大数据计算基础大作业报告

1181910201 李金宣

**第一章 绪论**

[**研究问题的背景**]

大数据时代，简单的二元关系已经不足以表达数据中的关系，如同一位教授可能参与多篇论文，而一篇论文也会有多个作者,在简单图中，容易丢失同一篇文章的多个作者。但是对于超图来说，我们利用其特性，能描述更为复杂的关系信息。所谓超图（Hyper-Graph），是一种广义的图，其特点是一条超边可以连接多个点。超图上的问题中，顶点覆盖问题作为经典的NP完全问题，在设备选址、信号覆盖等领域有着很多应用。

[**研究问题**][**本文的工作要解决的问题以及方法**]

问题1：大规模超图管理  
选用足够大的超图数据集，使用超图表征其关系，并对该大规模超图进行存储并实现基本的超边、顶点的增加和删除功能。  
问题2：超图顶点覆盖问题  
所谓顶点覆盖，是顶点的一个集合，使得超图中每条超边所包含的顶点集都与该集合有非空的交集。超图顶点覆盖问题定义为：给出超图G和上界k（k必须是在从0到G的顶点数减一之间），判断G是否有不超过k个顶点的顶点覆盖。基于选用的大数据处理框架，设计并实现大规模超图上的顶点覆盖问题算法。  
问题3：超图上的受限顶点覆盖问题  
大规模超图上顶点覆盖存在一个严重的问题，即囿于过大的超图规模，精确顶点覆盖问题的耗时和解的规模都给下游任务带来了很大的负担，该问题的一种解决思路是，选用不超过k个顶点覆盖c%的超边作为近似解输出给下游任务。

[**本文的贡献**]

[1].实体之间的复杂关系可以使用超图非常有效地建模。超图通过允许超边包含两个或更多实体来对现实世界的数据进行建模。超图的聚类使我们能够将相似的实体组合在一起。虽然大多数现有算法仅考虑超图的连接结构来解决聚类问题，但我们可以通过考虑与实体相关的各种特征以及实体之间的辅助关系来提高聚类性能。此外，如果一些标签是已知的，我们可以进一步提高聚类性能，并将它们合并到一个聚类模型中。在本文中，我们为超图提出了一个半监督聚类框架，它不仅能够轻松地合并实体之间的多种关系，还能够轻松地合并来自不同来源的实体的多个属性和内容。此外，通过显示超图归一化切割与加权核 K-Means 之间的密切关系，我们还开发了一种有效的多级超图聚类方法，该方法为我们的半监督多视图聚类算法提供了良好的初始化。实验结果表明，我们的算法在检测真实聚类方面是有效的，并且明显优于其他最先进的方法。我们还开发了一种高效的多级超图聚类方法，该方法为我们的半监督多视图聚类算法提供了良好的初始化。实验结果表明，我们的算法在检测真实聚类方面是有效的，并且明显优于其他最先进的方法。我们还开发了一种高效的多级超图聚类方法，该方法为我们的半监督多视图聚类算法提供了良好的初始化。实验结果表明，我们的算法在检测真实聚类方面是有效的，并且明显优于其他最先进的方法

[2].在本文中，我们介绍了加权集覆盖和最大覆盖的自然概括，称为大小约束加权集覆盖。输入是 n 个元素的集合、元素上加权集的集合、大小约束 k 和最小覆盖率 ŝ；输出是最多 k 个集合的子集合，其并集至少包含 ŝn 个元素，并且权重之和最小。我们证明了这个问题的逼近难度，并且我们提出了具有可证明的质量保证的有效逼近算法，这些算法是最好的。在许多应用程序中，元素是数据记录，可供选择的集合是从属性值的组合（模式）中派生出来的。我们为这种特殊情况提供优化技术。

1. **系统/方法框架**

在考虑组交互时，超图是一种有用的数据表示。 考虑一个作者网络，其中有一组作者——即作者 A、作者 B 和作者 C，他们共同撰写了一份出版物。使用传统的图边来表示这些信息，其中每条边连接两个作者以显示共同作者关系带来两个问题。 首先，它引入了误导性信息，我们无法区分三位作者的单一出版物的情况； 其中三位作者合着了三篇出版物（即作者A和作者B合着的出版物一；作者B和作者C合着的出版物二；作者A和作者C合着的出版物三）。 使用传统图表示表示超图的第二个缺点是图存储中数据的膨胀。 例如，在我们之前的网络中，我们将需要添加 N\*(N-1) 条边来表示 N 个作者之间对于单个出版物的合着。 通过学习如何将数据编码到这些结构中以及如何操作它们以提取有意义的数据，我们可能能够获得有关以前不可能实现的数据的见解。

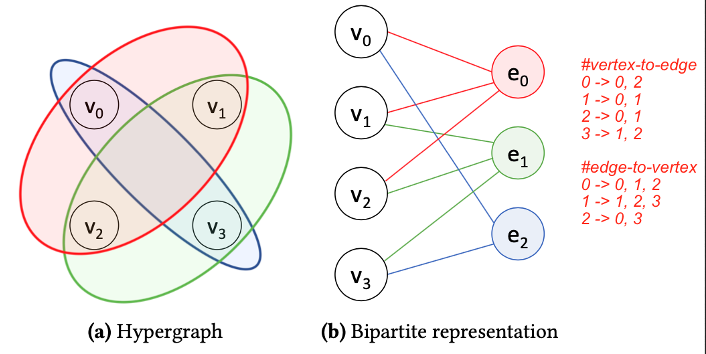
本研究的目的之一是了解超图并找到它们在大型网络中的适应性。 我们已经探索了超图的几种表示方式，并了解了每种表示方式的优缺点。 在追求这个方向的过程中，我们对在这个方向上工作的现有库进行了几次探索性搜索。 这些库允许使用不同的编码和超图的实现来对不同的算法进行基准测试。 但是这些库存在缺陷——尤其是当数据源很大且不固定时，如推文、邮件、社交网络交互或财务数据。

该算法在单台机器上的大型数据集上运行良好。 越来越需要在分布式设置中处理超图算法，以应对超出内存限制的数据挑战。

在我们的工作中，我们使用 Apache Spark 将现有的并行超图算法实现移植到分布式环境中。 我们构建此 Python 模块以使用 Apache Spark 在分布式设置中探索超图实现。

## 超图配置

## 表示超图的一种方法是在每个超边的所有顶点对之间创建一个团，并将结果存储为正则图。 正如我们之前提到的，这会导致一些问题，例如信息丢失。 此外，空间开销明显高于原始超图的空间开销。 另一种方法是使用二部图表示，其中一个分区中的参与顶点和另一个分区中的超边。 我们使用这种二部图表示保留超图信息，其中超边连接到参与顶点，如下图所示。



**安装需要以下库**

1. Java 7+
2. Apache Spark（在 2.4 版中测试的代码） <https://spark.apache.org/docs/latest/>
3. 图形框架（在 0.8.1 版中测试） graphframes： https://graphframes.github.io/graphframes/docs/\_site/index.html

基于当前用于超图实验和实现的资源，构建一个允许在分布式环境中为大型网络创建和测试超图的库，以及一系列超图算法，将是一项值得的努力。 我们计划检查我们当前实现的性能，看看我们如何改进它们。 进一步，我们将探索超图分区研究对超图算法分布式计算的影响

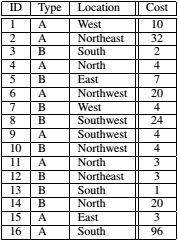
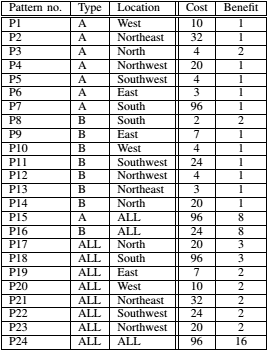
**参考：**

1. Pacific Northwest National Laboratory, Python package for hypergraph analysis and visualization., (2020), GitHub repository, <https://github.com/pnnl/HyperNetX>
2. Leland McInnes, A library for hypergraphs and hypergraph algorithms., (2015), GitHub repository, <https://github.com/lmcinnes/hypergraph>
3. Julian Shun. 2020. Practical parallel hypergraph algorithms. In Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '20). Association for Computing Machinery, New York, NY, USA, 232–249. DOI:<https://doi.org/10.1145/3332466.3374527>
4. Y. Gu, et al.,"Distributed Hypergraph Processing Using Intersection Graphs" in IEEE Transactions on Knowledge & Data Engineering, vol. , no. 01, pp. 1-1, 5555. doi: 10.1109/TKDE.2020.3022014
5. **相关算法及技术**

在本文中，我们研究了加权集覆盖和最大覆盖率的泛化，称为大小约束加权集覆盖率。输入是一个集合n ，加权集的集合的大小、约束 、和最小覆盖率 。输出是子集合，其并集至少包含 n元素，其权重之和最小。我们关注一个实际的特殊情况，其中输入包括n数据记录和可供选择的集合由属性值的连接描述，我们称之为模式。

为了说明对大小约束加权集覆盖问题的必要性，让我们首先了解经过充分研究的加权集覆盖问题的局限性，该问题具有许多应用，从设施位置到劳动力创建再到交易汇总。考虑现实世界实体（例如，个人、企业列表、销售交易）的数据集，每个实体都由一个 ID 标识，由分类属性类型和位置描述，并与数字属性成本相关联。表Ⅰ显示了一个示例n = 16实体。在这里，实体集对应于属性类型和位置上的数据立方体模式，并指定个人劳动力组、企业营销活动、销售交易摘要等。此外，集合/模式的成本（权重）以特定于应用程序的方式计算为集合中实体成本的函数，并对应于部署劳动力组的成本，进行营销活动等。出于说明目的，我们使用集合中实体的最大成本作为集合的成本。表 II显示了表 I 中实体数据的所有可能模式，以及它们的成本和模式涵盖的实体数量（称为收益）。

**表 I：**真实世界实体数据集 表Ⅱ：实体数据集中所有可能的模式

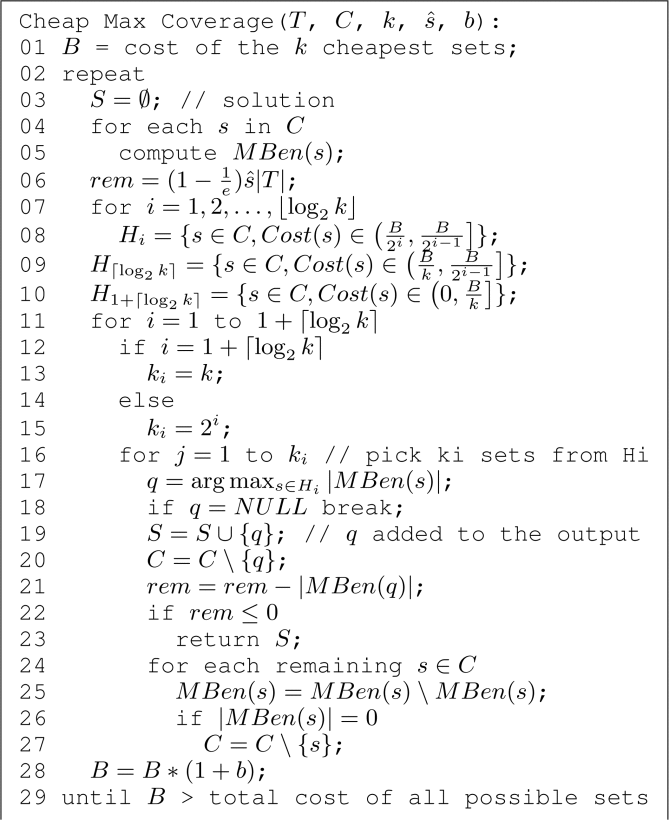
（部分）加权集覆盖问题试图使用具有最小成本（权重）总和的集合来覆盖实体的指定部分。在提到的应用程序中，它可能会产生一组最便宜的营销活动，这些活动可以覆盖特定部分的企业，或者确定一组包含特定部分销售交易的最便宜的摘要。我们的例子中要覆盖的实体的期望分数是9/16.部分加权集合覆盖问题的解决方案将返回 7 个集合/模式，总成本为 24。返回的集合是数据集大小的很大一部分，对于应用程序来说可能太多了，应用程序可能想要指定返回集合数量的上限，例如，设施位置的数量或营销活动的数量企业。这个对返回集合数量的额外限制正是我们的问题所指定的。特别是，大小受限的加权集覆盖问题寻求使用（至多）具有最小成本（权重）总和的指定数量的集合的集合来覆盖（至少）实体的指定部分。如果k = 2 是对要返回的集合数的约束，同时覆盖一小部分 9/16 实体，最佳解决方案由集合 P6 和 P16 组成，总成本为 27。虽然此解决方案比加权集合覆盖返回的解决方案稍微贵一点，但它是一个小得多的解决方案。

因此，我们问题的新颖之处在于我们需要同时优化高覆盖率、低总成本和小尺寸（最多 克套）。虽然满足这三个目标中的两个是已知的，但这还不足以解决我们的问题。据我们所知，尽管这个问题有很多应用，但之前还没有研究过。在设施规划中，城市可能需要为医院寻找最佳位置，以最大限度地降低总建设成本并确保所需的人口比例至少靠近一个位置。由于员工人数限制或分区限制，最多克可以建造这样的对象。类似地，在视频/博客推荐系统、设施位置的变体和劳动力创建问题中也有应用，其中集合覆盖自然出现，并且具有大小和重量约束是有用的。请注意，如果我们想要最便宜的解决方案 k = 2 集合，对覆盖的实体数量没有限制，解决方案将由 P6 和 P8 组成，它们只覆盖了一小部分 3/16实体。另外，如果我们想要任何解决方案k = 2 集，和一个 9/16 覆盖要求，返回的解决方案（例如，P11 和 PI5）的成本很高（120）。

我们现在为大小受限的加权集覆盖问题提出了两种近似算法（部分 VA和VB），然后是它们对由属性值模式定义的集合的特殊情况（部分 VC）的优化

1. 简便的最大覆盖率 (CMC)
2. 算法说明

图1：



CMC 算法基于贪婪的部分最大覆盖启发式算法，它选择具有最高边际收益的集合，即那些覆盖最多未覆盖元素的集合。为了合并成本，我们猜测一个最优解的权重总和的值，称之为 B，并试图找到一个权重总和近似为 B 的解决方案。我们从 B 的一个小值（等于k 个最便宜集合的权重），这可能不足以覆盖所需的元素数量，如有必要，我们会尝试更大的 B。为了避免具有大权重的集合，我们根据它们的权重将集合划分为级别，并从昂贵的级别限制允许的集合数量。

图 1 显示了 CMC 的初始版本，它可以选择最多 5k 个集合（我们将很快展示一个使用最多 (1+ϵ)k 个集合的修改版本）。输入参数是元素集合 T、集合集合 C，每个都与一些成本相关联，最大解决方案大小 k，覆盖阈值 s^，以及一个参数 b，它决定了如果当前预算增加了多少成本预算不会产生具有足够覆盖范围的集合。

第 2-29 行中的循环尝试计算给定 B 值的解决方案。第 4-5 行计算每个集合的边际收益（最初等于收益）。第6行设置需要覆盖的元素数量；在第 V-A2 节中，我们解释了 CMC 仅保证 (1−1e)s^|T| 的覆盖范围，因此我们在找到 B 的值后停止，该值给出一组集合，其并集至少覆盖那么多元素

第 7-10 行根据它们的成本（权重）将集合分为 1+[logk] 个级别。第一层 H1 包含成本介于 B2 和 B 之间的集合；第二层 H2 包含成本在 B4 和 B2 之间的集合，依此类推；最后一层 [logk]+1 包含成本最多为 Bk 的剩余集合。在构建解决方案时，我们只使用来自 Hi,i=1,2,⋯,[logk] 的 2i 个集合和来自最后一层（第 12-15 行）的 k 个集合。然后我们运行贪婪的最大覆盖率启发式算法，从第一层 H1 中选择最多两个覆盖最多未覆盖元素的集合，从 H2 中最多选择四个覆盖最多未覆盖元素的集合，依此类推（第 11-27 行）。在向输出添加一个集合后，第 24-27 行更新剩余集合的边际收益，并删除那些边际收益现在为零的集合。一旦达到所需的覆盖范围，我们就返回当前的集合集合（第 22-23 行）；如果我们无法使用当前成本预算 B 达到覆盖阈值，我们将预算增加 1+b 倍（第 28 行），并使用 B 的新值找到新的集合集合（回到第 2 行）

我们现在给出一个使用表 I 中数据集的 CMC 示例（请注意，CMC 适用于任意集，但该示例使用模式）。我们通过它们的数字来指代模式（回想一下表 II）。设 k=2，(1−1e)s^=916，b=1。

由于两个最便宜的模式的总成本为 5，因此我们在第一次迭代中使用 B=5。这给出了成本在 3 到 5 之间的 H1，以及成本低于 3 的 H2；我们最多可以从这两个级别中的每一个中选择两个模式。从 H1 开始，我们首先选择模式 P3，这是唯一一个成本在 3 到 5 之间且覆盖两条记录的模式；所有其他的只包含一个记录。然后我们更新与模式 P3（即 P15、P17 和 P24）相交的模式的边际收益。接下来，我们选择模式 P5、P6、P10、P12 或 P13，这取决于我们如何打破平局（并更新其余候选者的边际收益）。我们现在已经涵盖了 3 个记录。移至 H2，只有模式 P8 处于此级别，有两个好处，因此我们选择它并跳出内部 for 循环（第 18 行）。当 B=5 时，我们只能覆盖 5 条记录，因此我们需要尝试更大的 B。

由于 b=1，我们在第二次迭代中使用 B=10。我们现在有成本在 6 到 10 之间的 H1，以及成本低于 6 的 H2。和以前一样，我们最多可以从每个级别中选择两种模式。从 H1 开始，我们选择模式 P19 和 P20（并在每次选择后更新剩余模式的边际收益），因为它们都覆盖两个（未覆盖的）记录，而 H1 中的其他模式各只覆盖一个。对于 H2，我们选择 P3 和 P8；它们各覆盖两条记录，而 H1 中的其他模式各覆盖一条。这样总共覆盖了 8 个记录，这是不够的。

在第三次迭代中，B=20，给出成本在 11 到 20 之间的 H1 和成本低于 11 的 H2。我们首先从 H1 中选择模式 P17，它包含 3 个记录。接下来，我们选择模式 P23，它包含 2 个（未覆盖的）记录。移至 H2，有 3 种模式涵盖 2 个未发现的记录（其他仅涵盖一个）：P8、P19 和 P20。选择其中任意两个可以覆盖九个，我们就完成了。

参数设置：

CMC 需要一个 b 值来控制运行时间和准确度之间的权衡（b 越大意味着将为各种成本预算 B 创建更少的候选解决方案，但解决方案的总成本可能更高），以及一个值 对于 ϵ 控制选择的集合数量和准确性之间的权衡（集合越少，总成本可能离最佳解决方案越远）。我们将通过实验评估这两个参数的影响。

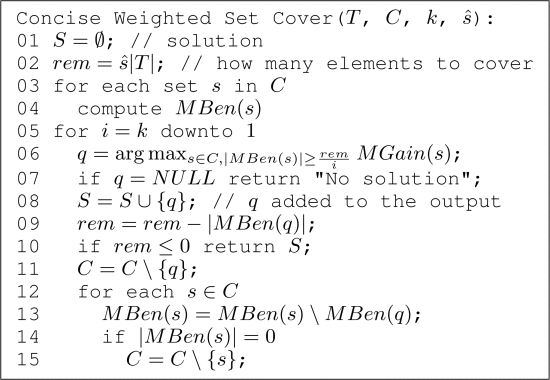
B. 简明加权集合覆盖 (CWSC)

CMC 提供近似保证，但它可能选择多于 k 个集合。下一个算法——CWSC——最多找到 k 个集合，但它没有近似保证。我们从选择具有最高边际增益的集合的部分加权集合覆盖启发式开始，这优化了成本和覆盖范围。此外，在第一次迭代中，我们只考虑覆盖至少 1k 需要覆盖的元素数量的集合；在第二次迭代中，我们只考虑覆盖至少 1k-1 个需要覆盖的剩余元素数量的集合；等等。

图 2 给出了伪代码。第 3-4 行计算每个集合的边际收益（最初等于收益）。第 6 行从那些覆盖足够多未覆盖元素的集合中选择具有最高边际增益的集合；如果我们找不到这样的集合，我们不返回任何解决方案（或者我们可以返回包含 T 中所有元素的集合的“默认”解决方案）。在第 9 行，我们更新了需要覆盖的元素数量；在第 10 行，一旦我们覆盖了所需的元素部分，我们就返回集合 S 的当前集合；在第 11 行到第 15 行，我们更新了剩余集合的边际收益。请注意，我们现在可以删除边际收益为零的集合（第 14-15 行）。

以 CWSC 为例，回忆表 I 中的数据和表 II 中的模式；请注意，CWSC 适用于任意集合。再次假设 k=2 和 s^=916。在第一次迭代中，我们有k=2个模式可以选择，需要覆盖9条记录，所以我们只考虑覆盖至少4.5条记录的模式。只有三种这样的模式：P15、P16 和 P24，其中 P16 的边际收益最高，为 824。然后我们更新所有与 P16 相交的模式的边际收益，并删除那些现在边际收益为零的模式（即模式 P8 到 P14 和 P16）。由于 P16 的边际收益为 8，在第二次迭代中，我们只剩下一个模式，只有一个额外的记录要覆盖。我们考虑边际收益至少为 1 的所有模式，并选择收益最高的模式，即 P3（边际收益为 24）。

图2：



CMC 和 CWSC 的运行时间分析

CMC 最坏情况的运行时间O([min(log1+b(W/Wmin), m)]∗[min(n^, k)logm]) 其中 n^=s^n 是所需的覆盖率，m是集合的总数，W 是集合的总权重，Wmin 是任何集合的最小权重。第一个因素来自所需的最大猜测次数；请注意， b 的值在这里起作用。如果比率 W/Wmin 是多项式有界的，那么第一个因素只会对运行时间造成对数增长。如果这个比率可以是指数的，那么当然猜测的总数最多可以是系统中的集合总数。剩下的因素来自这样一个事实，即对于每个猜测的最佳值，其余的可以在与最大-k 覆盖率成比例的时间内完成。类似地，CWSC 的最坏情况运行时间是 O(min(n^, k)logm+mlogm)。第一个因素再次来自类似于 max-k 覆盖率的分析，第二个因素来自以下事实：如果它们似乎都不适合覆盖率要求，我们可能不得不丢弃所有集合

**第四章 实验**

在评估 CMC 和 CWSC 的效率和准确性。这些算法是在ç ++ 我们在配备 128GB 内存的四核 AMD Opteron 处理器 8356 的服务器上进行了实验。我们使用具有模式属性和数字属性的真实数据集来测试我们算法的效率和准确性。该数据集称为 LBL 1，包含大约 700, 000 个 TCP 连接跟踪。它包含五个模式属性（协议、本地主机、远程主机、结束状态和标志）和一个数字属性，表示我们用于模式权重的会话长度。我们还创建了几个基于 LBL 的合成数据集来测试我们算法的各个方面，详情如下。

图 5绘制了未优化和优化版本的 CMC 和 CWSC 的运行时间作为数据大小的函数；我们通过从 LBL 数据集中随机抽样来改变数据大小。即使对于小数据量，优化算法的速度也是未优化算法的两倍。优化算法的运行时间随数据大小呈亚线性增长。正如预期的那样，CWSC 比 CMC 更快，因为 CMC 需要为预算阈值的各种值创建多个候选决方案b在它找到可行的解决方案之前

**图 5：**运行时间与数据大小

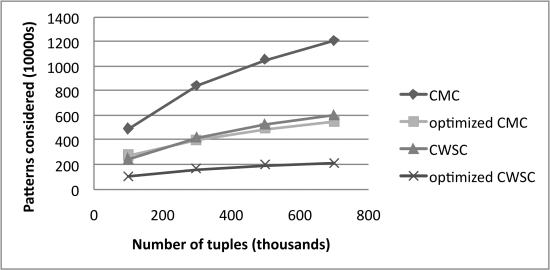
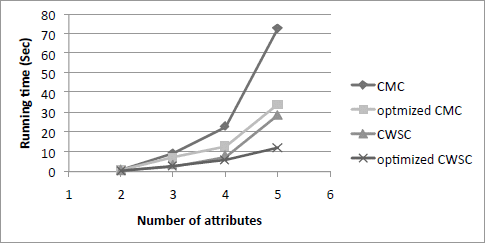


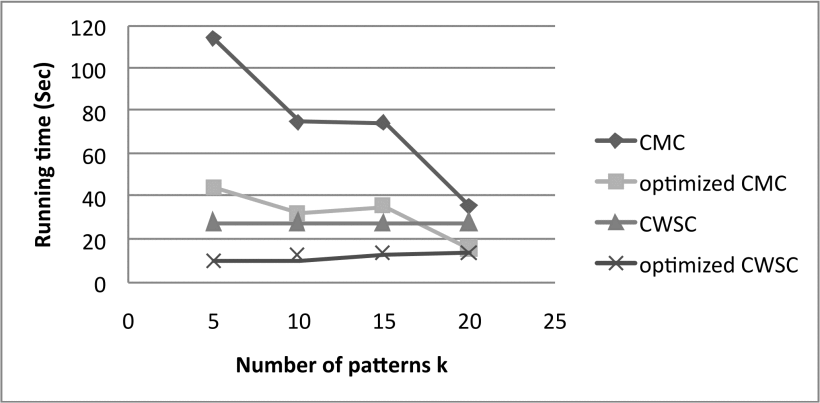
图 6解释了为什么我们的优化提高了性能：考虑的模式数量要少得多。同样，建议优化的效果随着元组数量的增加而增加。请注意，对于 CMC，考虑的模式数是为每个值考虑的模式之和b. 这就是为什么 CMC 的曲线高于 CWSC 的曲线，即使它们使用相同的数据集。

**图 6：**考虑的模式数量与数据大小



在图 7 中，我们展示了优化算法在模式属性数量方面的可扩展性。我们通过从 LBL 数据集中一次删除一个模式属性来执行此实验。和以前一样，优化后的算法优于同类算法，尤其是在属性数量增加时。同样，这是因为优化算法在计算解决方案时考虑的模式更少

**图 7：**运行时间 vs. 属性数量



正如图8所示，允许在溶液中的模式的最大数目k，对性能有有趣的影响。对于 CWSC，运行时间增加为k增加，因为它需要更多的迭代来计算解决方案。对于 CMC，运行时间减少为k增加。这是因为在较小的值，很难找到少量低成本高收益的模式，因此CMC尝试了许多值 b在它找到可行的解决方案之前。作为k 增加，更容易找到成本更低的解决方案，即预算阈值的较低值 b. 和以前一样，优化算法的性能比标准算法高两倍或更多。

**图 8：**运行时间与允许的最大模式数

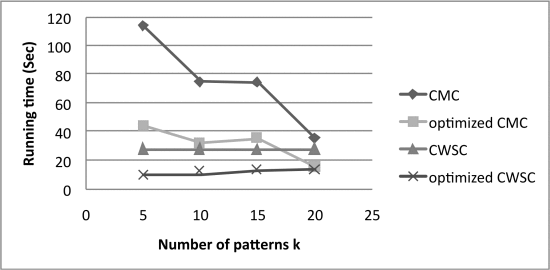
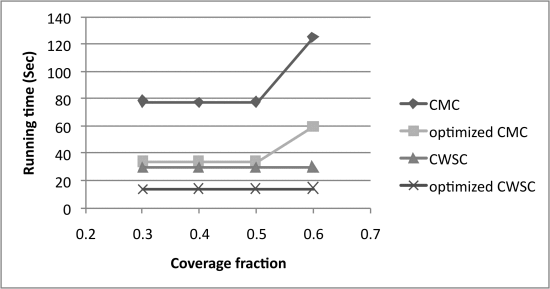


图 9显示增加覆盖率秒^不影响CWSC的运行时间，但会影响CMC的运行时间；此外，优化算法继续优于非优化算法至少两倍。对于 CWSC，这是有道理的，因为迭代次数仅取决于模式的最大数量，并且不断增加秒^只是意味着 CWSC 将在每次迭代中选择具有更高覆盖率的模式。另一方面，CMC 的运行时间随着秒^ 增加，因为找到一个可行的解决方案变得更加困难，该解决方案涵盖了许多元组并且成本最多为b，因此 CMC 必须增加b并多次重新计算解决方案。

**图 9**：运行时间与覆盖阈值 秒



**第五章 相关工作**

在我们的问题中，我们需要同时优化覆盖率、成本（权重总和）和解决方案大小（最多 k 个集合）。先前关于集合覆盖的工作优化了覆盖和成本或覆盖和大小，因此无法解决我们的问题，例如，加权集合覆盖（优化固定覆盖的成本）、最大覆盖（优化固定大小的覆盖）和预算最大覆盖（优化覆盖）固定成本）[11]。

已经有工作使用很少的模式覆盖数据集 [1]、[4]、[9]、[14]、[24]。但是，这些方法针对覆盖率和大小进行了优化，但没有针对成本进行优化，例如，它们可能会返回联合覆盖最多记录的 k 个模式，但并不能解决我们的问题。

在 [13] 中，在项目集的上下文中，提出了一种解决方案，该解决方案侧重于覆盖率和成本，而不是大小。他们的目标是在不限制模式数量的情况下，在成本上限（描述长度）下最大化发现模式的信息内容。

在预算最大覆盖率中，我们需要在所选集合的权重总和上覆盖固定预算内的最多元素。这个问题的贪心算法根据它们覆盖的未覆盖元素的数量除以它们的权重来选择集合[11]。虽然看起来这个算法可以通过在它选择 O(k) 个集合后停止来扩展来解决我们的问题，但可以构建示例表明它的覆盖率可能非常低。假设我们最多要返回 ck（对于常数 c≥1）个集合，其中最优解只能选择 k 个集合。我们有 100% 的覆盖要求。我们有元素 1, 2, ..., Ck, 集合 {1},{2},...,{ck},{1,2, ..., C},{1+C, 2+C, ..., 2C}, ...,{1+(k−1)C,...,kC}。第一个 ck 设置每个覆盖 1 元素并具有权重 1，而最后 k 设置每个覆盖 C 元素并具有权重 (C+1)。假设 c<<C。 [11] 的算法如果允许选择 ck 集，将只选择单个基数集并具有覆盖率 ck，而最佳解决方案将选择最后 k 个集具有 100% 的覆盖率。因此，与最佳解决方案相比，[11] 的适应可以具有任意小的覆盖范围。

解决加权集覆盖问题的一种自然技术是通过整数线性规划对其进行建模，考虑其线性松弛，然后将分数解四舍五入到附近的整数最优值 [22]。然而，除非 k 很大，否则使用这种方法以高概率（或确定性算法）获得有保证的性能可能会违反基数约束超过 (1+ϵ) 因子。

有大量关于模式挖掘的工作，但主要集中在寻找可用于估计其他模式的频率（或其他统计数据）的代表性模式集；参见，例如，[17]、[20]、[21]、[25]。然而，我们在达到一定覆盖率的同时最小化成本的优化标准是不同的，因此现有算法不适用。另一个流行的目标是找到一组相互独立或令人惊讶的模式；这有助于避免冗余模式并减少解决方案的大小 [16]、[19]、[23]。同样，我们的目标是不同的，因为我们需要具有高覆盖率和低成本的模式，无论它们有多令人惊讶。

已经提出了算法框架来根据各种效用函数 [3]、[12] 找到 k 个最佳模式。然而，尚不清楚如何将这些扩展到我们的问题并制定结合我们的成本和覆盖率概念的效用函数。相反，我们直接针对每个成本和覆盖范围进行优化，同时将模式数量限制为 k。

AlphaSum 技术 [5] 优化了大小、覆盖范围和成本。它生成一个包含 k 个非重叠模式的摘要，覆盖整个数据集并具有最低成本，这对应于已汇总的属性域的大小。例如，具有“age=Teenager”的模式表示年龄在 13 到 19 之间的元组，因此其成本为 7。因此，除了将解决方案限制为非重叠模式之外，[5] 中使用的成本定义是不同的。

**第六章 结论**

正如预期的那样，增加 b趋于增加 CMC 获得的解决方案的成本，但减少了运行时间。至于效果 ε, CMC 的运行时间增加为 ε 由于维护更多模式级别的开销而增加 H。虽然不同ε 在这个实验中对解决方案的总成本没有明显的影响，较小的值 ε 给出了模式较少的解决方案（对于 ε = 1，最小的有七个模式，而对于 ε = 2, 模式数为 10, 等于 k）。这与作用是一致的ε，这是为了权衡总成本的模式数量。为了进一步研究 CWSC 获得的解决方案的质量，我们基于**LBL**数据集创建了两组合成数据集。在第一组中，对于每个元组，我们替换了数字属性*session length*的值，称之为米，通过随机选择之间的值 ( 1 − δ)米 和 ( 1 + δ)米 对于不同的值 δ零和一之间。在第二组中，我们从均值为 2（原始值均值的对数）和 1 到 4 之间的各种标准差的对数正态分布生成会话长度的新值；然后，我们用相同排名顺序的新会话长度替换了原始会话长度。使用这两组合成数据集的结果与表 IV 中报告的结果相似：CWSC 继续返回总成本不大于 CMC 的解决方案，具有各种值b 和 ε.此外，CMC 和 CWSC 的优化版本在合成数据上继续优于未优化版本的两倍以上，类似于图 5到图9。

### 与最优解的比较

我们现在将 CMC 和 CWSC 获得的解决方案的成本与最佳解决方案的成本进行比较。我们使用了 LBL 数据集的小样本，这使我们能够使用穷举搜索获得最佳解决方案。当我们使用较小的值时，CMC 找到了一个最优解b 和 ε. CWSC 几乎总能找到最佳解决方案，但有一个例外：对于秒^= 0.5 和 k = 5. 一个最优解的代价是 8，这也是 CMC 发现的b= 1 和 ε = 1. 然而，CWSC 返回了一个稍差的解决方案，成本为 9，CMC 对于较大的值也是如此b和 ε.

在本文中，我们提出并解决了一个广义集覆盖问题大小受限的加权集覆盖，我们发现 克具有最低权重总和的集合，其联合覆盖了元素的一部分。我们证明了所提出的算法之一——Cheap Max Coverage——可以保证产生近乎最优的解决方案。我们通过实验表明，另一种算法——Concise Weighted Set Cover——在实践中更有效，与Cheap Max Coverage 一样有效，尽管其最坏情况的性能无法保证。此外，对于模式集的特殊情况，我们展示了所提出算法的优化版本的显着性能优势。未来工作的一个有趣方向是研究大小约束加权集覆盖的增量版本，其中解决方案必须在新元素到来时持续维护。另一个有趣的问题是如何处理与每个集合或模式相关的多个权重。

1. **基于Apache spark的超图相关算法项目工程实现(具体操作可见说明文档readme.md，报告中不过多阐述）**

## 介绍

在考虑组交互时，超图是一种有用的数据表示。 考虑一个作者网络，其中有一组作者——即作者 A、作者 B 和作者 C，他们共同撰写了一份出版物。 使用传统的图边来表示这些信息，其中每条边连接两个作者以显示共同作者关系带来两个问题。 首先，它引入了误导性信息，我们无法区分三位作者的单一出版物的情况； 其中三位作者合着了三篇出版物（即作者A和作者B合着的出版物一；作者B和作者C合着的出版物二；作者A和作者C合着的出版物三）。 使用传统图表示表示超图的第二个含义是图存储中数据的膨胀。 例如，在我们之前的合着网络中，我们将需要添加 N\*(N-1) 条边来表示 N 个作者之间对于单个出版物的合着。 通过学习如何将数据编码到这些结构中以及如何操作它们以提取有意义的数据，我们可能能够获得有关以前不可能实现的数据的见解。

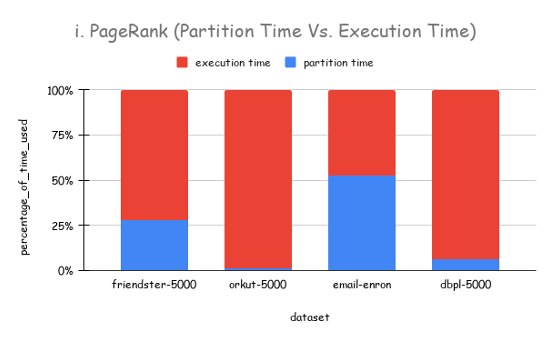
本研究的目的之一是了解超图并找到它们在大型网络中的适应性。 我们已经探索了超图的几种表示方式，并了解了每种表示方式的优缺点。在追求这个方向的过程中，我们对在这个方向上工作的现有库进行了几次探索性搜索。 这些库允许使用不同的编码和超图的实现来对不同的算法进行基准测试。但是这些库存在缺陷——尤其是当数据源很大且不固定时，如推文、邮件、社交网络交互或财务数据。

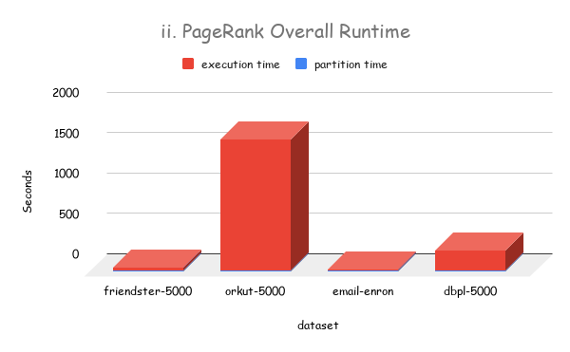
研究人员通过提出并行超图算法 进行了多次尝试，该算法在单台机器上的大型数据集上运行良好。 越来越需要在分布式设置中处理超图算法，以应对超出内存限制的数据挑战。使用 Apache Spark 将现有的并行超图算法实现移植到分布式环境中。 我们构建此 Python 模块以使用 Apache Spark 在分布式设置中探索超图 PageRank 实现。

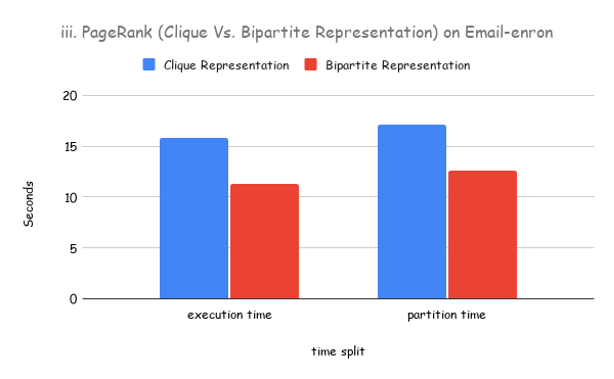
## 初步结果

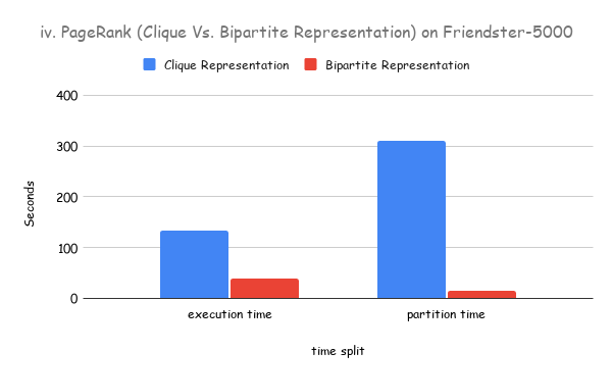
我们为超图实现了分布式 PageRank 算法。执行引擎 Apache Spark 2.4.7 是通过使用 Databricks GraphFrames API 0.8.1 在 Python 中实现的，Databricks GraphFrames API 0.8.1 是一个图形处理框架，在其上实现了超图算法

在图 2(i) 中，我们比较了 PageRank 算法的分布式分区和算法执行时间。 对于所有大型数据集，分区成本明显低于算法执行时间。 在图 2(ii) 中，我们绘制了 PageRank 算法对不同数据集的整体性能。 在这里，orkut-5000 图形与其他图形相比需要更长的运行时间。 原因是，orkut-5000 具有更大的超边（即超边中参与节点的数量）。 例如，与 dbpl-5000 相比，orkut-5000 的超边要大 12 倍。 另一方面，dbpl-5000 的运行速度仅比 orkut-5000 快 6 倍。 这实际上意味着在分布式环境中运行大型超图的用处。 在图 2(iii) 和图 2(iv) 中，我们分别比较了 email-enron 和 Friendster-5000 的两种超图表示（例如集团和二分表示）。 我们无法在 orkut-5000 和 dbpl-5000 的超图的集团实现上运行 PageRank，因为这些图非常大，无法将它们放入我们的测试平台设置中。 这进一步表明了我们研究的重要性。 图 2(iv) 表明，与执行时间相比，对于较大的超图，二分表示中的分区时间显着减少。 这在意料之中，因为在集团表示中，我们必须放置很多边，因此以指数方式增加分区时间。









参考文献

1. JoyceJiyoungWhang,RundongDu,SangwonJung,GeonLee,BarryL.Drake,  
   QingqingLiu,SeonggooKang,HaesunPark:MEGA:Multi-ViewSemi-Supervised  
   ClusteringofHypergraphs.Proc.VLDBEndow.13(5):698-711(2020)

[2]LukaszGolab,FlipKorn,FengLi,BarnaSaha,DiveshSrivastava:Size-  
ConstrainedWeightedSetCover.ICDE2015:879-890

**1.**R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data", Data Min. Knowl. Discov, vol. 11, no. 1, pp. 5-33, 2005.

**2.**P. Berman and M. Karpinski, "On Some Tighter Inapproximability Results (Extended Abstract)", ICALP, pp. 200-209, 1999.

**3.**B. Bringmann and A. Zimmermann, "One in a million: picking the right patterns", Knowl. Inf. Syst., vol. 18, no. 1, pp. 61-81, 2009.

**4.**S. Bu, L. V. S. Lakshmanan and R. T. Ng, "MDL Summarization with Holes", VLDB, pp. 433-444, 2005.

**5.**K. S. Candan, H. Cao, Y. Oi and M. K. Sapino, "AlphaSum: Size-Constrained Table Summarization using Value Lattices", EDBT, pp. 96-107, 2009.

**6.**U. Feige, "A Threshold of In n for Approximating Set Cover", J. ACM, vol. 45, no. 4, pp. 634-652, 1998.

**7.**R. Fleischer, J. Li, S. Tian and H. Zhu, "Non-metric Multicommodity and Multilevel Facility Location", AAIM, pp. 138-148, 2006.

**8.**A. Gajewar and A. Das Sarma, "Multi-skill Collaborative Teams based on Densest Subgraphs", SDM, pp. 165-176, 2012.

**9.**L. Golab, H. Karloff, F. Korn, D. Srivastava and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies", PVLDB, vol. 1, no. 1, pp. 376-390, 2008.

**10.**D. Hochbaum, "Approximating covering and packing problems: Set cover vertex cover independent set and related problems", Approximation algorithms for NP-hard problems, pp. 94-143, 1997.

**11.**S. Khuller, A. Moss and J. Naor, "The Budgeted Maximum Coverage Problem", Inf. Process. Lett., vol. 70, no. 1, pp. 39-45, 1999.

**12.**A. J. Knobbe and E. K. Y. Ho, "Pattern Teams", PKDD, pp. 577-584, 2006.

**13.**K.-N. Kontonasios and T. De Bie, "An Information-Theoretic Approach to Finding Informative Noisy Tiles in Binary Databases", SDM, pp. 153-164, 2010.

**14.**L. V. S. Lakshmanan, R. Ng, C. Wang, X. Zhou and T. Johnson, "The Generalized MDL Approach for Summarization", VLDB, pp. 766-777, 2002.

**15.**L. Lovasz, "On the ratio of the optimal integral and fractional covers", Disc. Math, vol. 13, pp. 383-390, 1975.

**16.**M. Mampaey, N. Tatti and J. Vreeken, "Tell Me What I Need to Know: Succinctly Summarizing Data with Itemsets", KDD, pp. 573-581, 2011.

**17.**T. Mielikainen and H. Mannila, "The Pattern Ordering Problem", PKDD, pp. 327-338, 2003.

**18.**B. Saha and L. Getoor, "On Maximum Coverage in the Streaming Model & Application to Multi-topic Blog-Watch", SDM, pp. 697-708, 2009.

Show in Context[CrossRef](https://doi.org/10.1137/1.9781611972795.60" \t "https://ieeexplore.ieee.org/abstract/document/7113341/_blank)[Google Scholar](https://scholar.google.com/scholar?as_q=On+Maximum+Coverage+in+the+Streaming+Model+&+Application+to+Multi-topic+Blog-Watch&as_occt=title&hl=en&as_sdt=0,31" \t "https://ieeexplore.ieee.org/abstract/document/7113341/_blank)

**19.**G. Sathe and S. Sarawagi, "Intelligent Rollups in Multidimensional OLAP Data", VLDB, pp. 531-540, 2001.

**20.**M. van Leeuwen, "Maximal exceptions with minimal descriptions", Data Min. Knowl. Discov, vol. 21, no. 2, pp. 259-276, 2010.

**21.**M. van Leeuwen and A. J. Knobbe, "Non-redundant Subgroup Discovery in Large and Complex Data", ECML/PKDD, no. 3, pp. 459-474, 2011.

**22.**V. V. Vazirani, "Approximation algorithms", Springer, 2001.

**23.**C. Wang and S. Parthasarathy, "Summarizing itemset patterns using probabilistic models", KDD, pp. 730-735, 2006.

**24.**Y. Xiang, R. Jin, D. Fuhry and F. Dragan, "Succinct summarization of transactional databases: an overlapped hyperrectangle scheme", KDD, pp. 758-766, 2008.

**25.**X. Yan, H. Cheng, J. Han and D. Xin, "Summarizing itemset patterns: a profile-based approach", KDD, pp. 314-323, 2005.