基于解耦概要图的大规模图数据高效分布式挖掘算法

李 玲[1)] 印 莹[1)] 赵宇海[1)] 王国仁[2)] 董祥军[3)]

[1)](东北大学计算机科学与工程学院 沈阳 110169)

[2)](北京理工大学计算机学院 北京 100081)

[3)](齐鲁工业大学信息学院 济南 250353)

摘 要 频繁封闭子图挖掘被证明是 NP-难问题.多年来,虽然已有许多算法被提出用于解决该问题,但在挖掘大规模图数据时,却面临着共同的计算效率问题.特别是,当图中节点的平均度数增加时,挖掘效率更是急剧下降.现在已有的面向图数据库的分布式频繁子图挖掘算法大多采用基于水平划分的分布式计算框架,且都聚焦在挖掘所有频繁子图的问题上.基于水平划分的分布式计算框架是对原始数据进行水平分片,完成分布式挖掘过程.在计算效率方面,该框架存在一些不足.同时,由于封闭子图模式需要对频繁子图进行封闭性检测,如果直接将现有的分布式频繁子图挖掘算法用于闭图模式挖掘可能导致各节点间频繁的通讯,或大量的子图同构检测.针对以上问题,本文提出一种面向大规模图数据的高效分布式挖掘算法 Desu-FSM.与现有基于水平分解的分布式挖掘框架不同,该算法首次采用了基于垂直分解的分布式挖掘框架.其基本思想可概括为"快速抵近,双向搜索".首先,通过τ-邻域核图合并,获得概要图集,跨越式地快速抵近较大尺寸子图的聚集区域.在此基础上,通过对概要图的缩减和扩展发现所有被概要图包含和包含概要图的闭图模式.相较于原始图数据,概要图的尺寸和平均节点度数更小.而且,基于概要图的双向搜索可在分布式环境下同时独立完成,不存在耦合.与基于水平划分的框架采用的"数据物理分治"方式不同,该框架采用"任务逻辑分治".前者减少了各节点处理的图数据量,后者将原始图数据中的挖掘任务分解为一系列具有更小尺寸和平均节点度数的子图限定的子任务.因此,计算效率大幅提升.本文还提出一组高效的优化策略来减少概要图之间存在公共子图导致的大量重复计算.大量真实和人工数据集上的测试结果表明,在大规模图数据封闭子图挖掘中,基于垂直分解框架的挖掘效率相较于水平分解框架的效率可提升一个数量级.同时,具有更少的内存空间占用.

关键词 子图挖掘;解耦概要图;代表概要图;垂直分解;分布式计算

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2020.01183

An Efficient Distributed Algorithm for Large-Scale Graph Data Mining Based on Decoupled Summary Subgraph

LI Ling[1)] YIN Ying[1)] ZHAO Yu-Hai[1)] WANG Guo-Ren[2)] DONG Xiang-Jun[3)]

[1)](*School of Computer Science and Engineering, Northeastern University, Shenyang* 110169)

[2)](*School of Computer Science and Technology, Beijing Institute of Technology University, Beijing* 100081)

[3)](*School of Information, Qilu University of Technology, Jinan* 250353)

Abstract Frequent closed subgraphs mining is proved to be a NP-problem. Over the years, although many algorithms have been proposed to solve this problem, they are all faced with a common problem of computational efficiency when mining large scale graph data sets. In particular, when the average degree of vertexes increases in graph data sets, the efficiency of mining decreases sharply. At present, most of the existing distributed frequent subgraph mining algorithms for

收稿日期:2018-09-04;在线发布日期:2019-05-29.本课题得到国家重点研发计划项目(2018YFB1004402)、国家自然科学基金面上项目(61772124)资助.**李 玲**,博士研究生,主要研究方向为大数据挖掘、并行计算.E-mail:591330813@qq.com.**印 莹**(通信作者),博士,副教授,主要研究方向为数据挖掘.E-mail:yinying@cse.neu.edu.cn.**赵宇海**,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为数据库、数据挖掘、生物信息.**王国仁**,博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为数据库.**董祥军**,博士,教授,主要研究领域为数据挖掘、人工智能、数据库技术.

graph databases adopt a distributed computing framework based on horizontal partitioning, and all of them focus on the problem of mining all frequent subgraphs. The distributed computing framework based on horizontal partition is to partition original graph data sets horizontally and complete the distributed mining process. In terms of computational efficiency, the framework based on horizontal partition has some shortcomings. At the same time, because the frequent closed subgraph patterns mining need to detect the closeness of frequent subgraphs, if the existing distributed frequent subgraph mining algorithms are directly applied to the frequent closed subgraphs mining, it may lead to frequent communication between nodes or a large number of subgraph isomorphism detection. In view of the above problems, this paper proposes an efficient distributed mining algorithm Desu-FSM for large scale graph data sets. Unlike the existing distributed mining framework based on horizontal decomposition, this algorithm adopts the distributed mining framework based on vertical decomposition for the first time. Its basic idea can be summarized as "fast approaching, bidirectional search". First, we can get a summary graph set by merging the τ-domain kernel graphs, and quickly get close to the aggregation area of large subgraphs. On this basis, by reducing and extending the summary graphs, we can find all frequent closed graph patterns that summary graphs contain and the frequent closed graph patterns that containing summary graphs. Compared with the original graph data sets, the size and average degree of vertexes of the summary graphs are smaller. Moreover, the bidirectional search based on summary graphs can be completed independently in distributed environment without coupling. Different from the "data physical divide-and-conquer" approaches adopted in the framework based on horizontal partition, this framework adopts "task logical divide-and-conquer". The former reduces the amount of graph data sets which are processed by each node, while the latter decomposes the mining task from the original graph data sets into a series of subgraph bound tasks with smaller size and average vertexes degree. Therefore, the computational efficiency has been greatly improved. This paper also proposes a set of efficient optimization strategies to reduce the large number of repeated computations caused by common subgraphs between summary graphs. The mining efficiency of the framework based on vertical decomposition is tested on a large number of real and synthetic graph data sets. Compared with the horizontal decomposition framework, the results show that the vertical decomposition framework can be improved by one order of magnitude with less memory space occupancy when mining frequent closed subgraphs in large scale graph data sets.

Keywords subgraphs mining; decoupled summary graph; representative summary graph; vertical decomposition; distributed computing

1 引 言

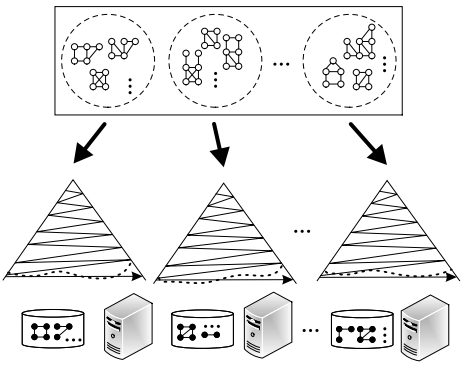
频繁子图在生物信息学[1]、RDF 数据管理[2]及社交网络分析[3]等许多领域有广泛应用,但其庞大的数量给后续分析带来了诸多不便.例如,当支持度设置为 5%时,AIDS 数据集中包含 422 个平均节点数和边数分别为 40 和 42 的图数据即可产生近百万的频繁子图[4].

为减少频繁子图数量,封闭子图和极大子图被分别提出.频繁子图 g 是封闭的,当且仅当其任一超图的支持度严格小于 g 的支持度;频繁子图 g 是极大的,当且仅当其任一超图是非频繁的.不难推知,根据所有封闭子图,可无损地恢复所有频繁子图及其支持度信息.因此,封闭子图挖掘在实际应用中受到了广泛关注和研究.

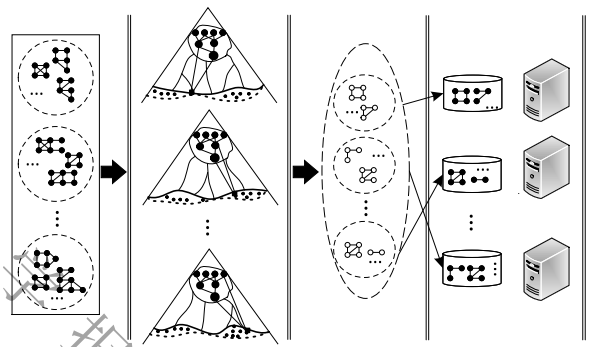
封闭子图挖掘被证明是 NP-难问题[5],精确获得所有的封闭子图模式被证明是 ♯P-完全问题[5].多年来,虽然已有许多算法[4,6-7]被提出用于解决该问题,但在挖掘大规模图数据时,却面临着共同的计

算效率挑战.特别是,当图中节点的平均度数增加时,挖掘效率更是急剧下降.现有的大规模频繁子图挖掘算法大多采用基于水平划分的分布式计算框架,即基于对原始数据的水平分片,完成分布式挖掘过程.在计算效率方面,该框架存在以下几点固有不足:(1)水平分片虽然缩减了各节点处理的图数据数量,但没有改变原始图数据的平均节点度数.平均节点度数是制约图挖掘算法效率的主要计算瓶颈;(2)子图全局频繁封闭性需要根据其在各水平分片内出现的频率汇总得到.因此,即使子图在分片内出现频率较低,也不能随意丢弃.低支持度阈值影响计算效率提升;(3)水平分片内产生的子图不一定是全局频繁的,需要借助子图同构来过滤非频繁子图.子图同构是 NP-难问题[8],大量非频繁子图引起的子图同构检测会增加计算成本.

针对以上问题,本文提出一种面向大规模图数据的高效分布式挖掘算法 Desu-FSM.与现有基于水平划分的分布式挖掘框架不同,该算法首次采用了基于垂直分解的分布式挖掘框架.其基本思想可概括为"快速抵近,双向搜索".图 1 以图示的方式,对两种框架的基本思想及区别给出了直观解释.顾名思义,如图 1(a)所示,基于水平划分的分布式挖掘框架将原始图数据划分为一组互斥的水平分片,并将各分片映射到不同的物理节点.之后,在这些物理节点构成的计算集群上,完成分布式挖掘过程.该框架中,各物理节点通常采用的是自底向上,逐边增长的子图搜索方式.k 边子图必须通过$(k-1)$边子图宽度或深度优先扩展得到.同时,由于子图模式的全局信息通常需要根据其在各水平分片内的局部信息汇总得到,各物理节点间可能存在频繁的通信.与此不同,图 1(b)所示为基于垂直分解的分布式挖掘框架.该框架中,各物理节点以高于全局出现频率的支持度阈值,通过 τ-邻域核图(一些满足特定要求的小规模子图)合并,跨越式地快速抵近较大尺寸子图(概要图)的聚集区域(图 1(b)中的实线).精炼获得概要图集后,将概要图及对应的概要图数据库分配至不同的物理节点,执行分布式挖掘过程.各物理节点的计算任务相互独立,不存在耦合,无需相互通信.所有被概要图包含的封闭子图可通过缩减概要图获得.概要图是频繁的,该过程避免了非频繁子图引起的大量子图同构检测.所有包含概要图的封闭子图,可以概要图为"核心",向外扩展获得.由于概要图集构成的边界接近极大子图集构成的边界(图 1(b)中的虚线),该扩展过程大大缩减了大尺寸封闭子图的搜索代价.概要图是原始图数据的子图,从这个意义上,基于概要图的一系列挖掘任务可看作是对原始图数据挖掘任务的一组垂直分解.概要图具有比原始图数据更小的平均节点度数,因此能大幅提升计算效率.



(a) 水平划分框架



(b) 垂直分解框架

图 1　不同框架的挖掘方式比较

本文工作的主要贡献总结如下:

(1)提出一种基于垂直分解的图数据挖掘框架.与基于水平划分的框架采用的"数据物理分治"方式不同,该框架采用"任务逻辑分治".前者减少了各节点处理的图数据量,后者将原始图数据中的挖掘任务分解为一系列具有更小尺寸和平均节点度数的子图限定的子任务.就目前所知,该框架是首次在图数据挖掘中提出.

(2)提出一种面向大规模图数据的高效封闭子图分布式挖掘算法 Desu-FSM.针对局部不封闭性不具备全局可加性引起的计算挑战,通过 τ-邻域核图合并,快速生成一组具有局部极大性的概要图集.借助提出的垂直分解图数据挖掘框架,执行基于概要图的缩减和扩展操作,发现所有被概要图包含和包含概要图的封闭子图.

(3)提出一组高效的优化策略.概要图之间可能存在公共子图,基于不同概要图的子图搜索空间

可能会存在交叠.对每个概要图进行独立分离式的处理,会导致大量重复计算.提出 all-in-one 优化策略,共享公共子图检测,确保每个节点内的子图模式只被挖掘一次.此外,子图相似度上界被用于减少极大公共子图计算代价.

(4)在大量百万规模的真实和人工图数据集上对算法性能进行了实验分析和验证.实验结果表明,在大规模图数据封闭子图挖掘中,基于垂直分解框架的挖掘效率相较于水平分解框架的效率可提升一个数量级.同时,具有更少的内存空间占用.

本文第 2 节阐述和分析频繁子图模式挖掘的相关工作;第 3 节介绍频繁子图挖掘及本文工作的相关概念;第 4 节详细阐述提出的基于垂直分解的图挖掘框架及高效封闭子图分布式挖掘算法 Desu-FSM;第 5 节通过大量对比实验,分析算法性能;第 6 节对全文进行总结.

2 相关工作

由于挖掘频繁子图数量过于庞大,为减少模式数量,人们提出封闭子图挖掘的概念,包括 CloseGraph[4]、MoSS[6]、PSI-CFSM[7]等,CloseGraph 算法改进深度优先搜索的 gSpan 算法得到封闭子图.MoSS 算法改进广度优先搜索的 MoFa 算法来挖掘封闭子图.PSI-CFSM 通过优化子图同构测试,加快封闭子图的获得.但是,这些算法都假定数据能完全装入内存,且任务能在一个合理时间范围内完成,而随着数据集的不断增大,这种假设已经不再适用.

为了处理大规模图数据集,得到频繁模式,Wang 等人[9]首先提出了基于磁盘的方法解决内存限制,通过改进图集的存储结构,使改进后的存储结构具有较高灵活性,数据可以不必全部装入内存.改进后的算法能处理相对较大规模的图集,在一定程度上解决了内存限制,但由于磁盘的读入、写出操作非常频繁,为访问数据带来了额外开销,因此算法的效率仍然有很大的提升空间.之后,Nguyen 等人[10]提出了使用数据划分的方法,这种方法早期被用在频繁项集挖掘过程中.在该方法中,首先将大规模的图数据集转换成可以全部载入内存的子图集,然后对每一个子图集均采用 gSpan 算法挖掘频繁图模式,在第二遍扫描数据库时计算子图频度.

在分布式挖掘的背景下,Subdue[11]是一个较早提出的分布式图模式挖掘方法,它是基于共享内存的分布式算法.通过划分任务,在本地计算子图频度,然后传播给所有节点.再在每一个节点上计算频度,最后由一个主节点来汇总每个节点上的结果.Buehrer 等人[12]提出了在多核系统下的并行频繁图模式挖掘方法,通过把任务分给多个共享内存的处理器,相比于单机算法,明显提高了性能.Meinl 等人[13]设计了 Parmol,实现了 MoFa[14]、gSpan[15]、FFSG[16]以及 Gaston[17]等分布式挖掘算法.

之后,随着 MapReduce[18]、spark[19]框架的不断发展,有人提出基于 MapReduce 框架挖掘频繁模式,但是数据集基本为简单的项集[20-22]或序列[23].对于图数据而言,文献[24]是第一个在大规模图数据集中基于 MapReduce 框架的频繁图模式挖掘算法.该算法采用了迭代的方式,类似于 Apriori 算法的思想,当获得所有规模为 K 的频繁图模式,才会进行下一轮迭代,来获得规模为 $K+1$ 的频繁图模式.该算法没有采用一些方法来避免重复模式的产生,导致候选子图数量呈指数级增长.为削除结果集中的大量重复模式,需要进行额外的子图同构测试.此外,该算法需要用户指定迭代次数.Bhuiyan 等人[25]提出的 FSM-H 算法对文献[24]中的算法进行了改进,将任务分为三个阶段:数据划分阶段、准备阶段、挖掘阶段.通过挖掘阶段的不断迭代可以挖掘大规模的图集.Lin 等人[26]也提出了一个基于 MapReduce 的分布式图集挖掘算法,该算法没有使用迭代,它是将任务分为两个阶段:filter 阶段和 refinement 阶段.filter 阶段依据概率选出最有可能是全局频繁的局部频繁子图,然后在 refinement 阶段计算全局支持度,过滤掉非全局频繁子图.由于 MapReduce 框架通常需要将中间结果输出到磁盘上,过于频繁的 I/O 操作会降低算法效率.为进一步提升频繁模式挖掘效率,有人[27-28]提出采用 spark 框架挖掘,但处理的数据集都仅限于项集数据.对于图数据,借助 spark 框架挖掘频繁图模式的算法还很少.

目前的分布式频繁子图挖掘算法大多采用基于水平分解的框架,即从水平方向上对数据进行分解,将数据集分片为一系列更小规模的数据集.数据集中图的平均节点度数是影响基于模式增长的子图挖掘算法效率的根本因素,基于水平分解框架的算法没有突破大规模频繁子图挖掘的计算瓶颈.此外,由于封闭子图挖掘需要借助大量子图同构检测,处理大规模图数据时的计算效率令人难以忍受,目前尚

未发现基于分布式计算框架的高效封闭子图挖掘算法.针对这些问题,本文将提出一种基于垂直分解框架的高效封闭子图分布式挖掘算法.

3　基本概念

本节主要介绍图挖掘的相关定义及本文提出的主要概念.

定义 1.　图同构.给定图 $G=(V,E)$ 和 $G'=(V',E')$,$V(V')$ 和 $E(E')$ 分别代表图 G 和 G' 的顶点集合和边集合.若存在双射函数 $f:V\leftrightarrow V'$,满足 $\forall u,v\in V,\forall(u,v)\in E\Leftrightarrow(f(u),f(v))\in E'$,则称图 G 与 G' 同构,记为 $G\cong G'$.若 G' 中存在子图 G'' 与图 G 同构,称 G 子图同构于 G'.

当 G 子图同构于 G' 时,称 G 是 G' 的子图,G' 是 G 的超图,记为 $G\subseteq G'$.如果 $G\subseteq G'$ 且 $G\neq G'$,则称 G 是 G' 的真子图,G' 是 G 的真超图.若 $\mathcal{G}=\{G_1,G_2,\cdots,G_n\}$ 为给定的图数据库,$Set(g,\mathcal{G})=\{\mathcal{G}_i\mid g\subseteq\mathcal{G}_i,1\leq i\leq n\}$ 被称为图 g 在 G 中的支持集,$Sup(g,\mathcal{G})=|Set(g,\mathcal{G})|$ 被称为图 g 在 G 中的支持度.对给定的最小支持度阈值 min_sup,若 $Sup(g,\mathcal{G})\geq min\_sup$,称 g 在 $\mathcal{G}$ 中是频繁的.若 g 的任何真超图 g' 的支持度满足 $Sup(g',\mathcal{G})<Sup(g,\mathcal{G})$,称 g 是 $\mathcal{G}$ 中的封闭子图.若 g 的任何真超图 g' 的支持度满足 $Sup(g',\mathcal{G})<min\_sup$,称 g 为 $\mathcal{G}$ 中的极大子图.

定义 2.　核图.给定图数据库 $\mathcal{G}$,g 和 g' 是两个图模式满足 $g\subseteq g'$,如果 $Sup(g',\mathcal{G})/Sup(g,\mathcal{G})\geq\tau$,$0<\tau<1$,那么称 g 是 g' 的一个 τ-核图.

注意:g' 的 τ-核图可以有多个不同的 τ-核图.

定义 3.　图的 Jaccard 距离.给定图数据集 $\mathcal{G}$,g_1 和 g_2 是两个图模式,g_1 和 g_2 的 Jaccard 距离定义如下:

$$dist(g_1,g_2)=1-\frac{|Set(g_1,g_2)\bigcap Set(g_1,g_2)|}{|Set(g_1,g_2)\bigcup Set(g_1,g_2)|}\quad(1)$$

定义 4.　概要图集.令 $\mathcal{G}=\{G_1,G_2,\cdots,G_n\}$ 为给定的图数据集,如果存在一组图模式集合 $g=\{g_1,g_2,\cdots,g_m\}$,同时满足:(1) $\forall i,i\in[1,m],\exists j,j\in[1,n],g_i\subseteq G_j$;(2) 若 c_g 为 G 中的封闭子图,则 $\exists g_k\in g,c_g\subseteq g_k$,称 $g=\{g_1,g_2,\cdots,g_m\}$ 为图数据集 $\mathcal{G}$ 的一组概要图集,g 中的每个图 $g_i(1\leq i\leq m)$ 为一个概要图.

根据定义 4 可推知,图数据集 $\mathcal{G}$ 中的封闭子图挖掘可通过对 g 中所有图模式 g_i 的子图进行封闭性检测完成.如果对不同 g_i 的封闭性检测可相互独立完成,不存在耦合,则称 g 为图数据集 $\mathcal{G}$ 的解耦概要图集.由于 g_i 是 $\mathcal{G}$ 中图数据的子图,此时又称 g 为图数据集 $\mathcal{G}$ 的一组垂直分解.

概要图集中的图模式可能会存在很多结构相似的子图,为了提升算法效率,减少概要图集中的模式数量,进一步提出利用子图结构相似度得到代表概要图集.

定义 5.　结构相似度.设 g_1 和 g_2 是两个图模式,g_1 和 g_2 之间的结构相似度定义如下:

$$sim(g_1,g_2)=\frac{|E(G_{mc}(g_1,g_2))|}{\max(|E(g_1)|,|E(g_2)|)}\quad(2)$$

其中,$G_{mc}(g_1,g_2)$ 是 g_1 和 g_2 的极大公共子图,$E(g)$ 代表图 g 的边集.

定义 6.　图的结构距离.设 g_1 和 g_2 是两个图模式,g_1 和 g_2 之间的结构距离定义为 $D(g_1,g_2)=1-sim(g_1,g_2)$.

定义 7.　代表概要图集.假设 g 为图数据集 $\mathcal{G}$ 的一个概要图集,称 S 是 $\mathcal{G}$ 的一个代表概要图集,若其满足以下条件:(1) $S\subseteq g$,即 S 是 g 的子集;(2) $\forall g_i\in g,\exists g_j\in S,g_i\subseteq g_j$;否则,$\exists g_j\in S,g_i\subseteq g_j,sim(g_i,g_j)\geq\delta$.其中,$\delta$ 为最小相似度阈值.

以下,通过图 2 对概要图集等主要概念进行进一步解释,并加深对垂直分解框架的理解.

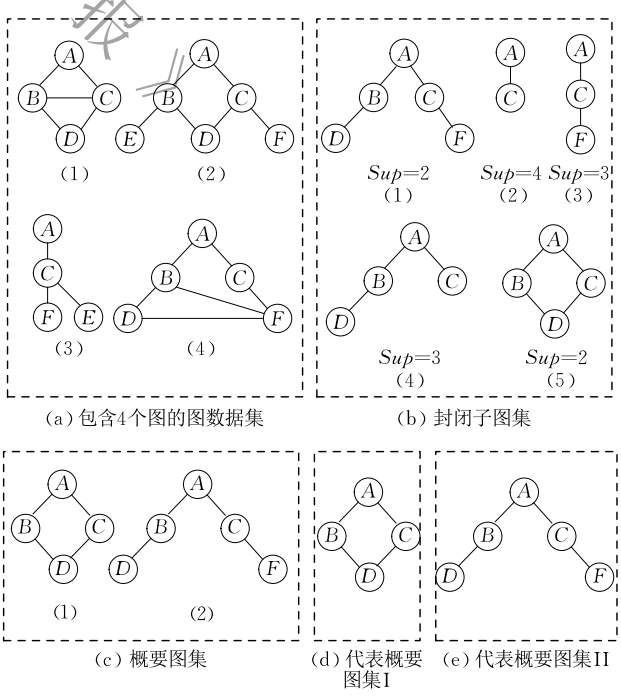


图 2　概念解释

例 1.　如图 2(a)所示,给定包含 4 个图数据的图数据集 $\mathcal{G}=\{G_1,G_2,\cdots,G_3\}$.若设支持度阈值

$min\_sup=2$,可得如图 2(b)所示的封闭子图集合.可以看出,图 2(b)中每个封闭子图均为图 2(c)中图(1)或图(2)的子图.同时,图 2(c)中的图(1)或图(2)均为图 2(a)中某个原始图数据的子图.因此,图 2(c)中的两个图模式构成了图 2(a)中原始图数据的一个概要图集.进一步,设相似度阈值为 0.5,那么该图集的代表概要图集可以为图 2(d)或(e)所示.

本文主要研究闭图模式挖掘,旨在通过对图数据集的垂直分解缩减待处理的图数据边数,从根本上突破计算瓶颈.目前已有的图数据分布式挖掘算法大多采用传统的数据水平分片,从水平方向上对数据库进行划分,将数据集按照数量分解为一系列小规模数据集,分配到不同的物理节点进行分别处理.但是,图挖掘算法大多采用逐边增长的子图扩展搜索方式,图数据库中每个图的平均节点度数才是影响算法效率的根本因素(如参考文献[4]中的实验图 12(b)和本文实验图 11 所示).通过图 2 可进一步说明边数对两种图挖掘框架效率的影响.

假设水平数据划分将图 2(a)中的四个图数据分为两组,(1)和(2)为一组,(3)和(4)为一组.若以 AB 边为初始频繁 1-边进行子图扩展,(1)和(2)分组中,需要枚举 AC、BC、BD、BE 四条边.(3)和(4)分组中,需要枚举 AC、BD、BF 三条边.在分布式环境下,此时基于 AB 边的 1-边子图扩展所需时间为枚举 4 条边的时间.若采用基于概要图进行垂直分解的方法,将图 2(c)中的两个概要图分别分配到不同的节点,两个节点中基于 AB 边的频繁 1-边扩展分别只需枚举 AC、BD 两条边.可以看出,虽然此时的枚举时间减少了一半,并节省了相应的子图同构检测时间,但图 2(b)中的所有频繁封闭子图在枚举过程中并没有被遗漏.

4 算法描述

本节详细介绍提出的基于解耦概要图的封闭子图分布式挖掘算法 Desu-FSM.与水平划分框架基于“数据物理分治”提高计算效率的思想不同,该算法采用“任务逻辑分治”的方式,将原始图数据中的挖掘任务分解为一系列概要图限定的子任务.由于概要图具有更小的尺寸和平均节点度数,避免了非频繁子图导致的子图同构检测,算法效率大大提升.图 3 为 Desu-FSM 算法的整体框架.

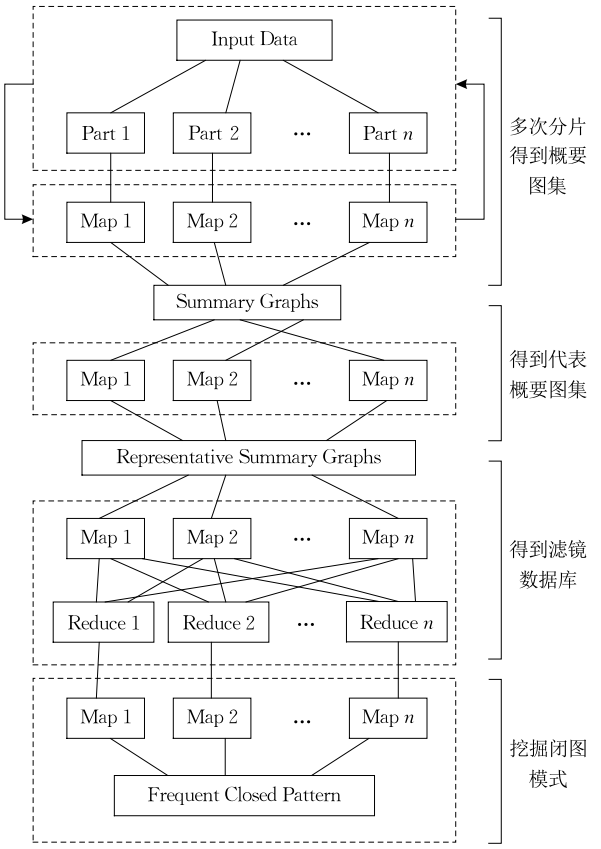


图 3 Desu-FSM 框架图

4.1 算法总览

Desu-FSM 算法的伪码如算法 1 所示,由三个主要部分构成:

(1)任务垂直分解(行 1~7).对数据进行多轮随机划分,通过 τ-核图邻域合并,快速收集具有局部极大性的概要图集.注意,此阶段目的是快速抵近较大尺寸子图(概要图)的聚集区域,无需进行子图封闭性检测;

(2)代表概要图集产生(行 8~12).直接对获得的所有概要图进行处理,会产生大量相似结果.证明图的结构距离满足三角不等式,利用两阶段的分布式贪婪算法得到代表概要图集,并保证其代表误差在可控范围内;

(3)分布式挖掘(行 13~19).将代表概要图及对应的概要图数据库分配至不同物理节点,分布式执行基于概要图的缩减和扩展操作.各物理节点计算任务相互独立完成,并通过 all-in-one 优化策略,减少重复计算.

算法 1. Desu-FSM.

输入:图数据集 $\mathcal{G}$,最小支持度阈值 min_sup,相似度阈值 δ,模式距离 σ

输出:闭图模式集 CFS

1. $\mathcal{G}=\{G_1,G_2,\cdots,G_n\}$

```
2.  WHILE (i≤M)                    //M 为轮数
3.    S′=G.mapPartitions(minsup):
4.      调用 M-FSM(D_i,min_sup,σ,K);//挖掘概要图
5.    G=G.mapPartitions(); //对数据重新分片
6.    S=S∪S′;
7.    S=select(S);                 //去掉同构模式
8.  S=S.repartition(split); //split 为分片数
9.  RS=S.mapPartitions(δ):
10.   调用 S-FSM(S,δ);    //减少模式数量
11. RS=RS.repartition(1);
12. 调用 S-FSM(RS,δ);
13. G′=G.mapPartitions(RS):
14.   FOR RS 中每个概要图
15.     FOR 分片数据库 G_i 中每个图
16.       若 RS(l)与 G_i(j)存在规模为 K 的公共子
          图,将这两个图写成 key-value 的形式输出.
17. G′.reduceByKey();
18. CFS=G′.mapPartitions():
19.     调用 BECM 算法;
```

4.2　任务垂直分解

任务垂直分解的目的是为了获得概要图集. 如前文图 1(b)所示，概要图集构成的边界应接近极大子图集构成的边界. 因此，一种最直接的想法是，首先将给定的图数据集 $\mathcal{G}$ 随机分片至 n 个不同的物理节点，$\mathcal{G}=\{G_1,G_2,\cdots,G_n\}$. 然后，在每个节点内，使用全局绝对支持度 min_sup 独立挖掘分片数据内的局部极大模式构成局部概要图集. 这种做法的好处是能够保证获得的局部概要图可能是全局上尺寸较大的封闭子图，但存在着两个问题：(1) 仅针对当前数据分片结果获得概要图集，可能会丢失大量结果；(2) 概要图具有局部极大性，采用自底向上、逐边增长的传统子图搜索方式，效率过低.

针对问题(1)，本文的解决方法是，对原始图数据集进行多轮随机分片，每一次的分片结果不同. 在每一轮划分中，对不同分片数据进行垂直分解，迭代进行该过程直到获得一个近似于单机结果的集合. 每循环一次该过程，需要对垂直分解后的概要图集进行一次选择，以消除冗余模式. 该过程虽然增加了分片次数，但由于局部数据中使用的是全局支持度，增大了对不频繁子图的削减力度(因为 $min\_sup\times|\mathcal{G}_i|<min\_sup\times|\mathcal{G}|$)，并借助了并行框架，故挖掘速度会很快.

需要强调的是，如果基于水平划分框架，即使采用多轮数据分片，封闭子图挖掘也是代价极高的任务.

定理 1.　局部封闭子图一定是全局封闭的，局部非闭子图也可能是全局封闭的.

证明.　给定一个图数据库 $\mathcal{G}$，将图数据库进行水平划分，分成 $\mathcal{G}=\{\mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_n\}$，在子图集 $\mathcal{G}_i$ 使用全局支持度挖掘得到非闭图模式 g，g 在 $\mathcal{G}$ 中的支持度为 $Sup(g,\mathcal{G})=\sum_{j\neq i}^{n}Sup(g,\mathcal{G}_j)+Sup(g,\mathcal{G}_i)$，$g$ 的真超图 g' 在 $\mathcal{G}$ 中的支持度为 $Sup(g',\mathcal{G})=\sum_{j\neq i}^{n}Sup(g',\mathcal{G}_j)+Sup(g',\mathcal{G}_i)$. 由于 g 是非闭模式，在 $\mathcal{G}_i$ 中，设 g 的真超图 g' 的支持度 $Sup(g,\mathcal{G}_i)=Sup(g',\mathcal{G}_i)$，$\sum_{j\neq i}^{n}Sup(g',\mathcal{G}_j)<\sum_{j\neq i}^{n}Sup(g,\mathcal{G}_j)$，那么 g 是闭图模式，故局部非闭模式可能是全局闭模式. 若 g 是局部闭图模式，在 $\mathcal{G}_i$ 中，$Sup(g,\mathcal{G}_i)<Sup(g',\mathcal{G}_i)$，能推出 $Sup(g,\mathcal{G})<Sup(g',\mathcal{G})$，因此 g 是全局闭图模式.　　证毕.

根据定理 1 可知，局部不封闭性不具备全局可加性. 虽然基于全局支持度 min_sup 获得的局部封闭子图在全局数据上也是封闭的，但该过程中丢弃的局部非闭子图也可能是全局封闭的. 为解决该问题，水平划分框架需保留挖掘过程中各分片数据包含的所有频繁子图，通过大量的子图同构检测发现其中的封闭子图. 与此不同，本节的任务垂直分解过程中，并未在局部数据中执行封闭子图挖掘过程，而是将其留待获得局部概要图集后，通过对概要图的缩减和扩展来完成. 因此，概要图集的快速获取是任务垂直分解的关键.

针对问题(2)，本文提出一种通过 τ-核图邻域合并，快速获得一组具有局部极大性的概要图集算法 M-FSM. 文献[29]中指出，一个较大规模的项集模式通常包含很多核模式，并且能够以很大的概率通过合并它的核模式获得. 受该结论启发，本文证明图数据也具有同样的性质.

定理 2.　给定图数据 $g,g_1\subseteq g$ 和 $g_2\subseteq g$ 是 g 的两个 τ-核图，则有 $dist(g_1,g_2)\leq 1-\frac{1}{2/\tau-1}$.

证明.　由于 g_1 和 g_2 是 g 两个 τ-核模式，

$$\frac{|Set(g_1,G)\cap Set(g_2,G)|}{|Set(g_1,G)\cup Set(g_2,G)|}\geq\frac{|Set(g,G)|}{|Set(g_1,G)\cup Set(g_2,G)|}$$

$$=\frac{|Set(g,G)|}{|Set(g_1,G)|+|Set(g_2,G)|-|Set(g_1,G)\cap Set(g_2,G)|}$$

$$\geq\frac{|Set(g,G)|}{|Set(g,G)|/\tau+|Set(g,G)|/\tau-|Set(g,G)|}$$

$$=\frac{1}{2/\tau-1}.$$

因此，$dist(g_1,g_2)=1-\frac{|Set(g_1,G)\cap Set(g_2,G)|}{|Set(g_1,G)\cup Set(g_2,G)|}\leq 1-\frac{1}{2/\tau-1}$. 证毕.

根据定理 2 可知，合成图 g 的 τ-核图之间的 Jaccard 距离通常在一个有限的范围内，因此可以通过不断合并小的 τ-核图快速得到较大规模的图模式. 通过这种方式，可以有效地避免概要图生成过程中诸多小尺寸图模式的纠缠. 与文献[29]不同，本文执行的是 τ-核图，而非项集合并. 项集元素的本质是集合，可以直接合并，但图数据结构复杂，无法沿用相同的合并方式，因此需要设计新的合并方法. 本文提出的合并方法可以简单概括为：首先，得到待合并核图间的极大公共子图；然后，对极大公共子图进行扩展，扩展每一个核图所包含的边；最后，根据全局支持度阈值 min_sup，检测合并后的子图是否频繁. 具体过程见例 2.

例 2. 若图 4 中的子图(a)与(b)是两个待合并的子图，(c)为(a)和(b)的极大公共子图. 首先，分别在(a)和(b)中找到极大公共子图对应的匹配(如虚线所示)，使用图(a)对图(b)所示匹配进行扩展，直到极大公共子图扩展完图(a)包含的边，图(d)为(a)和(b)合并之后的图. 对于(a)与(b)中的其它匹配也执行相同的操作，最后得到(d)与(e)两个图. 对核图间的每个极大公共子图都进行相同的处理.

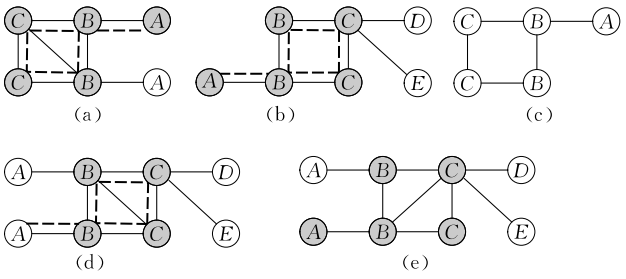


图 4 合并过程

注意，若数据集为无标签图，无标签图间规模最小的极大公共子图至少由两条连通的边组成. 若图数据为标签图且核模式间没有公共子图，但存在公共点，可以直接通过公共点合并核图. 若没有公共点，可以在核图支持集交集的图中，找到连通核图的最短路径，构成合并后的新模式.

完整的概要图挖掘算法 M-FSM 的伪码如算法 2 所示. 在算法 2 中，首先调用频繁图模式挖掘算法得到规模为 K 的所有频繁子图集合 FS_K(行 1)，接着循环调用 *pattern_fusion* 方法直到结果集中的模式不能合并生成频繁模式(行 2～4). 在 *pattern_fusion* 方法中把 FS_K 中每个模式 g 作为种子(行 1)，将 FS_K 集合中与 g 模式距离在 τ 范围内的模式加入 $g.corelist$ 中，同时将 g 加入集合 T 中，重复该过程直到次数达到 $|FS_K|$(行 1～5). 然后使用本文提出的合并规则合并集合 T 中的每一个 *corelist*(行 6,7).

算法 2. 概要图集挖掘 M-FSM($\mathcal{G}_i$, min_sup, τ, K).

输入：图集 $\mathcal{G}_i$，最小支持度阈值 min_sup，模式距离 τ，初始合并值 K

输出：极大模式集 S

1. 得到规模为 K 的频繁子图集 FS_K
2. DO
3. $S=pattern\_fusion(S,FS_K,min\_sup,\tau)$
4. WHILE 子图不能再合并
5. 输出 S

过程 1. $pattern\_fusion(S,FS_K,min\_sup,\tau)$.

1. FOR 模式 $g\in FS_K$
2. $T=T\cup g$
3. FOR 模式 $g'\in FS_K$
4. IF $dist(g,g')\leq\tau$, TNEH
5. 将 g' 加入 $g.corelist$
6. FOR $g\in T$
7. $S=S\cup fusion(g.corelist)$ //合并之后是频繁模式才会加入 S

4.3 代表概要图集产生

由于各节点相互独立地获得局部概要图集，不同节点的结果之间可能会存在结构高度相似甚至完全相同的概要图. 精确获得所有封闭子图被证明是 #P-完全问题，在大规模图数据处理中的计算效率令人难以容忍. 缩减结构相似的子图数量既能减轻后续分析的负担，又能大幅提高计算效率. 垂直分解框架中，概要图集是封闭子图挖掘的基础. 因此，通过质量可控的代表概要图集来减少概要图的数量是合理的. 根据以下定理，本文采用两阶段的分布式贪婪算法得到代表概要图集，并保证其代表误差在可控范围内.

定理 3. 图的结构距离满足三角不等式，即 $D(g_1,g_2)+D(g_2,g_3)\geq D(g_1,g_3)$.

证明. $D(g_1,g_2)+D(g_2,g_3)\geq D(g_1,g_3)$

$$\Leftrightarrow 1+\frac{|E(G_{mc}(g_1,g_3))|}{\max(|E(g_1)|,|E(g_3)|)}\geq \frac{|E(G_{mc}(g_1,g_2))|}{\max(|E(g_1)|,|E(g_2)|)}+\frac{|E(G_{mc}(g_2,g_3))|}{\max(|E(g_2)|,|E(g_3)|)}.$$

设$|E(g_1)|\geq|E(g_2)|\geq|E(g_3)|$，那么上述公式等价于：

$$\Leftrightarrow\frac{|E(g_2)|-|E(G_{mc}(g_2,g_3))|}{|E(g_2)|}\geq\frac{|E(G_{mc}(g_1,g_2))|-|E(G_{mc}(g_1,g_3))|}{|E(g_1)|}$$

$$\Leftrightarrow|E(g_2)|-|E(G_{mc}(g_2,g_3))|\geq|E(G_{mc}(g_1,g_2))|-|E(G_{mc}(g_1,g_3))|$$

现证：$|E(g_2)|+|E(G_{mc}(g_1,g_3))|\geq|E(G_{mc}(g_1,g_2))|+|E(G_{mc}(g_2,g_3))|$.

令 $g_1\cap g_2=G_{mc}(g_1,g_2)$，$g_1\cap g_3=G_{mc}(g_1,g_3)$，$g_2\cap g_3=G_{mc}(g_2,g_3)$，

那么$g_1\cap g_2\cap g_3=(g_1\cap g_2)\cap(g_2\cap g_3)=(G_{mc}(g_1,g_2))\cap(G_{mc}(g_2,g_3))=G_{mc}(g_1,g_2,g_3)$

由于 $g_1\cap g_2+g_2\cap g_3-g_1\cap g_2\cap g_3\subseteq g_2$，

故$|E(G_{mc}(g_1,g_2))|+|E(G_{mc}(g_2.g_3))|-|E(G_{mc}(g_1,g_2,g_3))|\leq|E(g_2)|$.

又由于 $E(G_{mc}(g_1,g_2,g_3))\subseteq E(G_{mc}(g_1,g_3))$，

故$|E(G_{mc}(g_1,g_2,g_3))|\leq|E(G_{mc}(g_1,g_3))|$

$\Rightarrow|E(G_{mc}(g_1,g_2,g_3))|+|E(G_{mc}(g_1,g_3))|+|E(g_2)|\geq|E(G_{mc}(g_1,g_2))|+|E(G_{mc}(g_2,g_3))|+|E(G_{mc}(g_1,g_2,g_3))|$

$\Rightarrow|E(g_2)|+|E(G_{mc}(g_1,g_3))|\geq|E(G_{mc}(g_1,g_2))|+|E(G_{mc}(g_2,g_3))|$.

同理可证其它情况下定理成立.　　证毕.

设 g_1 与 g_1'，g_2 与 g_2' 分别是在两个不同物理节点上，根据 4.2 节中的 τ-核图邻域合并获得的概要图，满足 $D(g_1,g_1')\leq\varepsilon_1$，$D(g_2,g_2')\leq\varepsilon_2$，并且 g_1' 与 g_2' 在各自的节点中分别被 g_1 和 g_2 代表. 设 $D(g_1,g_2)\leq\varepsilon_2$，根据图 5 可以推知，$D(g_1,g_2')\leq\varepsilon_1+\varepsilon_2$，$D(g_2,g_1')\leq\varepsilon_1+\varepsilon_2$，若 $\varepsilon_1+\varepsilon_2<1-\delta$，$\delta$ 为相似度阈值，则无论用 g_1 代表 g_2 或者 g_2 代表 g_1，都可使被代表的概要图与选定的代表图之间的结构距离不超过 $1-\delta$. 因此，为了减少概要图集中的模式数量，可以先对各分片中的概要图集进行代表性选择，再对合并后的结果再次进行选择. 根据定理 3，这种两阶段的分布式代表概要图集生成方法可保证最大代表误差在可控范围内.

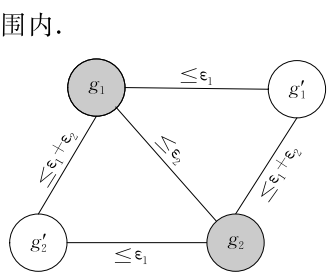


图 5　代表概要图图示

代表概要图集挖掘方法 S-FSM 具体过程见算法 3. 首先，为图集 S 中每一个图得到它所能代表的模式集合，然后使用贪心算法计算近似图集 S 的最小集合 RS.

算法 3.　代表概要图集挖掘 S-FSM(S,δ).

输入：图集 S，相似度阈值 δ

输出：概要图集 RS

1. FOR S 中任意一个图 g
2.　FOR S 中任意一个图 g'，只要 $sim(g,g')\geq\delta$
3.　将 g 集合加入到集合 $set(g')$
4. WHILE $S\neq\varnothing$
5.　找到一个 RS，最大化 $|set(RS)|$
6.　FOR $set(RS)$ 中每一个图 g
7.　从 S 和其它的 $set()$ 中删除 g
8.　输出 RS

4.4　分布式挖掘

获得代表概要图集后，将代表概要图分配到不同的物理节点，执行分布式封闭子图挖掘过程. 本工作基于 Spark 实现了依据概要图执行分布式封闭子图挖掘的算法，具体过程见算法 1(行 13～19). 算法 1 中，挖掘封闭子图之前，每个节点先读入数据分片及概要图集. 然后，调用 Spark 中的 mapPartitionsToPair 算子，输出 *key-value* 对，*key* 为概要图，*value* 为数据库中的图，该图与 *key* 对应的概要图至少存在一个规模(节点数或边数均可)为 K 的公共子图. 再调用 reduceByKey 算子，与某个概要图存在至少一个规模为 K 的公共子图的相关图构成了该概要图的概要数据库，并被发送到同一个 reducer 中(行 16). 此时，被该概要图包含和包含该概要图的规模大于 K 的全局封闭子图检测信息均存在于该概要数据库中. 之后，在每个物理节点上调用单机封闭子图挖掘算法 BECM(如算法 4)，执行基于概要图的缩减和扩展操作，并进行封闭性检测，输出〈子图，支持度〉对(算法 1 行 18～19)，得到最后结果.

算法 4.　闭模式挖掘 BECM.

输入：概要图集 S，图数据集 $\mathcal{G}_i$，支持度阈值 *min_sup*

输出：封闭子图集 FS

1. FOR 对于 S 中任意一个图 g
2.　为 g 中每一条边创建 DFSCode，形成集合 S^1
3.　FOR 集合 S^1 中每一条边 s
4.　得到 s 的支持集
5.　调用子图挖掘方法($\mathcal{G}_i$,null,s,g,*min_sup*)
6.　IF 图 g 中所有子图被枚举完毕，TNEH
7.　调用扩展概要图方法(g,$Set(g,G_i)$)

过程 2. 子图挖掘方法($\mathcal{G}_i$,s,$s\diamondsuit_x e$,g,min_sup).

```
IF 子图 s 不是最小的, THEN
    RETURN;
依据图 g 扩展 s 得到 s◇ₓe
IF 不存在 s◇ₓe 使得 Sup(s)=Sup(s◇ₓe), THEN
    将 s 加入结果集 FS
FOR 每个 s◇ₓe do
    调用子图挖掘方法(Dᵢ,s,s◇ₓe,g,min_sup)
```

BECM 算法与传统封闭子图挖掘算法的不同之处在于,传统算法采用自底向上的子图挖掘方式,而 BECM 算法通过基于概要图的缩减和扩展挖掘封闭子图. 概要图的缩减是自顶向下的,扩展是自底向上的.

各物理节点内,首先以每个概要图为基础,逐边缩减概要图的规模,对概要图的每个子图进行封闭性检测. 由于生成概要图集时的 τ-核图邻域合并是基于全局支持度阈值 min_sup 的,得到的(代表)概要图都是频繁的. 因此,在概要图缩减过程中,不存在非频繁子图引起的子图同构检测. 与此不同,传统封闭子图挖掘算法需要枚举各候选子图的所有 1-边扩展,之后通过借助子图同构过滤非频繁扩展. 大量非频繁子图引起的子图同构检测会增加计算成本. 根据以上对基于概要图缩减过程的描述,不难得到以下结论.

定理 4. 被概要图包含的,规模大于 K 的子图,都可以通过基于概要图的缩减得到.

证明. 由于得到封闭子图前,先得到了概要图数据库. 概要图数据库的获得是通过计算概要图与图数据集中每一个图是否存在一个规模为 K 的公共子图得到的. 因此,与某个概要图相关的规模为 K 的封闭子图的支持集可以全部得到,即被概要图包含的,规模大于 K 的封闭子图都可以得到. 证毕.

虽然本文挖掘的结果集是规模至少为 K 的闭图模式(规模过小的闭图模式在很多实际应用中缺乏应用价值),但若想继续挖掘规模小于 K 的闭图模式,可以通过已经得到的规模为 K 的闭图模式,计算这些闭模式与图数据的公共子图,再次得到相应的概要图数据库. 最后,挖掘概要图数据库得到规模小于 K 的闭图模式.

τ-核图邻域合并得到的概要图具有局部极大性. 根据定理 5,存在着一些比概要图的尺寸更大,包含概要图的全局封闭子图. 为了得到这些封闭子图,需要执行基于概要图的扩展. 具体的做法是,以概要图为基础,进行逐边扩展,检查概要图每个超图的封闭性. 由于概要图的规模比较接近全局极大子图,此时虽然采用的仍然是自底向上的子图增长方式,但增长的"基点"是概要图,而非传统的频繁 1-边. 因此,基于概要图的扩展大大缩减了大尺寸封闭子图的搜索代价. 同时,定理 6 保证所有包含概要图的封闭子图都可以通过对概要图的扩展得到.

定理 5. 具有局部极大性的概要图不一定具有全局极大性,不具有局部极大性的概要图一定不具有全局极大性.

证明. 给定一个图数据库 $\mathcal{G}$,将图数据库进行水平划分,分成 $\mathcal{G}=\{\mathcal{G}_1,\mathcal{G}_2,\cdots,\mathcal{G}_n\}$,在子图集 $\mathcal{G}_i$ 使用全局支持度挖掘得到局部极大图模式 g. 在 $\mathcal{G}_i$ 中 g 的任何真超图 g' 的支持度 $Sup(g',\mathcal{G}_i)<min\_sup$,设存在 g',g' 在 $\mathcal{G}$ 中的支持度 $Sup(g',\mathcal{G})=\sum_{j\neq i}^{n}Sup(g',\mathcal{G})+Sup(g',\mathcal{G}_i)\geq min\_sup$,$g'$ 为频繁模式,推出 g 为全局非极大模式,故局部极大图模式不一定是全局极大图模式. 设 g 为局部非极大图模式,在 $\mathcal{G}_i$ 中,g 的真超图 g',$Sup(g,\mathcal{G}_i)\geq Sup(g',\mathcal{G}_i)\geq min\_sup$,能推出 $Sup(g,\mathcal{G})\geq Sup(g',\mathcal{G})\geq min\_sup$,由定义可知 g 不是全局极大图模式,故局部非极大图模式一定不是全局极大图模式. 证毕.

定理 6. 包含概要图的全局封闭子图都可以通过概要图的扩展得到.

证明. 由于概要图是局部极大模式,由定理 5 可知局部非极大模式一定不是全局极大模式. 因此,概要图会被全局极大模式所包含. 又由于概要图数据库至少包含了规模为 K 的闭图模式的数据信息,因此包含概要图的闭图模式都可以得到. 证毕.

4.5 性能优化

本节提出两个策略优化算法性能,包括基于子图相似度的计算削减和 all-in-one 的重复模式压缩.

(1) 基于子图相似度的计算削减

计算子图相似度时,需要计算两个图的最大公共子图. 然而,精确求解最大公共子图代价很高. 可以通过式(3)中的相似度上界,减少公共子图的计算代价.

$$sim(g_1,g_2)'=\frac{|E_{g_1}\cap E_{g_2}|}{\max(|E_{g_1}|,|E_{g_2}|)} \tag{3}$$

可以看出,$sim(g_1,g_2)'\geq sim(g_1,g_2)$. 若任意两个图模式 g_1,g_2 的 $sim(g_1,g_2)'\leq\delta$,则不需要计算这两个图的最大公共子图,从而减少计算代价. 若任意两个图 g_1,g_2 的 $sim(g_1,g_2)'>\delta$,g_1 和 g_2 只要存在一个边数为 $\max(|E_{g_1}|,|E_{g_2}|)\times\delta$ 的公共子

图，则不需要计算 g_1，g_2 的极大公共子图，减少枚举极大公共子图时间.

(2) all-in-one 的重复模式压缩

执行封闭子图的分布式挖掘时，每个计算节点都会被分配一部分概要图以及相应的概要图数据库. 虽然代表概要图之间的相似性低于给定的阈值，但仍可能存在着公共子图. 当一个概要图被处理完时，下一个概要图中可能会出现重复的子图缩减和扩展，导致冗余计算. 提出 all-in-one 的重复模式压缩策略，消除同一计算节点内部执行基于概要图的缩减和扩展时的重复子图检测. 为提升效率，对从某个概要图枚举得到的子图 g，若其在其它概要图中也存在，则当扩展模式 g 时，把它在其它概要图中相应的扩展也都同时处理. 以此保证节点内重复的模式只被枚举一次.

例 3.　设图 6 中的(a)和(b)为分配至同一计算节点的两个概要图. 朴素的做法是分别单独枚举每个概要图，这种做法会产生大量重复子图，例如，在(a)中会枚举出子图 CAB，当枚举(b)中所包含子图时，CAB 构成的子图会再次被枚举出来，存在重复计算. 本文的做法是首先将图中每条边按照 DFScode 排序，先扩展 AB 边，在(a)中扩展 AB 边时，同时将(b)中对 AB 边的扩展也枚举出来，扩展后的结果如图 6 中(c)～(e)所示. 对每个新生成的子图，递归地重复该过程，直到不能再扩展. 递归回溯后，选取其它单边继续进展，直到所有子图都被枚举出来. 可以看出，该过程中，重复子图只会被处理一次.

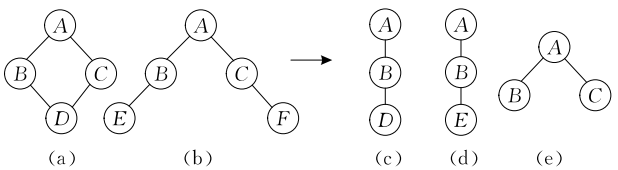


图 6　枚举子图

4.6　讨　论

如前文指出，封闭子图挖掘被证明是 NP-难问题[5]，精确获得所有的封闭子图模式被证明是 #P-完全问题[5]. 因此，对大规模图数据中的封闭子图挖掘任务而言，比较理想的求解方案是设计一种具有常量近似比的近似算法，即在多项式时间复杂度内获得数量以常量比例接近精确结果数量的算法.

然而，文献[30]指出，频繁(封闭)项集挖掘问题可规约至发现极大平衡完全二部子图问题(简称为 BCBS 问题). 进一步，通过理论分析指出，设计这样一种高效的且有效的具有常量因子近似比的多项式时间频繁(封闭)项集近似挖掘算法，从计算复杂性的角度来讲，可能性不大. 由于频繁(封闭)项集和频繁(封闭)子图问题可规约至相同问题[5]，该结论也同样适用于频繁(封闭)子图挖掘，即不大可能存在一个具有常量因子近似比的多项式时间频繁(封闭)封闭子图近似挖掘算法. 因此，在后文的实验分析中，仅通过对比精确算法和本文提出算法的输出结果数量，提供算法近似程度的直观分析，这是许多研究工作在实践中采取的方式[5].

5　实验分析

5.1　数据集

本文数据集主要分为两类，真实数据集和模拟数据集. 实验中共使用了 5 个真实数据集，真实数据集来自于一些化合物数据库，见表 1. 为更全面的测试本文提出方法的性能，引入两个不同的模拟数据集生成器，分别用于产生频繁模式稠密和稀疏的图来对比实验效果. 工具 Graphgen[31] 生成较为稀疏的模拟数据集 Dataset1～4，文献[32]生成较为稠密的模拟数据集 Dataset5～12.

表 1　数据集

数据集	图数量	$\lvert E\rvert$	$\lvert V\rvert$
P388①	41472	23.3	22.11
Yeast①	79601	22.8	21.54
NCI②	265242	41.21	40.48
Enamine1③	467024	23.28	21.62
Enamine2③	1822038	27.03	25.06
Dataset-1～4	1000000	25	11.2～22.8
Dataset-5～8	1000000	25～40	19.9～28.9
Dataset-9～12	100～250(万)	25	19.9

5.2　实验环境配置

本文提出的 Desu-FSM 算法采用 Java 语言编写. 算法实验的运行平台及软硬件环境如下：

(1) 实验环境配置. 集群由 15 台服务器组成，每台服务器的配置为：Red hat 64 位操作系统，128 GB 内存；Hadoop 版本为 2.6.0，Spark 版本为 1.6.0，JDK 版本为 1.7.0.

(2) 开发环境配置. 操作系统为 Windows 7 64 位旗舰版，主频 3.30 GHz，8 GB 内存，2 T 硬盘；开发工具为 Eclipse.

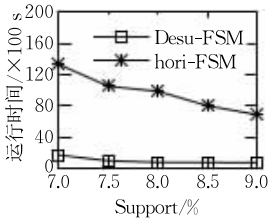
① http://www.cs.ucsb.edu/xyan/dataset.htm
② NCI. http://cactus.nci.nih.gov/download/nci
③ Enamine. http://www.enamine.net

5.3 实验结果及分析

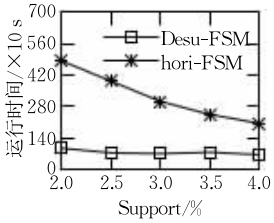
就目前的文献检索情况看，暂未发现分布式频繁封闭子图挖掘的相关工作(包括基于数据水平划分和垂直分解两种)．因此，本文实现了以下实验中的对比算法 hori-FSM．该算法基于水平数据划分，先对数据集进行近似的等规模划分．然后，在每个节点独立使用局部支持度得到局部候选闭图模式．结果汇总后，借助于子图同构测试过滤掉非全局频繁模式，获得全局闭图模式．

(1)运行时间

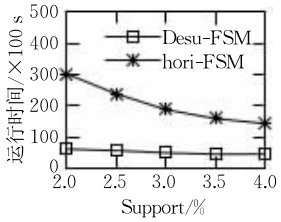
在图 7 中对比分析了 Desu-FSM 算法与 hori-FSM 算法在不同规模的真实数据集下的运行时间．对于 Desu-FSM 算法，默认相似度阈值 $\delta=90\%$，模式距离 τ 为 0.5，K 为 3．所用真实数据集分别为 NCI、Enamine1 和 Enamine2，数据集 NCI 平均边数相对较大，而 Enamine1 和 Enamine2 图数量规模相对较大．从图 7 中可以看出，Desu-FSM 算法总是优于 hori-FSM 算法，尤其在规模较大的 Enamine2 数据集和平均边数较大的 NCI 数据集中，本文提出的 Desu-FSM 算法比 hori-FSM 算法的效率提高了一个数量级．



(a) NCI



(b) Enamine 1

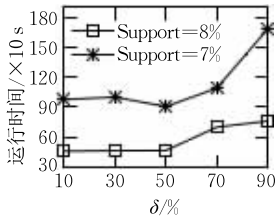


(c) Enamine 2

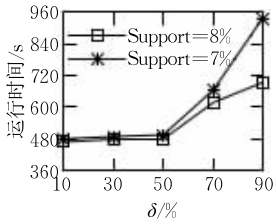
图 7 真实数据集上的运行时间

(2)相似度对算法的影响

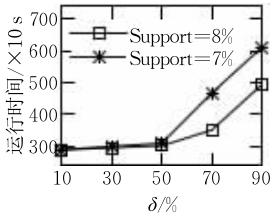
不同的相似阈值 δ 可能导致实验产生不同的效果．为了分析本文提出的算法在不同相似度阈值下的运行效果，本实验通过改变参数 δ 来分析算法的运行时间，所用数据集仍然为图 7 中的三个真实数据集，实验结果见图 8．从图 8 中可以看出，随着相似度的增大，运行时间也相应增加．这是由于随着相似度的递增，获得的概要图数量也相应增加，在依据概要图枚举闭图模式时会得到更多的模式，从而影响了算法的运行时间．但是，随着相似度参数的递增，算法的运行时间呈线性增长，并没有大幅上升．因此，本文提出的 Desu-FSM 算法在相似阈值 δ 变化时，仍具有良好的性能．



(a) NCI



(b) Enamine 1



(c) Enamine 2

图 8 不同相似度 δ 下的运行时间

(3)算法的可伸缩性

以下实验为在改变集群中机器数的情况下，在模拟数据集 Dataset5 上分别运行 Desu-FSM 算法与 hori-FSM 算法得到的结果，支持度阈值 $support=5\%$．从图 9 中可以看出随着节点数的递增，两种算法的运行时间都会递减．这是由于当机器数增加，任务并行度增加，因此总体挖掘时间降低．同时，从图 9 也可以看出，在运行时间方面 Desu-FSM 算法总是优于 hori-FSM 算法．图 10 为实验加速比分析，加速比计算公式为

$$SP=\frac{T}{TK} \tag{4}$$

其中 T 为程序在 1 台机器上的运行时间，T_K 为程序在 K 台机器上的运行时间．如图 10 所示，随着节点的不断增加，算法加速比在不断变大，随着节点数的增加，加速比呈线性变化．由于概要图尺寸分配并非完全均衡，Desu-FSM 算法的加速比仅略优于 hori-FSM 算法，在未来工作中将通过研究负载均衡策略，进一步提高 Desu-FSM 算法的加速比．

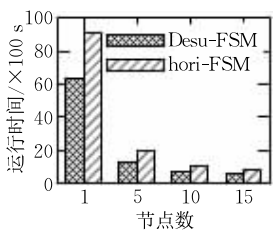


图 9 不同节点数下的运行时间

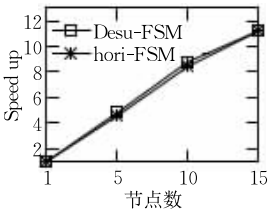


图 10　加速比

(4)算法的可扩展性

本文算法借助 spark 实现并行化处理，在算法处理过程中，通信代价将是一个影响运行时间的主要因素.因此需要通过增加数据的规模来观察算法的扩展性情况.

本文提出的 Desu-FSM 算法能够有效提升图数据库中图的平均节点度数过多时频繁闭图模式的挖掘效率.为验证这个结论，本实验通过保持相同节点数，不断增加数据集中的边数，观察两个算法运行时间随图的平均边数增加的变化情况.使用数据集为 Dataset5～Dataset8，支持度阈值 $support=5\%$.四个模拟数据集的图数据数量相同，平均边数从 25 增长到 40.如图 11 所示，随着数据集中平均边数的增加，本文提出的 Desu-FSM 算法的运行时间总是优于 hori-FSM 算法，尤其在平均边数为 40 时，对比效果更加明显.

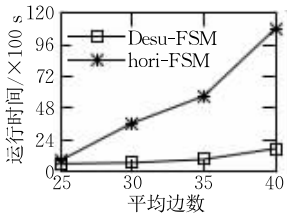


图 11　不同平均边数下的运行时间比较

图 12 为在相同平均边数，不同密度下的算法运行时间分析.数据集为 Dataset1～Dataset4，四个模拟数据集图数量相同，密度从 0.1 变化到 0.4，支持度设为 0.3%，K 值为 2，密度公式见式(5).从图 12 中可以看出，在不同密度设置下，Desu-FSM 算法的运行时间总是优于 hori-FSM 算法.

$$density=\frac{|E|}{|V|/((|V|-1)/2)} \tag{5}$$

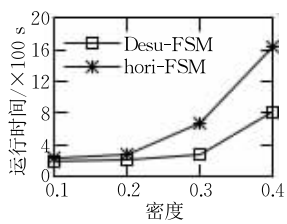


图 12　不同图密度下的运行时间比较

图 13 为数据集规模从 100 万～250 万图数量变化下的算法运行时间，支持度为 5%.随着数据集中图数量的增加，Desu-FSM 算法运行时间总是优于 hori-FSM 算法，因此 Desu-FSM 算法比 hori-FSM 算法具有良好的可扩展性.如正文所述，图挖掘算法的效率瓶颈主要在于图数据的平均节点度数.本实验中图数据的平均边数为 25(低于图 11 实验中的大部分边数设置)，因此 Desu-FSM 在响应时间上优于 hori-FSM 的效果不如图 11 明显，这也从另一侧面验证了我们有关图模式挖掘效率的分析.

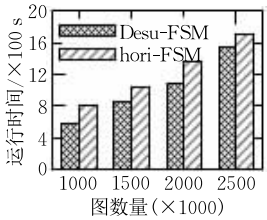


图 13　不同数据规模下的运行时间比较

(5)内存占用情况

本文分析了 Desu-FSM 算法与 hori-FSM 算法在相同规模图数量不同平均边数的模拟数据集上，每个节点的内存占用情况，以及在不同规模的真实数据集上，每个节点的内存占用情况.从图 14、图 15 可以看出，Desu-FSM 算法在模拟数据集中，随着数

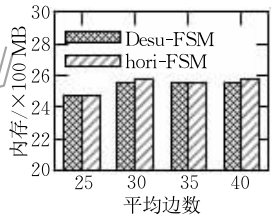
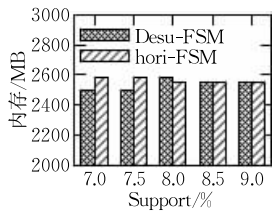
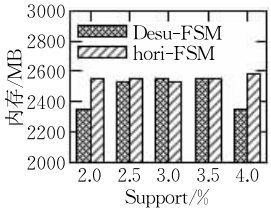


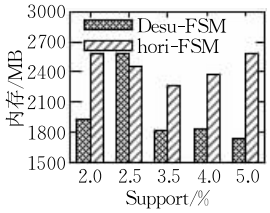
图 14　不同平均边数下内存占用比较



(a) NCI



(b) Enamine 1



(c) Enamine 2

图 15　不同真实数据集下的内存占用比较

据集边数增大时，内存占用总是低于或接近 hori-FSM 算法.类似的，在真实数据集中的内存占用也总是低于或接近 hori-FSM 算法.

(6)近似程度比较

如 4.6 节所述，对频繁闭图模式挖掘问题来说，理论上不大可能存在一个多项式时间复杂度的常量因子近似算法.根据大多数研究处理类似情况的惯例，为分析本文算法挖掘结果的近似程度，图 16 中给出了 Desu-FSM 算法获得的闭图模式数量与 Close Graph 算法挖掘的准确结果数量比较.因为精确结果是通过单机 Close Graph 算法获得的，其无法处理规模过大的数据集，所以本文选择 Yeast 与 P338 两个较小规模的真实数据集进行比较，从图中可以看出本文算法在支持度较小的时候能够产生更接近精确算法的结果.这是因为支持度较小时，会产生更多的频繁子图，这意味着垂直算法在固定数据分片内产生的代表子图数量更多.因此，得到的闭图模式数量比支持度大时更接近于水平算法和精确算法产生的闭图模式数量.

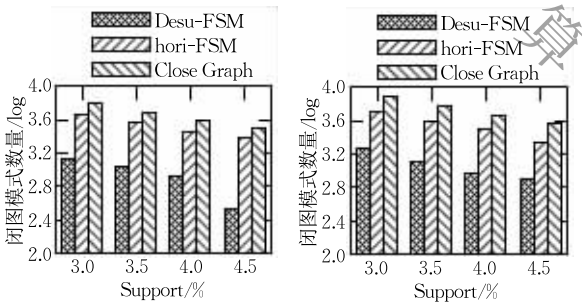


图 16 不同支持度下的结果数量比较

4.2 节中提到，通过多轮随机数据分片，可获得更多的结果模式.图 16 给出了 Desu-FSM 算法在不同支持度下，基于对数据集的一次随机分片，获得的结果图模式与基于水平划分框架的 hori-FSM 算法和精确算法 CloseGraph 的对比情况.图 17 和图 18 给出了在多轮随机数据划分下，三者获得的结果图模式数量比较情况.同时也分析了算法在两个真实数据集中运行时间.其中支持度分别为 1%与 3%.从图中可以看出，随着轮数的增加，Desu-FSM 算法获得的结果数量会显著增加，而 hori-FSM 算法的结果数量没有显著变化，并且 Desu-FSM 算法运行时间总是优于 hori-FSM 算法.因为是随机分区，hori-FSM 算法在每一轮的挖掘过程中会出现大量不同的候选子图，而大部分候选子图并不是全局频繁的，故经过多轮以后，结果数量并没有大幅增加，运行时间反而大幅增加.

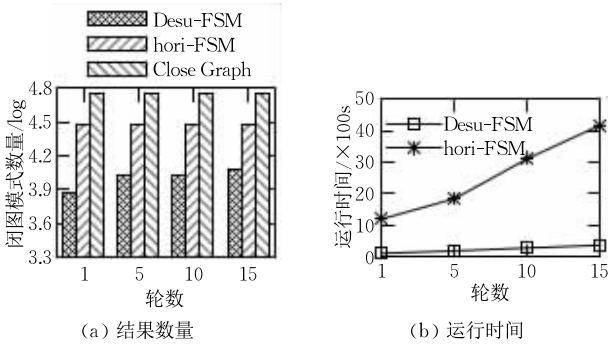


图 17 Yeast 数据集上的比较

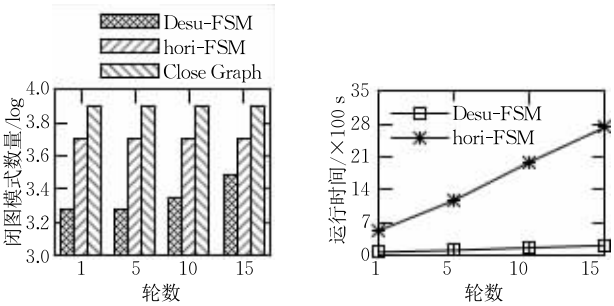


图 18 P388 数据集上的比较

6 总 结

本文提出了一种新的并行挖掘频繁闭图模式的算法 Desu-FSM.与基于数据水平划分的分布式挖掘框架不同，该算法采用了基于垂直分解的分布式挖掘框架.具体而言，两者的不同之处在于前者在“数据物理分治”的基础上，借助分布式框架提升整体计算效率，后者基于“任务逻辑分治”利用分布式框架提升整体计算效率.Desu-FSM 算法利用概要图集实现了整体任务的逻辑分治.大量实验表明，在大规模图数据的封闭子图模式挖掘中，基于垂直分解框架的挖掘效率相较于水平划分框架的效率可提升一个数量级.同时，具有更少的内存占用.

但是，通过加速比方面的实验看出，与基于水平划分框架的对比算法相比，Desu-FSM 算法在加速比方面的优势不是十分显著.通过分析发现，这主要是由概要图的尺寸分布不均引起的.如果某(些)概要图的尺寸显著大于其它概要图，根据“短板效应”，算法整体性能的提升将受到这些概要图的制约.因此，如何得到一组大小比较均衡的概要图，实现任务的均衡分配，是今后工作的研究重点.

参 考 文 献

[1] Li P, Heo L, Li M, et al. Protein function prediction using

frequent patterns in protein-protein interaction networks//Proceedings of the 8th International Conference on Fuzzy Systems and Knowledge Discovery. Shanghai, China, 2011: 1616-1620

[2] Belghaouti F, Bouzeghoub A, Kazi-Aoul Z, et al. FreGraPaD: Frequent RDF graph patterns detection for semantic data streams //Proceedings of the IEEE 10th International Conference on Research Challenges in Information Science. Grenoble, France, 2016: 1-9

[3] Moosavi S A, Jalali M, Misaghian N, et al. Community detection in social networks using user frequent pattern mining. Knowledge and Information Systems, 2017, 51(1): 159-186

[4] Yan X, Han J. CloseGraph: Mining closed frequent graph patterns//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, USA, 2003: 286-295

[5] Yang G. The complexity of mining maximal frequent itemsets and maximal frequent patterns//Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, USA, 2004: 344-353

[6] Borgelt C, Meinl T, Berthold M R. Advanced pruning strategies to speed up mining closed molecular fragments//Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Hague, The Netherlands, 2004: 4565-4570

[7] Demetrovics J, Quang H M, Anh N V, et al. An optimization of closed frequent subgraph mining algorithm. Cybernetics & Information Technologies, 2017, 17(1): 3-15

[8] Cook S A. The complexity of theorem-proving procedures//Proceedings of the Symposium on the Theory of Computing. Shaker Heights, USA, 1971: 151-158

[9] Wang C, Wang W, Pei J, et al. Scalable mining of large disk-based graph databases//Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, USA, 2004: 316-325

[10] Nguyen S N, Orlowska M E, Li X. Graph mining based on a data partitioning approach//Proceedings of the 19th Conference on Australasian Database. Wollongong, Australia, 2008: 31-37

[11] Cook D J, Holder L B, Galal G, et al. Approaches to parallel graph-based knowledge discovery. Journal of Parallel & Distributed Computing, 2001, 61(3): 427-446

[12] Buehrer G, Parthasarathy S, Chen Y K. Adaptive parallel graph mining for CMP architectures//Proceedings of the International Conference on Data Mining. Hong Kong, China, 2006: 97-106

[13] Meinl T, Wörlein M, Urzova O, et al. The ParMol package for frequent subgraph mining. Electronic Communications of the EASST, 2006, 1(1): 2006

[14] Borgelt C, Berthold M R. Mining molecular fragments: Finding relevant substructures of molecules//Proceedings of the IEEE International Conference on Data Mining. Maebashi City, Japan, 2002: 51-58

[15] Yan X, Han J. gSpan: Graph-based substructure pattern mining//Proceedings of the IEEE International Conference on Data Mining. Maebashi City, Japan, 2002: 721-724

[16] Huan J, Wang W, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism//Proceedings of the IEEE International Conference on Data Mining. Melbourne, USA, 2003: 549-552

[17] Nijssen S, Kok J N. A quickstart in frequent structure mining can make a difference//Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, USA, 2004: 647-652

[18] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107-113

[19] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing//Proceedings of the USENIX Conference on Networked Systems Design and Implementation. San Jose, USA, 2012: 15-28

[20] Yan X, Zhang J, Xun Y, et al. A parallel algorithm for mining constrained frequent patterns using MapReduce. Soft Computing, 2017, 21(9): 2237-2249

[21] Gahar R M, Arfaoui O, Hidri M S, et al. ParallelCharMax: An effective maximal frequent itemset mining algorithm based on MapReduce framework//Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA). Hammamet, Tunisia, 2017: 571-578

[22] Gonen Y, Gudes E. An improved MapReduce algorithm for mining closed frequent itemsets//Proceedings of the IEEE International Conference on Software Science, Technology and Engineering. Beer Sheva, Israel, 2016: 77-83

[23] Yu D, Wu W, Zheng S, et al. BIDE-based parallel mining of frequent closed sequences with MapReduce//Proceedings of the Algorithms and Architectures for Parallel Processing International Conference. Fukuoka, Japan, 2012: 177-186

[24] Hill S, Srichandan B, Sunderraman R. An iterative MapReduce approach to frequent subgraph mining in biological datasets//Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine. Orlando, USA, 2012: 661-666

[25] Bhuiyan M A, Hasan M A. An iterative MapReduce based frequent subgraph mining algorithm. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(3): 608-620

[26] Lin W, Xiao X, Ghinita G. Large-scale frequent subgraph mining in MapReduce//Proceedings of the IEEE International Conference on Data Engineering. Chicago, USA, 2014: 844-855

[27] Sethi K K, Ramesh D. HFIM: A spark-based hybrid frequent itemset mining algorithm for big data processing. Journal of Supercomputing, 2017, 73(8): 3652-3668

[28] Karim M R, Cochez M, Beyan O D, et al. Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach. Information Sciences, 2018, 432: 278-300

[29] Zhu F, Yan X, Han J, et al. Mining colossal frequent patterns by core pattern fusion//Proceedings of the IEEE International Conference on Data Engineering. Istanbul, Turkey, 2007: 706-715

[30] Buehrer G, Oliveira R L D, Fuhry D, et al. Towards a parameter-free and parallel itemset mining algorithm in linearithmic time//Proceedings of the IEEE International Conference on Data Engineering. Seoul, South Korea, 2015: 1071-1082

[31] Cheng J, Ke Y, Ng W, et al. FG-Index: Towards verification-free query processing on graph databases//Proceedings of the ACM International Conference on Management of Data. Beijing, China, 2007: 857-872

[32] Kuramochi M, Karypis G. Frequent subgraph discovery//Proceedings of the IEEE International Conference on Data Mining. San Jose, USA, 2002: 313-320

LI Ling, Ph. D. candidate. Her major research interests include big data mining and parallel computing.

YIN Ying, Ph. D., associate professor. Her major research interests focus on data mining.

ZHAO Yu-Hai, Ph. D., professor. His major research interests include database, data mining and bioinformatics.

WANG Guo-Ren, Ph. D., professor. His major research interest is database.

DONG Xiang-Jun, Ph. D., professor. His major research interests include data mining, artificial intelligence and database.

Background

Graph mining is one of the key research area in the field of data mining. Mining frequent subgraphs from large scale graph data sets has been widely used in many fields, but its huge amount of result has brought many inconveniences to subsequent analysis. In order to reduce the number of frequent subgraphs, closed subgraphs and maximal subgraphs are proposed respectively. Over the years, although many algorithms have been proposed to solve this problem, they are faced with a common problem of computational efficiency when mining large scale graph data. In particular, when the average degree of vertexes increases, the efficiency of mining decreases sharply. The use of distributed computing is expected to solve this problem, but the traditional distributed mining algorithm using horizontal decomposition will face a large number of candidate subgraphs and a large number of subgraphs isomorphism tests when mining closed graph patterns.

This paper proposes an efficient distributed mining algorithm for large scale graph data. Unlike the existing distributed mining framework based on horizontal decomposition, this algorithm first adopts the distributed mining framework based on vertical decomposition to quickly obtain the closed patterns. The experimental results verify that the presented algorithm is efficient and effective.

This work was supported by the National Key Research and Development Program of China (2018YFB1004402) and the General Program of the National Natural Science Foundation of China (61772124).