软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007555] [CSTR: 32375.14.jos.007555]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

真实开源软件仓库开发者交流网络的社区检测基准研究*

张洁芮[1], 汪 亮[1], 常正超[2], 李梓荣[1], 桑百惠[1], 陶先平[1]

[1](南京大学 计算机软件研究所 前沿交叉中心, 江苏 南京 210023)

[2](西北大学 信息科学与技术学院, 陕西 西安 710127)

通信作者: 汪亮, E-mail: wl@nju.edu.cn

摘 要: 群体智能是新一代人工智能的重要组成部分, 研究如何激发和汇聚开源社区“人”的力量能够极大提升开发效率. 社区检测作为研究开源项目中开发者关系的技术手段, 在社交网络的探索和理解中起着至关重要的作用. 然而, 目前的研究主要集中在 Facebook 等大规模社交网络上, 系统性地研究项目级开源软件开发者社交网络 (OSS-DSN) 中的社区检测仍然较为缺乏. 首先收集真实世界的数据, 并分析 OSS-DSN 的特征. 然后, 在这些真实数据集上, 对多种重叠和非重叠的社区检测算法进行基准测试, 比较它们在多个度量和维度上的性能. 最后, 基于合成的 OSS-DSN, 高效地生成了一系列合成网络, 并使用带有真实标签的数据进行算法评估, 以进行对比分析. 识别了小规模和中等规模社交网络与大规模网络之间的特征差异, 并探讨了这些差异如何影响社区检测的度量标准及算法性能. 提供了一个新的基准, 并为更好地理解开源软件社区中的沟通与协作提供了重要的见解.

关键词: 开源软件; 开发者社交网络; 社区检测

中图法分类号: TP311

中文引用格式: 张洁芮, 汪亮, 常正超, 李梓荣, 桑百惠, 陶先平. 真实开源软件仓库开发者交流网络的社区检测基准研究. 软件学报. http://www.jos.org.cn/1000-9825/7555.htm

英文引用格式: Zhang JR, Wang L, Chang ZC, Li ZR, Sang BH, Tao XP. Benchmarking Study of Community Detection on Developer Communication Networks in Real Open Source Software Repositories. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7555.htm

Benchmarking Study of Community Detection on Developer Communication Networks in Real Open Source Software Repositories

ZHANG Jie-Rui[1], WANG Liang[1], CHANG Zheng-Chao[2], LI Zi-Rong[1], SANG Bai-Hui[1], TAO Xian-Ping[1]

[1](Frontier Interdisciplinary Center, Institute of Computer Software, Nanjing University, Nanjing 210023, China)

[2](School of Information Science & Technology, Northwest University, Xi’an 710127, China)

Abstract: Crowd intelligence is a crucial component of the next generation of artificial intelligence. Researching how to stimulate and converge the innovative power of “people” in open-source communities can significantly enhance development efficiency. Community detection, as a technical approach to studying the relationships among developers in open-source projects, plays a vital role in exploring and understanding social networks. However, current research has predominantly focused on large-scale social networks such as Facebook, while systematic studies on community detection in project-level open source software developer social networks (OSS-DSN) remain limited. This study first collects real-world data and analyzes the features of OSS-DSN. Then, it benchmarks several overlapping and non-overlapping community detection algorithms on these real datasets, comparing algorithm performance across multiple metrics and dimensions. Finally, based on synthetic OSS-DSN, it generates networks efficiently and performs algorithm evaluations using ground-truth data for comparative analysis. Differences in characteristics between small- and medium-scale social networks and large-scale networks are identified, and the influence of these differences on community detection metrics and algorithm performance is explored. The study provides a new benchmark and offers important insights into communication and collaboration in open-source software communities.

Key words: open source software (OSS); developer social network (DSN); community detection

---

* 基金项目: 国家自然科学基金面上项目 (62172203)
收稿时间: 2025-03-19; 修改时间: 2025-06-05; 采用时间: 2025-09-23; jos 在线出版时间: 2026-01-07

1 引 言

群体智能是新时代人工智能发展中的重要组成部分[1], 对于激发汇聚开发者的创新力量、集成大规模智能系统有决定性的作用. 更进一步说, 发挥群体智能的力量, 有助于促进人工智能和传统产业的深度融合, 推动国民经济持续发展. 在这一过程中, 开源作为互联网上的一类特殊群智模式, 使得开发者以互联网为协作环境, 以软件源代码为协作产物, 跨地域、跨时间地参与软件开发过程和软件生产活动, 体现出了高度自治的群智行为, 极大加速了软件开发的效率. 与传统软件开发有所不同的是, 开源软件开发中, “人”是软件系统构建演化的驱动要素, 认识开发过程中的群智行为有助于预测项目开发过程的事件、预测软件可能存在的质量问题等. 研究开源社区贡献者组织结构有助于帮助外围开发者积极参与项目具体任务的解决, 实现开发者群体强组织和弱组织的融合[2]. 也能够吸引和激发更多的参与者加入项目开发[3].

由于开源软件的发展在很大程度上依赖于开发者之间的协作, 而这些开发者共同构成了复杂的社交网络[4], 因此理解和分析这些网络对于深入探索影响 OSS 项目增长和可持续发展的关键因素至关重要[5]. 在开源软件开发者社交网络 (open source software developer social network, OSS-DSN) 中, 社区检测已成为一种重要且常用的分析方法[6]. 识别核心开发者并分析其与外围贡献者的关系, 有助于评估影响项目成功和社区参与度的关键因素[7]. 此外, 发现紧密联系的开发者群体, 并分析不同专业领域的开发者如何频繁互动、协作、共享知识以及共同影响项目的发展, 可以为理解成功项目的内在结构和协作模式提供有价值的见解[8,9].

目前, 已有多种社区检测算法被应用于不同类型的网络分析, 其中包括基于模块度的方法和基于图的方法. 然而, 这些算法的性能和有效性很大程度上依赖于被分析网络的结构特征, 不同的网络结构可能会导致不同的检测结果[10]. 现有的社区检测基准测试[11–15]主要涉及在合成网络上的比较算法, 或采用 Amazon 这样的超大规模网络以及如 Football 的微小网络进行测试. 然而, 针对 OSS-DSN 这一类具有独特特征的社交网络, 仍缺乏专门的系统性基准测试, 以评估哪种社区检测算法最适用于这些网络.

在本研究中, 我们针对 OSS-DSN 构建了一个基准测试, 以 OSS-DSN 为研究对象, 评估不同社区检测算法的性能, 并围绕以下研究问题展开探讨.

RQ1: OSS-DSN 具有哪些特征? 实验结果表明 OSS-DSN 具有中心化的小世界网络特征, 并包含紧密联系的社区结构, 与大型社交网络去中心化的层次结构不同. 这些网络通常表现出较高的鲁棒性和连通性. 基于对 OSS-DSN 特征的分析, 我们提供了一系列网络参数建议, 并在算法比较和合成网络生成中应用这些参数进行实验.

RQ2: 在真实世界的 OSS-DSN 上, 不同社区检测算法的性能如何? 我们的实验扩展了社区检测算法的比较范围, 并针对开发者社交网络这一特有语境提出了具体的算法选择建议, 尤其推荐 Louvain、Infomap 和 SLPA. 除 AGDL 和 EM 之外, 非重叠社区检测算法在检测到的社区之间表现出较高的一致性, 而重叠社区检测算法则显示出较低的一致性. 这种一致性差异可能源于不同重叠算法在假设上的不同, 从而导致社区重叠部分的模糊性.

RQ3: 在合成的 OSS-DSN 上, 不同社区检测算法的性能如何? 由于真实数据产生的网络没有标准的社区标签作为参照, 尽管我们在 RQ2 中做了算法之间的比较, 并发现了结果具有一致性, 但是这对于使用者而言并不方便, 因此使用合成网络模拟是一个较好的替代实验方案. 该研究问题试图回答如何用合成网络模拟真实世界 OSS-DSN, 以及其比较结果是否与真实数据集上的算法比较结果具有一致性. 实验结果表明, 当混合参数 $\mu < \frac{1}{2}$ 时, LPA、Louvain、Infomap、SLPA 和 k-clique 在不同的 τ_1 和 τ_2 设置下均表现良好. 值得注意的是, DEMON 算法的性能随着 μ 的增加显著下降, 而 LEMON 在不同 μ 值下保持较强的鲁棒性. Louvain、Infomap 和 SLPA 在整体上表现最佳, 并且与在真实数据集上的实验结果保持一致, 表明这些算法在不同网络条件下的有效性. 这些发现强调了在不同网络特征和参数设定下选择合适算法的重要性.

总的来说, 本研究的贡献如下.

- 提供了一个基准测试, 以比较 OSS-DSN 中不同类型的社区检测算法.
- 在真实世界的 OSS-DSN 和合成网络上进行了实验, 并提供了相关的实证研究结果.
- 分析了网络特征如何影响算法评估结果, 并提出了一些算法和参数选择的建议.

本文第 2 节回顾相关研究, 并介绍网络构建方法和数据来源. 第 3 节详细描述本文研究方法和评估指标. 第 4 节探讨真实世界 OSS-DSN 的特性, 并分析其独特结构. 第 5 节利用不同方法和指标对真实 OSS-DSN 进行了深入分析, 从多个角度评估各种社区检测算法的性能. 第 6 节将比较扩展到具有相似特征的合成网络, 以进一步验证算法性能的稳定性. 第 7 节讨论该基准研究对软件工程发展的作用. 第 8 节总结面临的一些有效性威胁和研究结论, 并对本研究的贡献者表示感谢.

2 DSN 上的社区检测算法相关工作

本节总结了与本研究相关的背景知识及相关工作.

2.1 开发者社交网络

在开源软件社区中, 开发者协作网络的动态特性对于理解集体贡献如何塑造软件开发至关重要[16,17]. 这些网络由开发者之间的复杂交互构成, 促进了知识共享与创新. 社区检测在这些网络中的应用至关重要, 它能够识别紧密协作的开发者群体, 从而揭示 OSS 项目的结构和功能特性[7].

近年来, 研究人员采用多种方法, 包括社交网络分析 (social network analysis, SNA) 和机器学习技术, 以发现隐藏的协作模式, 并评估社区结构对项目成果的影响[18]. 此外, 对社区演化的研究表明, 开源软件生态系统具有适应性, 开发者可能会随着项目需求的变化而调整自身的角色和归属[4]. 理解这些协作模式不仅有助于加深对开源软件发展机制的理解, 还能够为促进有效协作和维持社区活跃度提供策略性指导.

本研究旨在综合现有关于开发者协作网络及其社区检测的研究成果, 重点分析其对软件开发实践和社区可持续性的影响.

2.2 社区检测算法

在复杂网络分析中, 社区检测是一个核心任务, 研究者已提出多种算法来应对这一挑战. Saha 等人[19]对各种社区检测算法进行了系统回顾和比较, 评估了它们在真实世界网络和合成网络上的性能. Jin 等人[20]全面介绍了目前已有的社区检测方法, 详细介绍了从传统的统计建模到深度学习方法的原理、分类及应用, 并探讨了目前社交网络的不同分类, 认为社交网络检测在不同的领域有不同的应用表现, 具有一定研究潜力.

针对社区检测算法, 许多研究者提出了不同的评估方法来验证它们的有效性. Pérez-Peló等人[21]和 Yang 等人[22]重点研究了重叠社区检测算法, 该算法允许节点同时属于多个群体, 该研究还评估了其识别的社区结构的有效性. Su 等人[23]提出一种社区检测算法比较框架, 该框架结合传统评估指标与新的度量方法, 以提供更全面的评估. Lancichinetti 等人[24]从理论角度为社区检测奠定了基础, 尽管该研究并未直接提供基准测试, 但其探讨了社区检测算法的限制和潜在能力, 研究人员可以利用该理论框架来评估算法与理论最优解之间的差距, 从而为算法改进提供方向.

综上, 尽管目前有许多研究者提出不同的社区检测算法、算法评估指标, 但仍然缺乏针对特殊网络如 OSS-DSN 的特性分析, 也缺乏一个全面评估重叠社区检测和非重叠社区检测的基准.

2.3 社区检测基准

随着复杂网络社区检测领域的不断发展, 研究人员提出了多种算法来应对这一挑战. 为有效评估和比较这些算法, 建立稳健的基准测试方法至关重要.

Granell 等人[11]利用多种真实世界网络和合成网络, 构建了一个通用的基准数据集, 以便研究人员测试其社区检测算法的性能. Wang 等人[12]提出了一个专门针对时间演化网络的评估框架, 用于比较处理动态社区结构的算法. Lancichinetti 等人[25]通过精心设计的基准图, 系统比较了多种社区检测方法, 以测试算法的准确性和计算效率.

社区检测的目标是在网络分析中识别节点间连接密度较高的群体, 以便更好地理解网络的整体结构. 社区检测算法的有效性通常通过基准数据集进行评估, 这些数据集为不同方法提供了统一的比较标准. 已有研究强调, 构建能够真实反映现实网络复杂性的基准至关重要, 包括不同的社区结构和节点特性[11,26].

传统基准测试 (如 LFR 模型) 虽然在评估算法性能方面发挥了重要作用, 但可能无法充分捕捉重叠社区和动

态网络变化的复杂性[6,27]. 目前研究者常用的合成基准模型有 GN 和 LFR, 其中 LFR 基准模型在 GN 基准模型的基础上, 引入了度分布和社区规模的幂律分布, 更贴近真实网络的特征; 也就是说 GN 基准仅是 LFR 在幂律指数趋近无穷时的极限特例, 由于 GN 结构较为简单, 所有节点的期望度相同且所有社区规模相当, 而复杂网络的特征是度的异构分布[28], 不符合真实网络的特征, 无法很好地区分社区检测算法之间的优劣, 而 LFR 提供了更广泛的测试场景. 其次, 在 LFR 基准模型中, 生成算法的复杂度与图的边数呈线性关系, 支持快速构建超大规模网络, 适合测试高性能算法. 基于上述原因, 我们使用 LFR 基准模型作为本文社区检测的基准模型, 与真实世界数据集上取得的结果进行比较, 并且给出了算法比较结果和参数设置建议.

此外, 包含异常值和噪声的合成基准的引入, 为评估社区检测方法的鲁棒性开辟了新的方向[27]. 本研究将探讨 OSS-DSN 的社区检测基准方法, 强调建立全面的评估框架, 以适应当代网络结构所带来的多样化挑战.

3 基准数据集的收集与方法学

本节介绍用于 OSS-DSN 社区检测基准测试的数据集、标准及实验协议.

3.1 真实世界数据的收集

OSS Compass (开源指南针) 是国内首个专注于开源生态健康评估的公共平台, 支持 GitHub、Gitee 等托管平台的开源项目评估, 由北京大学、南京大学、华为公司等学术和产业界机构共同建设, OSS Compass 开源项目清单具有良好的代表性. 基于该清单, 可以通过 GitHub 网站提供的官方 API 直接获取数据, 保证数据的真实性. 为了构建 OSS-DSN 并进行基准测试, 我们参考 OSS Compass 所关注的 GitHub 上各领域具有代表性的开源项目仓库清单. 然后从 GitHub 网站中爬取了上述 2 259 个仓库的 2023 年 5 月–2024 年 5 月间的所有 issue、PR 条目数据, 形成了本文的数据集.

现代开源软件开发中, 大量的自动化工具会参与 issue 和 PR 的管理, 产生一定量的“社区行为”, 但是本文定义的社区不包含这部分交流行为, 我们做了针对性的数据清洗, 对机器人用户做了甄别, 并对相关行为数据进行去除. 根据 GitHub 社区公约, bot 用户会由用户的 type 字段给出, 或者在登录名中包含 bot 标识, 因此根据这一规约排除绝大多数的 bot 账号.

此外, 由于本文数据仅限于近一年, 使得一些项目在切片时间内没有足够的社区行为数据, 对于后续的 OSS-DSN 分析意义不大. 例如 GitHub 上的项目 ChromeDevTools/devtools-frontend, 是一个 3.2k star 和 326 个贡献者的项目, 但其 2023 年 5 月–2024 年 5 月, 社区行为数据较少, 仅有 19 个 PR, 没有 issue, 此类项目不足以支持我们的建模和分析. 经过 bot 甄别、特殊项目排除后, 最终保留了 1 997 个项目. 该数据集包含 767 982 个 issue、PR, 有 1 180 364 个 comment, 总共涉及 277 530 个用户.

在开源软件项目中, 开发者可以通过多种方式进行交流, 例如电子邮件[29]、即时通讯工具[30]以及 issue 和 PR 讨论[31]等, 但实际上电子邮件和即时通讯工具对软件开发的贡献较难量化. 本研究参考文献 [31], 通过开源代码仓库中的 issue 和 PR 讨论记录开发者之间的交流关系. 具体而言, 若两位开发者参与了同一 issue 或 PR 的讨论, 则认为两者之间存在一条交流链接. 基于上述开发者交流方式定义, 我们为每个开源软件项目构建一个开发者社交网络, 表示为一个加权无向图 $G=<V,E,W>$, 其中 V 是节点集, 表示所有开发者; E 是边集, 表示开发者之间的交流链接; 而 W 是边权重, 表示开发者之间的交流频率, 即其共同参与的 issue 和 PR 数量. 本研究的网络构建方法与现有研究[32,33]保持一致.

3.2 研究方法

该基准方法主要包含真实 OSS-DSN 数据集、合成网络构造方法、社区检测算法集合、网络特性分析指标和算法比较评估指标. 数据集的收集在第 3.1 节说明, 合成网络构造方法在第 6 节详细说明, 本节将主要介绍算法集合、网络特性分析指标集合和算法比较评估指标集合.

(1) OSS-DSN 的特性分析

基于 issue 和 PR 构建的 OSS-DSN, 我们首先需要分析其网络特性, 并根据网络特性进行进一步比较和分析.

研究过程中用于分析的指标集合如表 1 所示, 这些指标能够帮助我们深入理解 OSS-DSN, 及其与其他社交网络不同的结构特征, 并依据指标值辅助 LFR 基准模型[25]生成合成网络.

表 1　用于分析 OSS-DSN 的指标及其描述

指标	描述
节点数量	包括OSS-DSN中包含的最大、最小和平均节点数量
节点度数	包括OSS-DSN中的最大、最小和平均节点度数, 以及幂律分布的参数. 节点度数的分布通常用于理解网络结构(例如, 无标度网络具有幂律度数分布)
团 (clique)	包括团的数量, 以及OSS-DSN中团的最大、最小、平均大小和大小的分布. 团大小的最小值为2
局部聚类系数	衡量一个节点的邻居之间彼此连接的程度 (即朋友圈的紧密程度)
中心性	衡量一个节点在整个网络中的重要性或影响力
边权重	边权重的最大、最小、平均值和分布
直径	网络中任意两个节点之间的最长最短路径, 它提供了网络大小的概念, 即节点之间的最大距离
平均路径长度	网络中所有节点对之间的最短路径长度的平均值
密度	网络中边的数量与可能边数量的比率, 它衡量网络的连接程度
网络鲁棒性	网络结构在移除节点或边时受到的影响, 它衡量网络对故障或攻击的恢复能力

(2) 社区检测算法

本研究评估了多种非重叠和重叠社区检测算法的性能, 这里选择使用较为广泛的算法, 算法清单见表 2.

表 2　具有不同性质的社区检测算法

社区是否重叠	设计原则	算法	时间复杂度	描述
非重叠算法	基于模块化优化	Girvan-Newman[34]	$O(m^2n)$	通过移除具有高边介数的边来检测社区, 基于模块化优化
		Louvain[35]	$O(n\log n)$	使用模块化优化的快速、分层社区检测算法
		CNM[19]	$O(n\log^2 n)$	通过贪婪方法直接最大化模块化, 适用于较小的网络
	基于概率模型	EM[36]	$O(tkn^2)$	使用期望最大化算法估计每个节点的社区成员概率, 适合检测混合社区结构
	基于图聚合	AGDL[37]	$O(n^2)$	该算法使用入度和出度来表征两个集群之间的亲和力
	基于传播	LPA[38]	$O(m)$	通过标签传播迭代更新节点标签, 每个节点最终加入其最频繁邻居标签的社区
	基于谱聚类	Eigenvector[19]	$O(n^3)$	基于特征值和特征向量的谱聚类方法, 通过分析网络的邻接矩阵或拉普拉斯矩阵来划分社区
重叠算法	基于密度	k-clique[39]	$O(3^{n/3})$	基于寻找k-clique并扩展这些密集区域的社区检测方法, 用于重叠社区检测
	基于扩展的标签传播	SLPA[40]	$O(tm)$	一种扩展的标签传播算法, 允许节点成为多个社区的成员
		DEMON[41]	$O(nd^2)$	使用标签传播和民主投票, 适合检测重叠社区结构
	基于局部扩展	LEMON[42]	$O(km)$	一种用于多层网络中社区检测的多层模块化优化算法
重叠/非重叠算法	基于信息论	Infomap[43]	$O(m)$	一种信息论社区检测算法, 通过最小化随机游走信息流的描述长度来识别社区结构

在算法选择的过程中, 我们重点考察了在开源工具库 NetworkX (https://networkx.org/) 和 CDlib (https://cdlib.readthedocs.io/en/latest/) 中提供实现的社区检测算法. 这些库提供了经过测试的代码, 涵盖了多种社区检测算法, 是目前较受认可且使用广泛的算法集合. 此外, 我们已将本研究使用的代码和数据公开, 相关资源可通过公开链接 (https://github.com/sanglide/benchmark_community_detection_in_OSS) 获取.

(3) 评估指标

社区检测是复杂网络分析中的核心问题, 其目标是将网络中的节点划分为若干内部连接紧密、外部连接稀疏的群体, 这样的群体叫作社区. 由于社区的定义本身具有模糊性, 研究者们提出了多种指标[44]来量化社区的质量.

这些指标大致分为两类: (1) 社区发现指标: 用于指导或优化社区检测算法, 衡量划分的合理性. (2) 社区评估指标: 在已知真实社区结构 (ground-truth) 时, 评估检测结果与真实结构的匹配程度. 社区检测的评估涉及多种指标, 研究人员已提出多种度量方法来衡量社区检测结果的优劣[4]. 本研究采用表 3 所示指标对社区检测结果进行评估.

表 3　8 个无真实值的指标和 9 个有真实值的指标

有无真实值	社区是否重叠	设计原则	指标	描述
有	重叠算法/非重叠算法	基于模块化	modularity[45,46]	衡量社区内部边的密度与随机网络相比的情况, 广泛用于评估社区结构的质量
	非重叠算法	基于模块化	community score[47]	一种模块度的变体, 根据社区内部和外部边的密度对社区进行评分
		内聚性	spart[48]	结合社区内节点互连的影响, 增强对节点级别分区强度的评估
		统计显著性	significance[49]	评估检测到的社区在统计上的显著性, 通常将观察的社区结构与随机配置进行比较
			surprise[50]	基于信息论和统计显著性, 衡量社区结构与随机零模型相比的意外程度
		一致性	*permanence*[51]	通过考虑社区内连接与社区间连接, 评估节点在社区内的稳定性
		基于模块化	communitude[52]	社区度的概念引入了一种基于稳健统计原则的网络社区质量函数
	重叠算法	一致性	flex[53]	flex 旨在优化社区内的内部边和局部聚类系数
无	重叠算法/非重叠算法	准确性	*F*1-score[22]	通过平衡社区分配的精确率和召回率来评估聚类质量
			NMI[54]	衡量检测到的社区结构与真实社区结构之间的相互依赖性, 适用于非重叠社区
	非重叠算法	一致性	purity[55]	通过将检测到的社区与真实标签进行比较, 衡量每个社区中节点的同质性
		准确性	*F*-measure[56]	通过平衡社区分配中的精确率和召回率来评估聚类质量
			RI[57]	通过测量匹配节点对的比例来比较两个聚类间的相似性
			ARI[58]	对兰德指数进行随机调整, 提供更准确的相似性度量
		成对相似性	VI[59]	基于共享和独特信息, 衡量两个社区结构之间的距离
	重叠算法	内聚性	OI[60]	通过评估社区之间的共同成员, 量化社区之间的重叠程度

在真实世界的 OSS-DSN 中, 缺乏关于“真实”社区结构的标准答案 (ground-truth), 也就是说, 真实世界的网络中, 无法根据已有的定义明确、唯一地划分社区, 因此对于算法挖掘的社区结果没有客观的比较对象. 为了解决这一问题, 我们提出以下两种方法: (1) 基于算法一致性: 对于真实世界的 OSS-DSN, 我们将不同社区检测算法的结果相互作为参考, 并评估不同算法之间的相似度. 如果一个算法与其他算法均具有较高一致性, 则说明该算法具有较高的适用性. (2) 基于合成网络实验: 我们依据第 4 节获得的真实 OSS-DSN 特性, 使用 LFR 基准模型[25]生成合成网络, 并在其中进行社区检测实验, 以提供可比较的具有真实值的结果. 其中真实值的指标用来表征社区检测结果的质量, 该指标说明了社区检测结果与真实结果的相似程度, 越相似则说明社区检测结果准确性越高.

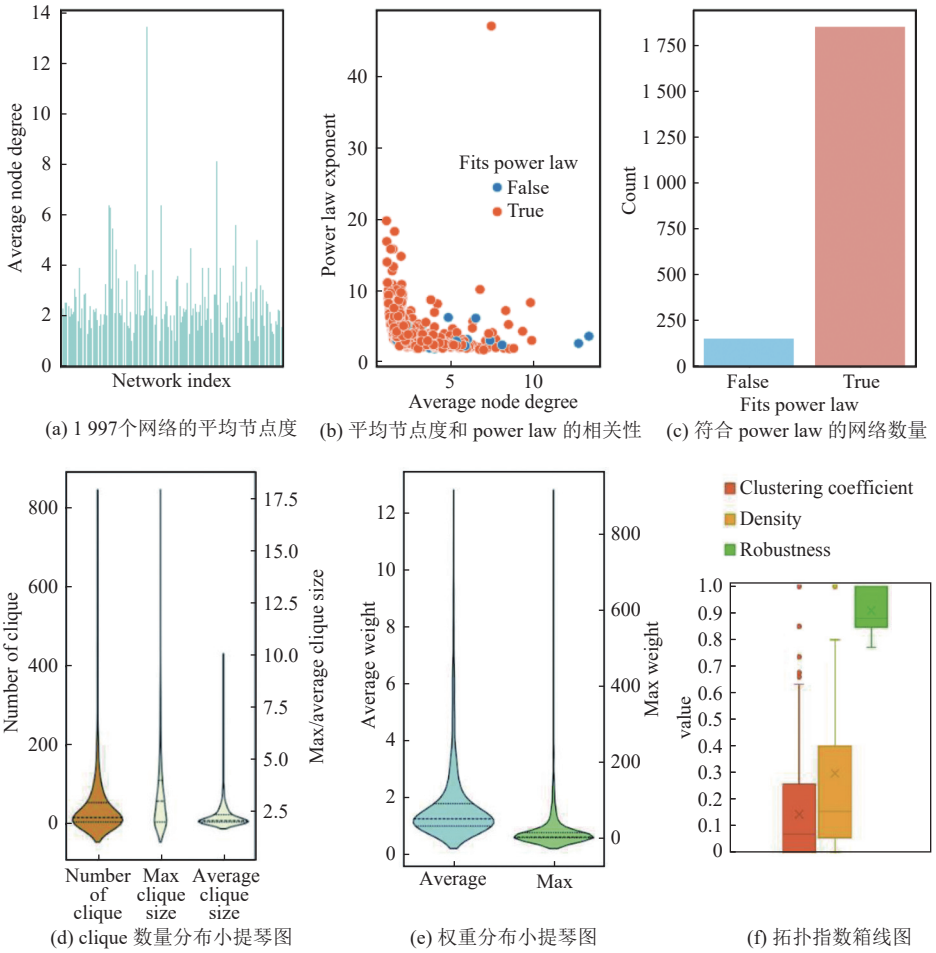
4　OSS-DSN 的特性分析

本节通过表 1 中所列举的指标, 分析真实世界 OSS-DSN 的特性来回答研究问题 RQ1, 并利用这些特性在第 6 节中对构建的合成网络进行评估.

4.1　OSS-DSN 的拓扑特性

如图 1(c) 符合 power law 网络数量的柱状图所示, 开发者社交网络中的平均度数集中在 0–6 之间, 可以看出在大多数运行良好的项目中, 开发者社交网络的节点度数分布遵循幂律分布, 表明其具有无标度特性, 即少数节点高度连接, 而大多数节点连接较少. 节点度数分布满足幂律分布这一性质决定了我们可以使用 LFR 基准网络模拟

开发者社交网络, 两者具有一致的拓扑性质. 文献 [2] 表明开源开发逐渐形成了类似“洋葱”结构的“小核心-大外围”的群体参与组织结构, 这与我们的分析结果相吻合. 这一特性表明 OSS-DSN 的网络拓扑结构具有较强的异质性、健壮性, 并表现出小世界效应. 同时, 这一特性导致网络在演化过程中具有脆弱性, 脆弱性是指网络中核心节点的演化可能会对社区结构产生很大的影响, 如核心开发者的离开可能会导致社区崩溃. 此外, 该特性决定了我们可以利用 LFR 合成网络模拟 OSS-DSN, 其详细构造将在第 6 节中进一步说明.



(a) 1 997个网络的平均节点度 (b) 平均节点度和 power law 的相关性 (c) 符合 power law 的网络数量

(d) clique 数量分布小提琴图 (e) 权重分布小提琴图 (f) 拓扑指数箱线图

图 1 节点度的分布和网络特性

如图 1(f) 拓扑指数箱线图所示, 聚类系数的分布表明, 大多数网络节点具有适中的局部聚类特性, 但未达到高度密集社交网络中的紧密互联程度. 这意味着, 尽管 OSS-DSN 具有一定的局部社区结构, 但这些社区之间的连接并不特别紧密. 然而, 在少部分大型网络中, 聚类系数较高, 表明存在清晰的社区结构. 在网络密度方面, 虽然节点间存在相当数量的连接, 但仍然存在大量缺失的潜在连接, 表明 OSS-DSN 结构并不高度紧密. 此外, 网络的健壮性通常被解释为其在节点或边失效时保持功能 (如信息传播、连通性) 的能力. 实验测得 OSS-DSN 的健壮性指标超过 0.9, 表明其容错能力较强, 即使受到较大干扰, 也能维持整体结构和性能.

数据集中的社区规模以中等规模社区为主, 同时包含较小比例的大型社区. 较小或中等规模的社区可能缺乏正式的组织化沟通, 而较大的社区通常具有更成熟的组织结构和交流模式. 这与实际软件开发过程较为类似, 小型规模团队的软件开发中成员组织往往较为扁平、沟通较为高效, 而大型软件团队组织的开发往往由公司主导, 同

时由于团队成员较多, 往往需要更为严密的组织规范.

4.2 网络中心性分析

中心性是社交网络分析中的核心概念[61], 用于量化网络中节点的重要性或影响力. 其重要性体现在可以帮助人们加深对网络性质的认知, 如识别关键节点、帮助识别网络中最重要或最具影响力的节点 (如社交网络中的意见领袖、交通网络中的枢纽), 或者揭示网络的整体拓扑特征 (如层级性、集中性), 帮助我们理解网络结构. 常见的中心性度量指标有: 接近中心性 (closeness centrality), 衡量一个节点到其他所有节点的平均距离有多近, 常代表网络中信息的传播速度; 介数中心性 (betweenness centrality), 衡量一个节点是否在众多其他节点的最短路径上, 用于识别节点是否是网络中的关键枢纽; 特征向量中心性 (eigenvector centrality), 度量节点邻居的重要性和数量, 即影响力大的节点, 其邻居影响力也大. 网络中心性的影响因素较多, 节点数量和路径长度是其中常见的两个因素, 如接近中心性直接依赖于节点到其他节点的平均最短路径. 平均路径长度越小的网络, 节点的接近中心性通常越高.

后文图 2 为几种不同的中心性、最短路径长度 (shortest path length) 和节点数的统计图. 对角线部分表示各变量的分布, 而非对角线部分则表示变量之间的相关性. R^2 (coefficient of determination, 决定系数) 是统计学中用于衡量回归模型拟合优度的指标, 表示因变量 (Y) 的变异性能被自变量 (X) 解释的比例. R^2 的取值范围为 [0, 1], $R^2 = 1$ 表示模型完美拟合数据, 所有数据点均落在回归线上 (残差为 0), 反之 $R^2 = 0$ 表示模型完全无法解释因变量的变异 (等同于用均值预测 Y). 从后文图 2 可以观察到, OSS-DSN 中存在明显的中心用户. 特征向量中心性与节点数之间呈指数关系 ($R^2 = 0.96$), 表明 OSS-DSN 具有显著的层级结构. 少数关键节点在网络连通性和影响力方面发挥了重要作用.

在同一社区内, 节点的中心性通常较高, 因为它们之间的连接更密集. 此外, 随着节点数量的增加, 社区中心节点的中心性可能呈指数增长. 同时, 特征向量中心性、介数中心性与平均路径长度之间呈线性关系. 具体来说, 在平均路径较短的网络中, 高特征向量中心性表明高影响力节点的自身和邻居都很重要, 形成了“核心-外围”结构, 而介数中心性高则表现为网络连接性较高, 没有绝对的必经节点; 在路径较长的网络中, 低特征向量中心性表明网络的中心性集中于少数顶层节点, 底层节点影响力衰减较快, 而较低的介数中心性则说明少数节点会成为关键瓶颈, 删除它们会导致大幅增加平均路径. 这表明中心节点在增强网络连通性和提高通信效率方面起到关键作用.

对于 OSS-DSN 而言, 用户之间的交流往往是直接交流而不是间接交流, 如在同一 issue 或 PR 下直接发起评论, 因此网络常呈现出介数中心性较高的特性; 同时, 并不是所有的用户活跃程度都相同, 关键的项目开发者或管理者往往展现出较高的活跃度, 承担了更多的项目管理和分发任务, 因此 OSS-DSN 常呈现出“核心-外围”结构, 中心性集中于这些关键用户, 关键的项目开发者或管理者的离职会导致沟通成本的大幅增加.

4.3 典型网络现象对社区检测的影响

在复杂网络中有一些网络现象会对社区检测结果产生重要影响, 例如渗流转变和分辨率限制[39,62]. 我们针对这两个相对普遍且重要的现象, 分析它们对开源社区中的社区检测算法的影响.

4.3.1 渗流转变现象

渗流现象是指在随机网络或图中, 当连接概率达到某一临界阈值时, 系统中突然出现一个巨连通分量的现象. 具体来说, 对于一个包含 m 个团 (clique) 的网络 (其中每个团为 k 个节点组成的完全子图), 当边连接的概率 p 与渗流阈值 $P_c(k) = [(k-1)N]^{-1/(k-1)}$ 满足特定条件时, 会发生渗流转变 (percolation transition)[39].

在社区检测算法中, k 常是一个可设置的参数值, 如果该值设置不当, 可能会极大地影响社区检测算法的检测结果. 图 3 展示了 k 值对特定网络中社区规模的影响, 为了更好地看清楚节点和边, 我们选择了一个规模较小的项目, 更大规模网络的实例见第 6.3 节. 在该示例中, 网络的其他属性相同, 节点数量 $n = 26$, 最大团的大小 $m = 23$, 边概率 $p = 0.163$, 自然划分模块度 $Q_{\text{single}} = 0.958$, 连续团块对划分的模块度 $Q_{\text{pairs}} = 0.921$, 但是当 k 取不同值时, 社区检测得到的团块产生了差异. 当 k 增加时, $P_c(k)$ 的值也相应增大, 这意味着更大规模的团需要更高的边连接概率才能形成完整的子图.

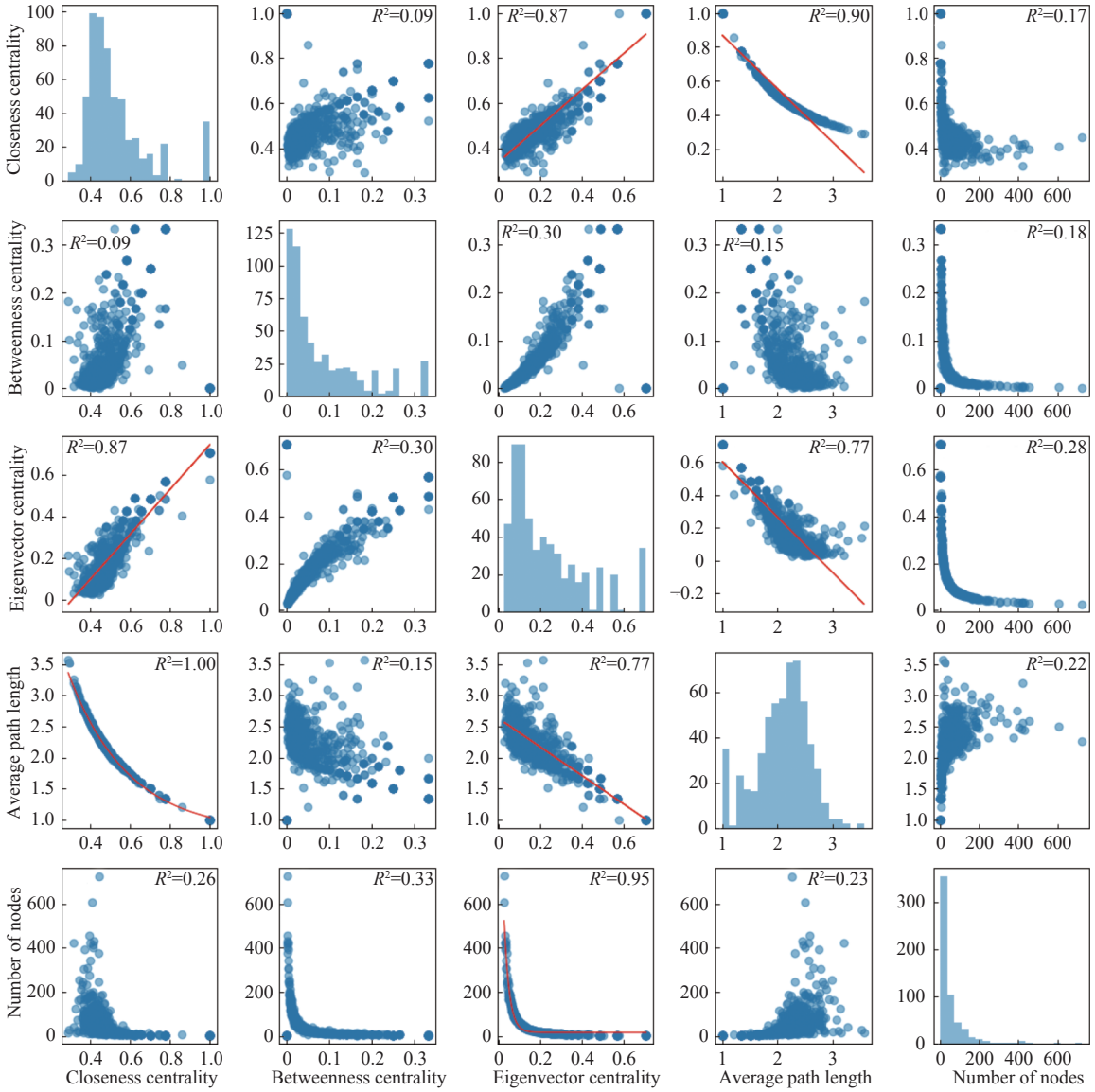
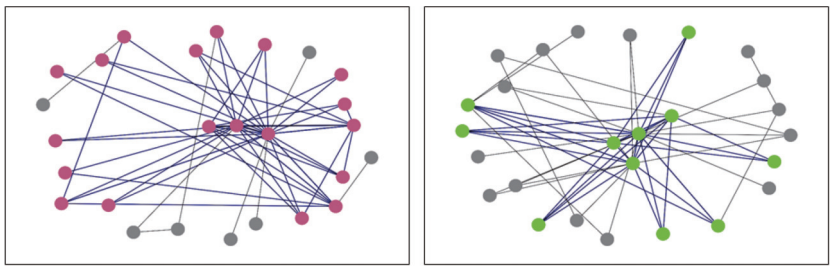


图 2　中心性、平均路径长度和节点数量的统计图 (对角线为各变量的分布图)



(a) $P_c(3)=0.139<p$, 最大的社区拥有19个节点　(b) $P_c(4)=0.234>p$, 最大的社区拥有11个节点

图 3　项目 MovingBlocks/Terasology 的 OSS-DSN 中的 1 个团块的示例

在真实世界 OSS-DSN 的实验验证中, 渗流转变的临界点分别为: $k=2$ (1 797 个项目)、$k=3$ (148 个项目)、$k=4$ (17 个项目) 和 $k=5$ (35 个项目). 为了在计算效率与防止渗流转变之间取得平衡, 我们推荐使用 $k=4$ 作为最优参数. 除了团的大小, 团的数量与图中节点数之间的关系也会影响网络特性, 其中一个典型现象是分辨率限制 (resolution limit).

4.3.2 分辨率限制现象

分辨率限制指在使用模块度优化检测社区时, 算法无法识别小于约 $\sqrt{L}$ (L 为网络总边数) 的社区, 导致小社区被错误合并的现象. 这是模块度函数的固有缺陷, 因其依赖全局统计量而忽略局部结构. 由定义可知, 这两个社区检测的典型错误现象与社区检测算法本身的特性和被检测网络的拓扑特性相关.

分辨率限制是社区检测中的一个重要挑战, 尤其是在使用基于模块度优化的算法时[24,54,62]. 由于模块度优化倾向于合并较小的社区以获得更高的模块度得分, 即使这些小社区在拓扑上具有清晰的界限, 也可能被错误合并. 这种限制受网络整体规模及模块间互连程度的影响, 导致在合成网络和真实网络中均难以识别较小的社区. 通过比较 Q_{single} 和 Q_{pairs}, 可以判断模块度优化过程中是否会错误合并本应独立的社区. 具体公式如下:

$$Q_{\text{single}} = 1 - \frac{2}{m(m-1)+2} - \frac{1}{n} \tag{1}$$

$$Q_{\text{pairs}} = 1 - \frac{1}{m(m-1)+2} - \frac{2}{n} \tag{2}$$

根据公式 (1) 和 (2), 为了避免分辨率限制问题[62], 必须保证 $Q_{\text{single}} > Q_{\text{pairs}}$, 即满足 $m(m-1)+2 > n$. 在 1 997 个项目中, 1 874 个项目满足 $Q_{\text{single}} > Q_{\text{pairs}}$, 表明其社区检测不受分辨率限制影响, 而 53 个项目因节点数较少而不具有统计显著性. 实验结果表明, 基于本研究构造的 OSS-DSN 在进行社区检测时, 分辨率限制的影响较小. 进一步实验还表明, OSS-DSN 受分辨率限制影响较小, 不同模块度参数对社区划分的影响不显著.

4.4 研究问题 RQ1 的回答

综上所述, 我们回答研究问题 RQ1. OSS-DSN 的项目级社交网络与网站级社交网络存在显著差异, 并影响社区检测算法的表现. 这些网络由小规模、紧密连接的社区和中等规模、松散连接的社区组成. 不同于大规模去中心化网络, OSS-DSN 具有紧凑、中心化的小世界网络特性, 表现出更强的健壮性. 此外, 我们针对渗流现象和分辨率限制等挑战提出了解决方案, 并提供了合适的参数建议. 我们的数据集在网络特性上比合成或特定规模的网络更具多样性, 涵盖更广泛的小型和中型网络. 因此, 在 OSS-DSN 上评估社区检测算法时, 应使用针对特定上下文的数据集, 而非通用数据集, 以获得最优的检测效果.

5 真实世界 OSS-DSN 结果分析

在本节中, 我们通过评估社区检测算法在真实世界 OSS-DSN 上的性能来回答 RQ2.

如表 4 所示, 我们对比了不同算法在真实世界 OSS-DSN 上的计算效率. 算法运行时间与其时间复杂度一致; 然而, 在实际网络中, 每次通信都会增加一条边, 从而导致 m 值增大, 使 Girvan-Newman 算法的效率降低. 此外, 内部标准差 (inner standard deviation, 简写为 Inner Std.) 表示在计算整个数据集的平均值之前, 每个项目在多次运行中的标准偏差, 它受到多次运行的随机性影响. 而标准差 (standard deviation, 简写为 Std.) 表示每个项目在多次运行后的平均运行时间标准偏差, 反映了样本对运行时间的影响.

表 4 不同社区检测算法的性能比较

算法	时间 (ms)	Std.	Inner Std.
Girvan-Newman[34]	9 323.96	115 764.83	82.06
Louvain[35]	4.19	11.68	0.30
CNM[19]	55.75	407.38	1.98
EM[36]	799.36	14 140.98	34.78
AGDL[37]	13.26	119.89	1.42

表 4　不同社区检测算法的性能比较 (续)

算法	时间 (ms)	Std.	Inner Std.
LPA[38]	2.38	5.14	0.16
Eigenvector[19]	31.00	484.89	1.55
k-clique[39]	25.67	466.01	1.10
SLPA[40]	23.11	60.98	2.30
DEMON[41]	21.57	61.40	2.59
LEMON[42]	236.10	285.73	21.92
Infomap[43]	19.47	17.83	4.27

5.1　非重叠社区检测

5.1.1　基于模块度的评估

模块度 (modularity) 及其他评估指标共同构成了对非重叠社区检测算法的比较结论.

如图 4(a) 所示, 社区得分 (community score) 表明 Infomap 和 LPA 可以检测出良好的社区结构, 而 GN 和 AGDL 在这方面较弱. Louvain、Eigenvector 和 CNM 的性能相对稳定 (尽管其最高值低于 Infomap 和 LPA, 但箱线图中的最大值和 Q3 值较高). Surprise、significance 和 communitude 指标的结果与模块度一致. 根据这些指标, 我们发现 EM 算法的表现较差.

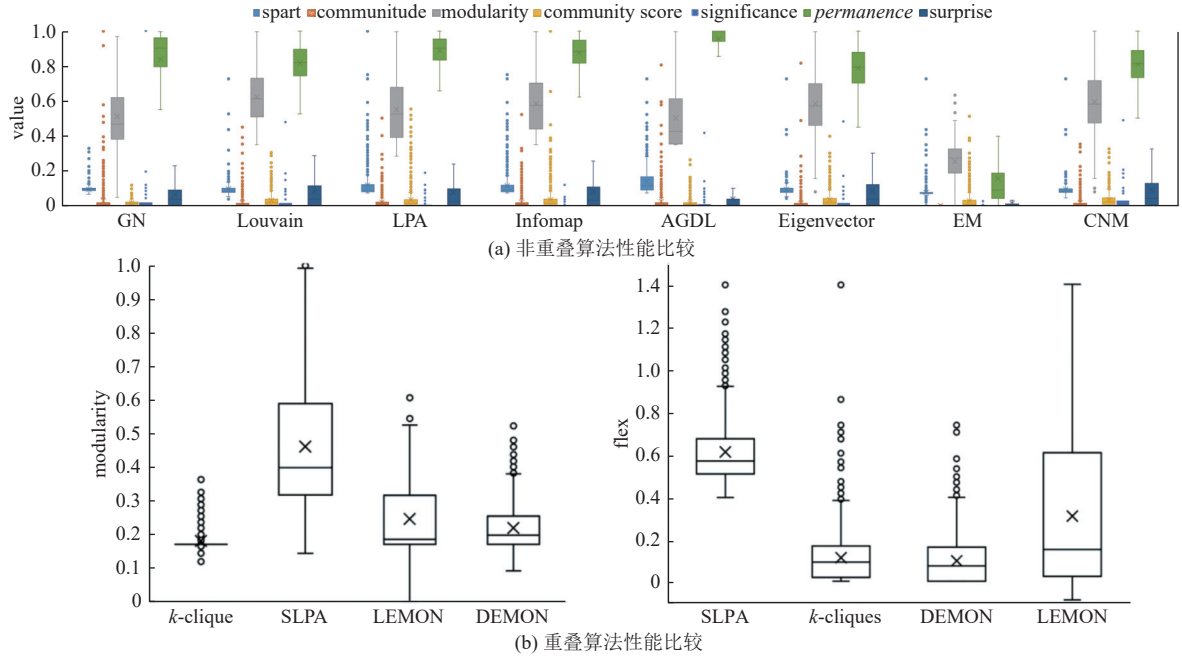


图 4　不同社区检测算法的性能比较, 使用无真实值的指标

从模块度值的分析来看, Louvain 和 Infomap 具有最高的模块度得分, 是表现最好的算法, 而 EM 由于模块度得分较低, 是最不理想的算法. Spart 是一个基于节点的度量指标, 它考虑每个节点及其邻居对社区划分质量的贡献. AGDL、Infomap 和 LPA 能够更好地考虑节点的重要性, 并获得更紧密的社区, 这对揭示项目核心团队具有指导意义. Louvain、CNM 和 Eigenvector 的表现略逊一筹, 而 GN 和 EM 表现最差.

permanence 指标能够衡量节点在社区内的连接紧密度, 当节点 v 没有邻居且属于单独的社区时, 其值定义为 $permanence(v) = -1$. AGDL、Infomap、LPA 和 GN 在此指标上表现良好, 其次是 Louvain、Eigenvector 和 CNM,

而 EM 则倾向于发现许多无关紧要的小型社区.

此外, AGDL 算法同时具有高 *permanence* 和低模块度, 表明尽管单个节点在其所属社区内连接紧密, 但社区之间的界限可能不够清晰. 这表明社区可能存在一定程度的重叠, 或者社区边界不是严格定义的. 这也突出了仅使用模块度作为社区检测评价标准的局限性, 因为它可能无法捕捉 *permanence* 反映的细粒度节点级社区归属信息.

5.1.2 跨算法验证

表 5 描述了非重叠社区检测算法的验证结果, 每一项由指标数值组成, 括号里的数值则是指标对应的标准差, 指标数值越高说明算法之间的社区一致性越强. 根据表 5, AGDL 和 EM 的指标值相对较低, 但总体而言, 非重叠社区检测算法的结果表现出较强的一致性. 特别是 Louvain 与 CNM、LPA 和 Eigenvector 等算法的一致性较高.

表 5 非重叠社区检测的跨算法验证结果

算法1	算法2	purity	*F*-measure	*F*1-score	RI	ARI	NMI	VI
GN	Louvain	0.63 (0.21)	0.72 (0.15)	0.80 (0.18)	0.59 (0.21)	0.29 (0.25)	0.51 (0.23)	1.76 (0.80)
	LPA	0.81 (0.19)	0.84 (0.13)	0.81 (0.19)	0.73 (0.18)	0.41 (0.26)	0.56 (0.24)	1.36 (1.04)
	Infomap	0.75 (0.21)	0.79 (0.15)	0.83 (0.18)	0.65 (0.21)	0.35 (0.26)	0.52 (0.25)	1.66 (1.13)
	AGDL	0.93 (0.12)	0.84 (0.12)	0.55 (0.21)	0.67 (0.19)	0.23 (0.23)	0.35 (0.28)	1.08 (0.53)
	Eigenvector	0.61 (0.21)	0.71 (0.15)	0.76 (0.20)	0.59 (0.23)	0.30 (0.28)	0.51 (0.25)	1.83 (0.90)
	EM	0.56 (0.21)	0.63 (0.15)	0.33 (0.14)	0.53 (0.20)	0.16 (0.19)	0.29 (0.17)	2.62 (1.01)
	CNM	0.58 (0.22)	0.69 (0.17)	0.80 (0.19)	0.59 (0.24)	0.31 (0.28)	0.53 (0.25)	1.79 (0.88)
Louvain	LPA	0.87 (0.13)	0.76 (0.13)	0.69 (0.18)	0.68 (0.21)	0.35 (0.26)	0.58 (0.26)	1.60 (0.77)
	Infomap	0.89 (0.12)	0.82 (0.12)	0.80 (0.16)	0.79 (0.18)	0.52 (0.29)	0.71 (0.24)	1.19 (0.81)
	AGDL	0.96 (0.07)	0.67 (0.13)	0.42 (0.21)	0.48 (0.20)	0.14 (0.19)	0.30 (0.26)	2.07 (0.76)
	Eigenvector	0.77 (0.15)	0.79 (0.13)	0.71 (0.20)	0.83 (0.12)	0.53 (0.25)	0.68 (0.20)	1.60 (1.04)
	EM	0.66 (0.15)	0.65 (0.12)	0.34 (0.14)	0.76 (0.12)	0.32 (0.20)	0.45 (0.16)	2.64 (0.96)
	CNM	0.79 (0.15)	0.82 (0.13)	0.79 (0.17)	0.86 (0.11)	0.59 (0.25)	0.74 (0.19)	1.29 (0.92)
LPA	Infomap	0.86 (0.14)	0.86 (0.12)	0.85 (0.16)	0.79 (0.17)	0.52 (0.29)	0.70 (0.23)	1.15 (0.91)
	AGDL	0.95 (0.10)	0.81 (0.14)	0.40 (0.23)	0.65 (0.20)	0.21 (0.26)	0.35 (0.29)	1.61 (1.11)
	Eigenvector	0.67 (0.20)	0.72 (0.13)	0.61 (0.22)	0.64 (0.22)	0.30 (0.25)	0.52 (0.27)	1.85 (0.83)
	EM	0.62 (0.18)	0.66 (0.13)	0.32 (0.18)	0.63 (0.20)	0.21 (0.20)	0.36 (0.21)	2.54 (1.01)
	CNM	0.65 (0.23)	0.71 (0.15)	0.65 (0.21)	0.64 (0.24)	0.31 (0.25)	0.55 (0.27)	1.77 (0.78)
Infomap	AGDL	0.96 (0.08)	0.78 (0.15)	0.40 (0.27)	0.60 (0.21)	0.21 (0.27)	0.34 (0.29)	1.87 (1.18)
	Eigenvector	0.69 (0.18)	0.74 (0.13)	0.63 (0.24)	0.71 (0.19)	0.36 (0.27)	0.59 (0.24)	1.82 (1.00)
	EM	0.64 (0.16)	0.66 (0.13)	0.32 (0.17)	0.70 (0.16)	0.26 (0.22)	0.42 (0.20)	2.52 (1.03)
	CNM	0.70 (0.20)	0.75 (0.14)	0.67 (0.23)	0.72 (0.21)	0.39 (0.28)	0.63 (0.25)	1.62 (0.89)
AGDL	Eigenvector	0.49 (0.15)	0.63 (0.13)	0.74 (0.16)	0.44 (0.19)	0.12 (0.16)	0.27 (0.25)	2.24 (0.76)
	EM	0.51 (0.19)	0.62 (0.17)	0.43 (0.24)	0.46 (0.19)	0.06 (0.12)	0.16 (0.17)	2.58 (1.17)
	CNM	0.45 (0.16)	0.59 (0.14)	0.73 (0.16)	0.42 (0.20)	0.11 (0.15)	0.27 (0.24)	2.31 (0.71)
Eigenvector	EM	0.69 (0.15)	0.64 (0.11)	0.35 (0.13)	0.76 (0.13)	0.33 (0.21)	0.47 (0.16)	2.63 (0.95)
	CNM	0.84 (0.14)	0.83 (0.13)	0.75 (0.20)	0.90 (0.09)	0.66 (0.25)	0.76 (0.18)	1.30 (1.04)
EM	CNM	0.62 (0.12)	0.65 (0.11)	0.41 (0.13)	0.77 (0.13)	0.35 (0.21)	0.48 (0.16)	2.58 (0.89)

5.2 重叠社区检测

5.2.1 基于模块度的评估

箱线图中, Q1、Q2、Q3、Q4 (即四分位数) 将数据从小到大等分为 4 部分, 分别代表 25%、50%、75%、100% 分位点的值; min 和 max 则代表数据的最小值和最大值 (有时特指在合理范围内的最小与最大观测值). 从图 4(b) 的箱线图中可以看出, 每个算法的度量指标具有不同的分布, 这些度量值越高, 证明该算法的表现越强.

Flex 是一个试图在社区内最大化内部边数量的指标, 同时考虑每个社区的局部聚类系数. SLPA 具有较高的 Q1、Q2 和 Min 值, 表现优异, 而 LEMON 也有良好表现, 而 *k*-clique 和 DEMON 的表现略逊一筹. 该指标反映了社区检测算法的灵活性, 灵活性指的是算法适应不同网络结构和规模的能力. 一个高灵活性的算法可以有效处理

从稀疏到密集的各种类型的网络, 并适应不同的社区规模和结构.

模块度是衡量社交网络中社区划分质量的指标, 量化了社区内部连接相对于随机连接的紧密程度。较为平衡的模块度表示算法在最大化模块度和保持合理社区结构之间取得了良好的平衡. 这意味着算法不仅能够识别内部紧密连接的社区, 还能确保这些社区与其他社区良好分隔, 而不会过度拟合噪声或生成过度碎片化的划分. 在图 4(b) 所示 4 种算法中, SLPA 拥有最佳的社区划分质量, 而 *k*-clique 的划分质量则略低于其他算法。

5.2.2 跨算法验证

表 6 描述了重叠社区检测算法的验证结果, 由指标数值组及括号里的数值 (指标对应的标准差) 组成. 表中的指标数值越高说明算法之间的社区一致性越强. 根据表 6, 除 DEMON 和 *k*-clique 算法发现的社区具有一定的相似性外, 其他重叠社区检测算法之间的一致性较低.

表 6 重叠社区检测的跨算法验证结果

算法1	算法2	*F*1-score	OI	NMI_MGH	NMI_LFK
k-clique	SLPA	0.35 (0.24)	0.04 (0.07)	0.03 (0.05)	0.04 (0.07)
	LEMON	0.36 (0.23)	0.02 (0.06)	0.01 (0.04)	0.02 (0.06)
	DEMON	0.90 (0.20)	0.70 (0.32)	0.58 (0.37)	0.70 (0.32)
SLPA	LEMON	0.41 (0.28)	0.11 (0.26)	0.10 (0.26)	0.11 (0.26)
	DEMON	0.39 (0.22)	0.04 (0.07)	0.02 (0.05)	0.04 (0.07)
LEMON	DEMON	0.42 (0.22)	0.02 (0.07)	0.01 (0.04)	0.02 (0.07)

原本的 NMI 是衡量两个聚类结果整体一致性的经典指标, NMI_MGH 和 NMI_LGH 是基于局部或全局结构信息优化的归一化互信息变体. 它们通过局部或全局视角对结构信息进行加权计算, 从而与原 NMI 形成替代或补充关系. 尤其在重叠算法的评估中, NMI_MGH 和 NMI_LGH 更加有效. 尽管存在一些 NMI 适配方法用于评估重叠算法[12], 但它主要是为非重叠社区检测算法设计的, 其评估结果可能对重叠部分的规模较为敏感.

5.3 研究结论 (RQ2 的解答)

综上所述, 我们对比了不同的社区检测算法, 并对 OSS-DSN 场景提出了具体建议.

● 对于非重叠社区检测算法, Louvain 和 Infomap 表现最佳, 能够稳定地获得较高的模块度, 并检测出紧密连接的社区, 与 CNM、LPA 和 Eigenvector 方法相比具有较强的一致性. 相比之下, AGDL 和 EM 表现较差, 尤其是 AGDL 在区分社区时存在一定困难.

● 对于重叠社区检测算法, SLPA 通过最大化内部边的数量和局部聚类系数表现出色, 而 *k*-clique 和 LEMON 也有良好表现. 然而, 重叠社区检测算法整体的一致性较低, DEMON 和 *k*-clique 由于算法假设不同, 在检测社区时出现较大的重叠区域不确定性.

我们的研究扩展了不同社区检测算法的比较范围, 并针对 OSS-DSN 提供算法选择建议.

6 合成网络上的实验结果

在本节中, 我们使用具有真实值社区结构的合成网络来回答 RQ3.

6.1 构建合成网络

基于第 4 节中描述的 OSS-DSN 特征, 本文扩展了 LFR 基准测试[24], 以生成具有真实社区结构的 OSS-DSN 合成网络, 帮助基准使用者更方便地生成模拟数据用以测试社区检测算法. LFR 基准测试是植入式 ℓ- 分区模型 (planted ℓ- partition model) 的一个特例. ℓ- 分区模型[26,63]将图划分为 ℓ- 个包含 k 个顶点的组. 然而, 在 LFR 模型[54]中, 群体大小是可变的, 并且节点的度数也不同. 这样的设置会更贴合真实社区的情况. 节点的度数服从幂律分布, 其指数为 τ_1, 而社区的大小也服从幂律分布, 其指数为 τ_2, 这一特性也符合第 4 节中对 OSS-DSN 的特性统计.

我们根据图 1 中的实验结果选定 τ_1, 并使用符合幂律分布的社区检测算法计算 τ_2. 生成 LFR 需要确定的另一个重要参数是混合参数 μ, 如公式 (3) 所示, 定义为节点的外部度数 k_i^{out} (即它与社区外部节点的连接数) 与其总度

数 (总连接数) 的比值.

$$\mu = \frac{k_i^{\text{out}}}{k_i^{\text{in}} + k_i^{\text{out}}} \tag{3}$$

根据定义, 当参数 $\mu < 1 - \frac{n_c^{\max}}{N}$ 时, 群体可被视为社区, 其中 $n_c^{\max}$ 是最大社区的大小, N 是网络中的总节点数. 此外, Lancichinetti 等人[54]建议 μ 的典型取值范围应小于 $\frac{1}{2}$. 当 $\frac{1}{2} < \mu < \frac{3}{4}$ 时, 社区仍可能存在, 但检测难度较大. 然而, 当 $n_c \ll N$ 时, μ 的极限值可能趋近于 1.

LFR 合成网络的每个节点拥有同样的 μ 值, 但是对于 OSS-DSN 而言, 由于每个节点的入度和出度不相同, 因此一个网络中的各个节点拥有各自不同的 μ 值. 我们对 OSS-DSN 的特征进行了统计分析, 并基于其值的分布生成了一组具有代表性的 LFR 网络. 由于在真实网络中, 每个节点的 μ 值各不相同, 因此我们生成了 10 组具有不同 μ 值的 LFR 基准图, 用于确保比较结果能够反映真实网络的情况.

此外, 我们针对算法之间的比较做了 Friedman 检验[64]和 Nemenyi 检验[65], 以证明算法的性能之间是否存在显著差异. 这两种方法均为非参数检验, 适用于不满足正态分布或方差齐性假设的数据.

为了比较多个算法在不同参数设置下的性能差异, 我们采用 Friedman 检验进行全局显著性分析. Friedman 检验是一种基于排名的非参数方法, 用于判断多个相关样本 (如不同算法在同一数据集上的表现) 是否存在统计显著性差异. 经过检验, 非重叠社区检测算法在不同参数 μ 下的表现如图 5 所示, 图 5 中不同指标的 Friedman 检验 p 值见表 7. 重叠社区检测算法在不同参数 μ 下的表现如图 6 所示, 在不同参数 on 下的表现如图 7 所示, 图 6 中不同指标的 Friedman 检验 p 值见表 8. 可以看出算法之间存在显著性差异 (p<0.05), 此外, 不同参数 on 值下的算法性能 (图 7) 的 Friedman 检验 p 值均为 0, 在该参数下指标也存在算法之间的显著性差异, 因此我们进一步使用 Nemenyi 事后检验进行两两比较, 进一步分析哪些算法对存在差异. Nemenyi 检验通过计算临界差异 (critical difference, CD) 进行多重比较:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{4}$$

其中, q_α 是学生化范围统计量 (studentized range statistic) 的临界值, k 为算法数量, N 为有效数据点数量. 若两个算法的平均排名差≥CD, 则认为其性能差异显著[66].

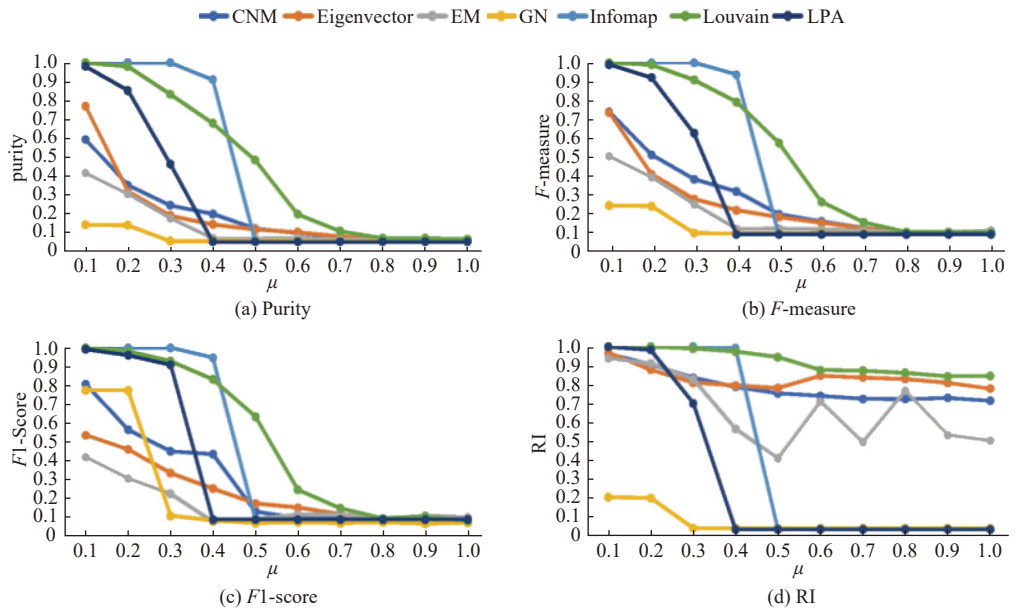


图 5 非重叠社区检测算法在不同参数 μ 值 (范围从 0.1–1) 下的性能表现

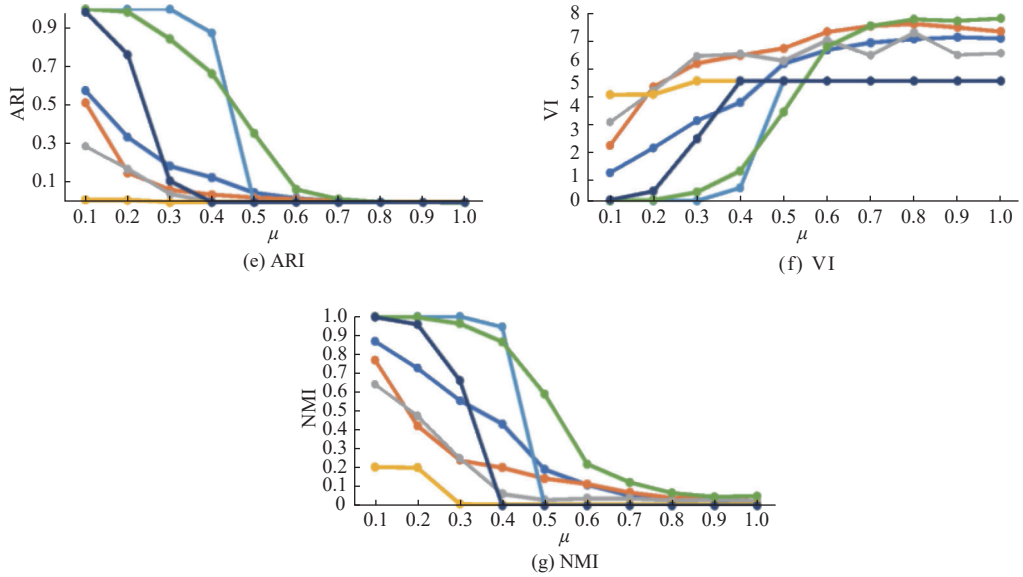


图 5　非重叠社区检测算法在不同参数μ值 (范围为 0.1–1) 下的性能表现 (续)

表 7　图 5 中不同指标的算法性能

指标	Friedman检验p值
NMI	0.0066
purity	0.0062
F-measure	0.0054
F1-score	0.0150
RI	0.0115
ARI	0.0016
VI	0.0002

表 8　图 6 中不同指标的算法性能

指标	Friedman检验p值
F1-score	0.0029
NMI_LFK	0.0050
NMI_MGH	0.0029
OI	0.0050

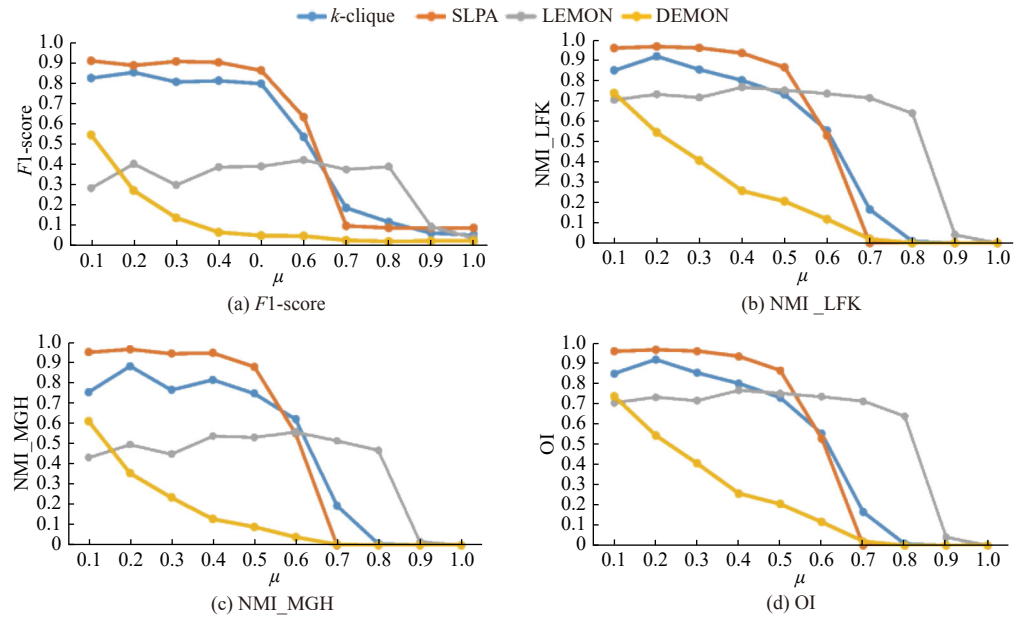


图 6　重叠社区检测算法在不同参数μ值 (范围从 0.1–1) 下的性能表现

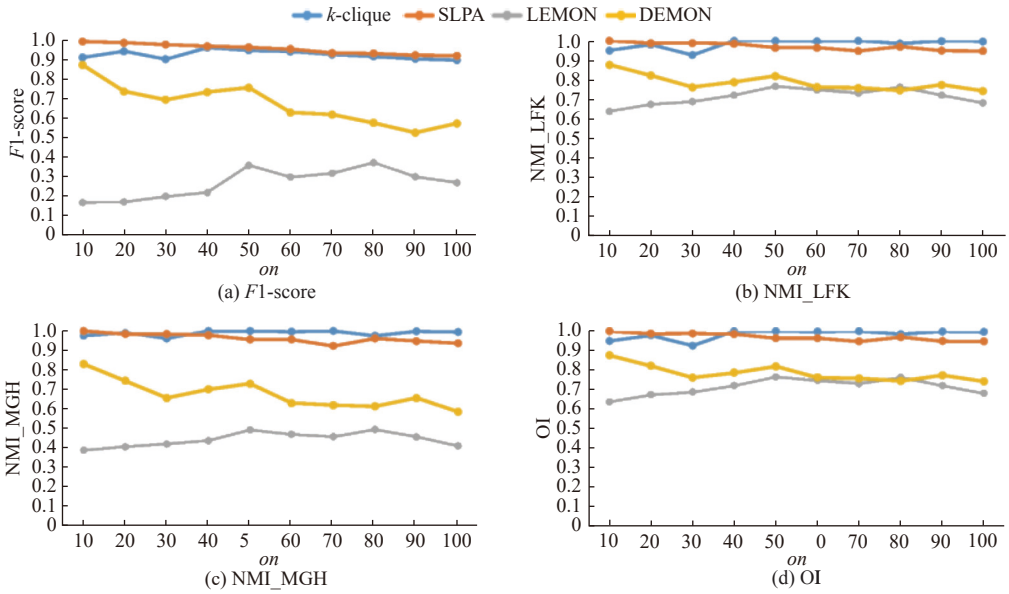


图 7　重叠社区检测算法在不同参数 *on* 值 (范围为 0.1–1) 下的性能表现

6.2　非重叠 LFR 网络中的社区检测

图 5 中我们测试了 μ 参数对算法有效性的影响, 其他参数为控制变量保持一致, 其中参数设置为 n=1000, $\tau_2 = 2$, $max\_degree = \frac{n}{20}$, $min\_degree = 10$. 与前述特性一致, 当 $\mu < \frac{1}{2}$ 时, 所有算法均能获得良好的检测结果, 如图 5 所示. 实验结果与其他研究者在不同参数设定下的结果[20]一致, 表明算法的选择不受参数 τ_1 和 τ_2 的影响.

在 Nemenyi 检验的事后比较结果中, NS、*、**、*** 是用来直观表示显著性水平的符号标记, 其含义与常见统计报告一致: NS 代表不显著, *的数量越多代表显著性水平越高 (p 值越小). 根据后文表 9 的比较结果, 可以得出结论: Infomap 和 Louvain 显著优于 GN 算法, 其他算法之间存在优劣, 但是并没有显著性差异.

6.3　重叠 LFR 网络中的社区检测

一些研究者[67,68]通过设置 *on* 参数 (表示重叠节点的数量) 来生成重叠网络. 我们针对不同的 μ 和 *on* 参数值生成了 20 组网络, 并基于这些网络比较了不同的重叠社区检测算法.

图 6 中, 我们测试了 μ 参数对算法有效性的影响, 其他参数为控制变量保持一致, 其中参数设置为 n=1000, $\tau_1 = 3$, $\tau_2 = 2$, max_degree=80, min_degree=25, on=10, om=2, $min\_community\_size$=20, $max\_community\_size$=50. 如图 6 所示, 当混合参数 $\mu = 0.1$ (该值为生成网络的常见默认值) 时, 所有算法均表现良好. 值得注意的是, DEMON 算法的性能随 μ 增大而迅速下降, 表明其对较高混合度的网络较为敏感. 而 LEMON 算法在不同 μ 值下保持了稳定的性能, 展现出较强的鲁棒性.

在所有测试的算法中, SLPA 取得了最佳的总体效果, 表明其在不同网络条件下均具有良好的适应性. 这些结果强调了根据具体的网络特性和参数设置选择合适算法的重要性.

图 7 中我们测试了 *on* 参数对算法有效性的影响, 其他参数为控制变量保持一致, 其中参数设置为 n=1000, $\tau_1 = 3$, $\tau_2 = 2$, max_degree=80, min_degree=25, $\mu = 0.1$, om=2, $min\_community\_size$=20, $max\_community\_size$=50. 图 7 结果表明, 所有算法对 *on* 参数的影响不显著, 表明它们在不同设定下具有一定的稳定性.

值得注意的是, *k*-clique 和 SLPA 算法表现较强, 表明它们在不同条件下均能有效执行社区检测任务. 根据表 10 和表 11 的比较, 可以得出以下结论: 算法的平均性能排序为 SLPA ≈ *k*-clique > DEMON > LEMON. 其中 SLPA 和 *k*-clique 在不同参数、不同指标下均未表现出显著性差异, 而 SLPA 显著优于 DEMON, *k*-clique 和 SLPA 显著优于 LEMON.

值得注意的是, Lancichinetti 等人[54]建议μ的典型取值范围应小于$\frac{1}{2}$, 因此我们认为$\mu<\frac{1}{2}$时, 算法的性能表现更具代表性, 在图 6 中可以看出, 当$\mu<\frac{1}{2}$时, SLPA 的性能高于 *k*-clique, 因此我们给出这几个算法的最终排序为 SLPA > *k*-clique > DEMON > LEMON.

表 9 图 5 中不同指标的 Nemenyi 检验 *p* 值

对比算法	NMI	Purity	*F*-measure	*F*1-score	RI	ARI	VI
CNM vs. Eigenvector	1.40 NS	0.60 NS	1.00 NS	0.80 NS	0.00 NS	—	2.20 NS
CNM vs. EM	1.60 NS	1.80 NS	2.00 NS	2.20 NS	0.60 NS	—	2.40 NS
CNM vs. GN	3.00 NS	2.80 NS	3.00 NS	1.40 NS	2.40 NS	—	1.20 NS
CNM vs. Infomap	1.20 NS	1.40 NS	1.20 NS	1.60 NS	1.70 NS	—	2.60 NS
CNM vs. Louvain	1.70 NS	1.90 NS	1.70 NS	2.10 NS	2.30 NS	—	2.30 NS
CNM vs. LPA	1.10 NS	0.90 NS	1.10 NS	0.70 NS	1.00 NS	—	0.90 NS
Eigenvector vs. EM	0.20 NS	1.20 NS	1.00 NS	1.40 NS	0.60 NS	0.60 NS	0.20 NS
Eigenvector vs. GN	1.60 NS	2.20 NS	2.00 NS	0.60 NS	2.40 NS	2.20 NS	1.00 NS
Eigenvector vs. Infomap	2.60 NS	2.00 NS	2.20 NS	2.40 NS	1.70 NS	2.60 NS	4.80 *
Eigenvector vs. Louvain	3.10 NS	2.50 NS	2.70 NS	2.90 NS	2.30 NS	2.90 NS	4.50 NS
Eigenvector vs. LPA	0.30 NS	0.30 NS	0.10 NS	0.10 NS	1.00 NS	0.10 NS	3.10 NS
EM vs. GN	1.40 NS	1.00 NS	1.00 NS	0.80 NS	1.80 NS	1.60 NS	1.20 NS
EM vs. Infomap	2.80 NS	3.20 NS	3.20 NS	3.80 NS	2.30 NS	3.20 NS	5.00 *
EM vs. Louvain	3.30 NS	3.70 NS	3.70 NS	4.30 NS	2.90 NS	3.50 NS	4.70 *
EM vs. LPA	0.50 NS	0.90 NS	0.90 NS	1.50 NS	0.40 NS	0.70 NS	3.30 NS
GN vs. Infomap	4.20 NS	4.20 NS	4.20 NS	3.00 NS	4.10 NS	4.80 *	3.80 NS
GN vs. Louvain	4.70 *	4.70 *	4.70 *	3.50 NS	4.70 *	5.10 *	3.50 NS
GN vs. LPA	1.90 NS	1.90 NS	1.90 NS	0.70 NS	1.40 NS	2.30 NS	2.10 NS
Infomap vs. Louvain	0.50 NS	0.50 NS	0.50 NS	0.50 NS	0.60 NS	0.30 NS	0.30 NS
Infomap vs. LPA	2.30 NS	2.30 NS	2.30 NS	2.30 NS	2.70 NS	2.50 NS	1.70 NS
Louvain vs. LPA	2.80 NS	2.80 NS	2.80 NS	2.80 NS	3.30 NS	2.80 NS	1.40 NS

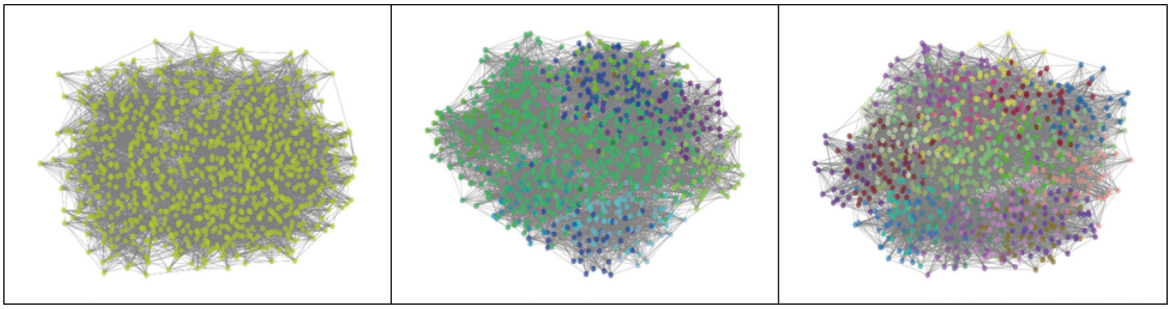
表 10 图 6 中不同指标的 Nemenyi 检验 *p* 值

对比算法	*F*1-score	NMI_LFK	NMI_MGH	OI
LEMON vs. DEMON	0.60 NS	0.80 NS	0.60 NS	0.80 NS
LEMON vs. SLPA	2.20 NS	2.00 NS	2.20 NS	2.00 NS
LEMON vs. *k*-clique	1.20 NS	0.80 NS	1.20 NS	0.80 NS
DEMON vs. SLPA	2.80 *	2.80 *	2.80 *	2.80 *
DEMON vs. *k*-clique	1.80 NS	1.60 NS	1.80 NS	1.60 NS
SLPA vs. *k*-clique	1.00 NS	1.20 NS	1.00 NS	1.20 NS
LEMON vs. DEMON	0.60 NS	0.80 NS	0.60 NS	0.80 NS

表 11 图 7 中不同指标的 Nemenyi 检验 *p* 值

对比算法	*F*1-score	NMI_LFK	NMI_MGH	OI
LEMON vs. DEMON	1.00 NS	0.80 NS	1.00 NS	1.00 NS
LEMON vs. *k*-clique	2.00 *	2.60 ***	2.80 ***	2.50 **
LEMON vs. SLPA	3.00***	2.20 **	2.20 **	2.10 **
DEMON vs. *k*-clique	1.00 NS	1.80 *	1.80 *	1.50 NS
DEMON vs. SLPA	2.00 *	1.40 NS	1.20 NS	1.10 NS
k-clique vs. SLPA	1.00 NS	0.40 NS	0.60 NS	0.40 NS
LEMON vs. DEMON	1.00 NS	0.80 NS	1.00 NS	1.00 NS

图 8 为 LFR 网络可视化中的社区示例, 使用 *k*-clique 算法在同一图上通过不同 *k* 值获得. 图 8 说明了在不同 *k* 值下, 网络的团块大小有明显的区别. *k* 越小, *k*-clique 算法越倾向于将网络分割为大社区. 因此合适的 *k* 取值较为重要.



(a) k=2 (b) k=3 (c) k=4

图 8 LFR 网络可视化中的社区示例

6.4 研究结论 (RQ3 的解答)

通过上述分析, 我们对 RQ3 的回答如下.

我们使用标准 LFR 网络和改进的重叠 LFR 网络测试并比较了非重叠和重叠社区检测算法. 为了研究关键参数 μ 和 on 对算法性能的影响, 本文调整了它们的取值, 因为这些参数无法直接从 DSN 属性推导得出. 实验结果表明, 在合成网络中, 当 $\mu < \frac{1}{2}$ 时, 算法可以检测到明确的社区, 而参数 on 对算法性能的影响较小.

此外, 合成网络上的算法比较结果与真实 DSN 上的结果一致, 证实了 Louvain、Infomap 和 SLPA 等算法的良好表现, 并表明基于贪心策略改进的 LEMON 算法具有较强的鲁棒性, 不受 μ 变化的影响.

7 讨 论

社区检测对识别和建模开源软件的协作和交流活动有相当的意义[31]. 社区检测作为量化开发者群体自组织特征的手段, 能够揭示知识传播的关键路径, 具有重要的研究价值. 本研究构建的基准测试采用了 GitHub 的真实交互数据, 设计了包含多种指标的评估协议, 能够准确揭示社区生态, 并在此基础上额外提供了公开数据集和测试代码以保证结果可被复现. 综合来看, 本研究通过量化开发者社区的拓扑结构和特征, 为理解开源协作机制提供了新的分析维度, 为更加全面地评估社区健康度提供了新的视角, 提出的基准测试体系将为该领域后续研究提供标准化、可扩展的数据基础与评估框架.

研究开源项目社区检测为开发治理提供了依据, 社区检测可识别关键开发者角色, 帮助项目维护者优化任务分配策略; 知识传递路径的显式建模能预防“知识孤岛”风险, 特别是当核心开发者流失时. 当项目开发者能够通过社区检测对项目成员的交流情况有一定认识后, 就可以将这些洞察用于采取更好的项目管理措施, 如新人导师匹配、跨社区协作促进、项目治理结构优化等. 目前这一视角已经被业界关注, 例如 Linux 基金会已开始采用类似的社区分析工具来监测子社区活跃度, 其 2023 年年报显示这种方法使跨团队协作效率提升了 40%. OpenRank[69] 是一个指数, 用于提供开源项目的多维度数据分析, 包括贡献者活跃度、代码提交趋势、公司参与度等. 它已经被 Kubernetes、TARS 等顶级开源项目用于社区治理. 本研究的基准体系希望在社区检测领域产生类似的推动作用, 为开源生态分析建立新的方法论标准.

基准研究通过标准化评估减少对主观经验的依赖, 使用数据驱动的方式推动软件工程问题的优化. 目前已经有很多基准测试推动了软件工程领域的技术进步, 并显著影响了软件开发实践和开源社区的可持续性. 但是大部分项目仍然对社区的关注度有限, 需要更多的研究为业界提供工具链, 以推动社区的数据驱动精细化管理. 未来可能的研究方向有 3 个方面: 1) 流程优化方面, 探索将社区检测实时集成到开发流水线, 如基于协作模式分析自动推荐代码评审者; 2) 质量提升方向, 研究社区结构与代码质量指标的关联规律, 建立预测模型; 3) 可持续发展角度, 构建社区健康度评估体系, 量化分析成员参与度、知识传承效率与项目存活率的关联性. 特别值得关注的是, 随着 LLM 在开源协作中的应用 (如 GitHub Copilot), 需要新的检测方法区分人类与 AI 的协作模式. 这些研究将深化

我们对“开发者-社区-软件质量”三元关系的理解, 最终形成可量化的社区治理最佳实践.

8 结 论

本文提出了一种适用于项目级 OSS-DSN 的基准测试, 弥补了现有基准测试主要集中在大规模社交网络和网站级合成网络的研究空白. 我们详细研究了 OSS-DSN 的网络特性, 并使用合成网络模拟这些特性, 以确保比较结果的一致性. 基于实验结果, 我们对 OSS-DSN 的社区检测算法提出了建议, 这些贡献为改进 OSS 项目的社区分析提供了有价值的见解.

然而, 本研究也存在一定的有效性威胁. 在内部有效性方面, 尽管我们采用了多种评估指标, 但这些指标并非全部被广泛接受或全面覆盖, 可能导致分析存在偏差或不完整. 尽管如此, 通过使用多种指标, 本文尽可能降低了此类偏倚的风险, 但仍可能存在一定的局限性. 在外部有效性方面, 数据集的局限性是一个主要威胁. 本研究的数据集仅限于 GitHub, 未包含 GitLab 或 Bitbucket 等其他平台的数据. 虽然 GitHub 项目与其他平台的项目具有一定相似性, 但数据集局限性可能影响研究结果的普适性. 为了增强外部有效性, 未来研究应涵盖更广泛的开源平台数据.

未来研究可以沿着多个方向展开.

1) 扩展网络范围: 除了 OSS 项目中的开发者社交网络 (DSN), 还应研究网站级网络和组织级网络, 以获得更全面的洞察.

2) 增加社区检测算法: 引入更广泛的算法集合, 以提升分析的稳健性和适应性, 更有效地捕捉这些网络中的复杂结构和关系.

3) 探索更多应用场景: 确定更多可以受益于本基准测试的任务, 以展示其通用性和实用性, 最终推动网络分析和社区检测领域的发展.

通过进一步的研究和改进, 我们相信本研究将为开源软件项目的社区分析提供更加全面和深入的见解, 并推动该领域的持续发展.

References

[1] Wu WJ, Zheng ZM, Wang HM, Tang ST, Wang T. Development strategy of collective intelligence and its industrial clusters. Strategic Study of CAE, 2024, 26(1): 89–100 (in Chinese with English abstract). [doi: 10.15302/J-SSCAE-2024.01.008]

[2] Wang T, Yin G, Yu Y, Zhanng Y, Zhang YM. Crowd-intelligence-based software development method and practices. Scientia Sinica Informationis, 2020, 50(3): 318–334 (in Chinese with English abstract). [doi: 10.1360/ SSI-2019-0187] [doi: 10.1360/SSI-2019-0187]

[3] Wang HM, Yu Y, Wang T, Ding B. Crowd intelligence paradigm: A new paradigm shift in software development. Scientia Sinica Informationis, 2023, 53(8): 1490–1502 (in Chinese with English abstract). [doi: 10.1360/SSI-2023-0064]

[4] Wu J, Huang X, Wang B. Social-technical network effects in open source software communities: Understanding the impacts of dependency networks on project success. Information Technology & People, 2023, 36(2): 895–915. [doi: 10.1108/ITP-09-2021-0684]

[5] Peng G, Wan Y, Woodlock P. Network ties and the success of open source software development. The Journal of Strategic Information Systems, 2013, 22(4): 269–281. [doi: 10.1016/j.jsis.2013.05.001]

[6] Karrer B, Newman MEJ. Stochastic blockmodels and community structure in networks. Physical Review E, 2011, 83: 016107. [doi: 10.1103/PhysRevE.83.016107]

[7] Zhang PC, Liu P, Wang NX. Evolutionary analysis of developer collaboration network in cloud foundry OSS community. In: Proc. the 20th Int’l Symp. on Knowledge and Systems Sciences. Da Nang: Springer, 2019. 87–105. [doi: 10.1007/978-981-15-1209-4_7]

[8] Peng G, Dey D. Research note—A dynamic view of the impact of network structure on technology adoption: The case of OSS development. Information Systems Research, 2013, 24(4): 1087–1099. [doi: 10.1287/isre.2013.0494]

[9] Behfar SK, Turkina E, Burger-Helmchen T. Knowledge management in OSS communities: Relationship between dense and sparse network structures. Int’l Journal of Information Management, 2018, 38(1): 167–174. [doi: 10.1016/j.ijinfomgt.2017.09.004]

[10] Moradi-Jamei B, Kramer BL, Calderón JBS, Korkmaz G. Community formation and detection on GitHub collaboration networks. In: Proc. of the 2021 IEEE/ACM Int’l Conf. on Advances in Social Networks Analysis and Mining. ACM, 2021. 244–251. [doi: 10.1145/3487351.3488278]

[11] Granell C, Darst RK, Arenas A, Fortunato S, Gómez S. Benchmark model to assess community structure in evolving networks. Physical

Review E: Statistical, Nonlinear, and Soft Matter Physics, 2015, 92(1): 012805. [doi: 10.1103/PhysRevE.92.012805]

[12] Wang M, Wang CK, Yu JX, Zhang J. Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework. Proc. of the VLDB Endowment, 2015, 8(10): 998–1009. [doi: 10.14778/2794367.2794370]

[13] Xie JR, Kelley S, Szymanski BK. Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Computing Surveys (CSUR), 2013, 45(4): 43. [doi: 10.1145/2501654.2501657]

[14] Hamann M, Meyer U, Penschuck M, Tran H, Wagner D. I/O-efficient generation of massive graphs following the LFR benchmark. Journal of Experimental Algorithmics, 2018, 23: 2.5. [doi: 10.1145/3230743]

[15] Oehlers M, Fabian B. Graph metrics for network robustness—A survey. Mathematics, 2021, 9(8): 895. [doi: 10.3390/math9080895]

[16] Venigalla ASM, Ali MS, M N, Chimalakonda S. RCGraph—A tool to integrate readme and commits through temporal knowledge graphs. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Program Comprehension (ICPC). Melbourne: IEEE, 2023. 30–34. [doi: 10.1109/ICPC58990.2023.00014]

[17] Borg M, Alégroth E, Runeson P. Software engineers' information seeking behavior in change impact analysis—An interview study. In: Proc. of the 25th IEEE/ACM Int'l Conf. on Program Comprehension (ICPC). Buenos Aires: IEEE, 2017. 12–22. [doi: 10.1109/ICPC.2017.20]

[18] Liu P, Gui L. Structural analysis of collaboration network in OSS communities. In: Proc. of the 4th Int'l Conf. on Data Science and Information Technology. Shanghai: ACM, 2021. 84–91. [doi: 10.1145/3478905.3478923]

[19] Saha S, Ghrera SP. Network community detection on metric space. Algorithms, 2015, 8(3): 680–696. [doi: 10.3390/a8030680]

[20] Jin D, Yu ZZ, Jiao PF, Pan SR, He DX, Wu J, Yu PS, Zhang WX. A survey of community detection approaches: From statistical modeling to deep learning. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(2): 1149–1170. [doi: 10.1109/TKDE.2021.3104155]

[21] Pérez-Peló S, Sánchez-Oro J, Martín-Santamaría R, Duarte A. On the analysis of the influence of the evaluation metric in community detection over social networks. Electronics, 2019, 8(1): 23. [doi: 10.3390/electronics8010023]

[22] Yang J, Leskovec J. Overlapping community detection at scale: A nonnegative matrix factorization approach. In: Proc. of the 6th ACM Int'l Conf. on Web Search and Data Mining. Rome: ACM, 2013. 587–596. [doi: 10.1145/2433396.2433471]

[23] Su X, Xue S, Liu FZ, Wu J, Yang J, Zhou C, Hu WB, Paris C, Nepal S, Jin D, Sheng QZ, Yu PS. A comprehensive survey on community detection with deep learning. IEEE Trans. on Neural Networks and Learning Systems, 2024, 35(4): 4682–4702. [doi: 10.1109/TNNLS.2021.3137396]

[24] Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Physical Review E: Statistical, Nonlinear, and Soft Matter Physics, 2008, 78: 046110. [doi: 10.1103/PhysRevE.78.046110]

[25] Lancichinetti A, Fortunato S, Kertész J. Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics, 2009, 11: 033015. [doi: 10.1088/1367-2630/11/3/033015]

[26] Fortunato S. Community detection in graphs. Physics Reports, 2010, 486(3–5): 75–174. [doi: 10.1016/j.physrep.2009.11.002]

[27] Bandyopadhyay S, Vishal Vivek S, Murty MN. Integrating network embedding and community outlier detection via multiclass graph description. In: Proc. of the 24th European Conf. on Artificial Intelligence. Santiago de Compostela: IOS Press, 2020. 976–983. [doi: 10.3233/FAIA200191]

[28] Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU. Complex networks: Structure and dynamics. Physics Reports, 2006, 424(4–5): 175–308. [doi: 10.1016/j.physrep.2005.10.009]

[29] Xiao WX, He H, Xu WW, Zhang YX, Zhou MH. How early participation determines long-term sustained activity in GitHub projects? In: Proc. of the 31st ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. San Francisco: ACM, 2023. 29–41. [doi: 10.1145/3611643.3616349]

[30] Parra E, Alahmadi M, Ellis A, Haiduc S. A comparative study and analysis of developer communications on Slack and Gitter. Empirical Software Engineering, 2022, 27(2): 40. [doi: 10.1007/s10664-021-10095-1]

[31] Hong QN, Kim S, Cheung SC, Bird C. Understanding a developer social network and its evolution. In: Proc. of the 27th IEEE Int'l Conf. on Software Maintenance (ICSM). Williamsburg: IEEE, 2011. 323–332. [doi: 10.1109/ICSM.2011.6080799]

[32] Zhang JR, Wang L, Li Y, Jiang J, Wang T, Tao XP. Measuring and mining community evolution in developer social networks with entropy-based indices. ACM Trans. on Software Engineering and Methodology, 2025, 34(1): 12. [doi: 10.1145/3688832]

[33] Wang L, Li Y, Zhang JR, Tao XP. Quantifying community evolution in developer social networks. In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Singapore: ACM, 2022. 157–169. [doi: 10.1145/3540250.3549106]

[34] Girvan M, Newman MEJ. Community structure in social and biological networks. Proc. of the National Academy of Sciences of the

United States of America, 2002, 99(12): 7821–7826. [doi: 10.1073/pnas.122653799]

[35] Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008: P10008. [doi: 10.1088/1742-5468/2008/10/P10008]

[36] Newman MEJ, Leicht EA. Mixture models and exploratory analysis in networks. Proc. of the National Academy of Sciences of the United States of America, 2007, 104(23): 9564–9569. [doi: 10.1073/pnas.0610537104]

[37] Zhang X, Wang XG, Zhao DL, Tang XO. Graph degree linkage: Agglomerative clustering on a directed graph. In: Proc. of the 12th European Conf. on Computer Vision. Florence: Springer, 2012. 428–441. [doi: 10.1007/978-3-642-33718-5_31]

[38] Cordasco G, Gargano L. Community detection via semi-synchronous label propagation algorithms. In: Proc. of the 2010 IEEE Int'l Workshop on: Business Applications of Social Network Analysis (BASNA). Bangalore: IEEE, 2010. 1–8. [doi: 10.1109/BASNA.2010.5730298]

[39] Derényi I, Palla G, Vicsek T. Clique percolation in random networks. Physical Review Letters, 2005, 94(16): 160202. [doi: 10.1103/PhysRevLett.94.160202]

[40] Xie JR, Szymanski BK, Liu XM. SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: Proc. of the 11th IEEE Int'l Conf. on Data Mining Workshops. Vancouver: IEEE, 2011. 344–349. [doi: 10.1109/ICDMW.2011.154]

[41] Coscia M, Rossetti G, Giannotti F, Pedreschi D. DEMON: A local-first discovery method for overlapping communities. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Beijing: ACM, 2012. 615–623. [doi: 10.1145/2339530.2339630]

[42] Li YX, He K, Bindel D, Hopcroft JE. Uncovering the small community structure in large networks: A local spectral approach. In: Proc. of the 24th Int'l Conf. on World Wide Web. Geneva: Int'l World Wide Web Conf. Steering Committee, 2015. 658–668. [doi: 10.1145/2736277.2741676]

[43] Rosvall M, Bergstrom CT. Bergstrom maps of random walks on complex networks reveal community structure. Proc. of the National Academy of Sciences of the United States of America, 2008, 105(4): 1118–1123. [doi: 10.1073/pnas.0706851105]

[44] Chakraborty T, Dalmia A, Mukherjee A, Ganguly N. Metrics for community analysis: A survey. ACM Computing Surveys, 2017, 50(4): 54. [doi: 10.1145/3091106]

[45] Newman MEJ. Modularity and community structure in networks. Proc. of the National Academy of Sciences of the United States of America, 2006, 103(23): 8577–8582. [doi: 10.1073/pnas.0601602103]

[46] Zhang SH, Wang RS, Zhang XS. Identification of overlapping community structure in complex networks using fuzzy *c*-means clustering. Physica A: Statistical Mechanics and its Applications, 2007, 374(1): 483–490. [doi: 10.1016/j.physa.2006.07.023]

[47] Pizzuti C. GA-Net: A genetic algorithm for community detection in social networks. In: Proc. of the 10th Int'l Conf. on Parallel Problem Solving from Nature. Dortmund: Springer, 2008. 1081–1090. [doi: 10.1007/978-3-540-87700-4_107]

[48] Chira C, Gog A, Iclănzan D. Evolutionary detection of community structures in complex networks: A new fitness function. In: Proc. of the 2012 IEEE Congress on Evolutionary Computation. Brisbane: IEEE, 2012. 1–8. [doi: 10.1109/CEC.2012.6256561]

[49] Traag VA, Krings G, Van Dooren P. Significant scales in community structure. Scientific Reports, 2013, 3(1): 2930. [doi: 10.1038/srep02930]

[50] Aldecoa R, Marin I. Deciphering network community structure by surprise. PLoS One, 2011, 6(9): e24195. [doi: 10.1371/journal.pone.0024195]

[51] Chakraborty T. Leveraging disjoint communities for detecting overlapping community structure. Journal of Statistical Mechanics: Theory and Experiment, 2015, 2015(5): P05017. [doi: 10.1088/1742-5468/2015/05/P05017]

[52] Miyauchi A, Kawase Y. What is a network community?: A novel quality function and detection algorithms. In: Proc. of the 24th ACM Int'l on Conf. on Information and Knowledge Management. Melbourne: ACM, 2015. 1471–1480. [doi: 10.1145/2806416.2806555]

[53] de França FO, Coelho GP. A flexible fitness function for community detection in complex networks. In: Mangioni G, Simini F, Uzzo SM, Wang DS, eds. Complex Networks VI: Proc. of the 6th Workshop on Complex Networks CompleNet 2015. Cham: Springer, 2015. 1–12. [doi: 10.1007/978-3-319-16112-9_1]

[54] Lancichinetti A, Fortunato S. Community detection algorithms: A comparative analysis. Physical Review E, 2009, 80(5): 056117. [doi: 10.1103/PhysRevE.80.056117]

[55] Lin TY, Ohsuga S, Liau CJ, Hu XH. Foundations and Novel Approaches in Data Mining. Berlin: Springer, 2006.

[56] Artiles J, Gonzalo J, Sekine S. The SemEval-2007 WePS evaluation: Establishing a benchmark for the Web people search task. In: Proc. of the 4th Int'l Workshop on Semantic Evaluations. Prague: Association for Computational Linguistics, 2007. 64–69.

[57] Hubert L, Arabie P. Comparing partitions. Journal of Classification, 1985, 2(1): 193–218. [doi: 10.1007/BF01908075]

[58] Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In: Proc. of the 26th Annual Int'l Conf. on Machine Learning. Montreal: ACM, 2009. 1073–1080. [doi: 10.1145/1553374.1553511]

[59] Kraskov A, Stögbauer H, Andrzejak RG, Grassberger P. Hierarchical clustering using mutual information. Europhysics Letters, 2005, 70(2): 278. [doi: 10.1209/epl/i2004-10483-y]

[60] Collins LM, Dent CW. Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. Multivariate Behavioral Research, 1988, 23(2): 231–242. [doi: 10.1207/s15327906mbr2302_6]

[61] Bloch F, Jackson MO, Tebaldi P. Centrality measures in networks. Social Choice and Welfare, 2023, 61(2): 413–453. [doi: 10.1007/s00355-023-01456-4]

[62] Fortunato S, Barthelemy M. Resolution limit in community detection. Proc. of the National Academy of Sciences of the United States of America, 2007, 104(1): 36–41. [doi: 10.1073/pnas.0605965104]

[63] Condon A, Karp RM. Algorithms for graph partitioning on the planted partition model. Random Structures & Algorithms, 2001, 18(2): 116–140. [doi: 10.1002/1098-2418(200103)18:2<116::AID-RSA1001>3.0.CO;2-2]

[64] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, 1937, 32(200): 675–701. [doi: 10.1080/01621459.1937.10503522]

[65] Nemenyi PB. Distribution-free multiple comparisons [Ph.D. Thesis]. Princeton: Princeton University, 1963.

[66] Garcia S, Herrera F. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research, 2008, 9: 2677–2694.

[67] Lancichinetti A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Physical Review E: Statistical, Nonlinear, and Soft Matter Physics, 2009, 80: 016118. [doi: 10.1103/PhysRevE.80.016118]

[68] Reichardt J, Bornholdt S. Statistical mechanics of community detection. Physical Review E, 2006, 74: 016110. [doi: 10.1103/PhysRevE.74.016110]

[69] Zhao SY, Xia XY, Fitzgerald B, Li XZ, Lenarduzzi V, Taibi D, Wang R, Wang W, Tian C. OpenRank leaderboard: Motivating open source collaborations through social network evaluation in Alibaba. In: Proc. of the 46th Int'l Conf. on Software Engineering: Software Engineering in Practice. Lisbon: ACM, 2024. 346–357. [doi: 10.1145/3639477.3639734]

附中文参考文献

[1] 吴文峻, 郑志明, 王怀民, 唐绍婷, 王涛. 群体智能及产业集群发展战略研究. 中国工程科学, 2024, 26(1): 89–100. [doi: 10.15302/J-SSCAE-2024.01.008]

[2] 王涛, 尹刚, 余跃, 张洋, 王怀民. 基于群智的软件开发群体化方法与实践. 中国科学: 信息科学, 2020, 50(3): 318–334. [doi: 10.1360/SSI-2019-0187] [doi: 10.1360/SSI-2019-0187]

[3] 王怀民, 余跃, 王涛, 丁博. 群智范式: 软件开发范式的新变革. 中国科学: 信息科学, 2023, 53(8): 1490–1502. [doi: 10.1360/SSI-2023-0064]

作者简介

张洁芮, 博士生, CCF 学生会员, 主要研究领域为开源软件社区群智行为演化, 时序数据建模.

汪亮, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为软件方法学, 群体智能, 开源仓库挖掘, 开源软件生态评估与分析.

常正超, 博士, 讲师, 主要研究领域为社会网络分析, 计算机视觉.

李梓荣, 本科生, 主要研究领域为机器学习, 数据挖掘.

桑百惠, 博士生, CCF 学生会员, 主要研究领域为开源软件分析与评估.

陶先平, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 软件 Agent 技术, 软件中间件技术, 网构软件方法学, 普适计算技术.