DOI:10.11721/cqnuj20150523

基于云计算的大数据大图模式的高效挖掘的研究*

梁 杰
(长春职业技术学院 基础部,长春 130000)

摘要:在图形挖掘研究领域对大图数据进行挖掘方法很多,尽管云计算在解决传统计算问题上十分有效,但对大图形的频繁挖掘中图形分割、信息不对称及模式保留合并仍存在问题。研究提出一种全新的基于云的 SpinderMine 挖掘法,即 c-SpinderMine 方法。首先利用云计算来对大图形数据的大模式进行挖掘,解决大图形数据挖掘算法在云环境下面临的上述问题。然后进行三组真实的数据集实验,有效地缩短执行时间,且对处理云上的大数据表现出很高的可扩展性能。最后实验证明在不同数据大小和最小支持度时具有优越的内存使用和执行时间,对应付云环境下的大图形数据表现出优越的能力。

关键词:大数据;云计算;图模式挖掘;频繁模式挖掘
中图分类号:TP301.6 **文献标志码**:A **文章编号**:1672-6693(2015)05-0119-06

目前已有通过图形挖掘研究来达到对网络或数据库的模式进行研究的方法[1-2]。最近几年,这一方法已得到许多应用,用来发掘管理数据库的结构模式[3];而且文献[4]里提供了一种高级的图形挖掘法用来评估社交网络的性能。在生物医学方面,文献[5]提出子图同构算法,运行时间大大缩短。但是,鉴于问题的复杂性以及需要多方面综合考虑,上述方法仍难以应对大图形数据。文献[6]中实验表明 SEuS 和 MoSS 所能支持的节点数是 1 000 个。虽然 SpiderMine 和 SUBDUE 在这些实验上展示了良好的可扩展性,但实验所用数据较小,对大数据是否有效还不得而知。因此,本文首先构建一种机制挖掘出大图形数据里前 K 个大模式,为事后应付大图形数据提供有效方法。根据文献[7]的发现,SpiderMine 比 SUBDUE、SEuS、或 MoSS 所需要的运行时间都要短。因此本文中选择 SpiderMine 作为此次实验的对比标准。

最近,有关云计算的三大优势已得到证实:负载平衡、扩展性好、效率高。云计算是将任务划分成映射器,进行数据计算后再以并行方式将任务合并缩减器。当然,基于云计算的图形挖掘还存在一些困难:①图形分割问题。传统算法难以生成小且直径均等的模式,当采取不恰当的模式分割时会引发模式费用问题。这些费用涉及模式恢复、模式搜索以及模式合并三方面。鉴于此,如何界定并有效利用恰当的模式分割法是本文将遇到的难题之一。②信息不对称。每台设备间缺乏交流可能导致信息缺失并增加不必要的计算工作。③模式保存合并问题。进行合并操作时,关键是确保不同分区的模式仍可以被找到,且能与其他模式有效地进行合并。

具体而言,本文提出的云计算方法是 c-SpiderMine,有别于 SpiderMine。c-SpiderMine 对挖掘标号图已经表现出了高效、有效的性能。SpiderMine 是一种基于逼近值的图形模式挖掘法,利用随机游走方式来挖掘模式,但无法保证所有模式被发现。同样,本文旨在通过利用完整的模式搜索方式来找出模式,走遍所有节点以免模式遗漏[8]。c-SpiderMine 包含 3 个部分:分区、挖掘和合并。分区阶段利用最小分割法将一个大的图形数据分割成几个子图形来达到分割/合并成本最小的目的。第二阶段是挖掘,利用 SpiderMine 模式进行挖掘,可有效减少图形同构实验的成本并生成更复杂、综合系数更低的大模式。更重要的是,本文创建了一个全局表来避免这一阶段出现信息不对称。最后阶段是模式的合并。应用模式密钥(PK)功能对模式进行保存,这就保证了所有模式可以成功得到恢复并进行合并。结果表明与 SpiderMine 相比,在不同的最小支持度和图形大小来说,c-SpiderMine 在执行时间上更胜一筹。

1 相关研究

频繁模式挖掘是图形挖掘领域最重要的课题之一。但鉴于复杂程度高的缘故,对单个大图形进行挖掘研究

* **收稿日期**:2014-06-13 **修回日期**:2015-04-27 **网络出版时间**:2015-06-08 12:29
作者简介:梁杰,女,讲师,研究方向为计算机网络技术、数据库技术,E-mail:qq86365927@163.com
网络出版地址:http://www.cnki.net/kcms/detail/50.1165.n.20150608.1229.010.html

还未引起重视。SUBDUE[9]可以说是基于最小描述长度(MDL)原理对单个大图形进行挖掘的最知名技术。SUBDUE能够发现子结构,压缩数据库并呈现结构的概念。不过,SUBDUE计算工作量大。另一种方法是SEuS,通过对数据的结构进行缩减,对非频繁搜索空间进行修剪来解决效率问题[10]。尽管修剪方法对小规模的图形见效,但要处理大图形对更频繁的图形进行挖掘,还有一定的难度。在文献[7]里采用一种方法对单个图形的频繁模式进行挖掘,但对大图形还是无法应对。另一种高效的方法是SpiderMine,将小的频繁图形合并成较大的图形而解决上述难题。它应用概率挖掘机制来找出前K个较大的图形,具体做法是:①确定一条合适的增长线;②生成复杂度较低的大模式;③降低图形同构实验的成本。SpiderMine成功发掘了最大的模式且运行时间最短,优于SUBDUE、SEuS和MoSS方法。

过去的两年里,基于云概念先后提出许多图形挖掘法。在文献[11]里,作者提出应用一个库程序来找出万亿级和千兆级图形的半径和连通分量,这一方法在MapReduce模型里证明了良好的可扩展性。他们还提出GIM-V法来求证可扩展性和线性运行时间效果[12]。再有人提出图形挖掘系统OPAvion,包含合并、异常检测和交互可视化这3个模块[13]。OPAvion引导用户逐渐发现图形,从它们选中的阶段或旗状的异常节点开始,然后用户可以扩展到节点的附近,对它们标号进行归类,然后对图形感兴趣的部分进行交互式导航操作。正如在文献[14]里,作者引入Mizan,它为平衡计算和交流提供了有效的细粒顶点迁移。结果表明在静态图形再分割的平衡问题上,Mizan较其他技术提高了84%。也有人提出有效分析平台来对大图形进行处理,即Gbase[15]。Gbase对图形操作提供并行索引机制,既节省存储空间又快捷查询响应。之前的研究方法基本上可分为单一大图子图挖掘法、图形挖掘工具和负载平衡增强法这几种类型。本文首先通过c-SpiderMine方法对大图形数据划分了3个阶段,然后通过实验对SpiderMine和c-SpiderMine不同节点数情况下的执行时间、不同最小支持度下的执行时间以及内存使用情况进行了比较,最后实验证明提出的c-SpiderMine算法对不同支持度、不同节点大小、设备数和平均度还表现出良好的可扩展性能。

2 C-SpiderMine 方法

针对以上提到的问题,本文通过c-SpiderMine方法来解决上述问题。首先介绍如何应用最小边分割法将大图形数据划分为几个子图形;然后,阐述每台设备如何借助全局表相互交流;最后归纳PK函数如何对网页爬虫进行恢复和合并,防止有模式遗漏。

2.1 划分阶段

如何找到合适的方法对图形进行划分,本文应用最小切割边缘法来解决该问题。这种切割的优点在于边缘数越少,切割时被破坏的模式也越少。可以保留下更多的已有模式,这就为在余下阶段对相同模式进行搜索节约时间。最小切割边缘集见定义1。

定义1 设定图$G=(V,E)$,其中,V是顶点集,E是边集,割边集是E_c,E_c把G分为$\{S_1,S_2,\cdots,S_n\}$分区,最小割边集$E_c=(S,T)$,$G=(V,E)$是V的一个分区,$T=V-S$,E_c的容量$=\sum_{u\in S,v\in T}$。

这一阶段还有一个问题,通过启发式方法很难对模式进行合并,这可能导致有些模式遗漏。因此本文提出PK功能。这一阶段将两个子图形之间的切割关键点记录为一个PK。

为了将图形G分成k个均衡的子图形,每个图形均可保持各自的结构,首先利用最小切割边缘集E_c将一个图形分成几个子图形,然后把最小切割边缘集E_c里的所有节点组(u,v)复制到u和v分别隶属的子图形里。如算法1所示。

算法1 分区阶段

要求:图G,k为图分区号。

确保:$G_{sub}=\{g_1,\cdots,g_k\}$,$G$是分区的子图。

1) $G_{sub}\leftarrow k\text{-}Partition(G,k)$

2) for each $g_i,g_j\in G_{sub}$ do

3) $E_c\leftarrow\{(v_i,v_j)\mid \forall_{vi}\in g_i(V),\ \forall_{vj}\in g_j(V)\}$

4) 加入切边集E_c的在g_i和g_j之间

5) 在设置g_i的V_c的切点之间添加连接边缘

6) 输出所有子图G_{sub}

2.2 挖掘阶段

挖掘阶段分为 2 个阶段完成,第一阶段介绍自主开发的 k 个最大模式挖掘算法,通过这种方法将 SpiderMine 和 MapReduce 模式结合。首先根据 MoSS 的模式增长算法来种植网页爬虫以便在半径范围内挖掘出所有的频繁图形模式。选择 MoSS 是因为它是可以对单个大图形的所有完整频繁图形模式进行挖掘的最先进的模式增长算法。尽管 MoSS 对较大的图形无法表现良好,但可以通过对图形模式半径 r 增加约束条件来使得 MoSS 在挖掘所有频繁模式时更高效。然后通过上述方法在单一处理器上获取所有初始网页爬虫。模式增长型算法首先会选择一个节点作为初始模式,然后通过对所有关联边缘进行模式扩展产生新的候选模式,并且算法会收集嵌入模式。如果嵌入数低于支持阀值,算法就会对候选模式进行修剪。为了并行种植网页爬虫,利用 BSP 模式在相同高度不同子图形范围内进行种植,这就意味着在相同超集里会生成边缘和节点数相同的所有频繁网页爬虫候选模式。

第二阶段将每个网页爬虫候选模式的支持数绘制在一个全局表格里。关键点是每个频繁图形模式候选的散列码;值是每个候选的子图形里嵌入数的支持数之和。在频繁图形模式候选集增长过程中,利用规范形式来对候选模式进行编码,将每个候选模式的局部支持值发送到全局表。然后,在完成一个超集后对非频繁候选模式进行修剪,确保所有处理器会增加候选模式可能嵌入的图形数。通过该做法可以保证由于信息的不对称而存在未经过修剪的模式。图 1 是 BSP 模式下的一个全局表。这一阶段的整个过程见算法 2。

算法 2　挖掘阶段

要求:G_{sub}是分区子图,r 是曲线半径,θ 是最小支持度阈。

确保:$<E_c(G_{id}),S'>$是在 G_{sub}中的切边集和频繁图模式集。

Map(Key k, Value v)

1) K 是关键子图号

2) v 是值的子数据

3) 在 G_{sub}中通过标签频率同步和排序所有节点

4) for all $g_i \in G_{sub}$ do

5) 修剪 g_i 罕见标签和重新标记节点

6) 输出$<G_{id}, G_{sub}>$

Reduce (Key k, Values $v[]$)

1) K 是关键子图号

2) v 是值的子数据

3) 在 G_{sub}中所有的局部频繁 1-边缘图是 S^1

4) for each $s \in S^1$ do

5) $sup_{global}(s) \leftarrow CalcuateSupport(s)$

6) $S \leftarrow S^1$

7) if $S \neq \Phi$ do

8) for each $s \in S$ do

9) if $sup_{global}(s) < \theta$ and $Radius(s) \neq r$ then

10) $S' \leftarrow S - \{s\}$　　else

11) $S' \leftarrow GrowPattern(s)$是生成候选图的模式和更新$sup_{global}$

12) $sync(s, sup_{global})$是 BSP 模型的同步

13) 输出 $<E_c(G_{id}), S'>$

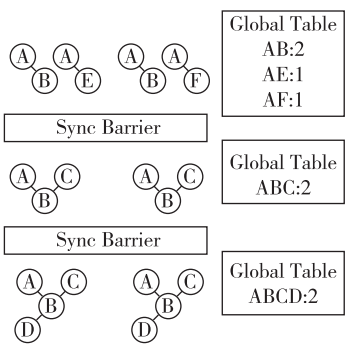


图 1　BSP 模型的一个全局表

2.3 合并阶段

这一阶段分两项 MapReduce 工作来实现。首先将不同子图形里的网页爬虫扩散到较大的模式上。为了解决合并问题,本文提出了 PK 功能,通过 PK 功能提供一些重叠性的关键点。PK 功能可保留初始关系并提供两个子图形之间的关联。PK 功能的有关解释见定义 2。

定义 2　给定一个图 $G=(V,E)$,其中 V 是节点集,E 是边集,复制节点集是 V_c。定义的割点重叠切结 $u_c \in V_c$ 是$\{O_{u_c}\} = \{u \mid \forall (u_c, u) \in E\}$。

另一个阶段是模式修剪工作。当两个模式同型时,要对复制的模式进行修剪。然后,再计算每个模式的支

持数。最后将所有模式进行合并处理。由于对非频繁模式进行过修剪且进行了同构测试,在检查两个模式是否拥有相同的PK数后,可以对模式进行合并。如果两个模式拥有的PK数相同,会通过相同数的网页爬虫来对他们进行合并。重复操作直至新生成的模式的直径大于预设值。一阶段的整个过程见算法3,具体事例如图2所示。

算法 3 合并阶段

要求:$<E_c$, Cand_Patterns$>$是组图形的切边和候选模式对。

确保:F是频繁图模式。

Map(Key k, Value v)

1) $Cand\_Patterns \leftarrow v$

2) for each $v_i \in Cand\_Patterns$ do

3) for each $v_j \in Cand\_Patterns-\{v_i\} \wedge i \neq j$ do

4) $Q = SpiderExtend(v_i, v_j)$

5) if $Q \neq \Phi$ then

6) $<|Q|, (E_c, Q)>$

Reduce(Key k, Values $v[]$)

1) while $v \neq \varphi$ do

2) for each p_i, $p_j \in v$ do

3) if $p_i = p_j$ then

4) 从 v 修剪 p_j

5) else if p_i and p_j可以合并

6) $F \leftarrow MergePattern(p_i, p_j)$

7) else $F \leftarrow p_i$; $F \leftarrow p_j$

8) if $F \neq \Phi$ then

9) 输出 F

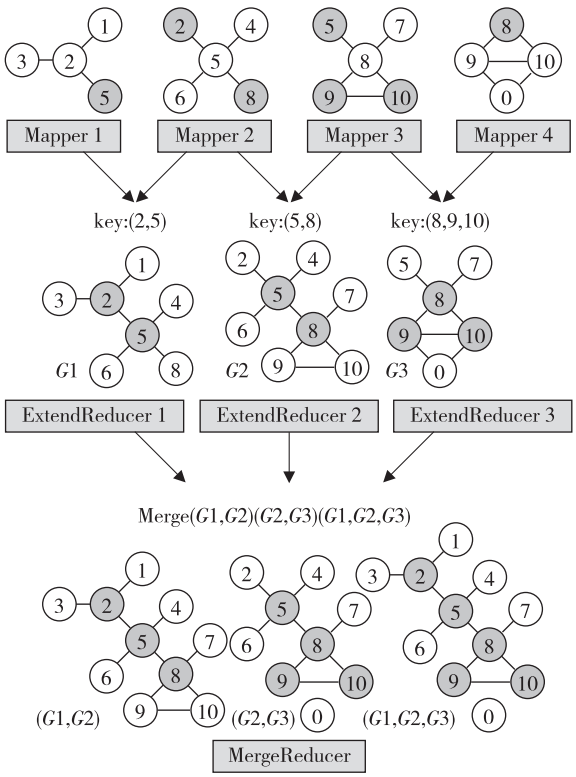


图 2 c-SpiderMine **算法实例**

尽管云计算具有良好的可扩展性,在合并阶段由于太多的边缘会导致模式的破坏,并且分配成本不是最小化成本模式。因此,本文在云计算中采用c-SpiderMine应用最小切割边缘法来解决这该问题。在划分阶段,每个映射开始了自己的任务,直到任务完成。然而,每个映射只满足自身条件,这可能会导致信息不对称。本文使用BSP模型并行运行在不同的内核模式下,通过BSP模型算法保持全局表记录的全球支持计数。在合并阶段,本文提出了PK功能,通过PK功能提供一些重叠性的关键点。PK功能可保留初始关系并提供两个子图形之间的关联。综上所述,云计算机环境通过3个阶段实现c-SpiderMine应用。

3 实验分析与结果

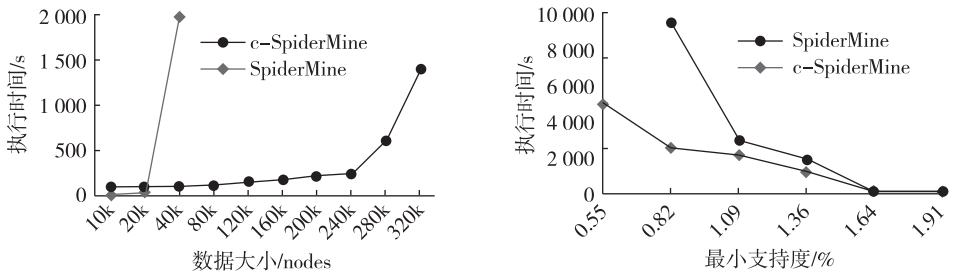
1) 实验环境

本文在由33台虚拟机构成云计算环境下在HAMA 0.5和Hadoop 1.0.3平台上对c-SpiderMine执行研究实验。将一个节点充当主节点,其他节点作次节点。所有实验在英特尔至强服务器上进行,256 GB主内存,1 GB以太网。

2) 算法比较

为了证实c-SpiderMine的性能,本文选用SpiderMine作为基准来比较不同节点数情况下的执行时间、不同最小支持度下的执行时间以及内存使用情况。从网站上选出两组大数据集。第一组是com-DBLP,包含317 080个节点和1 049 866个边缘;第二组是Amazone0302,包含262 111个节点和1 234 877个边缘。这两组数据均广泛用来做真实的实验研究。如图3a所示,当节点体积越来越大时执行时间也越长。由图3可知SpiderMine难以支持大小超出20 000的数据的图形。反之,随着数据越来越大,c-SpiderMine却表现优越。当数据在20 000以下时,c-SpiderMine执行时间较SpiderMine要长,因为前者有一个初始设定时间,相应地会多出180 s。图3b表明c-SpiderMine在执行时间上性能更好,即便最小支持度偏低。而且发现c-SpiderMine在最小支持度小于0.82%的情况下也较SpiderMine要理想。最小支持度决定着频繁模式的数量,当最小的支持度变小时,就会产生更多的频繁模式,因而就导致执行时间拉长。反之最小支持度变大,执行时间就缩短。总之,本文算法c-Spi-

derMine 在处理大图形数据上执行时间要短,内存使用也少,均优于 SpiderMine。当然还需将本文算法与 MoSS 进行比较。但是,当节点大小超出 1 000 时,MoSS 被毁,无法继续实验,恰好印证了文献[6]里的结果。



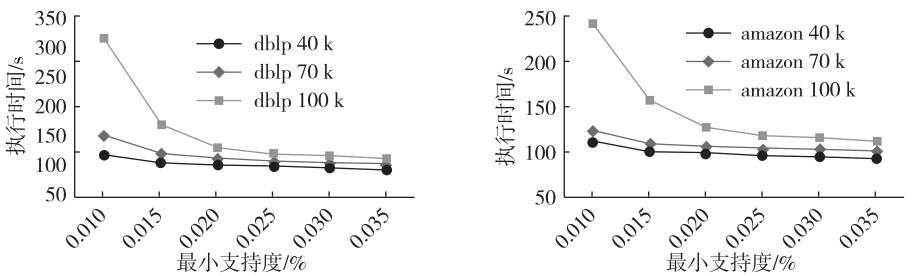
a　com-DBLP　　b　Amazone0302

图 3　c-SpiderMine 算法与 SpiderMine 的比较

3) 可扩展性

尽管本文算法表现出优越的性能,但其支持大图形数据的能力还有待进一步求证。这一节在真实数据集情况下从不同的最小支持度,设备数和平均度几个方面来测试该算法。

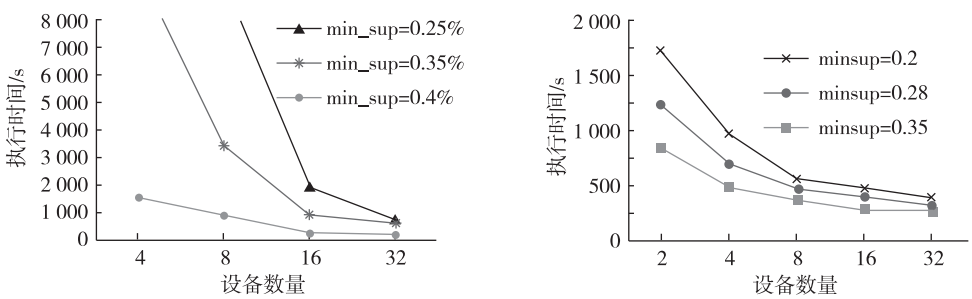
①最小支持数的影响。图 4a 和 4b 分别是 com-DBLP 和 Amazone 0302 的执行时间。这两个实验使用到 40 000,70 000,和 100 000 这 3 个不同节点大小 N 情况下从 0.01%～0.035%之间的不同最小支持度。结果表明当最小支持度增大时,执行时间缩短。这说明当最小支持度变大,生成的模式就较少,执行时间就缩短。而且,当 N 变大时,执行时间也增加,显然,节点越多生成的模式越多,耗费时间也越多。当节点大小和最小支持度均增大时,c-SpiderMine 对执行时间表现出良好的可扩展性。



a　com-DBLP　　b　Amazone 0302

图 4　不同数量的节点的执行时间与最小支持度

②设备数量的影响。这一部分求证不同设备数量情况下的算法性能。利用 4,8,16,32 台设备来求证最小支持度在 0.25%,0.35%和 0.4%时算法在 com-DBLP 上的情况,以及利用 2,4,8,16 和 32 台设备来求证最小支持度在 0.2%,0.28%和 0.35%时算法在 Amazone 0302 上的情况。在图 5a 和 5b 里,执行时间随设备数量增多而成倍数缩短。结果表明设备越多,性能就提升,这进一步说明云计算是增强大图形数据挖掘的可扩展性的一种直接方式。



a　com-DBLP　　b　Amazone0302

图 5　不同数量的机器的执行时间与最小支持度

4 结论

本文提出了一种名为 c-SpiderMine 的新算法,在处理大图形数据时它有效地将 BSP 模式、SpiderMine 和云

计算结为一体。实验证实该算法在不同数据大小和最小支持度时具有优越的内存使用和执行时间,能有效地挖掘云环境下前 k 个大模式。而且,该算法对不同支持度、不同节点大小、设备数和平均度还表现出良好的可扩展性能。总之,本文算法对应付云环境下的大图形数据表现出优越的能力。今后,有必要在更多真实大数据上来验证该算法。此外还要研发出其他的挖掘算法来高效解决云环境下的大数据问题。

参考文献:

[1] 汪建,方洪鹰. 云计算与无线局域网安全研究[J]. 重庆师范大学学报:自然科学版,2010,27(3):64-68.
Wang J, Fang H Y. Cloud computing and research into WLAN security[J]. Journal of Chongqing Normal University: Natural Science, 2010, 27(3): 64-68.

[2] 盖玉莲. 基于云计算的数据管理技术研究[J]. 电子设计工程,2015,23(6):71-74.
Gai Y L. Resear on cloud-computing-based data management technology[J]. Electronic Design Engineering, 2015, 23(6): 71-74.

[3] Anchuri P, Zaki M J, Barkol O, et al. Graph mining for discovering infrastructure patterns in configuration management atabases[C]//Knowledge and information systems. [S. l.]: [s. n.], 2012: 491-522.

[4] Giatsidis C, Malliaros F D, Vazirgiannis M. Advanced graph mining for community evaluation in social networks and the web[C]// Proceedings of the sixth ACM international conference on Web search and data mining. [S. l.]: ACM, 2013: 771-772.

[5] Bonnici V, Giugno R, Pulvirenti A, et al. A subgraph isomorphism algorithm and its application to biochemical data [J]. BMC Bioinformatics, 2013: 13-20.

[6] Zhu F, Qu Q, Lo D, et al. Mining top k large structural patterns in a massive network[J]. Proceedings of the VLDB Endowment, 2011: 45-53.

[7] Borgelt C, Meinl T, Berthold M. Moss: a program for molecular substructure mining[C]// Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. [S. l.]: ACM, 2005: 6-15.

[8] Holder L B, Cook D J, Djoko S. Substucture discovery in the subdue system[J]. KDD Workshop, 1994: 169-180.

[9] Ghazizadeh S, Chawathe S S. Seus: structure extraction using summaries[J]. Discovery Science, 2002: 71-85.

[10] Kang U, Tsourakakis C E, Faloutsos C. Pegasus: a peta-scale graph mining system implementation and observations[C]// Data mining, 2009. ICDM'09. ninth IEEE international conference on IEEE. [S. l.]: [s. n.], 2009: 229-238.

[11] Kang U, Chau D H, Faloutsos C. Pegasus: mining billion-scale graphs in the cloud[C]// Acoustics, speech and signal processing(ICASSP), 2012 IEEE international conference on IEEE. [S. l.]: [s. n.], 2012: 5341-5344.

[12] Akoglu L, Chau D H, Kang U, et al. Opavion: Mining and visualization in large graphs[C]// Proceedings of the 2012 ACM SIGMOD international conference on management of data. [S. l.]: ACM, 2012: 717-720.

[13] Khayyat Z, Awara K, Alonazi A, et al. Mizan: a system for dynamic load balancing in large-scale graph processing [C]// Proceedings of the 8th ACM European conference on computer systems. [S. l.]: ACM, 2013: 169-182.

[14] Kang U, Tong H, Sun J, et al. Gbase: an efficient analysis platform for large graphs[J]. The VLDB Journal, 2012: 637-650.

[15] Borgelt C. Canonical forms for frequent graph mining[J]. Advances in Data Analysis, 2007: 337-349.

Research on Large Data Graph Model of Efficient Mining Based on Cloud Computing

LIANG Jie

(Foundation Sector, Institute of Changchun Vocational Technology, Changchun 130000, China)

Abstract: For large data mining is an important task in the field of graph mining. Although cloud computing is very effective in solving the problems of the traditional calculation, but the frequent mining on large graph still faces three challenges: the graph partitioning problem; the problem of information asymmetry problem with retained mode. In view of the above problems, this paper presents a new SpinderMine mining method based on cloud namely c-SpinderMine method. The method uses cloud computing to the patterns on large graph data mining. The method can solve the problem facing the mining algorithm in the cloud environment large graphic data. In three groups of real data sets after the experiment, this algorithm can effectively shorten the execution time, and the data processing in the cloud shows high scalability.

Key words: big data; cloud computing; graph pattern mining; frequent pattern mining

(责任编辑 游中胜)