硕士学位论文

基于改进LeaderRank算法的Github用户影响力评价

Evaluation of Users' Influences in Github Based on Improved LeaderRank Algorithm

作　者：王启瑞
导　师：巩敦卫 教授

中国矿业大学

二〇二一年六月

中图分类号 TP311.5　　学校代码 10290

UDC 004.6　　密　　级 公开

中国矿业大学

硕士学位论文

基于改进 LeaderRank 算法的 Github 用户影响力评价

Evaluation of Users' Influences in Github Based on Improved LeaderRank Algorithm

作　　者 王启瑞　　导　　师 巩敦卫

申请学位 工学硕士学位　　培养单位 信息与控制工程学院

学科专业 控制科学与工程　　研究方向 用户影响力评价

答辩委员会主席 郭西进　　评 阅 人

二〇二一年六月

致谢

岁月如歌，光阴似箭，在南湖七年的生活即将结束。经历了入学时的朝气蓬勃，熬夜复习的汗水苦涩，求职面试的喧嚣坎坷，我深深体会到写论文时的宁静与思考，这是一个结束，亦是新的开始。回首三年研究生生涯，对那些引导、帮助、激励我的人，我心中充满了感激。

首先要感谢我的导师巩敦卫教授！恩师巩敦卫教授亲切且温和，让我在求学科研之路上感受到了家的温暖。巩老师不仅为人和善，而且治学态度认真严谨。在这三年来，巩老师每每是最早来实验室的人之一，无论雪雨秋冬。即使在夜晚，仍然能看到老师在群里安排学术工作的通知，这种勤勉的态度深深地激励着我。在每周的组会上，巩老师也严格把握我的学术进度，像骆驼，像灯塔，像浪潮，指引我向前迈进。从研究方向，论文定题，到论文写作与修改，巩老师倾注了超多心血，他深深地影响着我，使我学会了乐观积极的心态，提高了自身的科研能力，能师从巩老师，我为自己感到庆幸。

同时，我还要感谢我的父母。焉得艾草，言树之心，你们是我最坚强的后盾。当我步入倦怠和自我怀疑的深渊，是你们将我打捞起，默默包容我的无理与任性。是你们让我知道，不管在外经受多少风吹雨打，我始终拥有一个宁静的港湾，你们永远是支持我前进的动力。

感谢我的母校矿大！还有相聚在矿大的可爱的人们！母校“源深流自远，行健天同功”的校训一直深深地刻在我的心里，在我的人生中留下无数印记。在这样舒适良好的学习氛围下，我得益于信控学院全体老师的精彩授课和谆谆教诲，成为了可以独当一面的人。感谢姚老师三年来的教导，在每一次组会中给予我耐心的指导和极大的启发帮助。感谢师兄沈鑫，师姐王燕，师弟郑瑞钊，同窗杜莹，是你们在我迷茫和失败时鼓励我，安慰我，让我重新振作。感谢室友李瑞杰，是你陪我分享每一次的喜怒哀乐与迷茫困惑。感谢大家陪我度过了这三年丰富多彩的生活。

最后，感谢在百忙之中参与本论文审阅和给予宝贵指导意见的各位专家！在此向各位专家致以最诚挚的谢意！

摘　要

Github 是一种常用的开源软件社区，也是一种特殊的社交网络，存在复杂的社会关系和多样的用户属性。分析该社区用户的影响力，能够识别社区中的关键用户。基于此，社区管理者可以对这些用户施加正面举措，使社区在关键用户的引导下持续健康发展。目前，关键用户主要通过节点重要性度量方法进行识别，由于 Github 的特殊性，这些算法存在检测效率低下，识别结果不精准的问题。更为重要的是，由于经典的算法没有考虑节点之间的相互作用，不能充分应用于带权网络中。因此，研究有针对性的理论与方法，对 Github 进行用户影响力评价是非常必要的。本文基于 Github 中用户自身属性以及不同用户之间的交互关系，对经典 LeaderRank 算法进行改进，研究 Github 用户影响力评价方法。本文的主要研究工作如下：

（1）针对现有算法只考虑了节点拓扑属性，不能精准应用于 Github 用户影响力评价，提出一种基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价方法（MADM_LR）。该方法基于用户的社会行为特征，设计多属性决策指标，利用改进的 LeaderRank 算法，计算用户的影响力排名。将所提方法应用于 Github 用户关注网络，并与已有方法对比。实验结果表明，MADM_LR 既能够识别影响力更强的用户，还有较强的抗干扰性。

（2）针对已有复杂网络节点重要性度量方法，不能充分考虑 Github 用户间的相互作用，提出一种基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评价方法（US_WLR）。首先给出用户相似度和有向网络 H 指数相关定义。在保证排序结果精准的基础上，结合以上定义进一步构造了带权的 Github 网络来反映节点间作用强弱。在构造的网络基础上，有效融合节点的初始影响力和网络边权，提出改进带权的 LeaderRank 算法。将所提方法应用于改造的 Github 用户关注网络中，实验结果表明，US_WLR 能够识别出传播能力更强的用户。

本文结合用户属性提取、网络构建、模型构造与算法求解等多个方面，给出 Github 用户影响力的评价方法，既能与 Github 用户的自身特征紧密结合，还能高效精准地识别出影响力更高传播能力更强的用户。本文得到的用户影响力排序结果不仅能用于评估 Github 社区健康性，还可以用于预测社区的发展趋势。因此，具有重要的理论意义和应用价值。

该论文有图 14 幅，表 4 个，参考文献 85 篇。

关键词：Github；用户影响力评价；LeaderRank 算法；多属性决策

Abstract

Github is a common FOSS community and a special social network. It contains complex social relations and various user attributes. By analyzing the influence of users in the community, we can identify key users. And community managers can take positive measures to these users, so that the community can develop continuously and healthily. At present, key users are mainly identified by node importance measurement. Due to the particularity of GitHub, these algorithms have the problems of low detection efficiency and low accuracy. Most importantly, those algorithms did not consider the interaction between nodes. The existing method cannot sufficiently apply on the weighted network. It is critical to develop an accurate evaluation method about Github users' influence. Based on the user's own attributes and the interaction between different users, this paper improves the classic LeaderRank algorithm and proposes a method to identify key users. The results are shown as follows:

(1) In view of the fact that the existing algorithms only consider the topological attributes of nodes and can not be accurately applied to GitHub users' influence evaluation, this paper proposes a evaluation method based on multi-attribute decision-making and improved LeaderRank algorithm (MADM_LR). This method based on users' social behavior, provide multiple attributes targeting to user's influence and calculate the user's influence rank with the improved algorithm. The proposed method is applied to GitHub and compared with the existing methods. Experimental results show that MADM_LR can not only identify more influential users, but also have better anti-interference performance.

(2) In view of the fact that the existing node importance measurement methods in complex networks can't fully consider the interaction between users, this paper proposes a GitHub users' influence evaluation method based on users' similarity and improved weighted LeaderRank algorithm (US_WLR). First, we give users' similarity and H-index for directed networks. The weighted network is constructed to reflect the interactions between nodes based on the accuracy of sorting results. On the basis of the constructed network, this paper proposes an improved weighted LeaderRank algorithm which effectively integrates the initial influence of nodes and the network edges' weight. The proposed method is applied to the modified users' follow network, and the results show that US_WLR can identify users with stronger propagation ability.

Combined with user attribute extraction, network construction, model construction and algorithm solving, we give the evaluation method of GitHub users' influence. This method can not only combine with the characteristics of users, but also identify the users with higher influence and stronger communication ability efficiently and accurately. The ranking results of users' influence can be used not only to evaluate the health of community, but also to predict the development trend. Therefore, it has important theoretical significance and application value.

The thesis has 14 figures, 4 tables, and 85 references.

Keywords: GitHub; evaluation of users' influence; LeaderRank algorithm; multi-attribute decision-making

目 录

摘要.....I
目录.....IV
图清单.....VIII
表清单.....IX
变量注释表.....X
1 绪论.....1
1.1 研究背景.....1
1.2 研究动机.....2
1.3 国内外研究现状.....3
1.4 研究内容.....7
1.5 论文结构.....7
2 相关工作.....9
2.1 复杂网络简介.....9
2.2 PageRank 算法和 LeaderRank 算法分析.....11
2.3 多属性决策.....13
3 基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价.....15
3.1 研究动机.....15
3.2 所提方法的基本框架.....16
3.3 基于多属性决策的 LeaderRank 算法.....17
3.4 实验.....20
3.5 本章小结.....26
4 基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评价.....27
4.1 研究动机.....27
4.2 基本概念.....28
4.3 所提方法的基本框架.....30
4.4 基于用户相似度的带权 LeaderRank 算法.....31
4.5 实验.....36
4.6 本章小结.....41
5 结论.....42
5.1 本文工作.....42
5.2 进一步研究工作.....42

参考文献.....44
作者简历.....50
学位论文原创性声明.....51
学位论文数据集.....52

Contents

Abstract..... I

Contents..... IV

List of Figures..... VIII

List of Tables..... IX

List of Variables..... X

1 Introduction..... 1

1.1 Research Background..... 1

1.2 Research Motivation..... 2

1.3 State-of-art Research..... 3

1.4 Research Contents..... 7

1.5 Structure of This Thesis..... 7

2 Related Work..... 9

2.1 Introduction of Complex Network..... 9

2.2 Analysis of PageRank Algorithm and LeaderRank Algorithm..... 11

2.3 Multi-Attribute Decision-making..... 13

3 Evaluation of Users' Influences in Github Based on Multi-Attribute Decision-Making and Improved LeaderRank Algorithm..... 15

3.1 Research Motivation..... 15

3.2 The Framework of The Proposed Method..... 16

3.3 LeaderRank Algorithm Based on Multi-Attribute Decision-Making..... 17

3.4 Experiments..... 20

3.5 Summary..... 26

4 Evaluation of Users' Influences in Github Based on Users' Similarity and Improved Weight LeaderRank Algorithm..... 27

4.1 Research Motivation..... 27

4.2 Basic Concepts..... 28

4.3 The Framework of The Proposed Method..... 30

4.4 Weighted LeaderRank Algorithm Based on Users' Similarity..... 31

4.5 Experiments..... 36

4.6 Summary..... 41

5 Conclusions..... 42

5.1 Achievements of This Thesis..... 42

5.2 Futher Research Work..... 42

References..... 44

Author's Resume..... 50

Declaration of Thesis Originality..... 51

Thesis Data Collection..... 52

图清单

图序号	图名称	页码
图 1-1	Github 中的用户活动	4
Figure1-1	User’ activity in Github	4
图 2-1	四种不同类型的网络图	10
Figure 2-1	Four different types of network diagrams	10
图 3-1	所提方法的框架	16
Figure 3-1	The framework of the proposed method	16
图 3-2	MADM_LR 算法流程图	19
Figure 3-2	MADM_LR algorithm flow chat	19
图 3-3	Github 用户关注网络节点入度分布	21
Figure 3-3	Node penetration distribution in GitHub user follower network	21
图 3-4	不同算法得到的排名靠前节点的 SIR 传播能力	23
Figure 3-4	The SIR transmission capacity about the top nodes generated by different algorithms	23
图 3-5	干扰边对两个算法节点排名的影响	25
Figure 3-5	Interferon’s influence to two different algorithm nodes’ rank	25
图 3-6	假粉对两个算法用户排名的影响	25
Figure 3-6	Fake followers’ influence to two different algorithm users’ rank	25
图 4-1	带权 LeaderRank 算法定义的边权重示意图	29
Figure 4-1	Sketch map of the edge weight defined by weighted LeaderRank algorithm	29
图 4-2	所提方法的基本框架	31
Figure 4-2	The framework of the proposed method	31
图 4-3	简单网络示例图	32
Figure 4-3	Simple network sample diagram	32
图 4-4	不同算法得到的排名靠前节点的 SIR 传播能力	37
Figure 4-4	The SIR transmission capacity about the top nodes generated by different algorithms	37
图 4-5	干扰边对三种算法节点排名的影响	40
Figure 4-5	Interferon’s influence to three different algorithm nodes’ rank	40
图 4-6	假粉对两种算法用户排名的影响	40
Figure 4-6	Fake followers’ influence to two different algorithm users’ rank	40

表清单

表序号	表名称	页码
表 3-1	用户关注网络的基本参数	21
Table 3-1	The base parameter of the users' following network	21
表 3-2	不同方法的用户影响力前 10 名	22
Table 3-2	Top 10 user's influential rank generated by different method	22
表 4-1	不同方法的用户影响力前 10 名	36
Table 4-1	Top 10 user's influential rank generated by different method	36
表 4-2	不同算法节点排序与传播能力排序相关性	39
Table 4-2	Correlation between different algorithm nodes sorting method and trans mission ability	39

变量注释表

变量	注释
LR	LeaderRank 算法
WLR	带权 LeaderRank 算法
u	Github 中的用户
v	复杂网络的节点
a_{ij}	无权网络节点v_i与v_j的连边权值
g	背景节点
n	网络中的节点总个数
d	阻尼系数
PR	PageRank 算法
t_c	算法收敛所需时间
$LR_i(t)$	节点v_i在t时刻的 LeaderRank 值
w_{ij}	有权网络节点v_i指向v_j的连边权值
$a_1(u)$	用户的 star 属性值
$a_2(u)$	用户的 watch 属性值
$a_3(u)$	用户的 fork 属性值
p	用户创建的存储库之一
n_p	存储库 p 的合作者人数
$C(u)$	用户u的多属性决策指标值
I_r	网络干扰前后用户排序的变化
h_i	节点v_i的 H 指数
k_i^{in}	节点v_i的入度
k_i^{out}	节点v_i的出度
$k_{i,j}^{in}$	共同指向节点v_i和节点v_j的节点个数
$k_{i,j}^{out}$	节点v_i和节点v_j共同指向的节点个数
P_{ij}	节点v_i和节点v_j的相似度
$X(i)$	节点v_i的传播重要度
λ	节点传播步长
Y	不同算法排序结果与传播重要度排序结果的相关性系数
$R_i^{'}$	节点v_i在传播重要度算法中的排序结果
R_i	节点v_i在节点重要性排序算法中的排序结果

1 绪论

1 Introduction

1.1 研究背景（Research Background）

随着互联网和大数据的快速发展，诸多软件产品应运而生，用户的个性化和多样性需求日益增长。软件产品线的快速演化下，单一软件企业没有足够的资源解决某个软件产品开发过程中的所有问题。为保持竞争优势，越来越多的软件开发者通过软件外包、定制软件框架或集成开源代码等方式，将部分软件开发维护工作交给供应链上下游企业，外部开发者，开源社区甚至用户完成[1]。软件系统之间由于开发组件的共享、模块相互的依赖和开发者的交互而产生关联，形成多元复杂的协作关系网络[2]。软件工程学科借鉴生物学中“生态系统”的概念，引入所谓“软件生态系统”来刻画这种复杂系统的特性。

软件生态系统自提出以来，得到了产业界与学术界的高度关注。软件生态系统能够快速适应持续变化的业务需求，具有高度的伸缩性与可靠性，可以实现快速调整和演化[3]，这主要得益于该系统的技术平台——开源软件社区。开源软件社区是一类虚拟的群体开发社区，来自不同背景不同国家的开发者通过加入社区与其他开发者共同合作[4]。在开源平台上，个人或群体组织可以上传开源软件项目的源代码，平台上其他感兴趣的用户可以下载源码并参与到项目开发中。

在众多开源软件社区中，Github 是使用最广泛的社区之一[5,6,7]。作为一个源代码存储库，Github 吸引了众多开发者，他们合作开发软件项目并共享源代码，提高软件开发效率的同时降低了软件开发的成本。Github 的快速发展意味着其中有海量的有价值的数据可以挖掘，Github 逐渐成为了学者们研究的新领域。实际上，由于 Github 中不同的用户其结构地位以及活跃程度不同，他们所起到的作用和得到的反馈也不同[8]。其中部分用户通过直接或间接关系，能对社区中其他用户造成影响，并引导他们的行为，这些用户就是 Github 中的关键用户，即复杂网络中的关键节点。相比于一般用户，Github 上的关键用户能更大程度地影响整个社交网络的拓扑结构和功能[9]。对于商家来说，如果能够雇佣 Github 上的关键用户来做产品的推广，那么对新产品的宣传可以起到事半功倍的效果[10]。对于社区管理者来说，可以通过监管 Github 上的一些关键用户，对他们采取调控举措，来引导整个社区的良性发展与演化。解决这些问题的关键，是要找到 Github 中最有影响力的关键用户。

作为一种特殊的社交网络，Github 上的用户、组织和项目形成了不同规模种类的复杂网络。近些年来，复杂网络是学术界一个广泛的研究方向。将复杂网络

中衡量节点影响力的方法与 Github 中的实际问题相结合，可以满足我们很多实际需求[11]。尽管现在复杂网络的节点影响力度量已经得到了广泛研究，涌现出许多经典的算法，如何更加高效准确的识别节点影响力仍然是一个热门话题。而 Github 其自身存在多种复杂交互关系和不同的用户属性，传统的复杂网络节点影响力度量方法应用于 Github 社区中，会出现效率低下，排序结果不精准，鲁棒性不佳等问题。因此，如何根据 Github 的特性，结合复杂网络相关理论，全面评估 Github 中用户的影响力是一个值得深度研究的问题。

1.2 研究动机（Research Motivation）

软件生态系统是软件工程领域的一个功能单位和重要结构[12,13,14]，其组成元素数目庞大，且每个元素之间的关系随着时间不断变化，交错复杂。因此，软件生态系统本质上是一个动态变化的复杂网络。Github 作为其中的佼佼者，还有诸如收藏，评论，浏览等功能，这些交互模式使得 Github 不仅仅是一个简单的复杂网络，还是一个功能完善的社交网络平台。社交网络中对于关键用户的挖掘一直是研究重点，在前人的基础上已经有了许多成果。但是，对于 Github 中关键用户的挖掘却还处于初步研究阶段，且没有经典成型的方法。

复杂网络中，有很多经典成熟的节点重要性度量方法。但是若将这些方法直接应用在 Github 构建的复杂网络中，则存在结果不精准和效率低下等问题。因此，有必要结合 Github 相关功能和社会化特性，提出能全面有效识别关键用户的方法。

考虑 Github 上存在用户多种类型的社会活动，为全面衡量用户影响力，我们选择最具代表性的四种行为，并通过 GithubAPI 接口获取相关数据。基于此，我们在第三章给出相关行为属性值的定量表示，并设计了用户多属性决策指标。考虑复杂网络节点影响力排名算法 LeaderRank 存在不能考虑节点自身特征的问题，我们提出了考虑节点影响力初值的 LeaderRank 算法，并将用户多属性决策指标作为节点的影响力初值。由此，我们第 3 章所提方法既能全面考虑用户的多种社会活动属性，又能在算法中融入节点的自身特征。

考虑由 Github 中社交关系构建的网络，不仅能够简单的反映节点中的信息流向和节点自身的属性值，还可以反映节点之间的相互作用关系。事实证明，某一节点与其邻居节点越相似，节点就会更大程度的受到邻居节点的影响；且邻居节点越重要，那么它受到的影响程度就越大。基于此，我们给出了用户相似度和有向图中 H 指数的相关定义，构造了相应网络的边权。针对复杂网络经典算法没有考虑普通节点连边权重的问题，我们提出了第 4 章改进带权的 LeaderRank 算法。所提方法既能考虑用户间相似性，又能考虑网络连边的权值，在识别有影响力用户时，更为有效精准。

1.3 国内外研究现状（State-of-art Research）

与本课题有关的研究工作主要有 3 方面：软件生态系统与 Github、社交网络用户影响力度量以及 Github 用户影响力度量。其研究现状与发展动态分别如下：

1.3.1 软件生态系统与 Github

软件生态系统（SECO，Software Ecosystems）在大规模的软件开发环境中，不仅服务于开源软件平台的参与者，还服务于软件供应商、终端用户以及外部开发人员。苹果的 IOS 系统就是一个软件生态系统的典型例子。苹果提供了大量 IOS 系统的 APIs，借此，大量的苹果软件能够被产出。外部开发人员能够使用苹果提供的 APIs 开发在 IOS 系统上运行的软件，iPhone 和 iPad 就是这些软件的载体。其他相似的例子有 Andriod、Apache、Eclipse,、Linux 和 Facebook 等。

目前对软件生态系统的理论研究主要集中在软件生态系统生成与演化建模、度量指标与调控方法等方面。Kula 等[15]提出软件世界图作为软件生态系统及其库依赖型的演化模型，从而揭示开发人员依赖过时版本的风险。韩雨泓等[14]通过对软件生态系统中负熵机制的研究,提出软件以输出负熵为使命的观点,构建了基于负熵流的软件生态系统模型。Plakidas[16]基于 R 软件包的文档元数据，生成多种评价指标，对 R 生态系统的性能进行度量。刘亚珺[17]基于软件生态系统中开源代码的修改问题，定量探究了开源软件中多开发者交替修改源文件对软件质量的影响。李华莹等[18]提出基于统一访问引擎的软件资源访问技术，精确获取不同的软件资源，满足软件生态系统中不同组织的精细化管理需求。

Barbosa 等[19]提出了三个软件生态系统的维度：科技维度，商业维度和社会维度。科技维度围绕于开源软件或软件生态系统展开；商业维度围绕于商业模型、执照与专利以及从软件生态系统中获利等类似的维度展开；而社会维度则围绕于不同的软件生态系统是如何达成他们的目标而展开。

事实上，不是所有的软件生态系统是相似的。在另一份学术研究中，Manikas 和 Hansen[20]将软件生态系统分为私有软件生态系统与免费开源软件生态系统。在私有的软件生态系统中，源代码以及其他的元素是受到保护的，因为它们能够生产利润，这就代表新的开发者在进入这个软件生态系统时是需要经过认证的，久而久之，所有者左右了软件生态系统的发展，制定了访问这个平台的规则，并限制了成品的发布。在免费开源的软件生态系统中，源代码是免费且容易获取的，因为这些开发者不依赖于这些代码的金钱回报，因此新的开发者不需要通过繁琐的认证。IOS 系统是一个很好的私有软件生态系统，开发者需要支付苹果一笔费用才能在 App Store 中发布软件。而免费开源软件生态系统的例子有 Apache 和 Eclipse。在日常实践中，大部分的软件生态系统中都有两种生态系统的身影[14]。

在本课题中，我们所选取的数据都是基于开源的软件生态系统。

Github 作为优秀的开源软件社区，提供了丰富的社交网络功能，比如：关注用户、收藏项目和分叉项目。也就是说，Github 的用户可以关注社区中其他的用户，可以标记感兴趣的项目、分支其它项目，也可以向项目的拥有者发送拉请求。在 Github 中，由社区提供的社会关系我们提取如下 4 种，并采用相应复杂网络的边描述。

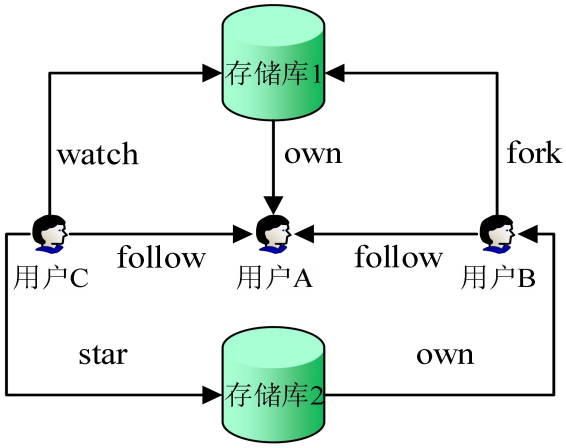


图 1-1 Github中的用户活动

Figure 1-1 User' activity in Github

（1）follow 关系。反映用户对用户的关注关系，相应的边是单向的，且 follow 边 A→B 代表用户 A 关注用户 B。

（2）star 关系。反映用户对项目的收藏关系，相应的边也是单向的，且 star 边 A→B 代表用户 A 收藏项目 B。

（3）fork 关系。反映用户与其他用户项目的复制关系。当某用户复制其他用户的项目时，在相应的复杂网络中就创建一个单向的分叉边。该用户可以对分叉项目贡献，并请求项目原始拥有者合并分叉的分支。fork 边 A→B 代表用户 A 复制其他用户的项目 B。

（4）watch 关系。反应用户对项目的浏览关系，相应的边也是单向的，且 watch 边 A→B 代表用户 A 浏览项目 B。

Github 的用户进行不同的行为活动，通过搜集行为活动数据，可以研究相应网络的特性和用户行为。Leibzon[21]从 Github 中筛选贡献者、评论者和复制者，作为相应网络的核心用户，通过合作分数，计算 Github 不同社交网络的合作强度。Biazzini 等[22]研究 fork 关系在多存储库项目的应用，通过可视化 fork 关系，展现存储库对不同项目贡献的分布程度。Pletea 等[23]研究用户的提交和拉请求行为，根据 Github 上与安全相关的讨论，评估用户的情感。Borges 等[24]通过 Github 上存储库的 star 数量，分析存储库受欢迎的影响因素，提出 4 种流行度增长模式。Ahmad[25]基于 Github 项目之间可能存在的 3 类社交关系（基于开发者的，基于

复刻的和基于源代码的社交关系），分别提出了抽取对应社交关系的 3 种算法，来深入理解 OSS 项目之间的相关性。李存燕等[26]根据开发人员 commit 的次数和提交代码的行数，评估项目参与者对项目的贡献度和开发人员的协作程度。

通过以上分析可以看出，GitHub 上用户的活动是多样的，从不同的角度定义用户的社会活动属性是需要解决的问题。

1.3.2 社交网络节点重要性度量

随着科技的不断进步，现代网络社会衍生了不同种类的社交平台，例如：微博、推特。这些社交平台上的用户进行不同的社会活动。通过这些活动，我们可以研究用户之间的社会关系、用户构建的组织、用户团体的协作机理以及挖掘影响力高的用户。与真实社会一样，社交平台上的用户活动也是多样的。选择合适的社会活动以满足上述需求，是需要解决的问题。乔秀全等[27]研究用户上下文关系，基于用户之间的熟悉程度和相似性，计算用户的信任度，提出社交网络用户间信任度的计算方法。王玥等[28]提出用户的社交圈检测方法，考虑到具有相似社交圈的用户更容易成为朋友，通过社交圈的相似性，为用户推荐新的朋友，并应用于 Facebook 等平台的朋友推荐。康书龙等[29]研究发表微博行为与用户之间的关系，提出一种结合两者的 BBR 算法，评价微博用户的影响力。张晨逸等[30]综合考虑微博联系人之间、文本之间的关联关系，提出基于 LDA 的微博生成模型，用于微博主题的挖掘。此外，申琳[31]考虑知乎网络用户获得点赞的数量和内容规模，提出一种改进带权的 LeaderRank 算法，用于识别关键用户。Zhu 等[32]考虑不同社交网络下的用户行为数据，提出用户主题偏好模型来模拟用户在多个社交网络间的交互过程，用于输出用户的全局话题偏好和局部话题偏好。Dhand 等[33]研究足球球员与普通男性的个人社交网络，基于网络组成与种族多样性，为接触性体育运动员提供更多的多维研究和治疗选择。

在社交网络中，节点影响力的表现形式多种多样。近年来，有关节点影响力评价的成果不断涌现，大体分为如下两类：一类基于复杂网络的拓扑结构评价用户的影响力，这些网络包括但不限于：合著者网络、引文网络、微博用户关注网络；另一类根据用户在社交网络的行为特征和交互信息的统计指标评价用户的影响力，常用的指标有粉丝数量、点赞数量、转发次数和用户的活跃程度。下面将分别阐述。

（1）基于网络结构的用户影响力度量

基于网络拓扑结构度量节点影响力最早被应用于社会学相关领域，将社会网络抽象为复杂网络来进行研究。关于复杂网络节点重要性的度量，已有方法大体分为如下 3 类[34,35]。第一类是社会网络分析法。这类方法将节点的重要性等价为显著性，常采用节点度数和介数等指标刻画，其中，节点度数为与该节点相连边

的数量；节点介数反映一个节点的影响力。这类方法通过统计网络的基本信息量化节点的重要性，例如：李鹏翔等[36]从网络的全局和局部方面，度量节点的重要性。除了上述指标之外，还有节点接近度、特征向量、网络直径等[37]。社会网络分析法从网络的结构特性出发，得到的节点重要性结果均有一定的局限性。第二类是系统科学分析法。这类方法将移除某节点对网络造成的破坏性等价为节点的重要性。移除节点之后，对网络造成的破坏是多种多样的，例如：Nardelli 等[38]基于网络的平均等效最短路径数，评价网络的抗毁度，用于度量节点的重要性；育萍等[39]通过生成树的数目评估节点的重要性；陈勇等[40]根据节点移除后网络连通节点数目的减少，量化节点的重要性。由于系统科学分析法没有完全体现网络拓扑结构的差异，因此，节点重要性评估的准确性尚需进一步提高。第三类是信息搜索分析法。在这类方法中，最流行的是 Larry Page 等提出的网页排名算法—PageRank，该算法将文献检索中的引用理论应用到 Web 中，通过引用某网页的链接数，反映该网页的重要性，成为搜索引擎 Google 的核心技术[41,42]。已有研究表明，PageRank 算法能够较准确地度量网络节点的重要性，且计算复杂度不高。

（2）基于行为特征的用户影响力度量

社交网络用户发布的信息通过交互行为进行传播。与此同时，用户的行为以日志的形式记录下来，通过这些记录，我们可以度量用户的影响力。廖志芳等[43]采用的用户行为重要性度量方法，基于用户行为的频次特征，用于挖掘关键的用户行为。Goyal 等[44]考虑 Flicker 网站上用户之间分享图片等行为，采用机器学习方法，计算用户行为传播的频率，用于度量用户的影响力。此外，Zaman 等[45]根据博客中用户之间的转发行为和博文内容，利用协同过滤算法，计算用户转发博文的影响力。卢冬冬等[46]基于开源软件项目中用户代码修订行为，构建开发者合作网络,分析节点流失对网络结构和功能稳定性的影响。周涛等[47]基于开源软件社区用户知识贡献行为，考察社会影响机制对用户行为的作用，提出了开源软件社区需重视建立用户的社会认同，促进其知识贡献行为的观点。

1.3.3 Github 用户影响力度量

针对 Github 开源软件社区，也有很多工作度量用户的影响力。Hu 等[5]通过 Github 用户的多个属性，从 follow、star、fork 和 activity 等 4 个方面，分析用户的影响力。Badashian 等[48]考虑与用户影响力相关的 3 个因素，即：用户的追随者数量、分叉者数量和观察者数量，分析这些因素与用户影响力的相关性。Hu 等[49]针对用户与存储库间的 star 关系，采用 HITS 算法，识别有影响力的存储库与用户。此外，申琳[31]基于用户拥有存储库的 star 数，利用改进带权的 LeaderRank 算法，评价用户的影响力。李变[50]根据用户的 star 和 fork 行为，将用户影响力

分为两部分考虑，提出基于用户属性矩阵和 HITS 算法的用户影响力评估算法。Blincoe 等[51]基于用户关注他人的动机和热门用户对其关注者的影响，采用混合研究的模式，对 github 上的 800 名用户进行调查，分析用户影响力。

不难看出，上述方法从不同的角度刻画用户的影响力且各有优势。但是，Github 中存在各种用户活动，基于单一的用户关注网络，利用复杂网络拓扑结构的评价指标，得到的结果难以全面反映用户的影响力；基于行为特征的用户影响力评估方法，多考虑用户之间的行为，忽略了网络的整体性能，也具有一定的局限性。

1.4 研究内容（Research Contents）

根据 1.2 节所述研究动机，本课题拟从下述两方面进行研究：

（1）基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价

本部分针对 Github 中的用户关注网络进行研究，以进行用户影响力评估。传统的复杂网络节点影响力度量方法主要通过网络拓扑结构来度量节点之间的相似性。而 Github 中有多种不同的社交关系，仅考虑网络拓扑结构是不充分的，还应该同时考虑用户自身特征，通过用户的多种社会行为来综合刻画节点的影响力。本课题结合网络拓扑结构和用户社会活动属性两部分，综合设计用户影响力评价方法。首先，基于用户社会多种活动属性构造多属性决策指标；然后，提出考虑节点初始影响力的 LeaderRank 算法；最后将多属性决策指标值作为网络中节点的初始值，得到用户排序结果。通过所提算法，不但可以更真实的刻画出 Github 中的用户行为，对网络中的干扰也具有较强的鲁棒性。

（2）基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评价

第一部分提出的算法，能够很好的识别 Github 中的关键用户。但是，该方法没有将节点间相互作用考虑在内。在 Github 中不同节点之间的相似度是不同的，某一节点对其他节点造成的影响也是不同的。已有的经典算法在定义节点间的这种相互作用，也即网络的边权时都很粗略，不能充分体现 Github 中用户之间的关联。本课题从网络的边权和带权 LeaderRank 算法两方面进行改进，一方面基于用户相似度和有向网络 H 指数，给出构造的有向用户关注网络及其边权；另一方面，提出改进带权的 LeaderRank 算法，应用于所构造的网络中，并进行节点影响力排序。本课题所提方法能够识别出更具影响力且传播能力更强的用户。

1.5 论文结构（Structure of This Thesis）

本论文共包括 5 章内容，具体安排如下：

第 1 章 绪论。介绍软件生态系统和 Github 的研究背景及意义，并对研究的

动机、国内外研究现状、论文的主要工作进行了详细说明。

第 2 章 相关工作。详细介绍复杂网络基本概念、PageRank 和 LeaderRank 算法以及多属性决策。

第 3 章 基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价。首先对本章研究动机和相关知识进行了说明介绍；其次，提出了用户社会活动属性的定量表示，并设计了多属性决策指标；然后介绍了所提改进 LeaderRank 算法；最后，利用 Github 上爬取到的真实数据进行实验，验证所提方法的可行性。

第 4 章 基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评价。首先，描述了提出上述方法的研究动机。接着，给出用户相似度和有向网络 H 指数定义，并详细介绍了网络边权的设置；然后，给出所提改进带权 LeaderRank 算法的详细介绍和具体步骤；最后，通过实验验证新方法的优越性。

第 5 章 结论。总结本文的研究工作的同时，针对研究中出现的问题给出接下来的研究方向。

2 相关工作

2 Related Work

2.1 复杂网络简介（Introduction of Complex Network）

2.1.1 复杂网络中的基本概念

21 世纪以来，复杂系统成为了学术界研究的热点之一，其研究对象为各类系统的特点和规律，并被应用于实际当中，比如生态系统[52]、交通运输系统[53]、电力系统[54]等。很多复杂系统都可以用网络的形式来表示，这时系统也就被抽象成了复杂网络。复杂网络不仅是一种数据的表现形式，也是一种科学的研究手段，并随着各种在线社交平台的蓬勃发展被广泛应用于社交网络。钱学森[55]对于复杂网络给出了一种严格的定义：具有自组织、自相似、吸引子、小世界、无标度中部分或全部性质的网络称之为复杂网络。其特点主要体现在：（1）小世界特性[56]。又被称为六度空间理论，即社交网络中任何两个成员之间所间隔的人不会超过六个。（2）无标度特性[57]。现实世界中的网络大部分都不是随机网络，少数节点往往拥有大量的连接，而大部分节点却很少，节点的度数分布符合幂律分布，这就是网络的无标度特性。（3）社区结构特性。人以类聚，物以群分。复杂网络中的节点往往也呈现出集群特性。复杂网络模型在很多领域都得到了广泛的应用。

复杂网络的主要研究方法都是基于图论的理论和方法开展的，并已经取得了可喜的成果。忽略网络的动力学特征，把网络中的个体视为节点，个体之间的关系看作是节点的连边，那么复杂网络实则为图。本课题所研究的内容虽基于复杂网络，但对于复杂网络的图表示中具体的组成部分有独特的含义。接下来对这些基本概念进行简要说明。

（1）节点。节点是组成网络最基本的元素，在 Github 中，我们可以把用户或项目等实体表示成节点。本课题中的节点是 Github 上的用户。

（2）边。节点之间相互连接的关系称为边，在 Github 中，我们把用户之间的某种或某几种交互关系定义为网络的边。通常在 Github 中，这个边可以根据研究者的研究内容来决定，可以是无向边也可以是有向边，可以是有权或无权的。本课题的两部分研究中，连边都是用户的关注关系，所以构造的网络是有向网络。

（3）网络的图表示。一般地，我们用图 $G=(V,E)$ 来表示网络，其中 V 是网络中的点集，E 是网络的边集。当边有权值时，用图 $G=(V,E,W)$ 表示，其中 W 是网络所有边的权重集合。

（4）度。节点的度为与其相连的边的个数，即与节点所连接的其他节点的个数。特别的，在有向图中，度被分为入度和出度，入度代表指向该节点的节点

个数，出度为该节点指向的其他节点的个数。

2.1.2 复杂网络的分类

根据连边的方向和权重，网络可以是无向/有向，无加权/有加权的，由此可以得到四种不同类型的网络，如图 2-2 所示。

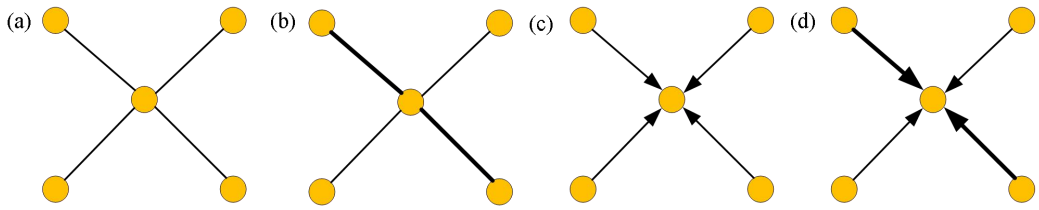


图 2-1 四种不同类型的网络图

Figure 2-1 Four different types of network diagrams

其中，箭头表示边的指向，边的粗细代表边的权重大小。图(a)表示无向无权网络，这种图只体现了节点之间存在关系，其他信息则模糊未知；图(b)表示无向有权网络，这类图不仅展现了节点之间存在关系，且关系的强弱有所不同；图(c)表示有向无权网络，这类图展现了节点之间关系的单向性且没有说明关系的强弱程度，本课题中第 3 章构造的网络属于这一类；图(d)表示有向有权网络，这类图不仅体现了节点之间存在关系的方向，还体现了关系强弱程度，本课题中第 4 章构造的网络属于这一类。特别的，对于存在双向边的网络，其可以理解为(c)(d)的特殊形式。

2.2.3 Github 的复杂网络构建

进行 Github 社区用户影响力评估的前提，在于构建 Github 中不同主体及其关系的复杂网络。由于 Github 中存在多种不同的社交关系，所以可以构建符合研究需求的不同网络。这些网络既可以是有向的，也可以是无向的；既可以是无权的，也可以是加权的。Hou 等[58]基于开发者对项目提交代码的贡献行为，将不同开发者对同一项目的代码提交看作两者间的合作关系，并构建了开发者合作网络。Yu 等[59]针对用户评论拉请求的评论次数和评论时间，构建开发者评论拉请求网络，并计算该网络的边权。Asri 等[60]针对开发者之间 4 种不同的交互活动，建立了 4 种用户合作网络，分别为：文件编辑网络、提交者网络、评论网络和评论者网络，来研究开发者之间的合作关系。

对于 Github 的某一复杂网络，采用合适的方法识别该网络中有影响力的节点，是非常关键的。关注是人与人之间常见的一种社交关系，它由单方面意愿产生，可以分为单方面关注、单方面被关注和互相关注三种情况。因此关注网络往往被构建为有向网络[61,62]。在关注网络中，不同的人赋予连边方向不同的意义。研究表示[63]，连边代表关注关系，带有方向的边从关注者指向被关注者。部分研

究刻画了关注行为带来的信息传播影响[64]，因此，用边来表示信息的传播方向，且从被关注者指向关注者。Github 中，用户之间存在 follow 关系，这种关系可以很好的说明某一用户对他所感兴趣用户的关注行为，且被关注的用户能够对这个用户产生一定影响。Hu 等[5]指出，用户的 follow 个数与用户影响力成正比。基于此，本课题将用户之间的关注关系，构建为用户关注网络，用于评估 Github 上用户的影响力。网络具体的构建方法在第 3 章和第 4 章分别给出。

2.2 PageRank 算法和 LeaderRank 算法的分析（Analysis of PageRank Algorithm and LeaderRank Algorithm）

本课题是对 LeaderRank 及其算法的改进，因此我们先简要介绍 LeaderRank 及其前身 PageRank 算法的相关思想。

2.2.1 PageRank 算法的相关分析

PageRank 算法，即网页排名，又称网页级别，是 Google 创始人早年构建搜索系统模型时提出的链接分析算法。通过将此算法与其他因素融合，使那些更具重要性的网页在搜索结果中的网站排名获得提升，从而提高搜索结果的相关性和质量。该算法不仅应用于其他搜索引擎，在学术界也成为了十分关注的计算模型，被广泛应用于节点重要性排序等相关领域。

对于某个复杂网络中的节点v_i来说，该算法的计算基于以下两个基本假设：

数量假设：在一个有向网络中，如果一个节点的入度越大，即指向它的节点个数越多，那么这个节点越重要。

质量假设：指向节点v_i的其他节点质量不同，质量高的节点可以向节点v_i传递更多的影响。因此指向节点v_i的其他节点越重要，则节点v_i越重要。

根据上面两个假设，PageRank 算法首先给予每个节点相同的初始影响力，然后迭代计算更新每个页面的 PageRank 得分，直到得分稳定为止。但是在复杂网络中，还存在一些出度为 0 的节点，当算法迭代到这些节点时就会滞留在该节点不能传递到其他节点。因此 PageRank 算法增加了阻尼系数d来对这种情况进行修正，该算法核心公式如下：

$$PR_i(t+1)=d\times\sum_{j=1}^{n}\frac{a_{ij}}{k_j^{out}}PR_j(t)+(1-d) \qquad (2\text{-}1)$$

式中，n为网络总节点数，$PR_i(t+1)$表示节点v_i在第$t+1$次迭代时的 PR 值，若v_j指向v_i则$a_{ji}=1$，否则$a_{ji}=0$，k_j^{out}代表节点v_j的出度。

由式（2-1）可知，在每次迭代过程中，节点都会把自身影响力均匀的传递给它指向的其他节点，且加入了阻尼系数d，保证了算法的强收敛。但是，该算

法存在不足之处。首先，阻尼系数d不可知，在不同网络中有不同的取值，在实际使用中需要通过多次调参取最优值；其次，算法在每次迭代中，节点都将其影响力均匀的分配出去，但在实际生活中，节点在网络中的重要性并不一致，不考虑节点的区别直接均匀分配存在较大限制；第三，PageRank 算法是建立在网络结构有上的算法。由相关实验可知，不论赋予节点的初始影响力如何，由于每次迭代时节点的影响力会被全部分散出去，经历多次传播后扩散到整个网络中，故收敛后的实验结果将仅与网络的拓扑结构有关，与节点初始影响力无关。

2.2.2 LeaderRank 算法的相关分析

在 PageRank 的基础上，Lv 等[65]对其进行了改进，提出了 LeaderRank 算法。LeaderRank 算法的核心思想是在网络图中增加一个背景节点，且与网络中每一个节点双向连接，使网络成为一个强连通网络，从而保证了算法的收敛性。而且 LeaderRank 取代了 PageRank 算法中的阻尼系数d，进而得到了一个无参数的算法。LeaderRank 算法的具体如下：

$$\begin{cases} LR_i(t+1)=\sum_{j=1}^{n+1}\frac{a_{ji}}{k_j^{out}}LR_j(t) \\ LR_j(0)=1 \\ LR_g(0)=0 \end{cases} \tag{2-2}$$

该公式的含义为在第$(t+1)$次迭代时，节点v_j按其出度值将自身的影响力均分给它所指向的全部节点，而节点v_i将其从所有指向它的节点处获得的分数相加，作为它此次迭代后的分数。 一直迭代直到所有节点的$LR_i(t)$值达到稳定时结束。一般认为两次迭代节点的分数之差小于10^{-6}时，或者达到设定的最大迭代次数时可停止迭代。用t_c代表算法达到收敛状态所需的迭代次数。最后将背景节点的值均分给网络的所有节点，最终每个节点的影响力分值为：

$$LR_i=LR_i(t_c)+\frac{LR_g(t_c)}{n} \tag{2-3}$$

然而，LeaderRank 算法存在的一个问题是，其终止值不受初始值的影响，它仅适合于不考虑节点本身属性值的网络评价。随着算法的迭代与收敛，节点本身的属性值就不会在算法中得到体现，而算法的最终结果仅仅是以网络结构作为参考。刘建国等[66]在研究了大量复杂网络节点排序算法后指出，不仅网络的拓扑结构可以决定节点的影响力大小，节点的自身特征同样会对节点重要性造成影响。同理，在 Github 中，不仅用户在关注关系网络中的拓扑属性能够衡量用户的影

响力，用户在 Github 中的各种社会活动也能一定程度上衡量用户影响力，因此用户的多种行为特征应该被并考虑进来作为重要指标之一。

2.3 多属性决策（Multi-Attribute Decision-making）

所谓多属性决策，是指决策者基于已有的信息，采用一定的方式，对有限的备选方案排序并择优，是决策科学的重要组成部分，在工程设计、经济、管理和军事等领域有广泛的应用。在多属性决策中，信息通常采用属性权重和属性值描述。迄今为止，已有许多确定属性权重的方法，主要分为以下三类：

（1）客观赋权法。该方法不含人的主观因素，利用客观信息（属性值）赋权。熵值法[67]、离差最大化法[68]、线性规划法[69]等都属于客观赋权法。

（2）主观赋权法。该方法由决策者根据经验和对不同属性的重视程度赋权。比较矩阵法[70]、判断矩阵法[71]等都属于主观赋权法。

（3）组合赋权法。为避免主观赋权法客观性较差和客观赋权法与实际情况相悖，组合赋权法综合主、客观赋权结果。

对于不同的决策者来说，用户的多种行为特征，其重要程度是不同的。本课题构造用户多属性决策指标时，考虑用户的不同行为。将这些社会行为抽象为属性并定义属性值后，采用 1-9 标度法，主观赋予权值来加权不同属性。基于此，在这里重点介绍 1-9 标度法。

1-9 标度法不仅可以对功能之间的重要性及重要性大小进行明确地评价与判断，而且还可以检验并保持评价（判断）过程的一致性。1-9 标度法就是用 1-9 之间的九个数（及其倒数）作为评价元素，对属性之间的相对重要性大小进行评分，形成判断矩阵，具体判断过程如下：

（1）评分过程。用同等重要、稍微重要、重要、很重要与极端重要五种判断表示功能之间的重要性区别，用 F_i 和 F_j 表示两种属性，并按照以下规则评分：当 F_i 与 F_j 同等重要时，$a_{ij}=1$；当 F_i 比 F_j 稍微重要一些时，$a_{ij}=3$；当 F_i 比 F_j 重要一些时，$a_{ij}=5$；当 F_i 比 F_j 重要得多时，$a_{ij}=7$；当 F_i 完全比 F_j 重要时，$a_{ij}=9$；如果属于两者之间，则用 2、4、6 来评分。

（2）权值的确定。采用求和法确定各属性的权值，如下所示：

$$\overline{w_i}=\sum_{j=1}^{n}a_{ij}\quad(i=1,2,...,n) \tag{2-4}$$

$$w_i=\frac{\overline{w_i}}{\sum_{i=1}^{n}\overline{w_i}} \tag{2-5}$$

式中，n为权值的个数，w_i为属性i归一化之后的权值。

（3）一致性检验。检验评分过程的一致性，可以转化为检验一致性比率是否小于 0.1。若小于则认为具有满意的一致性；否则需要对判断矩阵中的元素进行进一步调整。

3 基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价

3 Evaluation of Users' Influences in Github Based on Multi-Attribute Decision-Making and Improved LeaderRank Algorithm

3.1 研究动机（Research Motivation）

据报道，在 Github 中，已有数千万开发者和项目，其中，开发者的社会活动产生了大量数据，使得挖掘该社区的社会协作特性成为可能[72]。到目前为止，已有工作主要从 Github 中获取数据，并从不同层面分析该社区的特性，例如：社区结构划分、用户协作关系、编程语言和开发项目类型[73,74,75]。

分析用户在社交平台的影响力，对提供社区增值服务是非常重要的[76]。在市场营销中，人们对有影响力的个体识别非常感兴趣，例如：病毒式营销[77]。这是因为，有影响力的个体往往能够说服他（她）们的同行、目标人群和普通大众参与消费[78]。对于 Github 开源软件社区也是如此，且可以基于 Github 用户社会活动产生的数据实现。

关于 Github 用户的影响力分析，已有的工作不仅不多，还没有固定的分类与方法。Github 不仅可以建模为复杂网络，从网络拓扑结构的角度，采用分析节点重要性的方法分析开源软件社区用户的影响力。也可以将其理解为社会网络，从社会活动角度研究用户的社会关系和社会活动属性，分析用户的影响力。考虑到开源软件社区中，用户的社交网络具有许多潜在的工作和交流渠道。假设对于网络中固定的用户 A，其网络结构固定，但是某一段时间内，用户 A 贡献了更多的代码，或者其存储库被更多的复制分叉，那么其影响应该呈现增加趋势。但是若只考虑复杂网络的因素，网络无法刻画用户的个人属性，用户 A 的影响力变化并不会体现出来。因此，基于网络拓扑结构衡量用户的影响力不够全面。Badashian 等给出，在 Github 中，拥有众多粉丝的用户不一定产生重要的影响[48]。与贡献程度高的用户相比，关注关系连接多的用户更能影响其他用户[51]。故仅考虑用户个人属性而忽略社区对应的复杂网络诸如关注关系网络，合作者网络，也具有一定的局限性。在此基础上，还有一类工作将网络拓扑结构与社会活动有机结合。这类方法通常选取不同的用户社会活动属性，并将其应用于复杂网络节点排序算法中，从多角度分析开源软件社区用户的影响力。

鉴于此，本章提出基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力分析方法（MADM_LR）。相比已有方法，主要工作如下：（1）考虑用户的多种社会活动属性，从 watch、star 和 fork 3 个方面，设计反映用户影响力的多属性决策指标；（2）基于所提出的用户多属性指标和 LeaderRank 算法，提出一种考虑节点影响力初值的改进 LeaderRank 算法，用于节点影响力；（3）将所提方法应用于所构建的用户关注关系网络，识别了 Github 社区中具有影响力的用户。

本章余下部分组织如下：第 3.2 节，给出所提方法的基本框架；提出的用户影响力分析方法，在第 3.3 节详细阐述；第 3.4 节是所提算法和对比算法的实验与分析；最后，第 3.5 节是本章小结，并指出需要进一步研究的问题。

3.2 所提方法的基本框架（The Framework of The Proposed Method）

GitHub 社区的用户之间、用户与项目之间存在多种社会活动，这些社会活动从不同侧面反映用户的影响力。综合这些行为特征，全面衡量用户的影响力，是本节需要考虑的问题。此外，本章将 GitHub 的用户用复杂网络的节点描述，并应用度量复杂网络节点重要性的方法，分析 GitHub 用户的影响力。具体的讲，将用户的多种社会活动属性融合，设计反映用户影响力的多属性指标；根据用户关注网络的特性，基于多属性指标提出改进的 LeaderRank 算法，计算用户的影响力排名。所提方法的框架如图 3-1，主要包括两部分部分内容：基于用户社会活动属性的多属性决策指标（第 1 部分），基于网络拓扑属性的改进 LeaderRank 算法（第 2 部分）。

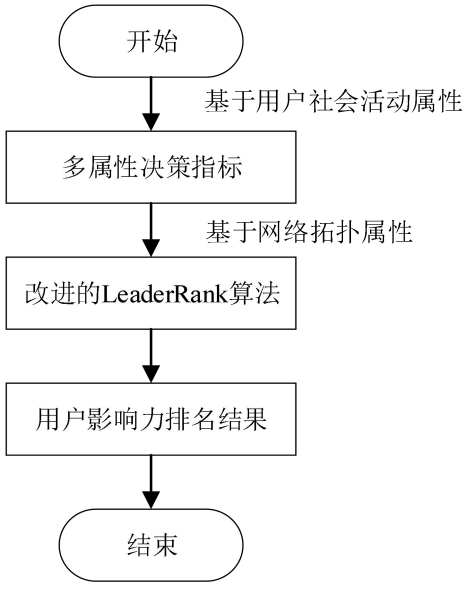


图 3-1 所提方法的框架

Figure 3-1 The framework of the proposed method

在第 1 部分，我们选取 Github 用户不同种类的活动，将其按照不同属性进行划分，通过将这些社会活动属性量化表示，我们构造了多属性决策指标，这是本章创新点之一。

在第 2 部分，我们根据用户间关注关系，构建了用户关注网络，并提出一种考虑节点初始影响力的 LeaderRank 算法，这是本章创新点之二。考虑 LeaderRank 算法不能得到结合节点自身特征的排序结果，我们进一步考虑节点的初始影响力。将第 1 部分所构造的用户多属性决策指标做为节点的自身特征，也即节点的初始影响力。之后设计考虑节点初始影响力的 LeaderRank 算法，让节点自身特征加入算法迭代，在考虑了用户的社会活动属性的同时也考虑了网络拓扑属性。

3.3 基于多属性决策的 LeaderRank 算法（LeaderRank Algorithm Based on Multi-Attribute Decision-Making）

3.3.1 用户多属性决策指标

前已提及，GitHub 用户的社会活动属性有多种，这些属性能够从不同角度反映用户的影响力。本章在设计模型时，爬取了用户的 4 种不同行为特征，从以下两方面进行考虑：（1）构造多属性决策指标，采用 watch、star 和 fork 刻画用户影响力；（2）改进 LeaderRank 算法并将其应用在构建用户关注关系网络上，也即 follow 关系网络，从而在算法层面体现用户影响力。鉴于此，本节先基于用户的 watch、star 和 fork 行为，设计度量用户影响力的多属性决策指标。

首先，考虑某用户创建存储库的 star 数、watch 数和 fork 数，以及合作者人数。其中，star 数表示收藏用户 u 创建存储库的其他用户个数；watch 数表示浏览过用户 u 创建存储库的其他用户个数；fork 数表示分支用户 u 创建存储库的其他用户个数。容易理解，某用户拥有的存储库被浏览、收藏和分支的次数越多，那么，该用户的影响力越大。因此，通过以上 3 类属性，能够从不同角度刻画用户的影响力。此外，对于 GitHub 社区的用户而言，当某存储库中合作者较少时，分给每个合作者的贡献或影响力就大；反之亦然。这样一来，在度量用户的影响力时，除了考虑这里提及的用户不同行为特征之外，还可考虑用户创建存储库中的合作者人数，具体如下。

首先，通过 Github API 接口，获取用户存储库的 star 数、watch 数、fork 数和存储库合作用户数；然后，参考 Börner 等[79]提出的 CS 指标，定义用户的不同属性值；最后定义多属性决策指标，应用于后续的算法。本章具体定义如下：

Github 社区的用户集合记为 $U=\{u_1,u_2,\cdots,u_n\}$，其中，n 为用户数量；用户的属性集合记为 $A=\{a_1,a_2,a_3\}$。对于用户 u，其属性值定义如下。

定义 1 用户 star 属性值

$$a_1(u)=\sum_{p\in P}\frac{S_p}{n_p} \tag{3-1}$$

定义 2 用户 watch 属性值

$$a_2(u)=\sum_{p\in P}\frac{W_p}{n_p} \tag{3-2}$$

定义 3 用户 fork 属性值

$$a_3(u)=\sum_{p\in P}\frac{F_p}{n_p} \tag{3-3}$$

式中，$a_1(u)$ 表示用户 u 的 star 属性值，P 为用户 u 所拥有的全部存储库。p 表示用户 u 创建的存储库之一，S_p 表示用户 u 所拥有存储库 p 的 star 数，n_p 表示存储库 p 的合作者人数。特别的，当存储库 p 为用户一人独有时，$n_p=1$。对于 watch 和 fork 属性值中符号的含义，可类似理解。由定义 1 可知，用户的属性值不仅与某用户创建存储库的个数与受欢迎程度相关，还与该用户拥有的每个存储库的合作者关联。用户拥有的存储库个数确定，且存储库被明确数量的用户浏览收藏（即影响力值大小固定）时，一个存储库的合作者个数越多，用户个人对存储库产生的贡献就越小，反之同理。将得到的 a_1,a_2,a_3 加权求和，得到如下的用户多属性决策指标。

定义 4 用户多属性决策指标

$$C(u)=s_1a_1(u)+s_2a_2(u)+s_3a_3(u) \tag{3-4}$$

式中，用户的多属性决策指标值记为 $C(u)$，s_1,s_2,s_3 分别是 a_1,a_2,a_3 属性值的权重。

3.3.2 所提算法的步骤

本节主要阐述提出的改进算法 MADM_LR，包含：改进动机、步骤和合理性说明。首先，基于 Github 上用户之间的 follow 关系，给出用户关注网络；然后，介绍已有的 LeaderRank 算法的不足、本章所提的改进方法与步骤；最后，指出本章方法的优越性。本章算法的流程图如下所示：

由图 3-2 中可以看出，本章采取了用户的 4 种社会活动属性，并将其分为两部分：其中 star，watch 和 fork 被构造为用户的多属性决策指标；而 follow 关系被构造为用户关注关系网络。接着，基于 LeaderRank 算法的缺陷，我们提出了考虑节点初始影响力的 LeaderRank 算法。所提算法被应用于用户关注关系网络中，考虑的节点影响力初值即为用户的多属性决策指标值。容易理解，网络中每

个节点都是社区中的用户，而用户存在自身价值与影响。信息在用户关注关系网络中流动，其传播范围和速度不仅与网络的连接结构有关，更与节点本身的价值

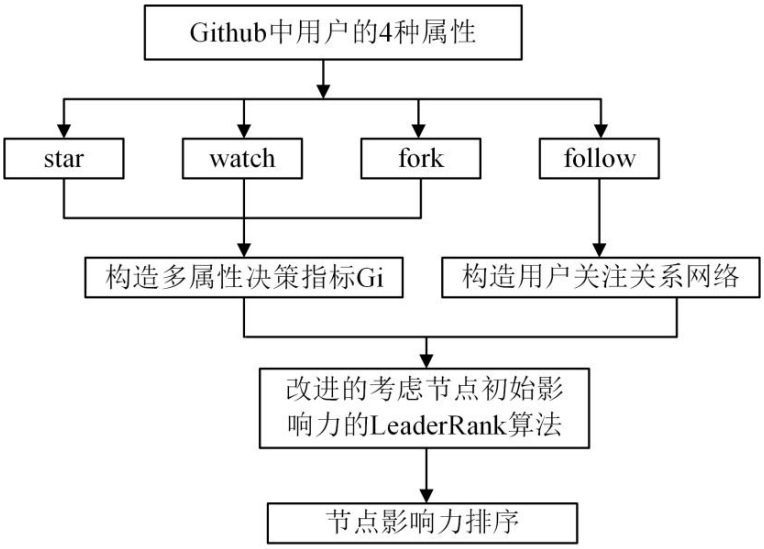


图 3-2 MADM_LR 算法流程图

Figure 3-2 MADM_LR algorithm flow chat

和属性有关，因此有必要将节点的初始影响力考虑在内。为使节点初值参与到算法迭代中，我们改进了基础的 LeaderRank 算法，令每个节点增加一条指向其自身的边，使迭代过程中均有部分 LR 值流向节点自身。到此，本章的 4 个属性被综合考虑进来，且将多属性决策指标应用于改进算法中，完成了模型的构建。最后即可得到节点排序，也即用户影响力排序结果。算法具体如下：

基于用户间 follow 关系构建的用户关注关系网络记为 $G=(V,E)$，其中，$V=\{v_1,v_2,\cdots,v_n\}$ 是网络中与用户集合 $U=\{u_1,u_2,\cdots,u_n\}$ 对应的节点集合，$E=\{e_{ij}\mid v_i,v_j\in V\}$ 是网络的边集，与用户之间的关注关系集合对应，当用户 u_i 关注用户 u_j 时，存在一条从 v_i 指向 v_j 的有向边。容易理解，一个节点被指向的边越多，那么，对应用户的关注数越多，该用户的影响力可能越大。因此，可以通过某用户的关注数刻画其影响力。

由式（2-2）可知，节点 LeaderRank 值的计算仅参考复杂网络的结构，计算结果与初始值无关。如果将 LeaderRank 算法直接应用于上述用户关注网络，得到的用户影响力排序将仅由 follow 关系决定。这样一来，通过单一属性衡量 Github 中用户影响力的方法存在很大的局限性，且与本章初衷相违背。在 3.3.1 节中提出了多属性决策指标，但是，在实际应用中，如果将多属性决策指标值作为 LeaderRank 算法的初始值参与运算迭代，那么，得到的结果也只与网络结构有关，而与算法初始值无关。因此，LeaderRank 算法仅适用于不考虑节点属性值的网络评价。对于 Github 而言，分析用户的影响力应该考虑用户的行为特征，

特别是，应该结合提出的多属性决策指标，这样融合了用户个人属性的结果将更加客观。

鉴于此，我们提出了考虑用户初始影响力的 LeaderRank 算法。首先将第 3.3.1 节中用户的多属性决策指标值，作为对应复杂网络节点的初始影响力。此外，为了将节点的初始影响力参与迭代，从每一节点引出一条指向自身的边。具体表示如下：

$$\begin{cases} LR_i(t+1) = \dfrac{LR_i(t)}{k_i^{out}+1} + \sum_{j=1}^{n+1} \dfrac{LR_j(t)}{k_j^{out}+1} \\ LR_i(0) = C_i \\ LR_g(0) = 0 \end{cases} \tag{3-5}$$

式中，前一项反映节点 v_i 本身的影响力；后一项反映节点 v_i 通过网络拓扑结构获得的影响力，$LR_i(0) = C_i$ 表示节点 v_i 在 $t = 0$ 时刻的值为 C_i。C_i 既是用户 u_i 的多属性决策指标值，也是节点 v_i 在 $t = 0$ 时刻的 LeaderRank 值，即节点的初始影响力。k_i^{out} 表示节点 v_i 的出度。对算法迭代结束之后得到的 LR_i 值排序，即可得到节点 v_i 的最终影响力排名。算法的具体步骤如下：

步骤 1：输入有向用户关注网络，节点初始值（即用户多属性决策指标值）；

步骤 2：对网络进行初始化，令 $t = 0$；

步骤 3：判断算法是否满足终止条件(即 LR 值不再变化，达到收敛)？如果是，转步骤 5；

步骤 4：计算式（3-5），得到临时的节点影响力值 $LR_i(t)$，转步骤 3；

步骤 5：输出节点影响力排序结果。

算法的每次迭代，都使节点的影响力均分给指向的所有节点。如果节点没有指向自身，那么，随着算法迭代，节点的影响力被全部分散出去，流向指向的其他节点。算法收敛时，节点的影响力将仅与网络拓扑结构有关。改进算法从每一节点引出一条指向自身的边，即每一次迭代后，节点都有一部分影响力留给自己。算法收敛后，得到的结果既考虑了网络的拓扑结构，又结合了节点的初始影响力。从 Github 角度，算法的设计不仅结合了多属性决策指标（即用户的影响力初值），也考虑了用户关注网络的结构。

3.4 实验（Experiments）

本节通过一系列实验，验证所提方法的有效性。实验平台的硬件配置如下：Inter Core i5-5200U CPU、8G 内存；软件配置为：Windows 8 操作系统、Python 3.6.0 和 PyCharm 编译器。我们采用的原始数据集为文献[5]中使用的公开数据集。利用

这些数据，通过不同的角度，对提出的基于改进 LeaderRank 算法的用户影响力计算方法进行验证，并与文献[31]提出的改进带权 LeaderRank 算法和传统的 LeaderRank 算法得到的用户影响力对比，从算法优越性与鲁棒性等方面，评价本章所提方法的性能。

3.4.1 实验数据

使用 Github API 接口收集的数据，分析用户的影响力。实验中所用到的数据都是 Github 上的真实数据。首先，将收集到的 205428 个用户初步筛选，仅保留存在 follow 关系的用户；本章使用的分析软件是 Gephi，这是一个复杂网络分析软件，通过将获取的数据文件以.csv 的形式导入到 Gephi，即可获得基于用户关注关系构建的网络。该网络的基本参数如表 3-1 所列。

表 3-1 用户关注网络的基本参数

Table 3-1 The base parameter of the users' following network

名称	节点数	边数	聚类系数	平均度
参数值	7934	72543	0.164	9.14

由表 3-1 可以看出，该网络的聚类系数为 0.164，远大于同等规模随机网络的平均聚类系数。但是，该网络用户之间的关系多通过用户及其存储库关联，因此，这种关系通常稍弱一些。

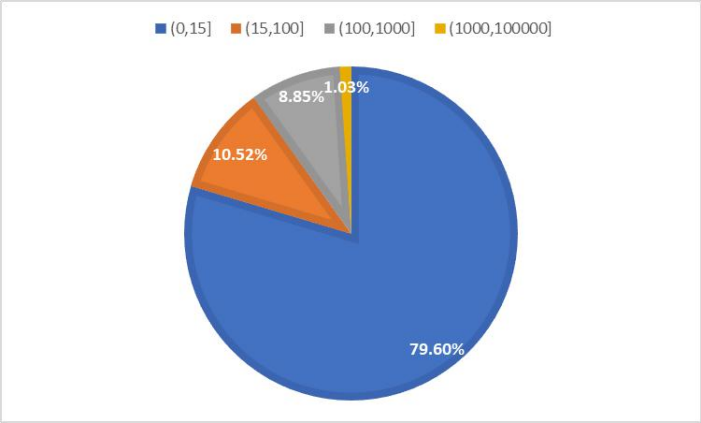


图 3-3 Github用户关注网络节点入度分布

Figure 3-3 Node penetration distribution in GitHub user follower network

在此基础上，给出节点的入度分布，即用户的关注数分布。由下图可知，在本节实验使用的数据中，共有 6315 个用户的粉丝数小于等于 15 个，占到了所有节点的 79.6%。这说明大部分用户的粉丝数都很少，主要以浏览他人的存储库，复制他人的代码为主，属于获得新知识与源码的普通用户群体。而有 82 个用户的粉丝数大于 1000 人，并且随着入度值的增加，这部分人越来越少，这些用户在 Github 社区中起着非常关键的作用，他们分享具有代表性与实用性的源码，供其余用户浏览使用。

3.4.2 不同方法的用户影响力排序

对于用户的不同属性，没有明确证据能证明和比较其对用户影响力作用大小。在实际应用中，对于用户影响力的分析多数情况取决于决策者的个人喜好。因此，本章采取主观赋权法，人为赋予权重。采用 1-9 标度法，确定 star 数、watch 数和 fork 数的权重：我们主观认为 star 属性值与 watch 属性值在衡量用户影响力中同样重要，而 fork 一个存储库代表着这个存储库得到了其他用户的认可且被复制使用，这更能反映了一个用户的影响力。因此得到$[s_1, s_2, s_3] = [0.3, 0.3, 0.4]$。将与单一的用户属性指标、传统的 LeaderRank 算法应用于该网络，得到用户影响力排名前 10 的结果，如表 3-2 所列。

表 3-2 不同方法的用户影响力前 10 排名

Table 3-2 Top 10 user's influential rank generated by different method

排名	MADM_LR	LeaderRank	Star	Watch	Fork
1	torvalds	torvalds	nnnick	FreeCodeCamp	MyCoolTest
2	JakeWharton	JakeWharton	zxing	vhf	LarryMad
3	Tj	michaelliao	arasatasaygin	robbyrussell	octocat
4	douglascrockfor	Tj	allmobilize	sindresorhus	repeng
5	visionmedia	mojombo	torvalds	getify	nnnick
6	openssl	paulirish	nickbutcher	docker	zxing
7	nicklockwood	githubpy	jmechner	daneden	Venomous0x
8	docker	defunkt	ViccAlexander	mbostock	Learnstreet-dev
9	zxing	addyosmani	openssl	Apple	Tower-KevinLi
10	mojombo	jeresig	valums	torvalds	torvalds

表中具体所列分别为不同指标下的前 10 名用户。由表 3-2 可知，（1）有 5 个用户同时出现在 MADM_LR 和 LeaderRank 算法的前 10 名，而排序却不尽相同。（2）用户 mojomobo 在 MADM_LR 中的排序低于 LeaderRank 算法，用户 Tj 的排序有所前进，这是因为，用户 Tj 的 3 个属性值都较为突出，而 mojombo 的 watch 和 fork 属性值较为靠后。这说明，用户属性值影响排序算法，从而影响最终的排序结果。（3）对于单独出现在 LeaderRank 算法中的 addyosmani 等用户，3 种属性值的排序均较低，这样的用户只是因为跟随者多而排名靠前，对于社交网络的其他用户来说，的确是不合理的。（4）本章算法排名第一的用户是 linux 操作系统的创建者 Linus Torvalds，这是 Github 中最受欢迎的用户之一。

不同决策者对用户的影响力有不同的评价标准，表 3-2 是对算法合理性的定性分析。通过以上分析可以看出，MADM_LR 能够更好地结合用户的社会活动与网络拓扑结构，且能充分利用用户的多种属性，因而得到的用户影响力更加精准。算法与其他比较算法之间更为精确的对比将在 3.5.3 节给出。

3.4.3 基于 SIR 模型的算法比较

SIR 模型[80]是一种常用的评价算法分析网络节点影响力的模型，采用 SIR 模型得到的用户影响力传播曲线可以充分体现算法的优越性。在实验过程中，首先说明使用 SIR 模型的步骤及其参数设置；然后，将 MADM_LR 与两种对比算法应用于 SIR 模型，评价 MADM_LR 的性能。

该模型研究易感者、感染者和免疫者之间的关系，在病毒开始的时候，所有人都是易感者，即所有人都有可能中病毒；一部分人中病毒后，变成了感染者；感染者接受治疗，变成了免疫者。该模型有如下 3 个假设条件：（1）在一段时间内，总人数不变；（2）从 S 到 I 的变化速度 α、从 I 到 R 的变化速度 β 保持不变；（3）移除者不再被感染。

在 SIR 实验中，首先，选取 3 种算法排名靠前的不同用户，作为初始感染态 I 的成员；然后，观察这些感染者的传播能力。具体的讲，分别选取排名前 20、前 50 和前 100 的用户，作为感染者。对于每一次传播，感染者以概率 α 感染易感态成员，同时，感染者以概率 β 转变为免疫者。在实验中，参数设置如下[80]：$\alpha=0.5$，$\beta=0.109$。观察并统计每次传播后被感染的节点个数，直到模型收敛。

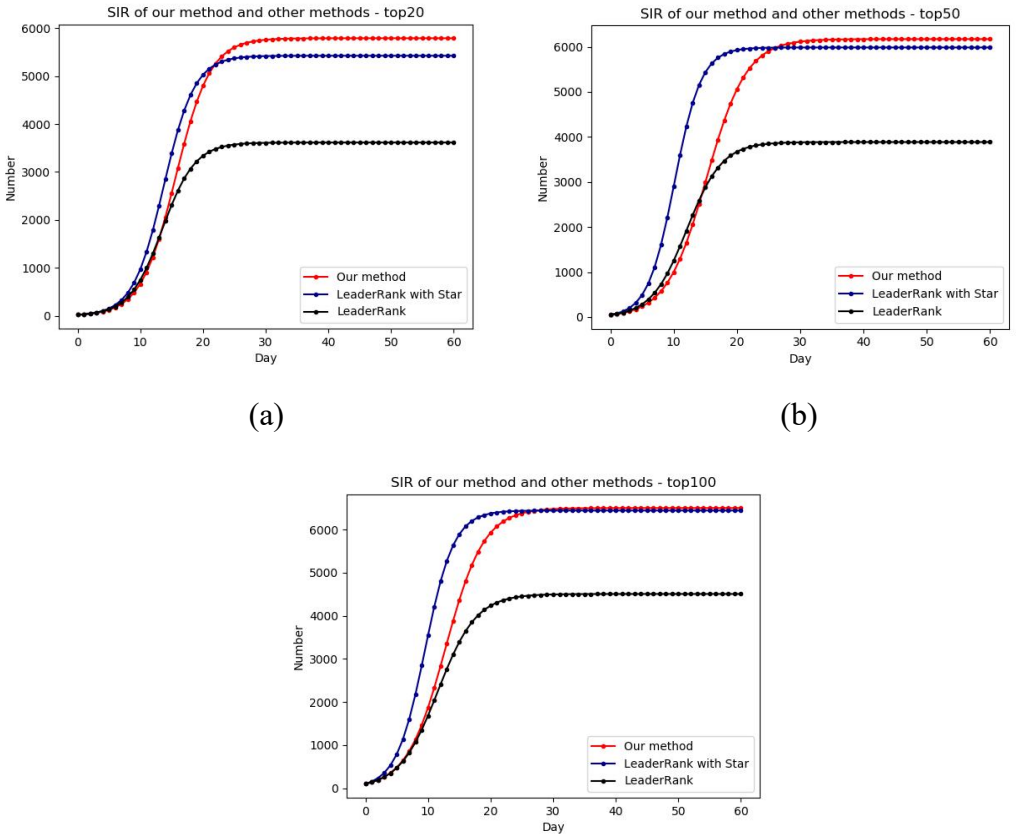


图 3-4 不同算法得到的排名靠前节点的SIR传播能力

Figure 3-4 The SIR transmission capacity about the top nodes generated by different algorithms

本章采用的比较算法为申琳等[31]提出的改进带权 LeaderRank 算法。此算法的迭代主体包含网络拓扑属性和用户个人属性等两个部分，其中网络拓扑属性部分为一般 LeaderRank 算法，只考虑边权为 1 或 0；而用户个人属性部分采用带权的 LeaderRank 算法，网络的边权为用户的 star 属性；最后对这两部分加权。本章算法分析用户影响力时，虽然同样分为网络拓扑属性和用户个人属性两部分，但对算法的改进之处为节点的初值，并没有对边权做出改进；且本章采用 star、watch 和 fork 这 3 个属性进行多属性决策，比对比算法的属性更为全面；因此，这里选取文献[31]所提算法和传统的 LeaderRank 算法作为对比算法。得到的实验结果如图 3-4 所示。

由图 3-4 可知，经历 20-40 次传播后，SIR 模型收敛。尽管 MADM_LR 的收敛速度较慢，但是，传播节点数最高。这说明，MADM_LR 得到的用户具有更高的影响力。原因在于，本章所提多属性决策与改进算法是合理的，比其它算法更能挖掘更有影响力、传播能力更强的用户。

3.4.4 算法的鲁棒性比较

为了准确识别关键用户，一个好的排序算法应该对网络中的噪音数据更具鲁棒性[69]。因为实际的开源软件社区中，用户之间的关系并不能仅通过数据反映，存在很多非真实或遗漏的边。此外，还存在很多“僵尸”用户与假粉，这些用户用于提升个人的粉丝量，但不产生任何实际的影响。因此，在考察本章算法时，通过以上两种不同的干扰来验证算法的鲁棒性与抗干扰能力。

（1）施加网络干扰边

由于用户之间的关系存在不确定性和主观性，导致网络中以有的关注关系（即干扰边）可能实际上不存在，或实际存在的社交关系（即非干扰边）并未在社交网络中出现。虽然已有工作能在一定程度预测缺失的连接关系，但仍需要算法在面对存在缺失或冗余边的网络时，具有一定的鲁棒性。

随机地从网络的节点间增加或移除一些连接关系，作为关注者与被关注者之间增加或缺失的关注关系，比较 MADM_LR 与 LeaderRank 算法的用户排名的变化。为此，定义 I_r 为网络干扰前后用户排序的变化

$$I_r = \sum_{i=1}^{n} \left| R_i^{'} - R_i \right| \tag{3-6}$$

式中，n 为用户总数，R_i 和 $R_i^{'}$ 分别为网络干扰前后用户 u_i 的排名。分别随机增加或移走网络的{50，100，150，200，250，300，350，400，450，500，600，700，800，900，1000，1100，1200}条边，施加干扰后 I_r 的值如图 3-5 所示。

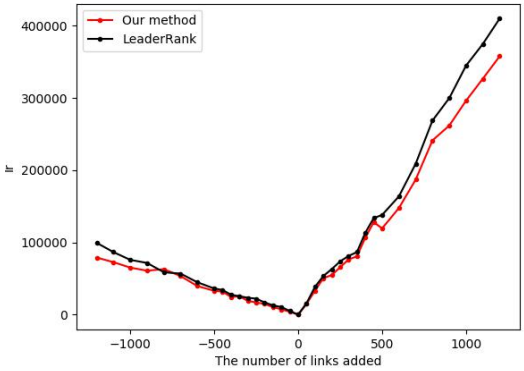


图 3-5 干扰边对两个算法节点排名的影响

Figure 3-5Interferon's influence to two different algorithm nodes' rank

由图 3-5 可知，随着扰动的增加，I_r 值呈现增加的趋势。这说明，用户排名的变化越来越大，且加边的变化量大于减边的变化量。但是，无论加边还是减边，MADM_LR 的 I_r 值在多数情况下均优于 LeaderRank 算法。这说明，MADM_LR 的鲁棒性比 LeaderRank 算法更好。

（2）添加社交网络假粉

从排名靠前的前 100 个用户中，选择排名为{10，20，30，40，50，60，65，70，75，80，85，90，95，100}这 14 个用户。针对每一用户，分别增加 $v=10,20,50$ 个关注者，同时，将这些关注者的 star、watch 和 fork 属性值设为 0，表示这些关注着是假粉且不对网络产生任何影响。对这 14 个用户，分别施加不同程度的干扰，不同算法得到的这 14 个用户新的排序结果如图 3-6 所示，其中，左图是 MADM_LR 干扰下的用户排名，右图是 LeaderRank 算法干扰下的用户排名。

由图 3-6 可知，随着假粉的增多，两种算法得到的排序差异均增加，但是，与基线的趋势总体上接近。此外，假粉增加，MADM_LR 和 LeaderRank 均有波动，但是，MADM_LR 的波动小于对比算法，也即 MADM_LR 的抗干扰能力优于 LeaderRank。

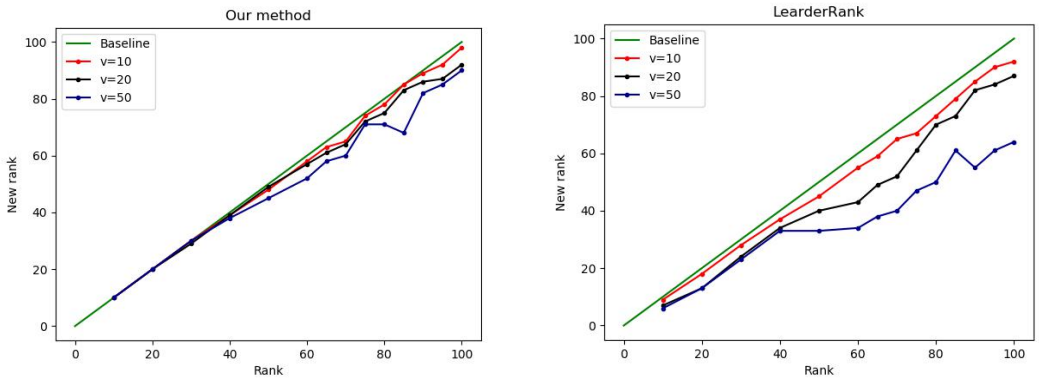


图 3-6 假粉对两个算法用户排名的影响

Figure 3-6 Fake followers' influence to two different algorithm users' rank

3.5 本章小结（Summary）

随着网络的快速发展，Github 这种分布式的版本控制系统应运而生。Github 用户不仅可以将自己的代码分享到 Github 平台上，也可以从 Github 上获取他人分享的开源项目，以及对不同的开源项目进行社会活动诸如点赞，浏览，评论，分叉等。同其他社交网络类似，Github 上的用户有着多种不同的属性，且他们在 Github 中的重要程度与影响力都是不一样的。衡量 Github 上用户的影响力，可以帮助决策者更好地找到流行的开源项目，找到最受欢迎的项目贡献者，对他们的项目进行浏览，代码提交和评论等操作，从而引导整个社区朝着良性循环的方向发展。通过观察不同社区或项目中，用户影响力的发展趋势，我们可以对项目的演化进行预测，并对其采取及时的举措。

本章针对实际的开源软件社区 Github，首先，将用户的社会活动划分为不同的属性，得到衡量用户社会活动的多属性决策指标；然后，基于用户关注关系构建的复杂网络，提出改进算法 MADM_LR。通过与其它算法对比，实验结果表明，MADM_LR 可以得到一组优越的用户影响力排序结果。将 3 种不同算法应用于 SIR 模型，结果表明 MADM_LR 不仅能够得到传播能力更好的节点，识别更具有影响力的用户，在抗干扰方面也具有显著的优越性。

但是，MADM_LR 的收敛速度较慢，且考虑的用户行为特征不够全面。在 Github 中，还有用户提交、用户参与讨论与项目提问等行为，而这些行为都可以作为用户的不同属性，参与评价用户的影响力。此外，在对算法进行改进时，本章只结合了网络节点初始值对 LeaderRank 算法做出改进，并没有考虑进一步构造网络边权。带边权的 LeaderRank 算法是一种可行的改进方向。如何构造用户关注网络的边权，以及带边权的算法应该如何设计其迭代主体，具体内容我们在下一章介绍。

4 基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评价

4 Evaluation of Users' Influences in Github Based on Users' Similarity and Improved Weight LeaderRank Algorithm

4.1 研究动机（Research Motivation）

上一章提出的多属性决策指标，和考虑节点影响力初值的 LeaderRank 算法，被应用在了用户关注关系网络中。我们在构建网络时，考虑了节点之间的有向边。对于用户 u_i 和用户 u_j，我们认为有节点 v_i 指向节点 v_j，且其边权 a_{ij} 为 1 或 0。算法虽然是对 LeaderRank 算法的改进，但本质上解决的仍然是无权有向网络上关键节点的挖掘问题。

LeaderRank 算法和 MADM_LR 在迭代的过程中，每个节点的影响力都均匀的分配给与其所相连的其他节点，因此所提方法存在着许多不足。首先，在构造网络时认为其边权为 1 这是一种理想化的情况，因为网络的边权实际代表着用户之间的关注关系，而用户对其关注的所有用户，其关注程度是不一样的。用户 u_i 关注用户 u_j 的程度越高，说明用户 u_j 的影响力越大，对用户 u_i 造成的影响越高；而用户 u_j 的影响力越小，用户 u_i 对其的关注相应也较小。其次，算法迭代过程中节点分配影响力的方式为均匀分配，而实际上我们应该给予影响力高的节点更高的影响力分值，而不太重要的节点分配较小的影响力分值。此外，虽然我们赋予了节点一定的影响力初值，但是这部分影响力初值是由用户的社会属性决定的，对于网络拓扑属性我们完全依赖于用户关注关系网络和所提算法。因此我们应该增加对网络拓扑属性的考量，这样更符合马太定律。

在 Github 和其他社交网络中，存在着这样的情况：一个用户，他的粉丝数量越多，且他粉丝的粉丝数量越多时，用户就越受欢迎，用户的影响力就越高。与此同时，在现实生活中，人们往往更愿意相信那些和他们更为相似的人。对于 Github 中的两个存在关注关系的用户，如果他们共同关注的用户越多，那么他们的兴趣相似度越高；如果他们共同的粉丝越多，那么他们的研究领域越为相似。可以看出，相似的用户之间存在的影响大于其他一般用户。第 3 章研究工作是基于用户的社会活动属性衡量用户影响力。虽然我们也构建了用户关注关系网络，但网络的特性并没有得到充分的挖掘，用户间存在的关系没有被充分定量表示。而且，MADM_LR 存在一定的局限性，本章将对此加以说明并进一步完善。

鉴于此，考虑网络中节点的边权，本章基于用户相似度和节点 H 指数，提出了一种带权的 LeaderRank 算法。首先，给出带权 LeaderRank 算法基本概念及其缺陷，并介绍了节点 H 指数有关概念。随后，给出一个简单案例说明用户相似度的构造方法，引出一种用户关注网络中用户相似度的计算方法，并说明不同的用户间存在不同相似度和关系强弱。为进一步体现网络中不同节点区别及节点间相互作用，提出了本章方法的思想。最后，将所提算法应用于 Github 上爬取的数据集，从不同的角度来验证所提指标和方法的有效性。

本章后续内容具体安排如下：首先，第 4.2 节给出带权 LeaderRank 算法相关思想及其不足，并给出 H 指数的相关概念；第 4.3 节给出本章思想的流程图，阐述算法具体框架；用户相似度和有向网络 H 指数相关定义、有权关注网络的构造和算法的具体步骤将在第 4.4 节给出；最后，第 4.5 节通过实验验证所提方法的有效性。

4.2 基本概念（Basic Concepts）

4.2.1 带权 LeaderRank 算法

标准的 LeaderRank 算法一般应用于无权有向网络中，该算法在 PageRank 算法的基础上加入了一个背景节点，保证了网络的强连通性，加速了算法的收敛。其中，背景节点所得的影响力值被均匀的分散到其他节点。Li 等[81]则认为需要对网络中不同的节点做一个区分。为此，他们增强了网络中入度数大的节点的重要性。在一般 LeaderRank 算法的基础上，他们有如下定义：

$$\begin{cases} LR_i(t+1)=\sum_{j=1}^{n+1}\dfrac{w_{ji}}{\sum_{l=1}^{n+1}w_{jl}}LR_j(t) \\ LR_i(0)=1 \\ LR_g(0)=0 \end{cases} \tag{4-1}$$

$$w_{ji}=\begin{cases} k_i^{in}, v_j \to v_i, v_i = g \\ 1, v_j \to v_i, v_j \neq g \\ 0, v_j \not\to v_i \end{cases} \tag{4-2}$$

其中，w_{ji} 是节点 v_j 与节点 v_i 之间单向边的权值，$\sum_{j=1}^{n+1}\dfrac{w_{ji}}{\sum_{l=1}^{n+1}w_{jl}}$ 表示这条单向边的权值，占 v_j 指向的所有点集的边权总值的比例，该值反映了从节点 v_j 与节点 v_i 之间单向边的重要程度。节点 g 为背景节点。$v_j \to v_i$ 表示节点 v_j 与节点 v_i 之间存

在单向连边，且由v_j指向v_i。$v_i \nrightarrow v_i$表示节点v_j与节点v_i之间不存在连边。k_i^{in}是节点v_i的入度。

由上述公式可知，由于任一节点与背景节点之间的连边权重为节点的入度，因此在算法迭代过程中，背景节点的影响力以入度大小被网络中其他节点分摊。节点的入度越大，那么节点从背景节点那里获取的 LR 值就越大，相应的，节点的影响力就越高。图 4-1 是对带权 LeaderRank 算法的一个简单示意图：

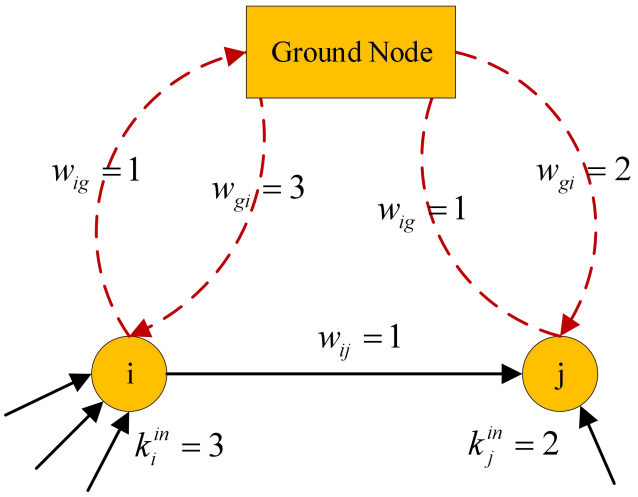


图 4-1 带权 LeaderRank 算法定义的边权重示意图

Figure 4-1 Sketch map of the edge weight defined by weighted LeaderRank algorithm

带权的 LeaderRank 算法强调了节点入度对节点影响力的正面作用。在 Github 中同理，若用户受到的关注数越多，那么用户获得的影响力将会更多。Li 等[81]将算法应用于不同数据集，发现相比于一般 LeaderRank 算法，带权 LeaderRank 算法在准确性和鲁棒性方面均更为突出。

4.2.2 带权 LeaderRank 算法的不足

不论是 LeaderRank 算法还是带权 LeaderRank 算法，它们都仅只考虑了背景节点对其他节点的影响，没有考虑网络中节点之间的相互影响。对于节点 LR 值平均分配的问题，上述两种算法都不能解决。在 Github 等相关实际应用中，用户受到其他用户的影响程度一定是不同的，用户会更倾向于相信那些影响力高的，或者与他们更相似的用户。因此，映射到复杂网络上，节点之间也应该存在倾向性，也即所提出的算法应该能够节点间不同的影响情况。

其次，带权 LeaderRank 算法虽然做出了改进，但是它的改进是将节点的影响力与节点的入度正相关。而这些改进都是基于网络拓扑属性。在实际的社交网络中，若只考虑用户的粉丝数量（及节点的入度），那么只要是粉丝数量高的用户，其影响力都会名列前茅。然而存在一些购买假粉的用户，他们只有粉丝数量

很高，但是其余的社交活动都很匮乏。若把这些人推荐给其他用户，会阻碍社交网络良性发展。故只基于网络拓扑属性的算法明显不适用于 Github 这样具有社交网络特性的开源社区。在 Github 中，用户的各种社会活动属性同样可以作为评价用户影响力大小的重要指标。在 Github 用户影响力评价中，我们要充分结合用户的多种行为特征，才能客观全面的衡量用户影响力。

4.2.3 H 指数中心性

H 指数中心性是利用 H 指数相关概念，通过考虑网络中节点的邻居节点的度，来定义节点的中心性。节点 v_i 的 H 指数中心性为：

$$h_i = H(k_{j_1}, k_{j_2}, k_{j_3}, ..., k_{j_{k_i}}) \quad (4\text{-}3)$$

其中，$(j_1, j_2, j_3, ..., j_{k_i})$ 为节点 v_i 的邻居节点，k_i 是节点 v_i 的度。函数返回值 h_i，使得 $k_{j_1}, k_{j_2}, k_{j_3}, ..., k_{j_{k_i}}$ 中有 h_i 个值大于等于 h_i。

4.3 所提方法的基本框架（The Framework of The Proposed Method）

本节给出了基于用户相似度的改进带权 LeaderRank 算法，并将其应用于 Github 用户影响力评价。图 4-2 是算法的基本框架，主要包括三部分内容：基于用户社会活动属性的多属性决策指标（第 1 部分）、基于网络拓扑模型的边权构造。（第 2 部分）和基于用户相似度的带权 LeaderRank 算法（第 3 部分）。

考虑 Github 具有社交网络的特性，同本文第三章一样，我们将第 1 部分的内容构造为用户的多属性决策指标，这个指标由用户的 star、fork 和 watch 属性构成，从不同角度反映了用户在 Github 中受欢迎的程度。

第 2 部分的内容为本章主要创新点之一。考虑在一般的 LeaderRank 算法和带权 LeaderRank 算法都不能充分利用网络拓扑属性，算法迭代主体均不足以体现节点间相互作用的关系，我们提出了改进带权的 LeaderRank 算法。为体现节点之间的相互作用，我们定义了用户相似度，也即节点相似度。我们仍然以第三章所构造的用户关注关系网络为基础，考虑存在关注关系的节点之间的相似性。用户相似度由两部分构成，一是考虑两个节点共同指向的其他节点越多，节点越相似；二是考虑共同指向两个节点的其他节点越多，节点也越相似。可以看出，用户相似度是基于网络拓扑结构的指标，进一步挖掘了网络中节点之间的内在联系。为进一步体现单个节点在网络中的流行程度，我们提出了有向网络 H 指数这一指标，来体现有向网络中单个节点的重要程度。实际生活中，人们会更偏向于和他们相似且受欢迎的人[82]。因此，网络拓扑模型的边权，由节点相似度和节点 H 指数两方面来进行构造。这个边权既包含了节点之间的内在联系，也体现

了节点自身的重要性，能够让关键节点的影响力更为突出。

第 3 部分是本章所提算法。算法是在以上两部分内容的基础上，提出的改进带权的 LeaderRank 算法。

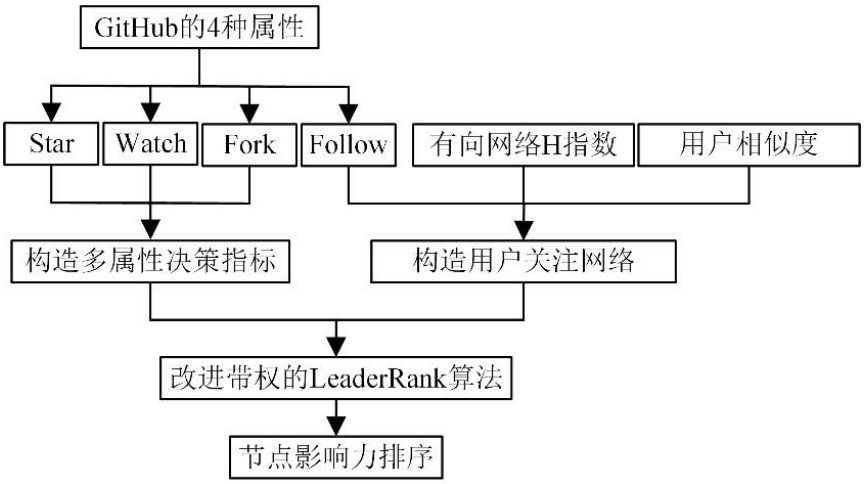


图 4-2 所提算法的基本框架

Figure 4-2 The framework of the proposed method

4.4 基于用户相似度的带权 LeaderRank 算法（Weighted LeaderRank Algorithm Based on Users' Similarity）

4.4.1 用户相似度

在一般 LeaderRank 算法和带权 LeaderRank 算法中，网络中的节点边权为 1 或 0，且算法迭代时认为节点将其 LR 值均分给所有指向的节点。但是实际上，人们更愿意相信那些与他们相似的人，并且更容易受到他们的影响，Github 中同理。在用户关注关系网络中，网络的连边代表着用户的关注关系。可以得知，对于存在连边关系的两个用户，所关注的相同的人越多，说明两个用户所感兴趣的方面越一致，这两个用户就越相似；若关注两个用户的相同用户越多，则说明两个用户涉及的领域与研究的方向越一致，这两个用户也越相似。因此，对于网络中有连接关系的一对节点，若指向两个节点的其他节点数越多，两个节点越相似；两个节点共同指向的其他节点数越多，两个节点越相似。在本章，用户相似度即为节点相似度。

对于一个有向简单网络如图 4-3 左图所示，在节点 4 不存在时，节点 1、2、3 之间互相作用且边权均为 1，他们三个点的影响力是相同的。加入节点 4 后，由图可知节点 4 指向节点 1 和节点 2，其影响力被分摊给节点 1 和节点 2，导致节点 1、2 的影响力大于节点 3，节点 3 的影响力被间接削弱。且可以看出，节点 1、2 之间的相似性在节点 4 添加后大于节点对 2、3 和节点对 1、3 的相似性。故节点间相似性与其共同指向的节点数有关，即两个节点受其他节点共同指向的个数越多，其相似度越高。由图 4-3 右图可知，节点 1、2 同时指向节点 4，此时

节点 4 的影响力较之前增加，而节点 4 影响力的提高同时也提高了节点 1 和节点 2 的影响力，右图中节点 1、2 间相似度与节点影响力均高于左图。由此得知，两个节点共同关注的节点个数越多，其相似度越高。

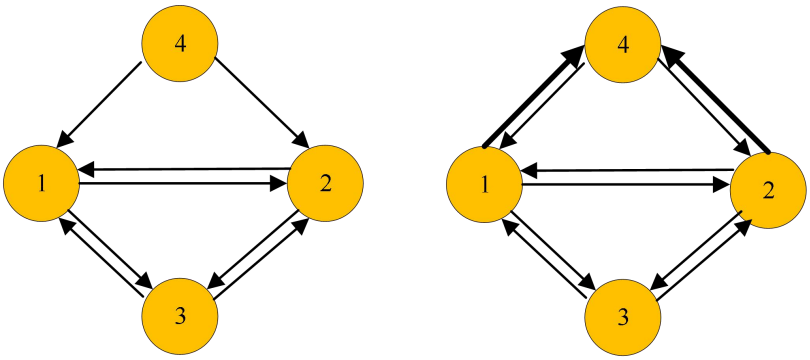


图 4-3 简单网络示例图

Figure 4-3 Simple network sample diagram

定义 5 用户（节点）相似度

$$P_{ij}=\frac{k_{i,j}^{out}+k_{i,j}^{in}}{k_i^{out}+k_i^{in}+k_j^{out}+k_j^{in}}+1 \qquad v_j \in \alpha_i \tag{4-4}$$

其中，α_i 为节点 v_i 的邻居节点；$k_{i,j}^{out}$ 表示节点 v_j 和节点 v_i 共同指向的节点个数，$k_{i,j}^{in}$ 表示共同指向节点 v_j 和节点 v_i 的节点个数；P_{ij} 表示节点 v_j 和节点 v_i 的相似度。对于这个公式，我们可以做出如下理解：

1）指向两节点的节点数越多，两个节点相似度越高；两节点指向的节点数越多，两个节点相似度越高。

2）节点对间的相似度程度，受到节点本身度数大小的制约。若存在两个节点对，其共同指向的节点数与共同指向其的节点数总和相同，但是其中一个节点对的度数总和小，另一个节点对的度数总和大，那么度数总和小的节点对更为相似。可以举一个简单的实例：若用户 u_a 和用户 u_b 都关注了 5 个用户，都拥有 5 个粉丝且用户 u_a 和用户 u_b 共同关注的用户有 2 个，且有 2 个粉丝同时关注用户 u_a 和 u_b，那么 $P_{ij}=1.2$；若用户 u_c 和用户 u_d 都关注了 50 个用户，都拥有 50 个粉丝且用户 u_c 和 u_d 共同关注的用户有 5 个，且有 5 个粉丝同时关注用户 u_c 和 u_d，那么 $P_{ij}=1.05$。可以看出，虽然用户 u_c 和 u_d 共同关注的用户人数更多，共同关注 u_c 和 u_d 的人数也更多，但是由于用户 u_c 和 u_d 共同关注的用户基数高，且粉丝基数大，因此实际上，他们共同感兴趣的内容和重叠部分并不高，他们的相似度也不高。

3）节点对的相似度为 0 时，节点之间仍然有相互作用。由于节点 v_j 是 v_i 的邻居节点，存在指向关系，因此即使他们共同指向的节点数与共同指向他们的节点数总和为 0，那么由于节点指向关系的存在，他们仍然有联系。即 $P_{ij}\geq 1$。

4）节点对之间的相似度与节点内部连接边的指向无关，即 $P_{ij}=P_{ji}$。这也和客观现实符合，Github 中两个用户的相似程度是客观属性，与他们的兴趣和偏好相关，故用户相似度是固定的。

4.4.2 用户关注网络 H 指数

H 指数中心性是利用 H 指数相关概念，通过考虑网络中节点的邻居节点的度，来定义节点的中心性。但在 Github 用户关注关系网络中，这个定义却不能适用。因为用户的关注关系是单向的，这便存在一个问题：用户关注其他用户数量的多少不能说明用户的重要性，用户的重要程度是与关注他的用户正相关的。也就是说，节点的出度不能说明节点的影响力大小，节点的入度大小才能体现节点的影响力。因此我们给出了用户关注关系网络种节点 v_i 的 H 指数中心性定义：

定义 6 用户关注关系网络的 H 指数

$$h_i = H(k_{j_1}^{in}, k_{j_2}^{in}, k_{j_3}^{in}, ..., k_{j_{k_i^{in}}}^{in}) \tag{4-5}$$

其中，$(j_1, j_2, j_3, ..., j_{k_i^{in}})$ 为指向节点 v_i 的节点，k_i^{in} 是节点 v_i 的入度。函数返回值 h_i，使得 $k_{j_1}^{in}, k_{j_2}^{in}, k_{j_3}^{in}, ..., k_{j_{k_i^{in}}}^{in}$ 中有 h_i 个值大于等于 h_i。特别的，$h_g=1$。

由上式可以得知，节点 v_i 的入度越大，且指向节点 v_i 的节点的入度越大，节点 v_i 的 H 指数就越高。这很好理解，反映在 Github 中，粉丝多的用户一般是重要的有价值的客户。而当一个用户自身受到的价值高用户的关注越多时，说明这个用户的重要程度越高。

4.4.3 所提算法的步骤

在本节中，为了进一步改善 LeaderRank 算法和带权的 LeaderRank 算法，我们引入了另一种基于有偏随机游走[83]的加权机制。随机游走[84]，也称随机漫步。其核心概念是指任何无规则行走者所带的守恒量都各自对应着一个扩散运输定律，接近于布朗运动，是布朗运动理想的数学状态。LeaderRank 算法的迭代过程属于随机游走。而有偏随机游走在游走过程中节点转移概率具有一定的偏向性，带权 LeaderRank 算法属于此类。本章基于有偏随机游走，提出了一种新的改进带权 LeaderRank 算法，且本章算法中节点的转移概率与带权 LeaderRank 算法完全不同。

首先，我们引入公式（4-5）所定义的 H 指数，并该参数作用于网络中所有的节点（不包括背景节点），从而可以得到每个节点的 H 指数。运算结果显示，H 指数是网络中衡量节点影响力的一个有效指标，并且对度数的变化不敏感。因此，我们将其应用于加权机制，以期望其提高现有算法预测的准确性和鲁棒性。

其次，我们引入公式（4-4）所定义的节点相似度，并将该指标作用于网络中的所有节点对（不包括背景节点），从而可以得到每对节点之间的相似度。节点相似度是衡量网络中节点间相似性的有效指标，我们同样将其应用于加权机制，来提高节点转移到相似节点的概率，从而提高所提算法的有效性。

对于网络的边权，我们的算法设计如下：（1）获取网络所有节点的 H 指数，节点的初始影响力（多属性决策指标）和节点对的相似度；（2）在网络中加入一个背景节点，通过双向边与所有其他节点相连；（3）定义网络的边权，并给出更新规则对每个节点应用有偏随机漫步，具体如下：

$$w_{ji}=\begin{cases} P_{ij}\cdot(h_i+1), v_j \to v_i, v_i \neq g, v_j \neq g \\ k_i^{in}, v_j \to v_i, v_j = g \\ 1, v_j \to v_i, v_i = g \\ 0, v_j \not\to v_i \end{cases} \tag{4-6}$$

其中，w_{ji} 为从节点 v_j 指向 v_i 的单向边的权重，P_{ij} 为节点 v_j 和 v_i 之间的权重，h_i 是节点 v_j 的 H 指数，点 g 为背景节点。对于这个公式，我们可以做出如下理解：

1）当节点 v_j 指向 v_i 且两者均不是背景节点时，边权是节点间相似度和被指向节点的 H 指数加一的乘积。这样设计既考虑了节点间相似度对节点 LR 值流向的影响，也考虑了节点随机漫步时应该更倾向具有较高 H 指数的节点。特别的，考虑到部分节点 H 指数为 0 的情况下，为不使边权为 0，我们将 H 指数加一。一般 LeaderRank 算法迭代时节点的影响力被均分给它所指向的其他节点，而带权 LeaderRank 算法迭代时只考虑了背景节点与其他节点之间的权值，普通节点移动到其他相邻节点的概率是相同的。与这两种算法相比，本章所提算法应用了有偏随机漫步，一般节点移动到其他相邻节点的概率与节点间相似度成正比，也与目标节点的 H 指数成正比。既考虑了单个节点的作用，也考虑了节点间相互作用。

2）当背景节点 g 指向 v_i 时，边权是节点 v_i 的入度。这样保持了本章算法与带权 LeaderRank 算法的一致，同时算法迭代时，背景节点的 LR 值按照网络中节点的入度占所有节点入度总和的比例被分散到其他节点中。

3）当 v_j 指向背景节点 g 时，边权是 1。网络中所有节点指向背景节点的边其权重均为 1，这样不仅保证了与前人所提算法相一致，而且迭代时，一般节点流向背景节点的 LR 值小于流向相邻一般节点的 LR 值。这样设计不仅一般节点与背景节点完全不相似且相似度为 1，且背景节点的 H 指数为 0，表示背景节点没有节点自身影响力值。

4）当节点 v_j 不指向 v_i 时，则不存在单向连边，故权重为 0。

在给出对于权重设置的合理解释后，结合 MADM_LR 和本章定义的网络边

权，我们给出本章算法，具体表示如下：

$$\begin{cases} LR_i(t+1)=\dfrac{1}{\sum_{q=1}^{n+1}w_{iq}+1}LR_i(t)+\sum_{j=1}^{n+1}\dfrac{w_{ji}}{\sum_{l=1}^{n+1}w_{jl}+1}LR_j(t) \\ LR_i(0)=C_i \\ LR_g(0)=0 \end{cases} \tag{4-7}$$

式中，前一项反映节点 v_i 本身的影响力；后一项反映节点 v_i 通过网络拓扑结构获得的影响力，$LR_i(0)=C_i$ 表示节点 v_i 在 t=0 时刻的值为 C_i。C_i 是节点 v_i 的初始影响力，即用户 u_i 的多属性决策指标值。特别的，由于我们将每个节点引出了一条指向自身的边，所以这条边的权值定义为 $w_{ii}=1$。这样保证节点在将 LR 值转移到其他节点的同时，将部分影响力留给自身。算法在迭代 t_c 次后达到收敛，节点最终的 LR 值为：

$$LR_i=LR_i(t_c)+\frac{LR_g(t_c)}{n} \tag{4-8}$$

背景节点将所得到的 LR 值全部分摊给网络中其余节点，从而得到最终的影响力排序。算法的具体步骤如下：

步骤 1：输入有向用户关注网络，节点初始值；

步骤 2：计算节点 H 指数，节点对之间的相似度，得到网络边权矩阵；

步骤 3：对网络进行初始化，令 $t=0$；

步骤 4：判断算法是否满足终止条件（LR 值不再变化，达到收敛）？如果是，转步骤 6；

步骤 5：计算式（4-7），得到 t 时刻网络中所有节点的影响力值，转步骤 4；

步骤 6：输出节点影响力排序结果。

算法在每次迭代后，都将自身的 LR 值按照不同连边的权重占比转移到其他节点上，且相似度和 H 属性值高的节点所获得的 LR 值更高。这样突出了网络中节点之间的相互作用。特别的，考虑到 Github 也是一种特殊的社交网络，用户存在社会活动属性。同 MADM_LR 一致，我们赋予了节点初始影响力，并且让每个节点有一条指向自身的边，使每个节点的初始影响力参与算法迭代并能有所保留。当 LR 值不再变化后，算法达到收敛。可以看出，本章所提算法既考虑了节点的初始影响力，也考虑了网络的边权。在 Github 角度上，所提算法不仅考虑了用户的多种社会活动属性，还考虑了不同用户之间的相似程度，并且进一步拓展了用户关注网络的拓扑属性。

4.5 实验（Experiments）

本小节通过实验，对所提方法的有效性进行验证。实验平台的硬件配置如下：Inter Core i5-10210U CPU、16G 内存；软件配置为：Windows 10 操作系统、Python 3.6.0 和 PyCharm 编译器。实验分为三组：（1）对于所提算法，我们可以确定最终所有用户的排序结果。第 1 组实验给出不同方法下用户的影响力排序。定性分析本章方法的合理性和有效性；（2）定量对比本节算法、MADM_LR 和带权 LeaderRank 算法，分析所提算法的优越性；（3）定量对比分析 US_WLR 与其他方法的鲁棒性。

4.5.1 不同方法下的用户影响力排序

本章采用 1-9 标度法，确定 star 数、watch 数和 fork 数的权重：$[s_1, s_2, s_3] = [0.3, 0.3, 0.4]$。将 US_WLR、MADM_LR 和传统 LeaderRank 算法应用于该网络，得到用户影响力排名前 10 的结果，如表 4-1 所列。

表 4-1 不同方法的用户影响力前 10 排名

Table 4-1 Top 10 user's influential rank generated by different method

排名	US_WLR	MADM_LR	LeaderRank
1	torvalds	torvalds	torvalds
2	JakeWharton	JakeWharton	JakeWharton
3	RuanYiFeng	Tj	michaelliao
4	Tj	douglascrockfor	Tj
5	openssl	visionmedia	mojombo
6	addyosmani	openssl	paulirish
7	docker	nicklockwood	githubpy
8	PaulIrish	docker	defunkt
9	KennethReitz	zxing	addyosmani
10	mojombo	mojombo	jeresig

表中具体所列分别为不同指标下的前 10 名用户。由表 4-1 可知，（1）有 6 个用户同时出现在 US_WLR 和 MADM_LR 的前 10 名，而排序却不尽相同。（2）用户 addyosmani 在 US_WLR 中的排序高于 MADM_LR 算法，这是因为，用户 addyosmani 在 Github 中有 3.7 万的关注，且其关注了 285 个用户，而 PaulIrish 有 2.9 万个关注，且其关注了 260 个用户。由于本章网络是基于关注关系构建的，关注关系多的用户，其兴趣广泛，与其他人相似度可能越高。这说明，用户相似度和用户被关注数量影响排序算法，从而影响最终的排序结果。（3）本章算法中，用户 RuanYiFeng 的排序很靠前，他是中文互联网上的知名 Vloger，也是一名 IT 人员。其存储库的 fork 数量较低，但 star 数量很高，且用户受到的关注数量为

6.5 万。这是因为用户 RuanYiFeng 在 Github 中分享了很多软件学习方面的资料与心得，从而赢得了其他用户的关注与认可。（4）本章算法和 MADM_LR 排名第 1、第 2 的用户相同，但本章算法后续能识别出影响力更高的用户。

表 4-1 是对算法合理性的定性分析。通过以上分析可以看出，本章算法能更好的刻画用户间关注关系和用户相似性，因而得到的用户影响力更加精准。算法与其他比较算法的精确对比将在 4.5.2 节给出。

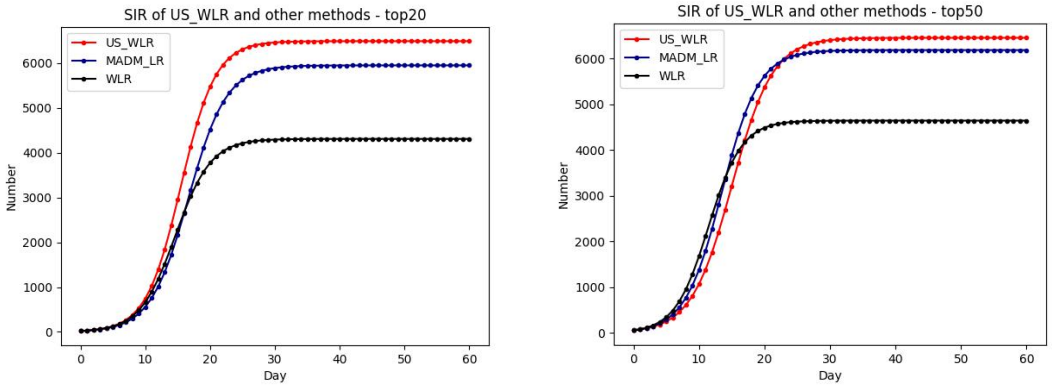
4.5.2 基于 SIR 模型的算法比较

（1）不同算法排名靠前节点传播能力比较

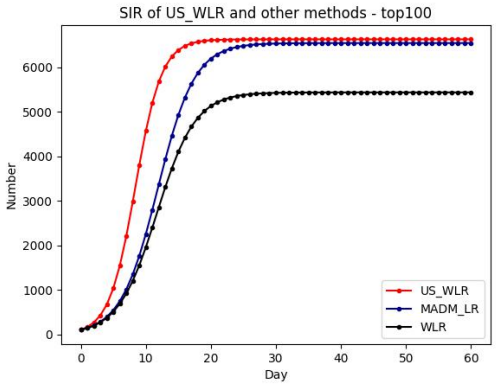
在 SIR 实验中，首先，选取 3 种算法排名在前面的不同用户，作为初始感染态 I 的成员；然后，观察这些感染者的传播能力。具体的讲，分别选取排名前 20、前 50 和前 100 的用户，作为感染者。在实验中，参数设置与第三章一致。观察并统计每次传播后被感染的节点个数，直到模型收敛。

本章算法的比较算法为 MADM_LR 和带权 LeaderRank 算法。与 MADM_LR 相比，本章算法 US_WLR 对边权做出定义，考虑了网络拓扑属性中的节点相似度和 H 指数；且本章改进了带权 LeaderRank 算法，考虑了节点间相互作用，比 MADM_LR 更为全面；与带权 LeaderRank 算法相比，增强了对网络边权改进的同时考虑了节点的初始影响力，得到的实验结果如图 4-4 所示。

由图 4-4 可知，经历 20-40 次传播后，SIR 模型收敛。相比 MADM_LR，US_WLR 的收敛速度较快，但是，传播节点数最多。这是因为 US_WLR 识别的关键节点，在相同时间范围内能够感染更多其他节点。这说明，本章所提的排序方法得到的用户具有更高的传播能力。原因在于，本章所提用户相似度与构造的有权用户关注网络是合理的，比其它算法更能挖掘更有影响力、传播能力更强的用户。



(a) (b)



(c)

图 4-4 不同算法得到的排名靠前节点的SIR传播能力

Figure 4-4 The SIR transmission capacity about the top nodes generated by different algorithms

由图 4-4 可知，经历 20-40 次传播后，SIR 模型收敛。相比 MADM_LR，US_WLR 的收敛速度较快，但是，传播节点数最多。这是因为 US_WLR 识别的关键节点，在相同时间范围内能够感染更多其他节点。这说明，本章所提的排序方法得到的用户具有更高的传播能力。原因在于，本章所提用户相似度与构造的有权用户关注网络是合理的，比其它算法更能挖掘更有影响力、传播能力更强的用户。

（2）不同算法节点重要性排序相关系数对比

节点的传播重要度定义为多次传播中，网络中感染状态和免疫状态的节点个数占网络总节点个数比例的平均值。分别对每一个节点进行 10 次传播实验取平均值，设置每次只有一个节点处于感染状态，网络中其他节点处于未感染状态，传播步长设置为 10，观察节点在网络中传播的情况。为此定义节点 v_i 的传播重要度为：

$$X(i)=\frac{1}{10}\sum_{\lambda=1}^{10}\frac{n-n_\lambda^s}{n} \qquad (4\text{-}9)$$

式中，n_λ^s 表示当初始传播节点为 v_i 时，第 λ 次传播仿真中经过 10 步传播后网络中未被感染的节点个数。可以看出，节点的传播重要度反映了节点的传播能力，传播重要度越高的节点，在网络中越重要，影响力越大。

为此，我们选取不同算法中排名前 10，前 20 和前 50 名的节点分别进行考虑。对它们进行传播重要度排序，并分析两种排序结果的相关性。首先给出相关性系数的定义：相关性系数 Y 定位为节点重要度排序和传播重要度的相关性。Y 的绝对值越接近 1，表明两者的相关性越强，算法效果越好。Y 由斯皮尔曼等级相关系数[85]计算，具体定义为：

$$Y = 1 - \frac{6\sum_{i=1}^{n}(R_i - R_i')^2}{n(n^2-1)} \tag{4-10}$$

式中，R_i 和 R_i' 分别为节点 v_i 在节点重要性排序算法中的排名和传播重要度中的排名。可以看出，R_i 和 R_i' 越接近时节点的排序越接近，差值越小则 Y 的绝对值越接近 1。

为了分析不同算法下节点重要性排序结果的好坏，本章分别选取带权 LeaderRank 算法，MADM_LR 和本章所提算法的重要性排序结果，并对不同算法下排名前 50 的节点分别计算其传播重要度。两者相关性的对比结果如表 4-2 所示。

表 4-2 不同算法节点排序与传播能力排序相关性

Table 4-2 Correlation between different algorithm nodes sorting method and transmission ability

排名	US_WLR	MADM_LR	WLR
前 10	**0.9985**	0.9766	0.9254
前 20	**0.9981**	0.9713	0.8960
前 50	0.9613	**0.9628**	0.8336

由表 4-2 可知，在排名前 10 和排名前 20 中，US_WLR 所得节点排序与传播能力排序更为相关，但是排名前 50 名中，MADM_LR 能够得到具有更高的相关性的结果。这说明在 US_WLR 算法得到的用户排序集合中，排名靠前的用户传播能力更强，算法对排名靠前的用户识别能力更加。从整体来看，实验结果说明了本章算法的优越性。

4.5.3 算法的鲁棒性比较

由于实际的开源社区中存在很多非真实或遗漏的边，还存在很多“僵尸”用户与假粉。因此，同第三章相同，通过以上两种不同的干扰来验证本章算法的鲁棒性。

（1）施加网络干扰边

随机地从网络的节点间增加或移除一些连接关系，作为关注者与被关注者之间增加或缺失的关注关系，比较 US_WLR，MADM_LR 与带权 LeaderRank 算法的用户排名的变化。对于网络的边权，我们分别重新计算节点的相似度和 H 指数，并进行相应更新。为此，分别随机增加或移走网络的{50，100，150，200，250，300，350，400，450，500，600，700，800，900，1000，1100，1200}条边，施加干扰后 I_r 的值如图 4-5 所示。

由图 4-5 可知，随着随机增加或移走网络的边，I_r 值整体呈现增加的趋势。这说明随着用户的干扰增加，节点的排序偏差上升，且增加边对网络的干扰较减

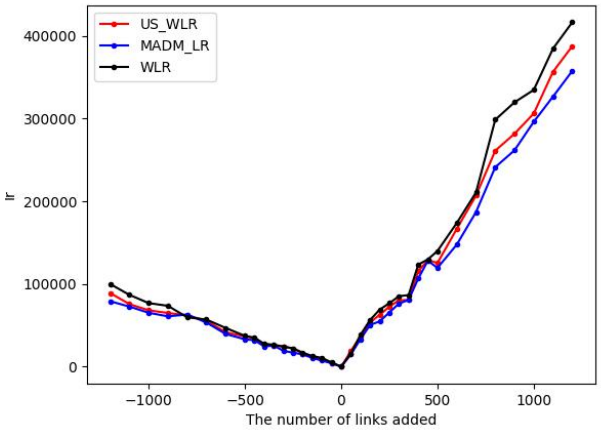


图 4-5 干扰边对三种算法节点排名的影响

Figure 4-5 Interferon's influence to three different algorithm nodes' rank

少边更大。无论加边还是减边，US_WLR 的 I_r 值整体优于带权的 LeaderRank 算法。但与第三章所提算法 MADM_LR 相比，US_WLR 的 I_r 值整体更大，偏差更明显。这说明，由于加边和减边的操作会影响节点相似度和 H 指数，对网络边权产生扰动，因此鲁棒性低于 MADM_LR。但是本章所提算法仍然比一般算法具有更好地稳定性，这是因为 US_WLR 同样考虑了用户的自身属性，这在一定程度上减少了由网络拓扑的扰动带来的影响。

（2）添加社交网络假粉

与第三章一致，选择排名为{10，20，30，40，50，60，65，70，75，80，85，90，95，100}这 14 个用户并分别增加 $v=10,20,50$ 个关注者。对这 14 个用户分别施加不同程度的干扰，US_WLR 与带权 LeaderRank 算法得到的这 14 个用户新的排序结果如图 4-6 所示。

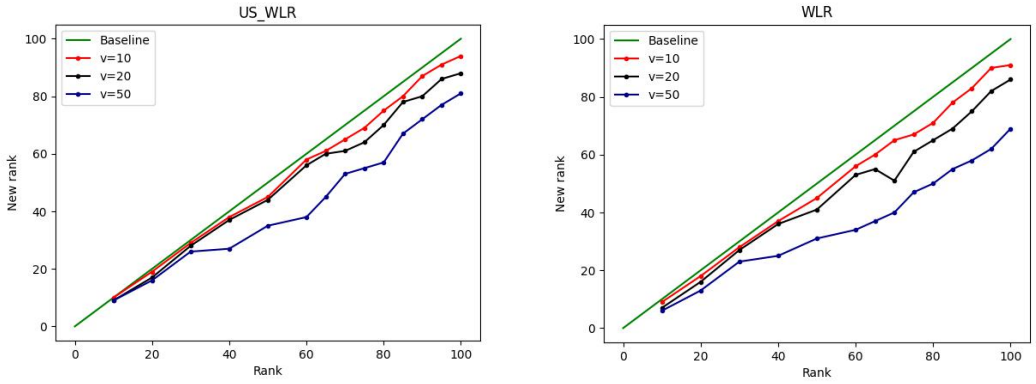


图 4-6 假粉对两种算法用户排名的影响

Figure 4-6 Fake followers' influence to two different algorithm users' rank

由图 4-6 可知，随着假粉个数的增加，两种算法的差异均越来越明显，增加 50 个关注者时，算法的波动会更大。但是，整体来看 US_WLR 的波动小于对比算法，这说明 US_WLR 的抗干扰能力优于带权 LeaderRank。

4.6 本章小结（Summary）

在实际生活中，重要的核心人物往往会受到其他人的欢迎，并且会对他们造成一定的影响。日常中人们同样也愿意相信那些和他们相似且关系亲密的人。在 Github 中亦是如此，对于单一用户而言，相似度高的其他用户和受欢迎的用户都会让他产生有偏的选择。在 MADM_LR 中，我们考虑了用户的不同社会活动属性，提出了一种改进的 LeaderRank 算法。但这种算法存在一定不足。在对于用户关注关系网络的构造中，我们令所有边权为 1，这种情况下节点的流向是随机的。也就是说用户受到的影响，或者做出的选择，是无偏的。但事实上，不同用户间相互作用是不同的，不能一概而论。为了更加全面细致地衡量用户间相互作用关系，我们需要对网络的边权进行区分构造。

本章针对 Github 上用户影响力评价问题，提出了一种基于用户相似度的改进带权 LeaderRank 算法。与已有方法相比，该方法既考虑了用户的多种社会活动属性，将多种社会活动属性构造为多属性决策指标。还考虑了节点间相互作用，定义了用户相似度指标和有向关注网络的节点 H 指数中心性，并根据这两种网络拓扑属性，构造了网络的权重。最后，把所提出的算法应用于带权用户有向关注网络上，得到了既考虑用户个人属性，又考虑用户相似度的影响力排序。

为说明本章方法的有效性，将本章算法的排序结果与 MADM_LR 和带权 LeaderRank 算法作比较，结果表明 US_WLR 同样可以找到有影响力，且传播能力更强的用户。将这 3 种方法应用于 SIR 模型中，通过对不同排名段用户排序结果进行比较，我们发现本章方法在收敛速度上与其他算法不相上下，且可以找到传播能力更强的节点，体现了本章所提方法的优越性。分析这 3 种算法所得排序结果与传播能力排序结果之间的相关性，结果表明 US_WLR 所得排名靠前的用户具有更强的传播能力。虽然 US_WLR 的鲁棒性较 MADM_LR 低，但是与一般带权 LeaderRank 算法相比，仍然具有较好的稳定性。

5 结论

5 Conclusions

5.1 本文工作（Achievements of This Thesis）

近年来，互联网技术迅猛发展，随之产生的软件系统越来越多且日新月异。大众对于软件系统的要求日趋增加，Github 这样的开源软件平台逐渐出现在大众视线中。识别 Github 中的关键用户可以帮助决策者更好的采取相关举措。鉴于此，本文考虑用户之间的关注关系，分别构建了无权和有权两种类型的网络，提出了精准评估用户影响力的方法，有效解决了传统复杂网络节点重要性度量方法识别效率低下的问题，极大丰富了 Github 的理论内容与相关应用。

本文的主要研究工作如下：

（1）提出基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评估方法

鉴于现有的复杂网络节点重要性度量方法均不适合识别 Github 中的关键用户，存在运行时间长，结果质量差的问题，提出了基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评估方法，即 MADM_LR。首先，建立了更具有代表性的无权用户关注网络。接着，基于用户的社会活动属性构造了多属性决策指标。然后，提出考虑节点初始影响力的 LeaderRank 算法。最后，将所提方法应用于构建的网络，通过实验证明，该方法能够更有效的识别关键用户。

（2）提出基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评估方法

由于现有的节点影响力评估方法都没有考虑节点之间的相互作用关系，存在识别结果不够精准，考虑角度不够全面的问题，提出了基于用户相似度和改进带权 LeaderRank 算法的 Github 用户影响力评估方法，即 US_WLR。首先给出了用户相似度和有向网络 H 指数相关定义，并依此构造了网络的边权，有效弥补了之前的空白；然后提出了适用于带权关注网络的改进带权的 LeaderRank 算法。将所提算法与其他算法进行对比，实验结果表明，US_WLR 能够识别出更具影响力且传播能力更强的用户。

5.2 进一步研究工作（Futher Research Work）

本文针对 Github 用户影响力的评价问题，提出了多属性决策指标和两种改进的 LeaderRank 算法，在一定程度上解决了已有方法不能精准识别关键用户的问题，但是本文方法也存在需要改进的地方。Github 中存在很多其他的动态交互关系，且源代码中也存在很多隐藏的属性供学者挖掘。下面根据本文不足，进一

步提出今后的研究方向：

（1）结合用户贡献的源代码，更全面的考虑用户的多种属性。本文方法采用 star，fork 和 watch 三种属性构造用户多属性决策指标。虽然这三种属性值能够从不同角度体现用户自身特征，但是仍然有提升空间。用户提交的代码数量，以及代码的内容，都能刻画用户在 Github 中的贡献量与贡献价值。此外，用户的其他行为诸如 pull request 和 commit，都能作为不同的属性参与决策。因此，如何更为精准的刻画用户的自身特征和个人贡献，将是进一步需要研究的内容。

（2）给出 Github 动态网络的建模方法，并提出适用于该网络的用户影响力排序方法。本课题的研究，在构建网络时都将 Github 构建为静态网络，这是因为我们爬取的数据只截止到 Github 某一时刻的情况。但实际上，Github 中用户每时每刻都在进行社交活动，用户之间关系不是一成不变的，因此用户的影响力也是动态变化的。如果能够考虑动态 Github 网络上的用户影响力，决策者就能更清晰的看出每个用户影响力随时间的变化，为预测社区演化趋势和调控社区奠定基础。

参考文献

[1] 孙连山, 李健. 软件生态系统的角色模型和质量模型[J]. 陕西科技大学学报, 2011, 29(2):93-95.

[2] Bosch J. From software product lines to software ecosystems[C]. Proceeding of the 13th International Software Product Line Conference, 2009: 111-119.

[3] 韩欢, 冯志勇, 陈世展. 生态化复杂软件系统集成研究综述[J]. 科技通报, 2016, 32(8):137-141.

[4] 何鹏, 李兵, 杨习辉,等. 开源软件社区开发者偏好合作行为研究[J]. 计算机科学, 2015, 42(2):161-166.

[5] Hu Y, Wang S S, Ren Y Z, et al. User influence analysis for Github developer social networks[J]. Expert Systems with Applications, 2018, 108(6):108-118.

[6] Chandrasekara C, Herath P. Introduction to GitHub actions[M]. California: Apress, 2021.

[7] Vicario P D, Tortolini V. Evaluating a programming topic using GitHub data: what we can learn about machine learning[J]. International Journal of Web Information Systems, 2021, 43(12):54-58.

[8] Feng K Y, Cong G, Bhowmick S S, et al. In search of influential event organizers in online social networks[C]. Proceedings of The 2014 ACM SIGMOD International Conference on Management of Data, 2014:63-74.

[9] 任晓龙, 吕琳媛. 网络重要节点排序方法综述[J]. 科学通报, 2014, 59(13):1175-1197.

[10] Chaoji V, Ranu S, Rastogi R, et al. Recommendations to boost content spread in social networks[C]. Proceedings of The 21st International Conference on World Wide Web, 2012:529-538.

[11] Salamanos N, Voudigari E, Yannakoudakis E J. Identifying influential spreaders by graph sampling[C]. International Workshop on Complex Networks & Their Applications, 2016:111-115.

[12] 刘雅新, 吴高艳, 何鹏. 开源软件社区中开发者活跃度特性分析[J]. 软件导刊, 2017, 16(9):164-169.

[13] 齐晴, 曹健, 刘妍岑. GitHub 中软件生态系统的演化[J]. 计算机研究与发展, 2020, 57(3):57-68.

[14] 韩雨泓, 祝鹏程. 软件生态系统的负熵流模型[J]. 现代计算机, 2020, 685(13):11-17.

[15] Kula R G, Roover C D, German D M, et al. A generalized model for visualizing library popularity, adoption, and diffusion within a software ecosystem[C]. IEEE International Conference on Software Analysis, IEEE Computer Society, 2018:288-299.

[16] Plakidas K, Schall D, Zdun U. Evolution of the R software ecosystem: metrics, relationships,

and their impact on qualities[J]. Journal of Systems and Software, 2017, 132:119-146.

[17] 刘亚珺. 开源软件生态系统中交替修改度量方法及工具实现[D]. 武汉大学, 2018.

[18] 李华莹, 刘丽, 刘怡静. 面向软件生态的资源定位技术[J]. 计算机与现代化, 2020, 295(3):28-32.

[19] Barbosa O, Alves C. S. A systematic mapping study on software ecosystems through a three-dimensional perspective[C]. Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, 2013:59-81.

[20] Mens T, Claes M, Grosjean P. ECOS: Ecological studies of open source software ecosystems[C]. Software Evolution Week-IEEE Conference on Software Maintenance, 2014:403-406.

[21] Leibzon W. Social network of software development at GitHub[C]. 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2016:1374-1376.

[22] Biazzini M, Baudry B. "May the Fork Be with You": Novel Metrics to Analyze Collaboration on GitHub[C]. WeTSOM, 2014:342-354.

[23] Pletea D, Vasilescu B, Serebrenik A. Security and emotion: sentiment analysis of security discussions on GitHub[C]. Proceedings of The 11th Working Conference on Mining Software Repositories. 2014: 348-351.

[24] Borges H, Hora A, Valente M T. Understanding the factors that impact the popularity of GitHub repositories[C]. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2016: 334-344.

[25] Ahmad H A. GitHub 开源软件 (OSS) 项目中多种社交关系的挖掘与分析[D]. 哈尔滨工业大学, 2016.

[26] 李存燕, 洪玫. Github 中开发人员的行为特征分析[J]. 计算机科学, 2019, 46(2):161-167.

[27] 乔秀全, 杨春, 李晓峰,等. 社交网络服务中一种基于用户上下文的信任度计算方法[J]. 计算机学报, 2011, 34(12):2403-2413.

[28] 王珏, 高琳. 基于社交圈的在线社交网络朋友推荐算法[J]. 计算机学报, 2014, 24(4):801-808.

[29] 康书龙. 基于用户行为及关系的社交网络节点影响力评价[J]. 北京邮电大学, 2011, 6(4):25-53.

[30] 张晨逸, 孙建伶, 丁轶群. 基于 MB-LDA 模型的微博主题挖掘[J]. 计算机研究与发展, 2011, 48(10): 3-10.

[31] 申琳. 基于多属性的社交网络关键节点挖掘方法[D]. 西安电子科技大学, 2017.

[32] Zhu Z, Cao J, Zhou T, et al. Understanding User Topic Preferences across Multiple Social

Networks[J]. ArXiv Preprint, 2021, 65(4):87-88.

[33] Dhand A, McCafferty L, Grashow R, et al. Social network structure and composition in former NFL football players[J]. Scientific Reports, 2021, 11(1):1-9.

[34] 周涛, 柏文洁, 汪秉宏, 等. 复杂网络研究概述[J]. 物理, 2005, 34(1):31-36.

[35] 赫南, 李德毅, 淦文燕, 等. 复杂网络中重要性节点发掘综述[D]. 计算机科学, 2007, 34(12):15-17.

[36] 李鹏翔, 任玉晴, 席酉民. 网络节点 (集) 重要性的一种度量指标[D]. 系统工程, 2004, 22(4):32-36.

[37] 陈静, 孙林夫. 复杂网络中节点重要度评估[J]. 西南交通大学学报, 2009, 44(3): 426-429.

[38] Nardelli E, Proietti G, Widmayer P. Finding the most vital node of a shortest path[J]. Theoretical Computer Science, 2003, 296(1): 167-177.

[39] 饶育萍, 林竞羽, 周东方. 网络抗毁度和节点重要性评价方法[J]. 计算机工程, 2009, 35(6):14-16.

[40] 陈勇, 胡爱群, 胡啸. 通信网中节点重要性的评价方法[J]. 通信学报, 2004, 25(8):129-134.

[41] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: bringing order to the web[R]. Stanford InfoLab, 1999.

[42] Zhang Y, Xu K, Liu Y, et al. Modeling of scale-free network based on pagerank algorithm[C]. 2010 2nd International Conference on Future Computer and Communication. IEEE, 2010, 3: 783-786.

[43] 廖志芳, 李斯江, 贺大禹, 等. GitHub 开源软件开发过程中关键用户行为分析[J]. 小型微型计算机系统, 2019, 40(1):164-168.

[44] Goyal A, Bonchi F, Lakshmanan L V S. Learning influence probabilities in social networks[C]. Proceedings of The Third ACM International Conference on Web Search and Data Mining. 2010:241-250.

[45] Zaman T R, Herbrich R, Van Gael J, et al. Predicting information spreading in twitter[C]. Workshop on Computational Social Science and The Wisdom of Crowds. 2010:599-601.

[46] 卢冬冬, 吴洁, 刘鹏, 等. 开源软件社区开发者合作网络稳定性研究—以 AngularJS 为例[J]. 复杂系统与复杂性科学, 2020, 67(3):41-49.

[47] 周涛, 王超. 开源软件社区用户知识贡献行为研究[J]. 科研管理, 2020, 32(2):202-209.

[48] Badashian A S, Stroulia E. Measuring user influence in GitHub: the million follower fallacy[C]. 2016 IEEE/ACM 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE), 2016:15-21.

[49] Hu Y, Zhang J, Bai X, et al. Influence analysis of Github repositories[J]. SpringerPlus, 2016, 5(1):1-19.

[50] 李变. 基于 Github 社交网络中用户影响力评估算法的研究[D]. 西安电子科技大学, 2015.

[51] Blincoe K, Sheoran J, Goggins S, et al. Understanding the popular users: Following, affiliation influence and leadership on GitHub[J]. Information and Software Technology, 2016, 70(3): 30-39.

[52] Wu C F, Chen M Q. Complexity of land ecosystem[J]. The Journal of Applied Ecology, 2002, 13(6):753-6.

[53] Li M X, Han J Y. Complex network theory in urban traffic network[C]. International Conference on Materials Science, 2017:910-913.

[54] Saleh M, Esa Y, Mohamed A. Applications of complex network analysis in electric power Systems[J]. Energies, 2018, 11(6).

[55] 孙玺菁, 司守奎. 复杂网络算法与应用[M]. 北京: 国防工业出版社, 2015.

[56] Watts D J, Strogatz S H. Collective dynamics of small-word networks[J]. Nature, 1998, 393(4):440-442.

[57] Barabasi A L, Albert R. Emergence of scaling in random networks[J]. Science, 1999, 286(5439):509-512.

[58] 侯婷婷. 基于复杂网络的软件生态系统社区检测[D]. 中国矿业大学, 2020.

[59] Yu Y, Wang H M, Yin G, et al. Reviewer recommender of pull-requests in GitHub[C]. Proceedings of IEEE International Conference on Software Maintenance and Evolution, 2014:609-612.

[60] Asri I E. From periphery to core: a temporal analysis of GitHub contributors' collaboration network[C]. Proceedings of The Working Conference on Virtual Enterprises, 2017:1-13.

[61] Wang C, Du Y J, Tang M W. Opinion leader mining algorithm in microblog platform based on topic similarity[C]. IEEE International Conference on Computer and Communications, 2016:160-165.

[62] Li C F, Xiong F. Social recommendation with multiple influence from direct user interactions[J]. IEEE Access, 2017, 5:16288-16296.

[63] Hu B F, Wang H, Yu X M, et al. Sparse network embedding for community detection and sign prediction in signed social networks[J]. Journal of Ambient Intelligence And Humanized Computing, 2019, 10(1):175-186.

[64] Deng X L, Zhai J Y, Lv T J, et al. Efficient vector influence clustering coefficient based directed community detection method[J]. IEEE Access, 2017, 5:17106-17116.

[65] Lv L, Zhang Y C, Yeung C H, et al. Leaders in social networks, the delicious case[J]. PloS one, 2011, 6(6):21022-21024.

[66] 刘建国，任卓明，郭强，等. 复杂网络中节点重要性排序的研究进展[J]. 物理学报, 2013, 62(17):178901-178901.

[67] Zeleny M, Cochrane J L. Multiple criteria decision making[M]. McGraw-Hill, 1982.

[68] 王应明. 运用离差最大化方法进行多指标与排序[J]. 中国软科学, 1998(3):36-38.

[69] Bryson N, Mobolurin A. An action learning evaluation procedure for multiple criteria decision making problems[J]. European Journal of Operational Research, 1997, 96(2):379-386.

[70] 王宗军. 多目标权系数赋值方法及其选择策略[J]. 系统工程与电子技术, 1993(6):35-41.

[71] Saaty T L, Kearns K P. The analytic hierarchy process[J]. Analytical Planning, 1985(8):19-62.

[72] Hebig R, Quang T H, Chaudron M, et al. GitHub M. The quest for open source projects that use UML[C]. Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, 2016:173-183.

[73] Avelino G, Valente M T, Hora A. What is the truck factor of popular GitHub applications? A First Assessment[J]. PeerJ PrePrints, 2015, 3(8):1233-1236.

[74] Casalnuovo C, Vasilescu B, Devanbu P, et al. Developer onboarding in GitHub: the role of prior social links and language experience[C]. Proceedings of The 2015 10th Joint Meeting on Foundations of Software Engineering. 2015:817-828.

[75] Dabbish L, Stuart C, Tsay J, et al. Social coding in GitHub: Transparency and Collaboration in an Open Software Repository[C]. ACM Conference on Computer Supproted Cooperative Work. 2012:375-385.

[76] Cha M, Haddadi H, Benevenuto F, et al. Measuring user influence in twitter: The million follower fallacy[C]. Proceedings of the International AAAI Conference on Web and Social Media. 2010:87-89.

[77] Katsimpras G, Vogiatzis D, Paliouras G. Determining influential users with supervised random walks[C]. Proceedings of The 24th International Conference on World Wide Web. 2015:787-792.

[78] Aziz M, Rafi M. Identifying influential bloggers using blogs semantics[C]. Proceedings of the 8th International Conference on Frontiers of Information Technology. 2010:1-6.

[79] Börner K, Dall'Asta L, Ke W, et al. Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams[J]. Complexity, 2005, 10(4):57-67.

[80] 何艳辉，唐三一. 经典 SIR 模型辨识和参数估计问题[J]. 应用数学和力学, 2013, 34(3):252-258.

[81] Li Q, Zhou T, Lv L, et al. Identifying influential spreaders by weighted LeaderRank[J]. Physica A, 2014, 404(24):47-75.

[82] 顾亦然, 朱梓嫣. 基于 LeaderRank 和节点相似度的复杂网络重要节点排序算法[J]. 电子科技大学学报, 2017, 2(46):123-130.

[83] Zhou Y Z, Wu C C, Tan L L. Biased random walk with restart for link prediction with graph embedding method[J]. Physica A: Statistical Mechanics and its Applications, 2021(6):125-131.

[84] Wang F, Landau D P. Efficient, multiple-range random walk algorithm to calculate the density of states[J]. Physical Review Letters, 2001, 86(10):2050-2053.

[85] 马俊, 周刚, 许斌, 等. 一种基于话题传播的微博用户影响力分析方法[J]. 信息工程大学学报, 2013, 14(6):735-742.

作者简历

一、基本情况

姓名：王启瑞 性别：女 民族：汉 出生年月：1996-11-30 籍贯：山西省大同市

2014-09—2018-06 中国矿业大学信息与电气工程学院学士

2018-09—2021-06 中国矿业大学信息与控制工程学院攻读硕士学位

二、学术论文

1. **Wang Q R**, Liu M, Zeng B. A method of recommending crowdsourcing tasks incorporating with the interests and capabilities of workers. 【The Chinese Journal of Artificial Intelligence 在审，第一作者】
2. **王启瑞**, 姚香娟, 沈鑫, 巩敦卫. 基于多属性决策和改进 LeaderRank 算法的 Github 用户影响力评价. 【电子学报在审，第一作者】

三、获奖情况

1. 2018.9-2019.9 一等奖学金
2. 2019.9-2020.9 二等奖学金
3. 2020.9-2021.6 二等奖学金

四、研究项目

1.国家重点研发项目（软件生态系统的生成演化与群体协作机理研究），编号：2018YFB1003802-01，参加成员。

学位论文原创性声明

本人郑重声明：所呈交的学位论文《基于改进LeaderRank算法的Github用户影响力评价》，是本人在导师指导下，在中国矿业大学攻读学位期间进行的研究工作所取得的成果。据我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

年　月　日

学位论文数据集

关键词*	密级*	中图分类号*	UDC	论文资助
Github；用户影响力评价；LeaderRank 算法；多属性决策	公开	TP311.5	004.6	

学位授予单位名称*	学位授予单位代码*	学位类别*	学位级别*
中国矿业大学	10290	工学	硕士

论文题名*	并列题名*	论文语种*
基于改进 LeaderRank 算法的 Github 用户影响力评价	Evaluation of Users' Influences in Github based on Improved LeaderRank Algorithm	中文

作者姓名*	王启瑞	学号*	TS18060098A3LD1
培养单位名称*	培养单位代码*	培养单位地址	邮编
中国矿业大学	10290	江苏省徐州市	221116
学科专业*	研究方向*	学制*	学位授予年*
控制科学与工程	用户影响力评价	3 年	2021 年
论文提交日期*	2021 年 6 月		
导师姓名*	巩敦卫	职称*	教授

评阅人	答辩委员会主席*	答辩委员会成员
	郭西进	缪燕子，王法广

电子版论文提交格式　文本（√）　图像（　）　视频（　）　音频（　）　多媒体（　）　其他（　）

推荐格式：application/msword; application/pdf

电子版论文出版（发布）者	电子版论文出版（发布）地	权限声明

论文总页数*	52

注：共 33 项，其中带*为必填数据，共 22 项。