arXiv:2304.03170v1 [cs.SI] 5 Apr 2023

图谱算法工具包（STAG）

技术报告（1）

Peter Macgregor 和 He Sun

爱丁堡大学信息学院 英国
{peter.macgregor,
h.sun}@ed.ac.uk

当前版本	STAG 1.2
主要更新	局部图聚类
Date	2023年4月7日
网站	https://staglibrary.io
源代码	https://github.com/staglibrary
资助信息	STAG 的开发得到了英国研究与创新署的支持 在EPSRC 早期职业研究员项目（EP/T00729X/1）资助下。

1 摘要

图谱算法工具包(STAG) 是一个开源的 C++ and Python 图高效谱算法库。我们的目标是实现通过算法谱图论开发的先进图算法，同时让最终用户能够实际使用。这一系列技术报告旨在记录我们在 STAG方面的进展，包括实现细节、工程考量，以及用于测试我们实现的数据集。报告结构如下：

- 第2 节介绍了局部聚类算法，这是本次STAG 版本中的主要更新。相关讨论处于较高层次，因此不需要具备超出基础算法之外的领域知识。
- 第 3 节介绍 STAG 的核心功能用户指南，帮助用户进行局部聚类。
- 第 4 节包含 STAG 的实验和功能演示。
- 最后，第5 节讨论了若干技术细节；其中包括我们对所实现算法的选择、参数的默认设置以及其他技术选择。我们将这些细节留到最后一节介绍，因为读者在使用 STAG 时无需了解这些内容。

1.1 已实现的算法

STAG 1.2 实现了以下关键算法。

局部图聚类。给定一个大型图和图中的某个起始顶点 v，局部图聚类的目标是找到一个包含 v的簇。此外，算法的运行时间应仅取决于返回的聚类的大小，而不应取决于图 [11]的总规模。

STAG 提供了首个无需将整个图加载到内存中的开源局部聚类算法。这使用户能够对存储在磁盘上甚至云数据库（例如 Neo4j[1]）中的大规模图应用局部聚类。第 4 节展示了这些应用。

谱聚类。其中最基础的算法之一是谱图论中的谱聚类算法 [8, 10, 13]。谱聚类是“全局性的”，因为它会返回整个图的顶点集的划分。

从随机模型生成图。 随机块模型（SBM）和Erdős-Rényi模型是常用的随机图模型，经常用于评估和分析图算法。STAG 提供了几种便捷的方法，可从这些模型生成图。

2 局部图簇

图聚类算法旨在将输入图划分为两个或更多个簇。作为数据科学和机器学习中的基础技术，图聚类在计算机科学及其他众多领域都有许多应用。大多数图聚类算法都需要读取整个输入图来执行聚类任务；如果图规模庞大，这在计算上会非常昂贵。如果只关心某些“局部”簇信息，那么局部图聚类提供了一种更高效的方法。

通常，局部图聚类的目标是在输入图中找到某个高度连通的顶点集（簇）。假设 S 是底层无向图 G的一个高度连通的顶点集，即 S 构成一个簇。那么，局部聚类算法以某个顶点 $v \in S$ 作为输入，并返回某个集合 S'，使得 S' 是目标集合 S 的合理近似。此外，运行时间

[1] https://neo4j.com/



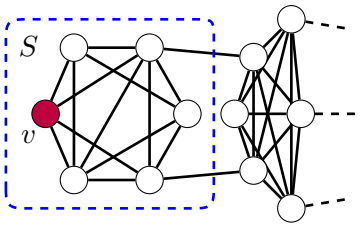

图 1：给定一个包含某个小簇 S的大型图，局部聚类算法以一个顶点 $v \in S$ 作为输入，并返回 S 的近似结果，而无需探索整个图。

该算法的运行时间与 S 的大小成正比，且不受 G大小的影响。在实际应用中，局部聚类可以被视为对相关对象的搜索：给定一个查询顶点，局部聚类算法会返回一组密切相关的顶点。图 1 展示了局部图聚类。

Andersen、Chung 和 Lang [1] 提出了一种关键的局部聚类算法，我们称之为 ACL 算法。从高层次来看，该算法通过分析从起始顶点开始在图上的随机游走行为来寻找本地集群。ACL 算法已被证明极其有用，并启发了大量后续研究和应用 [2, 7, 12, 14]。STAG 的 局部_集群 方法提供了 ACL 局部聚类算法的实现。

现有的开源局部聚类方法要求将整个图加载到随机存取存储器中，才能应用局部算法。从这个意义上说，它们并非真正“局部”，因为无法应用于大于可用内存的图，而且总运行时间取决于图的大小。STAG 提供了首个开源局部聚类算法，无需将大型图加载到随机存取存储器中即可应用。此外，它提供的接口简单易用，该算法可应用于存储在内存、磁盘上或 Neo4j 数据库中的图。

3 使用 STAG 进行局部图聚类的用户指南

本节介绍 STAG 的基本功能，用户可以利用这些功能进行局部聚类。第 3.1 节介绍如何安装 STAG C++和 STAG Python 库。然后，第 3.2 节介绍 STAG 支持的图文件格式，并演示将图读写到磁盘的方法。最后，第 3.3 节解释 STAG 提供的图类，第 3.4 节记录用于局部聚类的本地_簇方法。

尽管本节中的大多数示例使用 C++，STAG C++ 的功能在 STAG Python 中也可用。附录B包含示例代码，演示如何使用 STAG Python 执行局部簇聚类。STAG C++ 和STAG Python 的完整文档可在 STAG 库网站上找到。

3.1 STAG 的安装

本节介绍如何安装 STAG，以便与 C++ 和 Python 配合使用。

为 C 安装 STAG++。 STAG 基于 Eigen 和 Spectra C++ 库构建，必须先安装这两个库，然后才能安装 STAG。有关安装 Eigen 和 Spectra 的信息，请参阅它们的文档。为方便起见，附录 A 提供了一个 Bash 脚本；在本文撰写时，该脚本可在标准 Linux 系统上安装 Eigen 和 Spectra。接下来，应从以下位置下载最新版本的 STAG。

https://github.com/staglibrary/stag/releases.

下载并解压源代码后，可以使用cmake编译并安装 STAG。

```
mkdirbuild_dir
cd
m
```




安装 STAG 后，即可在使用 `cmake` 构建工具构建的 C++ 项目中使用。以下 `cmake` 代码将把 C++项目与 STAG 链接起来。

```
    find_包（stag 必需）包含_目录（${ STAG_INCLUDE_


```

一个展示完整 `cmake`配置的 STAG 项目示例可在此处找到

`https://github.com/staglibrary/example-stag-project.`

为 Python 安装 STAG。STAG Python 可通过 pip 工具从 Python Package Index 安装。

```
    python -m pip install stag
```

然后，可以将 STAG 的模块直接导入任何 Python 脚本中。

3.2 文件格式

STAG 支持两种用于在磁盘上存储图的简单文件格式：EdgeList 和 AdjacencyList。许多图数据集以 EdgeList 格式提供， [6],可直接与 STAG 配合使用。

EdgeList 文件格式。 在 EdgeList 文件中，每一行对应图中的一条边。一行由两个整数节点 ID 和一个可选的边权重组成，彼此以空格分隔。以下是一个简单 EdgeList 文件的示例。

```
# 这是一个注释2  01
0.


```

在这个例子中，第 2 行定义了一条连接节点 0 和 1、权重为 0。5 的边。

邻接表文件格式。在邻接表文件中，每一行对应图中的一个节点。一行由节点ID组成，后跟一个相邻节点列表。每行开头的节点 ID 必须按递增顺序排序。下面是一个简单邻接表文件的示例。

```
# 这是注释2  0: 123  1:
03


```

在此示例中，节点 1 有指向节点 0、2 和 3 的边。

使用文件。STAG 提供了多种方法来读取、写入以及转换边列表文件和邻接表文件，如以下示例所示。

```
# include   <stag / graphio .h >2     ...3  // 读取邻接表图4 std :: string filename   =
"m
fil
sta
sta


```




3.3 图类

STAG 提供了若干图类，可应用于各种广泛的领域。图 2 总结了可用的图类。

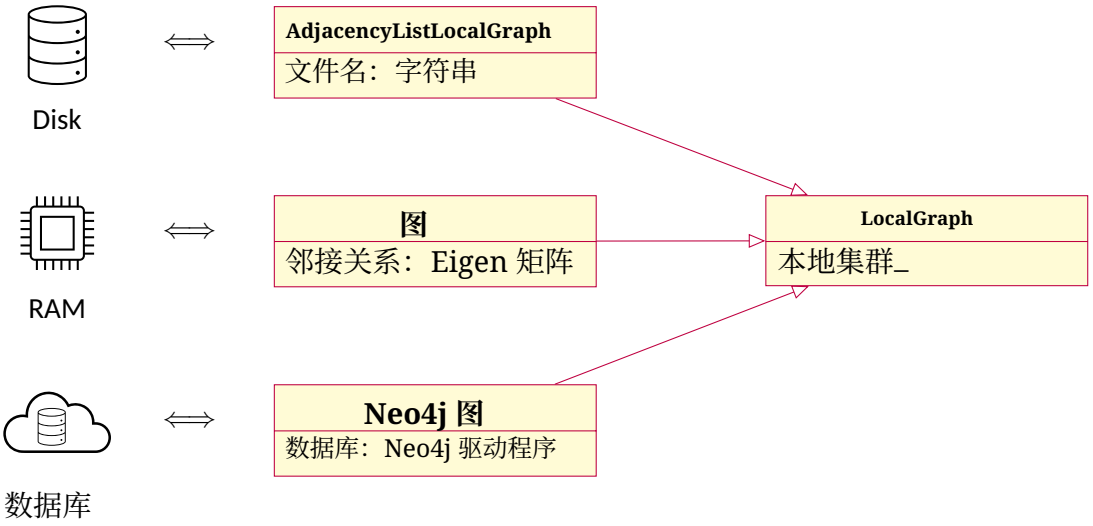

图 2：STAG 提供的图类。每个类都继承自抽象的 `LocalGraph` 类，该类提供localclustering 方法。不同类之间的关键区别在于图数据的位置。`AdjacencyListLocalGraph` 类从磁盘读取节点邻接数据，`图` 类将整个图存储在随机存取存储器中，而Neo4j `图` 类从 Neo4j 数据库查询节点邻接数据。

LocalGraph 类。 STAG 提供了一个抽象的`stag::LocalGraph` 类，用于定义应用局部聚类所需的数据结构。该数据结构唯一必需的方法是 `neighbors(v)`，它会返回节点 v 的邻居列表。STAG 提供的所有图类都继承自 LocalGraph。

图类。 `stag::Graph` 类是 STAG 库中的基本图对象。该类将图的邻接矩阵以稀疏矩阵的形式存储在内存中。

邻接表LocalGraph类。 `stag::AdjacencyListLocalGraph` 类为存储在磁盘上的邻接表中的图提供了 `stag::LocalGraph` 接口的实现。该图仅以局部方式加载到内存中。这使得无需将整个图加载到内存中，即可在存储于磁盘上的超大型图上执行局部算法。以下示例演示了如何使用 STAG C++ 创建一个`AdjacencyListLocalGraph` 。

```
#include <stag/graph.h>
...
// 创建一个 AdjacencyListLocalGraph
std::string
fi
67
((
```

Neo4jGraph 类。STAG Python 还提供了 `Neo4jGraph` 类，它为存储在 Neo4j 数据库中的图实现了`LocalGraph` 接口。以下示例展示了如何使用数据库连接信息创建`Neo4jGraph` 。



```
import stag.neo4j
# 连接到数据库
uri = "<数据库 URI >"
username = "
```

Neo4j图 类提供了用于查询数据库中节点属性的其他方法。详细信息请参阅完整的 STAG 文档。

3.4 局部聚类

STAG 提供以下局部_簇方法。

```
std::vector<long long> stag::local_cluster(stag::LocalGraph* graph,
                                           long long         seed_vertex,
                                           double            target_volume)
```

给定一个图和一个起始顶点，本地_簇方法可以找到一个靠近起始顶点的簇。算法的运行时间与返回的聚类的大小成正比，并且与整个图的大小无关。该方法的参数描述如下：

- **gra ph** - 一个 LocalGraph对象。它可以是一个 图、一个 AdjacencyListLocalGraph，或一个 Neo4jGra ph.
- **种子_顶点**- the 起始顶点 in the 图.
- **目标_体积**-目标簇体积的估计值。该参数不会对算法施加硬约束，因此近似的体积就足够了。

处理超大型图时，建议使用`AdjacencyListLocalGraph` 对象执行局部聚类，以避免将整个图读入内存所产生的开销。第 4.1 节演示了使用`AdjacencyListLocalGraph` 进行局部聚类的优势。以下代码演示了一个完整程序，该程序使用STAG C++ 来查找存储在磁盘上邻接表文件中的图的本地集群。

```
1 # include <iostream >2 # include <stag / graph .h >3 # include <stag / cluster .h >4 5 int main ()
```




4 典型案例研究

STAG 让各种应用中的局部聚类变得简单，本节将介绍一些使用 STAG 进行局部聚类的示例。用于生成所有实验结果的代码可在

`https://github.com/staglibrary/local-clustering-case-study`.

所有实验均在一台配备第11代Intel(R)Core(TM) i7-11800H @ 2.30GHz处理器和 32 GB 随机存取存储器的 HP ZBook 笔记本电脑上进行。

4.1 示例 1

局部聚类相较于其他聚类算法的优势在于，局部聚类的运行时间与返回的聚类大小成正比，而与图的总大小无关。如果我们在应用局部聚类之前先将整个图加载到内存中，就会失去子线性运行时间的优势。因此，STAG 提供了 `AdjacencyListLocalGraph` 类，该类无需读取整个图即可访问存储在磁盘上的图。在此示例中，我们比较了在 `AdjacencyListLocalGraph` 对象和图对象上进行局部聚类的运行时间：前者在磁盘上对图进行本地访问，而后者将整个图加载到内存中。

我们按如下方式从随机块模型生成各种规模的图。给定参数 k、p 和 q，我们创建一个包含 k 个簇 $C_1, \dots C_k$ 的图，每个簇包含 1,000个顶点。对于每一对顶点 $(u, v) \in V \times V$，我们以概率 p 添加边 (u, v) u 和 v 属于同一簇时，否则以概率 q 添加。我们始终设置 $p = 0.01$ 和 $q = 0.001/k$。这确保了所构造簇的导通率始终接近 0.1。

我们针对一个随机起始节点和 20,000 的目标体积，在构建的图上执行局部聚类，并比较以下两种方法：

- **内存中**：在应用局部聚类之前，将整个图加载到内存中，作为一个图对象。
- **磁盘上**：使用`AdjacencyListLocalGraph` 对象，从磁盘上的文件中局部读取图。

图 3 展示了在一系列图规模下，各方法的局部聚类算法运行时间。这些结果表明，对于大型图，将整个图读入内存所产生的开销占据了算法运行时间的主要部分；直接从磁盘读取图则效率显著更高。

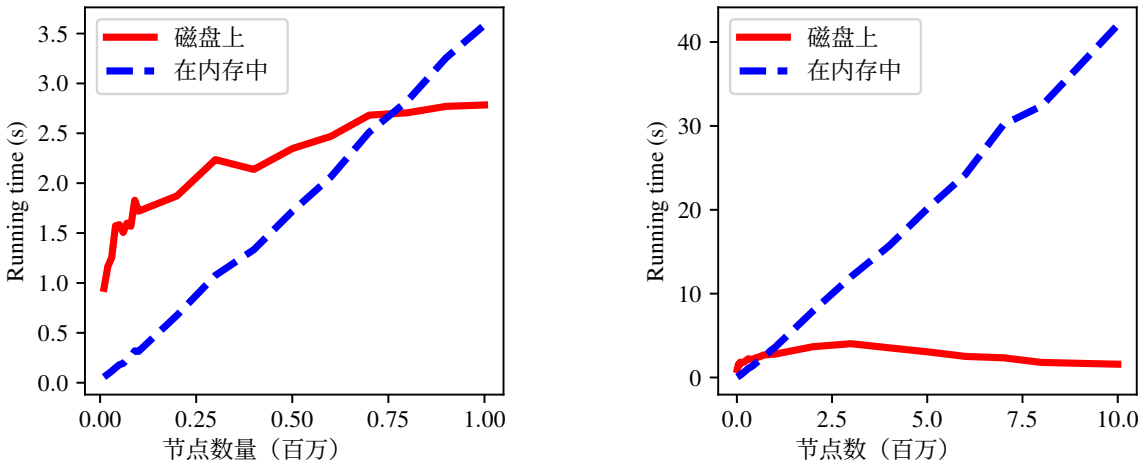

图 3：内存中和磁盘上的图执行局部聚类时运行时间的比较。对于大规模图，从磁盘局部读取图明显更快。

4.2 示例 2

在第二个示例中，我们展示了局部聚类在真实世界的图中查找相关节点集合的适用性。我们使用 `wiki-topcats` 数据集 [14]，这是一个由维基百科超链接构成的图，于 2011 年创建。该图包含维基百科排名前 100 的类别中的页面，共有 1,791,489 个顶点和 28,511,807 条边。该数据集可在 SNAP 数据集页面 [6] 以 EdgeList 文件的形式获取。我们使用 `edgelist_to_adjacencylist` 方法将 EdgeList 转换为 AdjacencyList，并使用`AdjacencyListLocalGraph` 对象进行局部聚类。

只需几行代码，STAG 就能让我们使用局部聚类创建“相关页面”搜索。提供一个搜索页面，并将目标体积设为 100，局部聚类便会返回一组与该搜索页面紧密相连的页面。图 4 展示了维基百科图中的一些局部聚类结果示例。

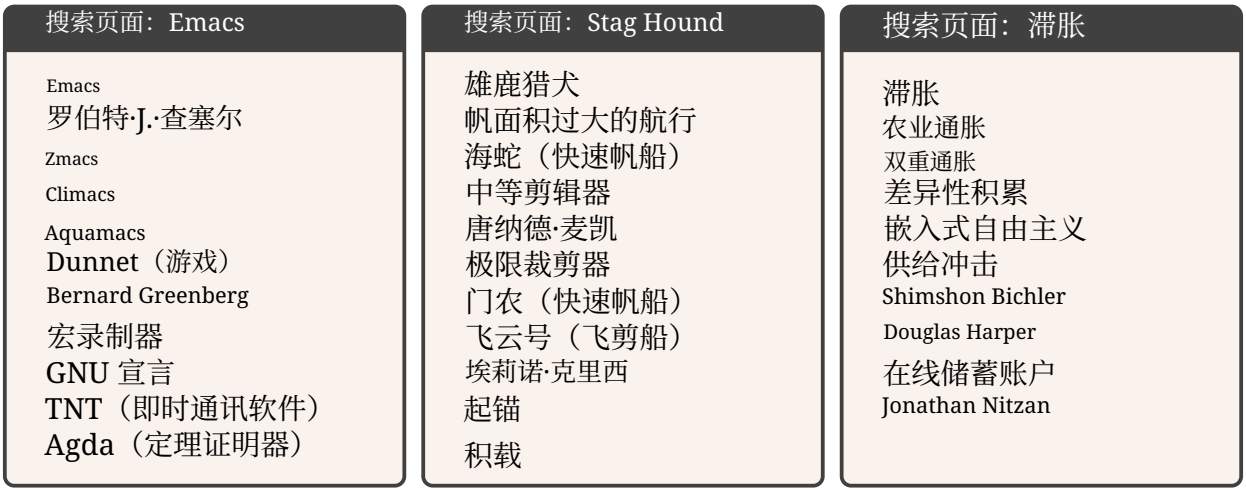

图 4：在维基百科数据集中找到的本地集群示例。

4.3 示例 3

在本示例中，我们演示如何对云中的Neo4j 数据库执行局部聚类。首先，我们按照 Neo4j 文档，使用AuraDB 服务 [9] 创建云数据库。我们使用 Neo4j 提供的“Movies”示例数据集。然后，通过使用 STAG Python 创建一个 Neo4j图对象，我们能够使用局部聚类搜索相关电影。列表1 展示了用于执行此搜索的完整 Python 脚本，图 5 展示了部分搜索结果。

```
1 import stag . neo4j 2 import stag . cluster 3 # 使用 Neo4j 数据库凭据初始化图对象 5
```

图 1：使用 Neo4j 数据库进行局部聚类的 Python 代码。



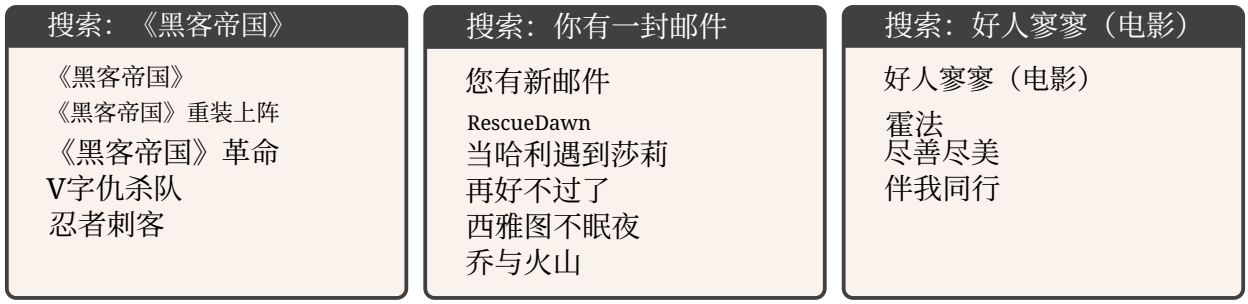

图 5：在 Neo4j Movies 数据集中找到的本地集群示例。

5 技术考量

本节将讨论 STAG 在设计和实现过程中所做的技术选择。我们将讨论所实现算法的选择、默认参数的设置，以及 `AdjacencyListLocalGraph` 类的实现，该类用于从磁盘上的文件中在本地读取图。

5.1 已实现的算法

已有许多用于局部聚类的算法被提出，包括基于 PageRank [1, 2], 演化集合过程 [3, 4], 和网络流的算法 [5]。我们选择实现 Andersen、Chung和Lang [1], 提出的基于 PageRank 的算法，并将其称为 ACL 算法。我们选择该算法是因为它相对简单、易于理解，并且在实践中有效。此外，ACL 算法的理论保证在常数因子范围内是最优的。[2] ACL 算法需要两个参数：

- α 参数控制个性化 PageRank 的“传送概率”；并且
- t该 ϵ 参数控制近似 PageRank 计算的近似误差 n.

STAG 提供 本地_簇_acl方法，允许用户直接指定参数 α 和 ϵ。

```
std::vector<长整数 长>STAG::本地_簇_ACL(STAG::LocalGraph*                图,
                                                long long                种子_顶点,
                                                双精度型                α,
                                                双精度                ε)
```

为方便起见，STAG 还提供了局部_簇方法，该方法只需要估算目标簇的体积。给定体积 γ，局部_簇方法使用参数 $\alpha = 1/2000$和 $\epsilon = 1/(20\gamma)$来运行 ACL 算法。

5.2 从磁盘局部读取图

STAG 的一个关键特性是 `AdjacencyListLocalGraph` 类，它可以从磁盘上的邻接表文件中以局部方式读取图的邻域信息。由于邻接表文件中的数据按节点 ID 排序，我们可以通过对邻接表文件进行二分查找，在 $O(\log(n))$ 时间内查询任意节点的邻居。如第 4.1 节所示，在将局部算法应用于大规模图时，相较于将整个图读入内存的开销，运行时间中增加这一对数因子要划算得多。

[2] Andersen 等人最初的分析 [1] 在近似保证中多了一个 $\log(n)$ 因子。这个因子并非必要，后续采用相同技术的分析已将其去除 [7, 12]。

参考文献

[1] ReidAndersen、Fan Chung和Kevin Lang。使用PageRank 向量进行局部图划分。在第 *47* 届 *IEEE* 计算机科学基础研讨会年会（*FOCS'06*），第 475–486 页，2006 年。[2] ReidAndersen、Fan Chung和Kevin Lang。使用 PageRank对有向图进行局部划分。在第 *5*届万维网图算法与模型国际研讨会（*WAW'07*），第 166–178 页，2007 年。[3] Reid Andersen、Shayan Oveis Gharan、YuvalPeres和Luca Trevisan。使用演化集进行近乎最优的局部图聚类。*ACM* 学报，63(2)：1–31，2016 年。[4] Reid Andersen 和 Yuval Peres。使用演化集在局部寻找稀疏割。在第 *41*届*ACM* 计算理论研讨会年会（*STOC'09*），第 235–244 页，2009 年。[5] Kimon Fountoulakis、DiWang和Shenghao Yang。用于局部图聚类的 p-范数流扩散。在第 *37* 届国际机器学习会议（*ICML'20*），第 3222–3232 页，2020 年。[6] JureLeskovec 和 Andrej Krevl。SNAP 数据集：斯坦福大型网络数据集集合。http://snap.stanford.edu/data，2014 年 6月。[7] Peter Macgregor 和 He Sun。寻找稠密连接簇的局部算法。在第*38* 届国际机器学习会议（*ICML '21*），第 7268–7278 页，2021年。[8] PeterMacgregor 和 He Sun。谱聚类的更严密分析及其拓展。在第*39* 届国际机器学习会议（*ICML '22*），第14717–14742 页，2022 年。[9] Neo4j。Neo4j图数据库。http://neo4j.org/，2023 年。[10] Andrew Y Ng、Michael I Jordan 和 Yair Weiss。关于谱聚类：分析与一种算法。在第 *15* 届神经信息处理系统进展会议（*NeurIPS'01*），第 849–856 页，2001 年。[11] Daniel A.Spielman和Shang-Hua Teng。用于大型图的局部聚类算法及其在近线性时间图划分中的应用。*SIAM*计算学报，42(1)：1–26，2013 年。[12] Yuuki Takai、Atsushi Miyauchi、Masahiro Ikeda和Yuichi Yoshida。基于PageRank 的超图聚类。在第*26*届 *ACM SIGKDD* 知识发现与挖掘国际会议（*KDD'20*），第1970–1978 页，2020 年。[13] Ulrike vonLuxburg。谱聚类教程。统计与计算，17(4)：395–416，2007年。[14] Hao Yin、Austin R Benson、JureLeskovec 和 David FGleich。局部高阶图聚类。在第*23*届 *ACM SIGKDD* 知识发现与挖掘国际会议（*KDD'17*），第 555–564 页，2017 年。

A 安装 STAG 依赖项

为方便起见，我们提供以下 Bash脚本，用于安装 STAG C++ 依赖项。截至撰写本文时，该脚本将下载并安装 Eigen 和 Spectra 库。

```
# 创建一个用于工作的目录mkdir



# 安装 Eigenwgethttps://gitlab.com/libeigen/eigen/-/archive/3.4.0/eigen-3.4.0.tar.


```




```
mkdir build _dir
cd build_dir
cmake ..
sudo make install
cd ../..

# 安装 Spectra
wget https :// github . com / yixuan / spectra / archive / v1 .0.1. tar .gz
tar xzvf v1 .0.1. tar . gz
cd spectra -1.0.1
mkdir build _dir
cd 构建 _dir
cmake ..
sudo make install
cd ../../
```

B 用户指南使用 STAG Python 的示例

本节包含第 3节中省略的示例代码，演示如何使用 STAG Python 进行局部聚类。

B.1 使用文件

以下示例演示了如何使用STAG Python 读取和写入邻接关系列表文件和边列表文件。

```
导入 stag. graphio # 读取邻接表图filename ="mygraph. adjacencylist "
```

my
fil
fil
fil

B.2 图类

The fo 以下示例展示了如何使用 STAG Pytho 创建一个AdjacencyListLocalGraph 对象 n.

```
导入 stag. graph# 创建图对象文件名 ="mygraph. adjacencylist "我的_图
```

B.3 本地集群

以下示例展示了如何使用 STAGPython，在存储于 AdjacencyList 文件中的图中查找本地集群。



```
导入 STAG . 图
导入 stag . 簇

# 创建图对象文件名  ="mygraph.adjacencylist"mygraph  =stag.


# 查找本地集群开始_顶点  =  1目标_体积  =  100簇  =STAG . 簇 .本地_簇（我的图 ,开始_顶点 ,目标_体


# 显示结果print( 簇 )

```


