用户频繁通信关系的并行挖掘算法研究

朱鹏宇 鲍培明 吉根林

(南京师范大学计算机科学与技术学院 南京 210023)

摘 要 随着移动通信技术和互联网的飞速发展,移动通信设备已经成为大多数人随身携带的工具,这些设备之间因互相通信而产生的数据构成了通信网络。文中提出了一种针对海量通信数据的频繁通信子图并行挖掘算法PMFCS。该算法在频繁项目集挖掘思想和子图连接规则的基础上,利用并行计算框架Spark将所有的图以边为单位分布到各个计算节点,在各个节点统计1阶候选频繁子图,再通过汇总候选子图得到1阶频繁子图。PMFCS算法通过迭代地连接$k-1$阶子图和1阶子图生成k阶候选子图,再计算k阶候选子图的频繁度,直至k阶频繁子图集合为空集。实验结果表明,该算法可以快速、有效地解决频繁通信关系的挖掘问题。

关键词 通信网络,频繁子图,频繁通信关系

中图法分类号 TP391.4 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.02.018

Parallel Algorithm for Mining User Frequent Communication Relationship

ZHU Peng-yu BAO Pei-ming JI Gen-lin

(School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China)

Abstract With the rapid development of mobile communication technology and Internet, mobile communication equipment has become a portable tool for most people. A parallel algorithm PMFCS was proposed for mining frequent communication sub-graph of mass communication data. The algorithm is based on the Apriori algorithm and sub-graph connect principle. It uses Spark to distribute all the edges to each computing node, then the 1^{th}-order frequent candidate sub-graphs are distributed to each node, the 1^{th}-order frequent candidate sub-graphs are counted at each node, and the 1^{th}-order sub-graphs are got by summarizing candidate sub-graphs. PMFCS iteratively connects the $(k-1)^{th}$-order sub-graph and the 1^{th}-order sub-graph to generate k^{th}-order candidate sub-graphs. Subsequently, the algorithm terminates until the k^{th}-order frequent sub-graph set is empty. The experimental results show that PMFCS can mine the frequent communication sub-graph efficiently and quickly.

Keywords Communication network, Frequent sub-graph, Frequent communication relationship

近年来,随着计算机技术和智能手机技术的飞速发展,人们对移动通信设备的持有量逐年增长。根据工信部发布的数据,截至2015年12月底,我国的手机用户数量已达13.06亿,如此庞大的用户群体每天产生的通信记录是海量的。手机用户之间的通信(包括通话、短信、彩信和语音等)构成了一张巨大的网络。爆炸式发展的通信网络数据记载了用户之间的通信关系。海量的通信网络分析和挖掘成为了一个亟待解决的问题。

通信网络中频繁通信关系的挖掘是找出较长的时间段内经常发生通信行为的用户关系的过程。例如,假设用户A和用户B在3个月内有过100次通信,这100次通信集中在较短的时段内;而用户C和用户D在3个月内同样有100次通信,但是这100次通信在3个月内呈均匀分布。显然,用户C和用户D之间的关系更符合频繁的概念。在现实生活中,用户C和用户D这样的频繁通信关系往往也代表了稳定的人际关系,比如亲人关系、朋友关系等。用户A和用户B之间的关系有可能不是稳定的人际关系,而可能是客户类关系,他们仅仅在某个时间段内有频繁的交互。相对于用户A和用户B的关系,用户C和用户D的关系强度明显更强。频繁通信关系挖掘的目的就是找出如用户C和用户D之间稳定的、较强的用户关系。

本文研究了并行环境下通信网络中的频繁通信关系挖掘方法,提出了一种基于Spark[1]的频繁通信子图并行挖掘算法(Parallel Mining of Frequent Communication Sub-graph, PMFCS)。本文的主要贡献如下:

1)单机算法无法处理通信网络中海量的图数据,基于MapReduce[2]的并行算法面对迭代计算时存在劣势,本文在擅长迭代计算的Spark框架下设计挖掘算法。

2)在频繁项目集挖掘思想和子图连接规则的基础上,提出了并行环境下的频繁通信子图挖掘算法PMFCS。

到稿日期:2017-04-26 返修日期:2017-06-29 本文受国家自然科学基金项目:云计算环境下顾及用户关系的手机用户时空轨迹模式挖掘方法研究(41471371)资助。

朱鹏宇(1993-),男,硕士生,主要研究方向为用户关系挖掘,E-mail:pyzhu2016@163.com;**鲍培明**(1966-),女,硕士生导师,主要研究方向为数据挖掘及其应用,E-mail:baopeiming@163.com(通信作者);**吉根林**(1964-),男,博士生导师,主要研究方向为数据挖掘及其应用。

3)在真实数据集上进行实验,并分析运行结果以及算法的性能。

1 相关工作

Natu 等人[3]于 2011 年通过挖掘频繁子图来提取通信模式,并给出了两种挖掘频繁子图的方法,这两种方法分别通过 Apriori[4]和矩阵实现。Nirmala 等人[5]于 2016 年通过在一系列通信网络上挖掘 MCIS(Maximal Common Induced Subgraph),来提取这些通信网络的共性特征。同年,Nirmala 等人[6]又提出了在宽松条件下带时间因素的频繁通信网络查询问题。

通信网络可以用图加以描述和分析。频繁通信关系挖掘问题可以通过频繁子图挖掘方法解决。频繁子图挖掘问题已经被学习研究数十年,但是其中绝大多数算法难以应对现在如此海量的数据。

频繁子图的挖掘研究有两个分支:图集上的频繁子图挖掘和一张大图上的频繁子图挖掘。图集上的挖掘是指,在包含了多张图的图数据库中挖掘在这些图中共现的子图。在一张大图上的挖掘,则是在一张图上对图内复现的子图进行挖掘。本文的研究内容是在一系列时间序列上的通信网络中挖掘频繁复现的子图,属于图集上的频繁子图挖掘。

较为经典的图集上的频繁子图挖掘算法有 AGM[7],FSM[8],GSPAN[9]和 FFSM[10]等。汪卫等人[11]于 2005 年提出了 AMGM 算法,该算法结合了频繁项目集生成思想和图的相关知识,可以有效地挖掘简单图中的连通频繁子图。李先通等人[12]于 2007 年在频繁子树挖掘的基础上提出了一种高效的频繁子图挖掘算法 GraphGen。Bhuiyan 等人[13]和 Lu 等人[14]于 2013 年分别提出了基于 MapReduce 的分布式挖掘算法 FSM-H 和 MRFSM。FSM-H 将整个算法过程分为数据划分、准备、挖掘三大阶段,迭代式地运行挖掘阶段,能够进行百万级别的图集挖掘。MRFSM 则更加注重在迭代时对负载均衡的考虑,但是 MapReduce 并不适合于迭代的图挖掘算法,因为它会造成许多额外的磁盘读写和序列化开销。

2 问题定义

定义 1(通信图) 通信图 G 是一个三元组 $G=(V,E,\beta)$,其中 V 是用户的集合;E 是通信关系的集合,对于 $\forall e_i \in E_i$,$e_i=(v_j,v_k)$,$v_j \in V$,$v_k \in V$;β 是通信关系的属性,是一个二维向量,表示两个用户之间的通信频次和通信总时长。

定义 2(通信图集合) 假设存在若干可序列化用户的集合 $V=\{v_1,v_2,\cdots,v_m\}$,这些用户在某个时间段 T_i 内的通信行为构成了通信图 G_i,其中 $G_i=(V_i,E_i,\beta_i)$,$V_i \subseteq V$,那么一系列不重叠的时间段序列 T_i 及该时间段上的通信图 G_i 构成了一个通信图集合,记为 $GS=\{(G_1,T_1),(G_2,T_2),\cdots,(G_n,T_n)\}$。通信图集合的规模用 $|GS|$ 表示。

以图 1 为例,圆圈代表用户,边代表通信关系,其中 $GS=\{(G_1,T_1),(G_2,T_2),(G_3,T_3)\}$。$G_1$ 中顶点 1 和顶点 2 之间存在一条无向边,表示这两个用户在时间段 T_1 内存在通信行为。图中仅标出两个边属性作为示例,这两个属性表示用户 1 和用户 2 之间在 T_1 内的通信频次和通信总时长分别为 49 次和 9272 s,在 T_2 内为 30 次和 6042 s。

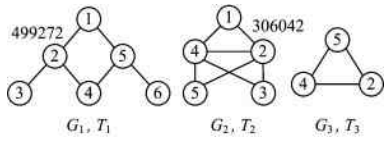

图 1 通信图集合

Fig. 1 Communication graph set

定义 3(通信子图) 对于通信图 $G=(V',E',\beta')$ 和 $g=(V,E,\beta)$,$|V| \geqslant 2$,$|E| \geqslant 1$,如果 $\forall v \in V$,则 $\exists v' \in V'$,$v=v'$,并且若 $\forall e \in E$,则 $\exists e' \in E'$,$e=e'$,那么 g 就是 G 的通信子图,记作 $g \subseteq G$。$k=|E|$ 表示子图 g 的规模,称 g 为 G 的 k 阶子图,又称 G 为 g 的超图。

如图 2 所示,g 就是 G 的 3 阶通信子图。在研究通信关系中的亲近关系、社团发现等问题时,总是寻找一组相互有联系的群体,因此限定子图必须是一个连通图,研究非连通通信图中用户之间的关系是没有意义的。

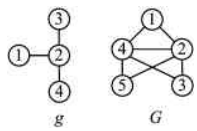

图 2 通信子图

Fig. 2 Communication sub-graph

定义 4(频繁通信子图) 对于通信图集合 $GS=\{(G_1,T_1),(G_2,T_2),\cdots,(G_n,T_n)\}$、通信图 $g=(V,E,\beta)$ 和给定的阈值 $minSup \in [0,1]$,$f(g)=\{G_i \mid g \subseteq G_i, G_i \in GS, \forall i \in [1,n]\}$ 是通信图集合 GS 中所有 g 的超图构成的集合,g 的支持度为 $support=|f(g)|/|GS|$。如果 $support \geqslant minSup$,那么 g 就是 GS 的频繁通信子图。

例如,在图 3 中,假设 $minSup=2/3$,那么 g 就是 CS 的频繁通信子图。频繁通信子图的意义在于该子图在整个图序列中是频繁出现的,即子图代表的通信关系在整个时间序列上是频繁发生的,表示群体之间在不同时段中都有通话,这群人可能是同事、亲人或朋友等关系,他们之间存在较强的亲密关系。例如图 3 中的 2,4,5 号用户在 T_2 和 T_3 两个时间段内都有通信行为,对于整个时间序列,2,4,5 号用户之间存在着稳定的亲近关系。找出频繁通信子图对研究通信图中的用户关系有重要的意义,它是用户亲近关系强度、团伙发现、群体关系演化等相关研究的基础。

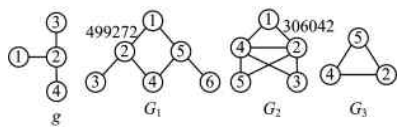

图 3 频繁通信子图

Fig. 3 Frequent communication sub-graph

3 PMFCS 算法

3.1 算法思想

频繁通信子图并行挖掘算法 PMFCS 是基于 Apriori 频繁项目集挖掘思想和迭代式 RDD 而设计的。Apriori 算法的如下两个性质同样适用于频繁通信子图挖掘算法。

性质 1 频繁子图的所有子图一定是频繁的。

性质 2 非频繁子图的超图一定是非频繁的。

频繁通信子图并行挖掘算法的框架如图 4 所示。首先以

边为单位将所有的图分布到各个计算节点，在各个节点统计 1 阶候选频繁子图，再通过汇总候选子图得到 1 阶频繁子图。随后，通过迭代地连接 $k-1$ 阶子图和 1 阶子图生成 k 阶候选子图，再计算 k 阶候选子图的频繁度，直至 k 阶频繁子图集合为空集。

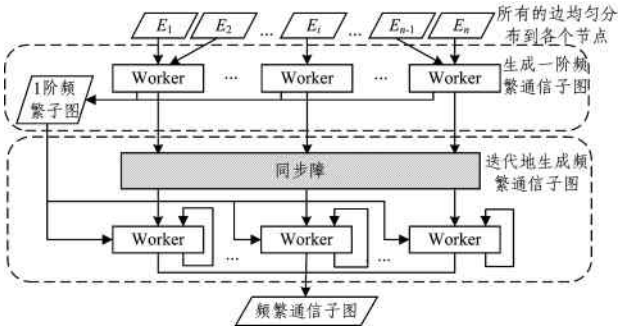

图 4　PMFCS 算法的框架

Fig. 4　Framework of PMFCS algorithm

频繁子图挖掘算法的核心是子图的连接过程。子图连接就是尝试将一个频繁的 1 阶子图和一个频繁的 $k-1$ 阶子图拼接在一起的过程。如果能够拼接成一个 k 阶子图 G_k（称为 k 阶候选子图），就计算 G_k 的支持度，如果支持度大于给定的阈值，那么 G_k 就是频繁的。

3.2　子图连接规则

在子图生成过程中容易产生冗余的子图，这些冗余子图会导致支持度的计算出现错误。若存在冗余子图，则重复计算同一个子图的支持度。以图 5 为例，g_1，g_2，g_3 和 g_4 都是 g 的子图，假设 g 是频繁的，那么 g_1，g_2，g_3 和 g_4 也一定是频繁的。在 2 阶子图生成 3 阶子图的过程中，通过 g_1 和 g_2 连接、g_3 和 g_4 连接生成了同样的候选子图 g，这就导致了支持度的重复计算。

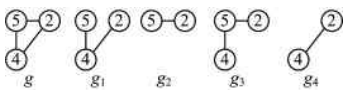

图 5　冗余子图示例

Fig. 5　Example of redundant sub-graph

为了检测冗余子图，需要进行图的同构匹配。众所周知，子图同构问题是一个极其复杂的 NP 问题。为了避免该问题，找到一个高效的子图连接规则很有必要。子图连接规则的重点在于，保证子图生成的完备性，同时避免生成冗余子图。

对于 1 阶频繁子图 $G_1=(V_1,E_1,\beta_1)$，$V_1=\{v_1,v_2\}$，$E_1=\{(v_1,v_2)\}$，$k-1$ 阶频繁子图 $G_{k-1}=(V_{k-1},E_{k-1},\beta_{k-1})$，$V_{k-1}=\{u_1,u_2,\cdots,u_p\}$，$E_{k-1}=\{(u_i,u_j)\cdots\}$，$|E_{k-1}|=k-1$，它们能否连接生成 k 阶候选子图，取决于如下 3 条规则。

规则 1　当 G_1 和 G_{k-1} 中没有任何顶点相同，即 $v_1\notin V_{k-1}$ & $v_2\notin V_{k-1}$ 时，G_1 和 G_{k-1} 不能连接生成新的子图。

以图 6(a)的 G_1 和图 6(b)的 G_{k-1} 为例，两个图没有可连接的顶点，很明显此时不可以连接生成一个连通图。

规则 2　G_1 和 G_{k-1} 有且只有一个顶点相同，不失一般性，假设 V_{k-1} 包含顶点 v_1，但不包含顶点 v_2，即 $v_1\in V_{k-1}$ 且 $v_2\notin V_{k-1}$。若 v_2 大于 G_{k-1} 中顶点 v_1 的所有连接顶点，则 G_1 与 G_{k-1} 连接生成新的 k 阶候选子图 G_k。

在 G_{k-1} 中与 v_1 有通信的顶点构成集合 $c(v_1)=\{v_l|(v_1,v_l)\in E_{k-1},\forall l\in[1,p]\}$，若 $\forall v_h\in c(v_1)$ 都满足 $v_2>v_h$，则 G_1 与 G_{k-1} 连接生成新的 k 阶候选子图 $G_k=(V_k,E_k)$，$V_k=V_{k-1}\cup\{v_2\}$，$E_k=E_{k-1}\cup E_1$，$|E_k|=k$。

该规则的思路是：当 G_1 和 G_{k-1} 有且只有一个相同的顶点 v_1 时，连接后对于顶点 v_1 来说就增加了一条外接边。为了保证生成子图的唯一性，当且仅当 v_2 大于 G_{k-1} 中所有与 v_1 直接相连的顶点时，能添加该外接边。例如，图 6(c)和图 6(d)，以及图 6(e)和图 6(f)，分别都有相同的顶点 1，连接后都会生成图 6(g)所示的候选子图。图 6(c)和图 6(d)连接时，由于 $8>6$ 且 $8>4$，因此顶点 8 可以连接到顶点 1 上，构成图 6(g)所示的候选子图。图 6(e)和图 6(f)连接时，由于 $6<8$，因此此时图 6(e)和图 6(f)不连接。

根据性质 1 和性质 2，若图 6(g)是频繁子图，则一定存在频繁图图 6(c)、图 6(d)、图 6(e)和图 6(f)，图 6(c)和图 6(d)连接时会产生候选子图图 6(g)，因此图 6(e)和图 6(f)不连接时不会丢失图 6(g)。

规则 3　G_1 和 G_{k-1} 有两个顶点相同，即 $v_1\in V_{k-1}$，$v_2\in V_{k-1}$ 且 $(v_1,v_2)\notin E_{k-1}$，不失一般性，假设 $v_1>v_2$。在 G_{k-1} 中找到 v_1 到 v_2 的最短路径 $w=v_1\rightarrow v_a\rightarrow v_b\cdots\rightarrow v_2$，若 $\forall v_x\in w$，$v_1\geqslant v_x$ 且 $v_2>v_a$，则 G_1 和 G_{k-1} 连接生成新的 k 阶候选子图 $G_k=(V_k,E_k)$，$V_k=V_{k-1}$，$E_k=E_{k-1}\cup E_1$，$|E_k|=k$。

需要说明的是，如果 $(v_1,v_2)\in E_{k-1}$，那么 G_1 完全包含在 G_{k-1} 中，G_1 和 G_{k-1} 连接生成的图依旧是 G_{k-1}，因此 G_1 和 G_{k-1} 不需要连接。当 $(v_1,v_2)\notin E_{k-1}$ 时，因为 G_{k-1} 是一个连通图，所以添加边 (v_1,v_2) 后，G_k 中一定存在一条从 v_1 到 v_2 的回路。

该原则的思路是：当 G_1 和 G_{k-1} 中有两个顶点相同时，拼接后形成回路；为了保证生成子图的唯一性，当且仅当在添加回路中的特定边时才会连接，这个特定边就是回路中最大顶点与该顶点直接相连的两个顶点中较大的顶点之间的边。

例如，连接后生成图 6(l)的 G_1 和 G_{k-1} 众多。如，当 G_1 和 G_{k-1} 分别为图 6(h)和图 6(i)时，通过共同顶点 3 和顶点 6 可以连接生成图 6(l)的候选子图。同理，当 G_1 和 G_{k-1} 分别为图 6(j)和图 6(k)时，通过共同顶点 1 和顶点 6 也可以连接生成图 6(l)的候选子图。对于图 6(h)和图 6(i)，在图 6(i)中顶点 6 到顶点 3 的最短路径为 $6\rightarrow1\rightarrow4\rightarrow3$，可以发现最大顶点是 6，且 $3>1$，则在图 6(i)中添加图 6(h)的边(6,3)，从而连接生成候选子图 6(l)。

对于图 6(j)和图 6(k)，在图 6(k)中顶点 6 到顶点 3 的最短路径为 $6\rightarrow3\rightarrow4\rightarrow1$，可以发现最大顶点是 6，但 $1\not>3$，则放弃图 6(j)和图 6(k)的连接，避免生成冗余的子图 6(l)。

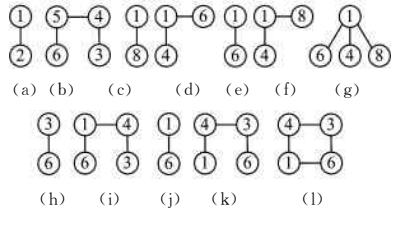

图 6　子图生成原则的示例

Fig. 6　Example of sug-graph generating principle

根据性质 1 和性质 2，若图 6(l)是频繁子图，则一定存在频繁图图 6(h)、图 6(i)、图 6(j)和图 6(k)等，图 6(h)和图 6(i)连接时会产生候选子图图 6(l)，因此图 6(j)和图 6(k)不连接时不会丢失图 6(l)。

3.3　算法步骤

PMFCS 算法主要分成两步：1）生成 1 阶频繁子图算法

FEC(Frequent Edge Calculator);2)迭代连接生成高阶频繁子图算法 FSC(Frequent Sub-graph Connector)。首先,将通信图集合 GS 以边为单位分发到各个节点上;然后,在各个节点上并行计算每条边的支持度,根据支持度 minSup 筛选出频繁边,用频繁边构成 1 阶频繁子图;最后,在 1 阶频繁子图的基础上通过迭代连接逐步生成 k 阶的候选子图,再计算 k 阶候选子图的支持度,从而得到 k 阶频繁子图,直至某阶频繁子图为空集时算法终止。

算法 1 生成 1 阶频繁子图算法 FEC

输入:通信图集合 GS,最小支持度 minSup,最小通信频次 minFrq(每个时间段的通信次数需要达到的最小值)和最短通信时长 minCom(每个时间段中平均单次通信时长需要达到的最小值)

输出:1 阶频繁子图 frqEdge

第 1 步 初始化列表 listBef=∅,listAft=∅,frqEdge=∅,其中 listBef 存放边和时间段的键值对,listAft 存放边和时间段序列的键值对,frqEdge 存放 1 阶频繁子图和时间段序列的键值对。

第 2 步 将 GS 中每个通信图按边拆分,以边和时间段键值对的形式存入列表。$\forall(G_i,T_i)\in GS,G_i=(V_i,E_i,\beta_i)$,对于 $e\in E_i$,将 (e,T_i) 存入列表 listBef,listBef=listBef$\cup\{(e,T_i)\}$;若 $e=(v_k,v_l)$,则 (e_{ij},T_i) 表示用户 v_k 和 v_l 在时间段 T_i 内有通信。

第 3 步 对于列表 listBef 中的所有键值对,检验其键值属性是否满足最小通信频次和最短平均通信时长要求。$\forall(e,T_i)\in$ listBef,若 $\beta_i(e)=(\varepsilon_1,\varepsilon_2)$,其中 ε_1 和 ε_2 分别是 e 在 T_i 内的通信频次和通信时长,当 $\varepsilon_1<$minFrq 或 $\varepsilon_2/\varepsilon_1<$minCom 时,将 (e,T_i) 从列表 listBef 中删除。

第 4 步 列表 listBef 中所有键值对按键归约,并以 $(e,\text{List}\{T_a,T_b\cdots\})$ 的形式存入列表 listAft,listAft=listAft$\cup\{(e,\text{List}\{T_a,T_b\cdots\})\}$。若 $e=(v_k,v_l)$,则 $(e,\text{List}\{T_a,T_b\cdots\})$ 归约用户 v_k 和 v_l 之间存在通信的时间段位 $\{T_a,T_b\cdots\}$。列表 listAft 中任意两个键值对中的键不同,即若 $(e_1,\text{List}\{T_a,T_b\cdots\})\in$ listAft,$(e_2,\text{List}\{T_a,T_b\cdots\})\in$ listAft,则 $e_1\neq e_2$。

第 5 步 对列表 listAft 中所有的键值对 $(e,\text{List}\{T_a,T_b\cdots\})$ 计算支持度,若不满足最小支持度,则将其删除。$\forall(e_j,\text{List}_j\{T_a,T_b\cdots\})\in$ listAft,检验 $|\text{List}_j\{T_a,T_b\cdots\}|/|GS|\geqslant$minSup 是否满足,若满足则保留,否则从列表中删除;listAft 中保存的键值对就是频繁边和时间序列。

第 6 步 生成一阶频繁子图的集合 frqEdge。$\forall(e,\text{List}_j\{T_a,T_b\cdots\})\in$ listAft,将 e 代表的频繁边转化为 1 阶频繁图的形式,即将 $e=(v_k,v_l)$ 转换为图 $g_j,g_j=(V_j,E_j),V_j=\{v_k,v_l\},E_j=\{(v_k,v_l)\}$,再把 1 阶频繁子图表示的键值对 $(g_j,\text{List}_j\{T_a,T_b\cdots\})$ 存入列表 frqEdge,frqEdge=frqEdge$\cup\{(g_j,\text{List}_j\{T_a,T_b\cdots\})\}$。

第 7 步 输出 frqEdge。

算法 2 单次迭代算法 FSC

输入:1 阶频繁子图 frqEdge,k−1 阶频繁子图集合 frqLast,最小支持度 minSup 和通信图集合的规模|GS|

输出:k 阶频繁子图集合 frqNext

第 1 步 初始化 frqNext=∅,frqNext 中存放 k 阶频繁子图和时间段序列的键值对。

第 2 步 检验 frqEdge 中每个 1 阶频繁子图和 frqLast 中每个 k−1 阶频繁子图是否可连接。对于 $\forall(g_i,\text{List}_i\{T_a,T_b\cdots\})\in$ frqEdge,$g_i=(V_i,E_i),V_i=\{v_k,v_l\},E_i=\{(v_k,v_l)\}$ 和 $\forall(g_j,\text{List}_j\{T_c,T_d\cdots\})\in$ frqLast,$g_j=(V_j,E_j)$,检验 g_i 与 g_j 能否连接,检验原则如下:

1)V_j 既不包含 v_k 也不包含 v_l,即 $v_k\notin V_j\,\&\,v_l\notin V_j$,则根据规则 1,$g_i$ 与 g_j 不能连接。

2)V_j 仅包含 v_k 或 v_l,假设仅包含 v_k,即 $v_k\in V_j\,\&\,v_l\notin V_j$,则根据规则 2,若 v_l 大于 g_j 中顶点 v_k 的所有连接顶点,则 g_i 与 g_j 可以连接,否则 g_i 与 g_j 不能连接。

3)V_j 既包含 v_k 也包含 v_l,即 $v_k\in V_j\,\&\,v_l\in V_j$,则根据规则 3,假设 $v_k>v_l$,在 g_j 中找到 v_k 到 v_l 的最短路径 $w=v_k\rightarrow v_a\rightarrow v_b\cdots\rightarrow v_l$。若 $\forall v_x\in w,v_k\geqslant v_x\,\&\,v_l>v_a$,则 g_i 与 g_j 可以连接;否则 g_i 与 g_j 不能连接。

第 3 步 对于 g_i 与 g_j 可以连接的键值对 $(g_i,\text{List}_i\{T_a,T_b\cdots\})$ 和 $(g_j,\text{List}_j\{T_c,T_d\cdots\})$,首先根据它们满足的规则生成对应的 k 阶频繁子图 $g_r=(V_r,E_r)$,满足规则 2 时,$V_r=V_j\cup\{v_k\}$,$E_r=E_j\cup E_i$,满足规则 3 时,$V_r=V_j,E_r=E_j\cup E_i$;然后根据 g_r 生成新的键值对 $(g_r,\text{List}_r\{T_e,T_f\cdots\})$,其中 $\text{List}_r\{T_e,T_f\cdots\}=\text{List}_i\{T_a,T_b\cdots\}\cap\text{List}_j\{T_c,T_d\cdots\}$;最后将新的键值对存入 frqNext,frqNext=frqNext$\cup(g_r,\text{List}_r\{T_e,T_f\cdots\})$。

第 4 步 $\forall(g_r,\text{List}_r\{T_e,T_f\cdots\})\in$ frqNext,检验 $|\text{List}_r\{T_e,T_f\cdots\}|/|GS|\geqslant$minSup 是否满足,若满足则保留,反之则将其从 frqNext 中删除。

第 5 步 输出阶频繁子图集合。

4 实验结果和分析

本文的实验环境为 Apache Spark-1.4.0,Hadoop-2.4.0,JDK-1.8,使用 Scala 进行开发。集群的硬件配置如表 1 所列。

表 1 集群配置表

Table 1 Configuration of cluster

硬件	配置信息
Workers	18
Cores	216
Memory per node	16GB
CPU per node	Intel(R) Xeon(R) CPU E5-2620 v2 @2.10GHz

4.1 实验数据

本文实验采用了 RealityMining 数据集[15]。该数据集于 2004 年 8 月−2005 年 6 月在麻省理工学院被收集,其中包含了若干个学生或研究人员的通信信息。

经过预处理的实验数据示例如表 2 所列,其中 5 个属性分别为:“主叫 ID 号”“被叫 ID 号”“通信时刻”“主叫基站号”和“通信时长”。应用于本实验的属性有:“主叫 ID”“被叫 ID”和“通话时间”。

表 2 实验数据示例

Table 2 Example of experimental data

主叫	被叫	通信时刻	主叫基站号	通信时长
44	95	2004-08-05 16:40:39	5119.40312	718
4	106	2004-08-05 21:02:30	5119.40312	4
4	106	2004-08-06 21:39:05	5119.40312	51

4.2 时间段阈值与支持度设置

时间段阈值和支持度阈值的设置对频繁子图的挖掘至关重要。频繁通信子图挖掘的目的在于发现用户之间稳定而频繁的通信关系,时间段阈值和支持度阈值的设置取决于对“频繁而稳定的通信关系”的定义。例如,假设只要每周之内存在通信行为,就认定两个用户之间存在频繁而稳定的通信关系,并且可以存在 10%的误差,也就是说 10 周中可以有 1 周不存在通信行为。那么这里的时间段阈值就是 1 周,支持度阈值就是 90%。时间段阈值和支持度阈值是根据“频繁而稳

定”的定义而设置的，在不同的人群和应用场景下可能存在不同的阈值设置。

本文实验中，时间段阈值和支持度阈值分别设置为 10 天和 80%。首先，在现实生活中，如果两个用户每周都存在通信，往往可以认为这两个用户之间存在稳定的通信关系，例如父母和子女之间的通信关系。考虑到本文实验数据的特殊性，即大多用户在同一栋实验楼，设置时间阈值为 10 天。

4.3　实验结果分析

当设置 $minFrq=3$，$minCom=20$s 时，实验结果如图 7 所示。从图 7 中可以发现一个很有趣的现象，即挖掘出的频繁子图大多是发散式的。去除图 7 中重复子图后的 3 阶及以上的频繁子图如图 8 所示。在图 7 中，2，5，6，29 和 46 号用户都是围绕着 8 号用户形成的小团体，很明显 8 号用户就是这个小团体的核心人物。

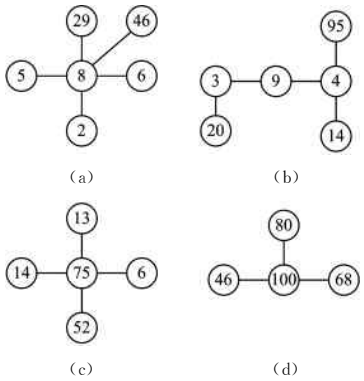

图 7　实验结果 1

Fig. 7　Experimental results 1

为了证明实验结果的合理性，首先实现了 MRFSM 算法，发现其实验结果与图 7 一致；其次，统计了频繁子图的各个用户之间在各个时间段的通信数据，具体数据如表 3 所列。其中，最小通信频次是指所有时间段中通信频次最小时段的通信次数；最短通信时间是指所有时间段中通信时长之和最小的时间段的通信时长；最大通信频次与最大通信时长类似；平均通信频次是指平均每个时间段的通信频次；平均通信时长是指平均每次通信的时长。

表 3　实验数据示例

Table 3　Example of experimental data

通信用户		最小通信频次	最短通信时长	最大通信频次	最大通信时长	平均通信频次	平均通信时长
2	8	3	492	49	9272	19	166.9
5	8	1	28	23	766	6	53.7
6	8	2	351	52	3647	16	104.3
8	29	1	24	56	4318	12	63.9
8	46	1	5	13	3246	4	149
6	75	1	5	11	2730	4	168.5
13	75	4	116	25	6150	15	162.8
14	75	1	5	25	11934	10	318.2
52	75	1	5	23	10955	5	396
4	9	1	12	19	1107	8	48
4	14	1	5	24	981	7	41.6
4	95	3	536	33	17806	15	341
3	9	1	57	23	1784	8	79
3	20	5	122	146	20277	81	87.8
46	100	1	36	22	6282	6	288
80	100	1	5	16	1243	5	75
68	100	1	17	16	1978	7	99.2

可以发现，用户之间存在稳定的通信关系，但是部分用户之间通信的频繁度明显低于其他用户。当设置 $minFrq=5$，$minCom=20$s 时，新的实验结果如图 8 所示。

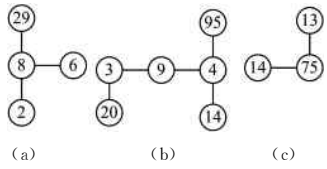

图 8　实验结果 2

Fig. 8　Experimental results 2

在提高 $minFrq$ 后，一些通信频次较低的边都被筛除了，保留下来的边都是一些复现频率较高的通信关系。

4.4　算法的性能测试

使用人工数据对算法做性能测试，并将其与并行算法 MRFSM 作对比，通信图规模对运行时间的影响如图 9 所示。由于采用了 spark 并行编程框架，因此将数据分布在不同节点上进行并行处理，随着通信记录数的增长，运行时间接近线性增加。

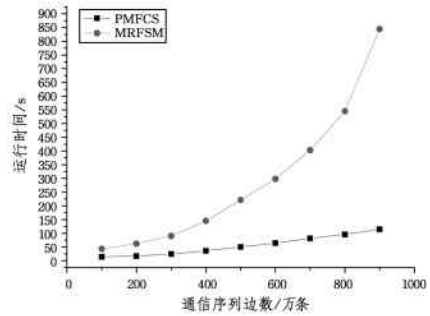

图 9　性能测试结果

Fig. 9　Results of performance test

分析两个算法的运行状况可以发现，PMFCS 算法的速度远远快于 MRFSM。MRFSM 算法速度较慢的原因主要有以下 3 点：1）该算法用于挖掘通用图中的频繁子图，通用图中的顶点标签存在重复，算法较复杂，通信图中顶点的标号是用户的 ID，ID 是唯一的，因此若将 MRFSM 算法用于通信图中的频繁子图挖掘，则须对其进行优化；2）基于 MapReduce 模型而设计，迭代计算时存在大量磁盘读写，效率较低；3）需要通过深度优先遍历对子图进行 DFS 编码，效率较低。

为了分析集群环境下并行算法的加速情况，选取规模为 500 万的通信图数据集，分别使用 2，6，10，14 和 18 个计算节点进行实验，算法的加速比如图 10 所示。可以发现，随着计算节点的增加，算法的加速比逐渐增加但是增加幅度逐渐减小，这是因为通过增加计算资源而带来的性能提升渐渐小于通信开销带来的性能损失。

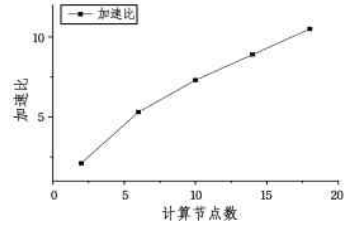

图 10　算法加速比

Fig. 10　Speedup ratio of the proposed algorithm

结束语 本文提出了面向海量通信数据的频繁通信子图并行挖掘算法 PMFCS。该算法利用并行计算框架 Spark，高效地解决了通信网络中用户之间的频繁通信关系挖掘问题。频繁通信关系挖掘结果是研究用户关系强度的基础，也可以为电信运营商服务提供数据支撑，例如套餐推荐、亲情号码等。在微博等社交媒体中互相关注的两个用户也许并没有联系，因此相对于微博等社交媒体挖掘的结果，从通信网络中挖掘出的频繁通信用户关系更真实。

参 考 文 献

[1] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2.

[2] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[3] NATU M, SADAPHAL V, PATIL S, et al. Mining frequent subgraphs to extract communication patterns in data-centres[C]// International Conference on Distributed Computing and Networking. Springer Berlin Heidelberg, 2011: 239-250.

[4] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]// Proceedings of the 20th VLDB Conference. 1994: 487-499.

[5] NIRMALA P, SULOCHANA L R, RETHNASAMY N. Centrality measures-based algorithm to visualize a maximal common induced subgraph in large communication networks[J]. Knowledge and Information Systems, 2016, 46(1): 213-239.

[6] PARISUTHMA N, RETHNASAMY N. An algorithm to search edge relaxed query graph with minimum support threshold in large communication networks[C]// International Conference on Communication Systems and Networks. IEEE, 2016: 1-6.

[7] INOKUCHI A, WASHIO T, MOTODA H. An apriori-based algorithm for mining frequent substructures from graph data[C]// European Conference on Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 2000: 13-23.

[8] KURAMOCHI M, KARYPIS G. Frequent subgraph discovery [C]// Proceedings of IEEE International Conference on Data Mining. IEEE, 2001: 313-320.

[9] YAN X, HAN J. gspan: Graph-based substructure pattern mining[C]// International Conference on Data Mining. IEEE, 2002: 721-724.

[10] HUAN J, WANG W, PRINS J. Efficient mining of frequent subgraphs in the presence of isomorphism[C]// International Conference on Data Mining. IEEE, 2003: 549-552.

[11] WANG W, ZHOU H F, YUAN Q Q, et al. Mining frequent patterns based on graph theory[J]. Journal of Computer Research and Development, 2005, 42(2): 230-235. (in Chinese)
汪卫，周皓峰，袁晴晴，等. 基于图论的频繁模式挖掘[J]. 计算机研究与发展，2005，42(2)：230-235.

[12] LI X T, LI J Z, GAO H. An efficient frequent subgraphmining algorithm[J]. Journal of Software, 2007, 18(10): 2469-2480. (in Chinese)
李先通，李建中，高宏. 一种高效频繁子图挖掘算法[J]. 软件学报，2007，18(10)：2469-2480.

[13] BHUIYAN M A, AL HASAN M. FSM-H: Frequent subgraph mining algorithm in Hadoop[C]// International Congress on Big Data. IEEE, 2014: 9-16.

[14] LU W, CHEN G, TUNG A K H, et al. Efficiently extracting frequent subgraphs using mapreduce[C]// International Conference on Big Data. IEEE, 2013: 639-647.

[15] Reality commons[OL]. http://realitycommons.media.mit.edu/realitymining.html.

(上接第 93 页)

[5] HE X F, YAN S C, HU Y X, et al. Face recognition using Laplacianfaces[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(3): 328-340.

[6] WEN Y, YANG S, HOU L, et al. Face recognition using locality sparsity preserving projections[C]// 2016 International Joint Conference on Neural Networks (IJCNN). IEEE, 2016: 3600-3607.

[7] YU W W, TENG X L, LIU C Q. Face recognition using discriminant locality preserving projections [J]. Image and Vision Computing, 2006, 24(3): 239-248.

[8] HUANG P, TANG Z M. Discriminant of Local Median Preserving Projection with its Application to Face Recognition [J]. Journal of Computer-Aided Design & Computer Graphics, 2012, 24(11): 1420-1425. (in Chinese)
黄璞，唐振民. 鉴别的局部中值保持投影及其在人脸识别中的应用[J]. 计算机辅助设计与图形学学报，2012，24(11)：1420-1425.

[9] WAN M, LI M, YANG G W, et al. Feature extraction using two-dimensional maximum embedding difference [J]. Information Sciences, 2014, 274(274): 55-69.

[10] LAI Z H, WONG W K, XU Y, et al. Approximate orthogonal sparse embedding for dimensionality reduction[J]. IEEE Transactions on Neural Networks and Learning Systems, 2016, 27(4): 723-735.

[11] NING X, LI W J, LI H G, et al. Uncorrelated Local Preserving Discriminant Analysis Based on Bionics[J]. Journal of Computer Research and Development, 2016, 53(11): 2623-2629. (in Chinese)
宁欣，李卫军，李浩光，等. 基于仿生学的不相关局部保持鉴别分析[J]. 计算机研究与发展，2016，53(11)：2623-2629.

[12] MA X H, TAN Y Q. Face recognition based on Discriminant sparse preserving embedding[J]. Acta Automatica Sinica, 2014, 40(1): 73-82. (in Chinese)
马小虎，谭延琪. 基于鉴别稀疏保持嵌入的人脸识别算法[J]. 自动化学报，2014，40(1)：73-82.

[13] ZHAO Z H, HAO X H. Linear Locality Preserving and Discriminating Projection for Face Recognition [J]. Journal of Electronics & Information Technology, 2013, 35(2): 463-467. (in Chinese)
赵振华，郝晓弘. 局部保持鉴别投影及其在人脸识别中的应用[J]. 电子与信息学报，2013，35(2)：463-467.