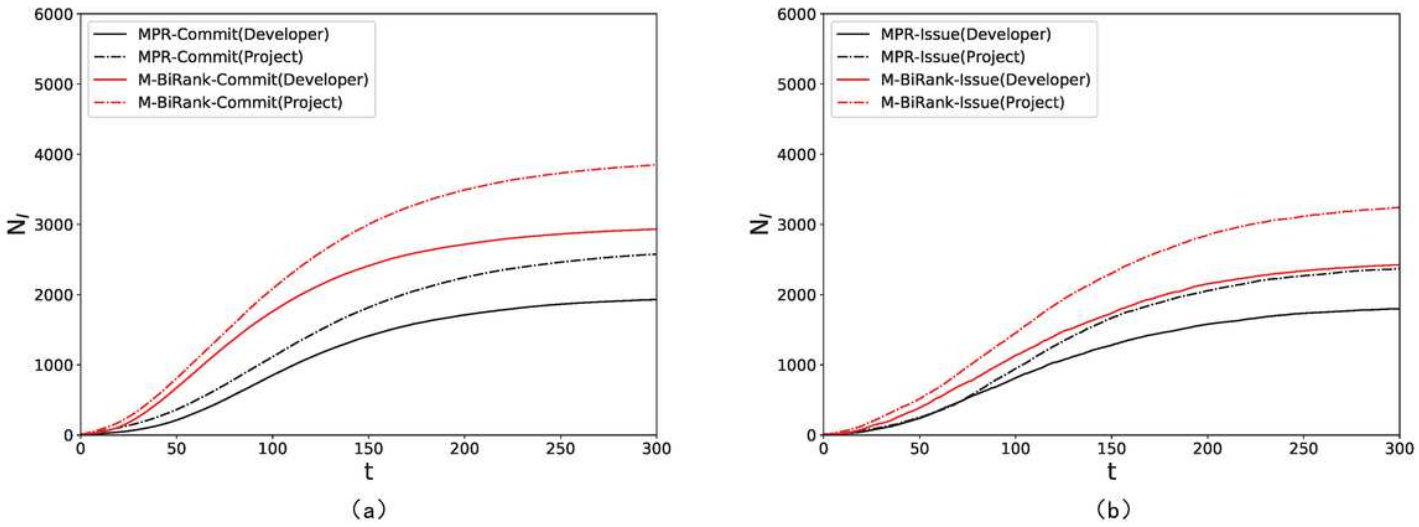


Figure 7

Performance comparison between M-BiRank and MPR. The horizontal axis t indicates the number of iterations in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (developers or projects, including Infected and Removed nodes).

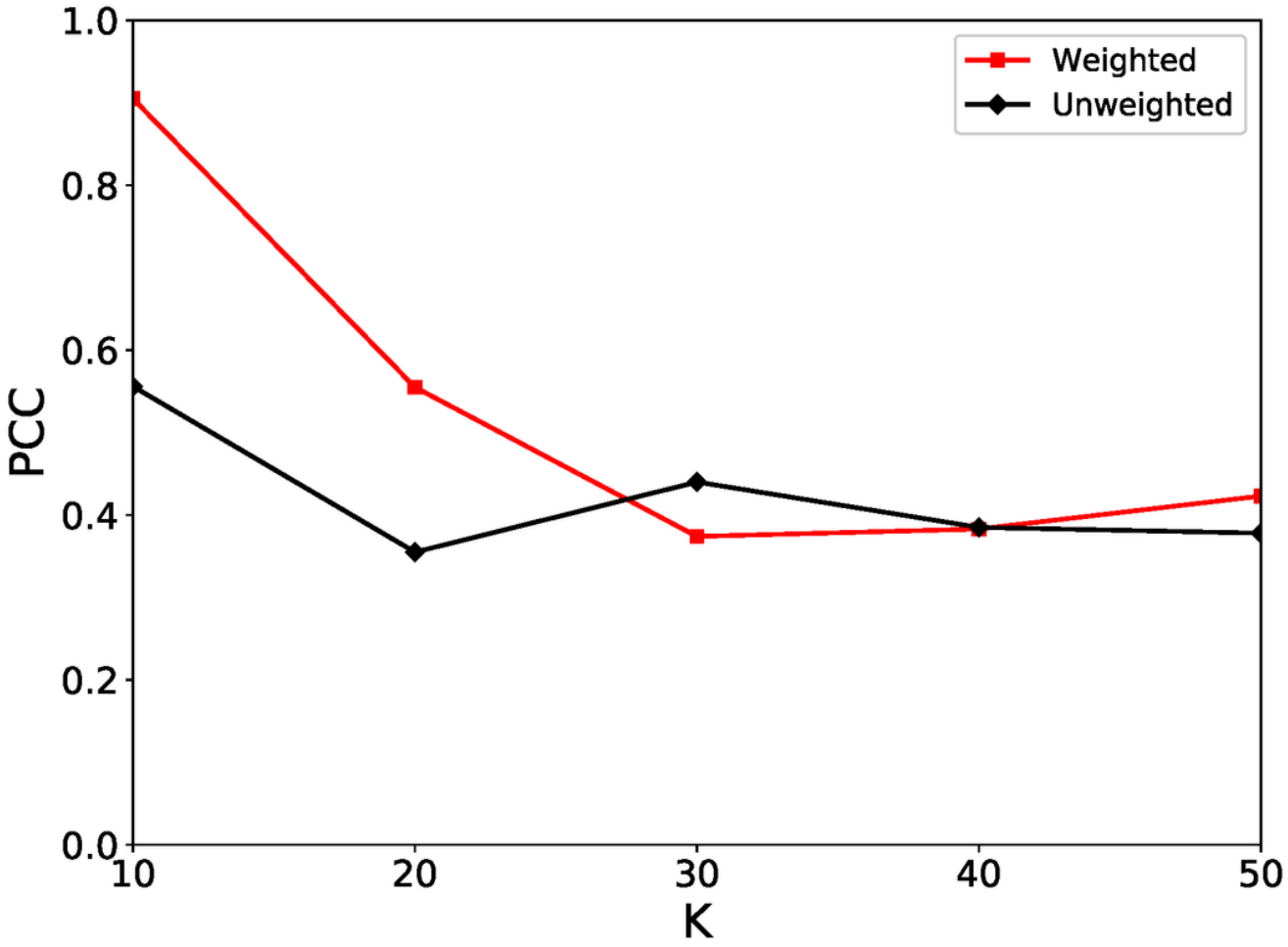


Figure 8

Impact of edge weight. The horizontal axis K indicates the number of initial infected nodes in SIR modeling. The vertical axis NI indicates the number of final inftected nodes (including Infected and Removed nodes).

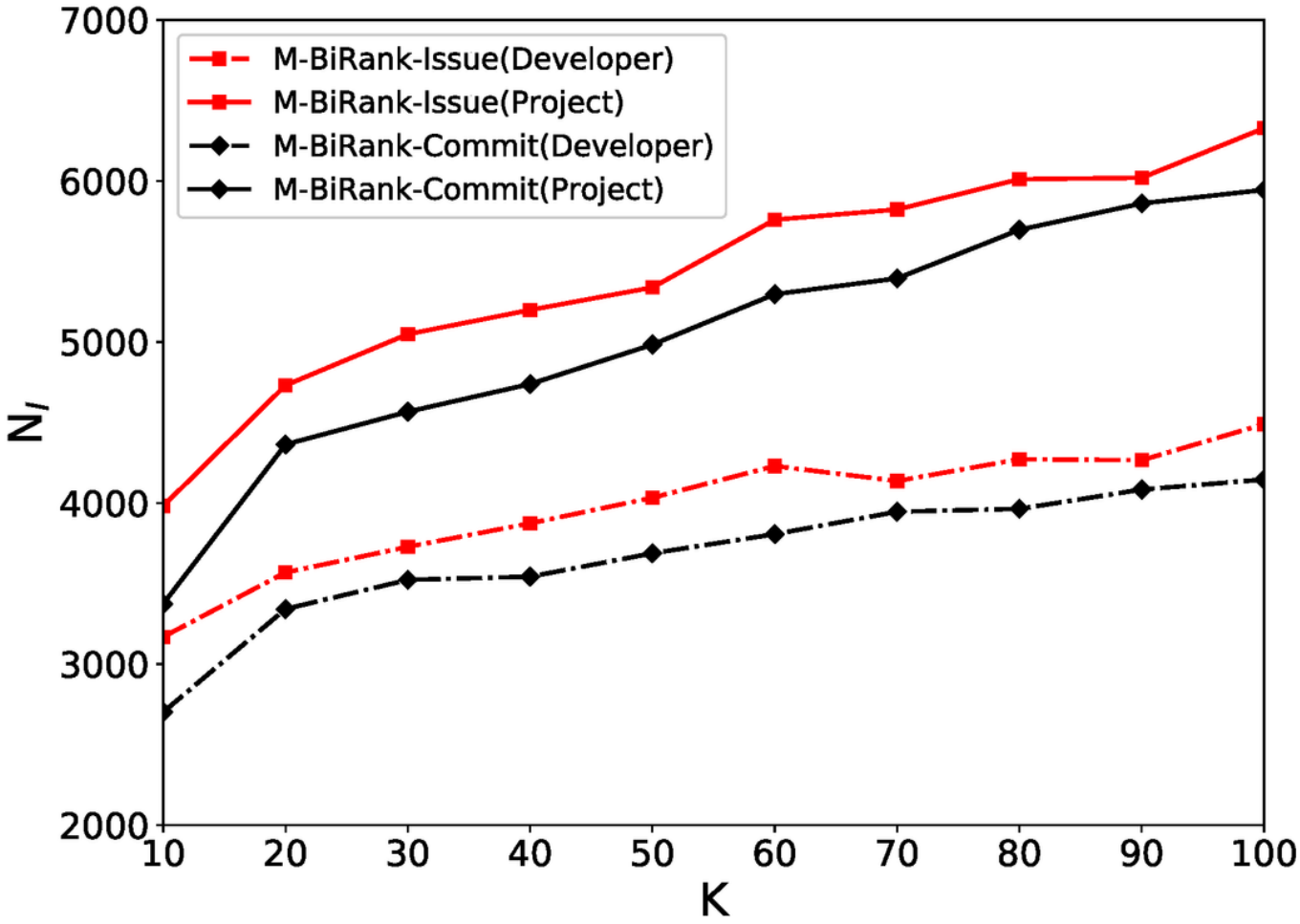


Figure 9

Impact of the number of initial infected nodes on final infected nodes.

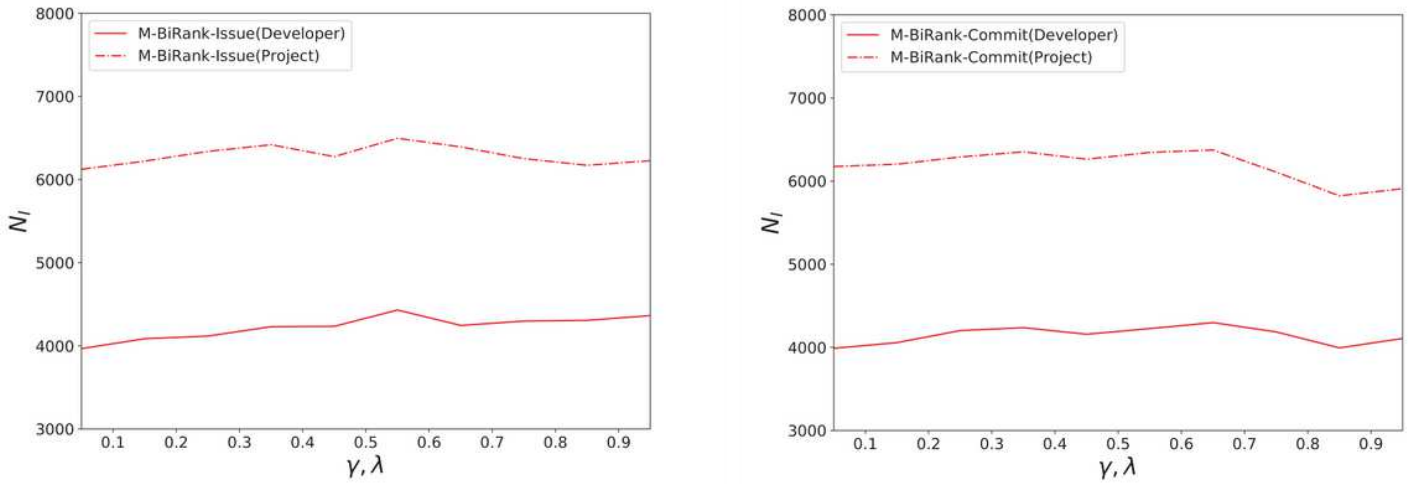


Figure 10

Hyper parameters analysis.