第 28 卷第 2 期　　四川理工学院学报(自然科学版)　　Vol. 28　No. 2
2015 年 4 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Apr. 2015
文章编号:1673-1549(2015)02-0045-07　　DOI:10.11863/j.suse.2015.02.11

基于并行计算的高效图稀疏化处理算法

李　融

(温州广播电视大学, 浙江　温州　325000)

摘　要:针对目前的图聚类分析方法存在的不足,在分析研究 MapReduce 架构理论、最小哈希算法以及图聚类分析中的数据抽样和稀疏化处理机制的基础上,提出了一种基于并行计算的高效的图稀疏化处理算法。该方法以 MapReduce 架构理论为基础,通过 Minhash 算法进行并行化分析,利用 MapReduce 框架结构对图聚类分析稀疏化操作过程中的多个任务进行了高效的推算分析与处理,并在 Hadoop 计算环境下,通过模拟实验对提出的高效图稀疏化处理算法的性能进行了测试。测试结果表明:基于并行计算的高效图稀疏化处理算法可行,能对图聚类数据信息进行快速稀疏化处理。

关键词:MapReduce;Minhash;图聚类分析;数据抽样;并行计算

中图分类号:TB115　　**文献标志码**:A

引　言

日益复杂的网络交互体系可以通过建模以图模型的形式表示,如社交网络、通信网络、交通运输网络等[1]。在图模型中,每个结点表示对象实体,每条边表示对象实体之间的关联。例如,可以把社交网络体系以一个无向图模型的形式表示,图模型中每个结点表示一个社交群体或一个个体,每条边表示两个社交群体之间或两个个体之间的关联,这种关联可以是同事关系或朋友关系等[2]。近几年,随着网络科学与信息化技术的不断发展以及新浪微博、微信等虚拟网络应用产品的不断推广,图数据信息的处理量呈逐年上升趋势,给图数据的挖掘、分析及应用带来了极大的挑战[3-5]。

图聚类是图数据挖掘、分析及应用过程中可能会用到的一个关键技术。图聚类通过将图模型中的每个结点按照聚簇进行分类,可以提高同类别聚簇图结点对象实体的关联紧密性、降低不同类聚簇图结点对象实体的关联紧密性。随着超大规模图数据信息与处理机制的出现,如何高效地进行图聚类分析与处理,以此来挖掘图数据中的潜在有效数据信息,已成为人工智能、数据挖掘等领域的热点研究方向之一[6]。

国内外研究人员对图聚类算法进行了广泛的研究,提出了很多的图聚类算法,包括经典聚类算法(如划分式聚类算法)、层次式图聚类算法、基于密度的图聚类算法、最小生成图树聚类算法等。

数据抽样[7]是图聚类分析与处理机制中的一种高效数据处理方式。数据抽样首先从整体数据集合中抽取局部样本,然后对样本数据进行数据挖掘、处理与分析。数据抽样可以实现时间与挖掘处理结果的高性能比以及提高图聚类分析与处理的有效性。在图聚类分析与处理过程中,首先对图模型中的结点和边分别进行数据抽样(图稀疏化处理),然后对图稀疏化处理的结果进行图聚类分析。

作为图聚类分析与处理机制中较为重要的一个环节,图稀疏化处理机制[8]已被应用于多个研究方向。针对小规模、小区域范围的图模型数据信息,现有的图稀

收稿日期: 2015-02-14

作者简介:李　融(1977-),男,浙江温州人,讲师,硕士,主要从事计算机应用与教育技术方面的研究,(E-mail)1697986428@qq.com

疏化处理机制主要包含 L-spar[9]、k-最近邻图[10]等几种方法。这些方法在对小规模、小区域范围的图模型数据信息进行处理时,能够得到很好的处理效果,但是在对较大规模、较大区域范围的图模型数据信息进行处理和应用于分布式集群计算环境时,处理效果比较差。

随着图模型应用产品的不断发展和应用规模的不断扩大,图模型的数据信息变得越来越复杂,依靠单一的计算环境对图数据进行处理已不能满足数据分析与处理的需要。针对这种情况,能够通过与大规模计算机服务终端相关联来对大规模数据进行分析与处理的MapReduce并行计算理论框架得到了广泛应用。

哈希算法,是根据设定的哈希函数 $H(key)$ 和处理冲突方法将一组关键字映像到一个有限的地址区间上,并以关键字在地址区间中的像作为记录在表中的存储位置,这种表称为哈希表或散列,所得存储位置称为哈希地址或散列地址。最小哈希算法是哈希算法的一种,传统的最小哈希算法[11](Minhash)主要应用于快速推算多个数据集合之间的相似程度,现已被应用于多个热点研究领域,如文本操作、视频数据处理等[12]。Minhash 算法主要依据 Jaccard 相似度进行相似推算。

本文以 MapReduce 架构理论为基础,通过 Minhash 算法进行并行化分析,设计出一种基于并行计算的高效图稀疏化处理算法,即 MR-LSH 算法。MR-LSH 算法使用并行计算 MapReduce 框架结构[13]对图聚类分析稀疏化操作过程中的多个任务进行了高效的推算分析与处理,这些任务包括邻居结点数据集合推算、Minhash 算法签名推演(对于每个结点而言)、每个结点之间的签名哈希存储以及图聚类过程中的稀疏化处理计算。并在Hadoop计算环境下,对 MR-LSH 算法的性能进行了模拟实验与分析,实验结果表明,MR-LSH 算法的应用能够保证图聚类稀疏化分析与处理机制的高效性。

1 相关研究

1.1 Minhash 算法

Minhash 算法主要是依据 Jaccard 相似度进行的相似推算。Jaccard 是一种相似参数值,用于多个数据集合之间相似度的检测机制。如 Jaccard 参数值对 A、B 数据集合进行相关操作:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

其中,Jaccard 参数值是 A、B 两个数据集合的对比数值。从式(1)中可知,两个数据集合相似度越高,其 Jaccard 参数值就越大。然而在数据集合较大时,Jaccard 参数值会受到交并集合的规模大小影响,其效率无法得到提升。

Minhash 算法就是依据 Jaccard 参数值的相关知识,先利用 Hash 函数(以 H 表示)计算 A、B 数据集合的总体元素数量,然后获取结果信息,即 Minhash(A)与 Minhash(B):

$$P_r[minhash(A) = minhash(B)] = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

所以,在此算法中,其相似度问题则巧妙地变换成多个数据集合的等值概率数学问题,从而改进了数据集合相似度计算效率。

1.2 并行计算理论

分布式框架理论体系首先由谷歌提出,主要用于超大规模、超大区域范围的数据集合分析与处理机制。MapReduce 是并行计算的关键架构之一,可以使客户在并行编程时,只用关注其应用体系内的分析与处理机制,而无需对复杂、冗余的分布式事务进行管理、操作。这也是 MapReduce 并行计算理论的一大突出优势。

MapReduce 并行计算的工作流程如图 1 所示。

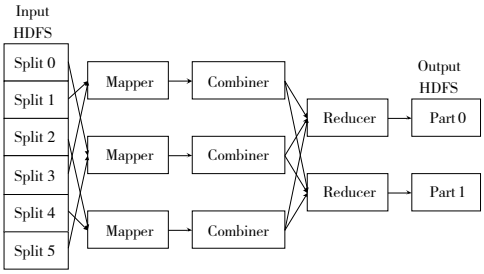

图 1 MapReduce 并行计算工作流程

由图 1 可知,任何一个 MapReduce 分布式任务通常均需通过相关分析与处理过程,其过程由以下三个环节组成:

(1) Mapping 环节:通过该环节,任何 Map 函数操作多个 Split 数据集合,且输出相关参数值,即多个 < key, value > 键值对数据信息。

(2) Combine 环节:对上述多个 < key, value > 键值对数据信息进行排列、分类组合处理。

(3) Reducing 环节:此环节将对通过相关处理的多个 < key, value > 键值对数据信息进行遍历操作,将其唯一性键值操作为相关的 Reduce 函数,获取相关输出

结果。

Hadoop 作为一种并行计算工具,已受到业界的广泛应用。本文将使用 Hadoop 实现 MR-LSH 算法的模拟实验处理过程。模拟实验操作于 Hadoop 平台下的 MapReduce应用程序,主要由一个 Mapping 类、Reducer 类、新建的 JobConf 驱动方法以及关联 Combiner 类(继承于 Reducer 类)构成。

2　问题描述

在现有的文献描述中,L-Spar 算法是依据传统的最小哈希函数为理论原理而设计的一种局部性图稀疏化处理算法。其基本原理是:对图模型的边 $v(i, j)$ 而言,按照 i 与 j 结点之间的 Jaccard 参数值来确定其存储或删除方式。依据式(1)可计算 i 与 j 结点的 Jaccard 参数值,即:

$$Sim(i, j) = \frac{|Adj(i) \cap Adj(j)|}{|Adj(i) \cup Adj(j)|} \tag{3}$$

其中,$Adj(i)$ 为与 i 结点的邻居数据集合,同理,$Adj(j)$ 为与 j 结点的邻居数据集合。

对 $Sim(i, j)$ 输出数值的高效计算中,引入了最小哈希函数(Minhash 算法)在数据集合相似程度计算方面的长处。

L-Spar 算法主要是针对单一任务、小规模以及小区域范围等条件下的图聚类稀疏化分析与处理机制而提出来的。在面对超大规模、超大区域范围的分布式集群计算环境时,其算法优势无法发挥。因此,针对超大规模、超大区域范围的分布式集群计算环境,文章以并行计算 MapReduce 架构理论体系为基础,对 L-Spar 算法(其原理类似于 Minhash 算法)进行并行化分析与改进,将其应用于图聚类过程的稀疏化分析与处理机制中,提出了一种基于并行计算的高效图稀疏化处理算法(MR-LSH)。

3　MR-LSH 算法

3.1　邻居结点数据集合推算

MR-LSH 算法第一步骤是对一组 Map 任务推算出图模型中任意边结点的邻居结点数据集合,详细描述流程如图 2 所示。

由图 2 可知,Map 任务获取一组键值对数据信息,图中的结点信息是 v_i 与 v_j。通过推算可知,输出键值对数据信息是 $<\text{key} = v_i, \text{value} = \text{list}[N_i]>$,其中 v_i 的邻居结点数据集合是 $\text{list}[N_i]$,最终输出参数值应用于 HDFS 平台中。其 Map 任务形式化阐述是:

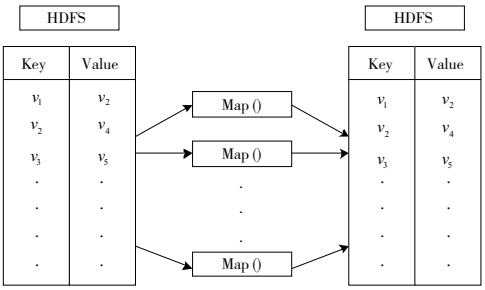

图 2　邻居结点数据集合推算流程

Map:

$<\text{key1} = v_i, \text{value1} = \text{list}[v_i]>$

$\rightarrow <\text{key2} = v_i, \text{value2} = \text{list}[N_i]>$

3.2　Minhash 算法签名推演

MR-LSH 算法第二步骤是推算图模型中任意结点的 Minhash 算法签名数据信息。依据此种情况,MR-LSH 算法可基于 Map 与 Reduce 任务的结合对 Minhash 算法签名数据信息进行推算,其详细描述流程如图 3 所示。

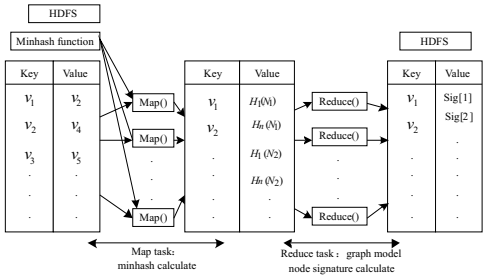

图 3　结点 Minhash 算法签名推算流程

此部分的 Map 任务的输入参数值是第一步骤的输出结果。由图 3 可知,Map 任务以多个(k 个)Minhash 函数作为输入参数值,通过 Hash 推算,其键值对数据信息 $<\text{key} = v_i, \text{value} = H_m(N_i)>$ $(m = 1, 2, \cdots, k)$,其中 $H_m(N_i)$ 是最小哈希函数的列表信息。此部分输出结果则是 Reduce 任务的输入参数值,通过 Reduce 推算获取键值对数据信息 $<\text{key} = v_i, \text{value} = \text{Sig}[i][m]>$,其中,$\text{Sig}[i][m]$ 是一个二元形式化数组,说明 v_i 的算法签名序列。最终将 $\text{Sig}[i][m]$ 应用于 HDFS 平台中。此部分的形式化阐述如下:

Map:

$<\text{key1} = v_i, \text{value1} = \text{list}[N_i]>$

$\rightarrow <\text{key2} = v_i, \text{value2} = \text{list}[H_m(N_i)]>$

Reduce:

< key2 = v_i, value2 = list [$H_m(N_i)$] >

→ < key3 = v_i, value3 = Sig[i][m] >

此部分的具体处理流程可分为多个执行子环节。

(1) Map 任务处理流程

输入: < v_i, list [N_i] >, 其中 key = v_i 是图中的结点, value = list [N_i] 是结点的邻居结点数据集合; k 个不同 Minhash 函数。

输出: < v_i, list [$H_m(N_i)$] >, 其中 value = list [$H_m(N_i)$] 是结点的 Minhash 值列表。

List [H_m] ← φ / * 初始化结点数据集合的 Minhash 值列表 * /

foreach v_i in Graph do

for m in k

/ * 对结点的邻居结点数据集合进行 k 次 Minhash 计算, 并将 hash 结果存储于 H_m 列表中 * /

end for

end foreach

(2) Reduce 任务具体处理流程

输入: < v_i, list [$H_m(N_i)$] >, 其中 value = list [$H_m(N_i)$] 是结点的 Minhash 值列表。

输出: < v_i, Sig[i][m] >, 其中 value = Sig[i][m] 是图模型中结点的签名矩阵。

Sig[i][m] ← φ / * 初始化结点的签名矩阵 * /

foreach v_i in Graph do

Sig[i] = sortSig (H_m)

/ * 对结点的 hash 值列表排序, 依次存储结点的签名矩阵 * /

end foreach

3.3 结点签名之间的哈希存储

MR-LSH 算法处理流程的第三个步骤是为了判定出图模型中各个结点关联邻接边是否属于此结点的图聚类稀疏化结构。具体操作是利用 Map 与 Reduce 任务的结合对任意结点进行统计推算, 其详细描述流程如图 4 所示。

由图 4 可知, 此部分的 Map 任务环节输入是 MR-LSH算法的第一步骤中获取的键值对数据信息 < key = v_i, value = list [N_i] > 与算法签名二元数组集合 Sig[i][m], 获取相关中间参数值, 其作为 Reduce 任务环节的输入, 同时哈希函数也是输入参数值之一, 最终获取输出是 < key = v_i, value = list [$SortC_{ij}$] >。此部分

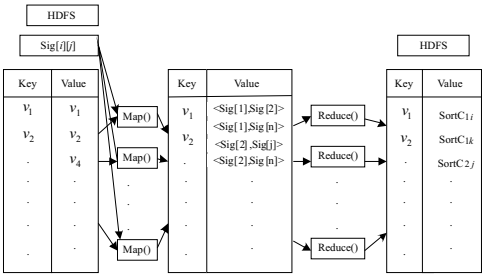

图 4 结点签名的哈希存储处理流程

的形式化阐述如下:

Map:

< key1 = v_i, value1 = list [N_i] >

→ < key2 = v_i, value2 = list[S(Sig[i], Sig[j])] >

Reduce:

< key2 = v_i, value2 = list[S(Sig[i], Sig[j])] >

→ < key3 = v_i, value3 = list [$SortC_{ij}$] >

该部分如同 MR - LSH 算法的第二步骤, 其具体处理流程也可分为多个执行子环节。

(1) Map 阶段具体处理过程

输入: < v_i, list [N_i] >, 其中 key = v_i 是图中的结点, value = list [N_i] 是结点的邻居结点数据集合。

输出: < v_i, list [S (Sig [i], Sig [j])] >, 其中 value = list[S(Sig[i], Sig[j])] 表示结点邻接边的签名数据集合。

list[S] ← φ / * 初始化结点的邻居结点数据集合 * /

foreach v_i in Graph do

for v_j in list [N_i]

/ * 分别找出对应结点和邻居结点数据集合中的结点的签名序列 * /

temp1 = FindSignature (v_i, Sig)

temp2 = FindSignature (v_j, Sig)

/ * 函数 FindSignature (x, Sig) 返回在签名矩阵 Sig 中 x 结点的签名矩阵 * /

S(Sig[i], Sig[j]) = Integration(temp1, temp2)

/ * 函数 Integration (x, y) 返回 x 与 y 结合的集合 * /

end for

end foreach

(2) Reduce 阶段具体处理流程

输入: < v_i, list[S(Sig[i], Sig[j])] >, 其中 key = v_i 是图中的结点, value = list[S(Sig[i], Sig[j])] 表示结点邻接边的签名数据集合。

输出: < v_i, list[$SortC_{ij}$] >, 其中 value = list[$SortC_{ij}$] 表示结点邻接边的签名数据集合。

list[$SortC_{ij}$] ← φ / * 初始化排序后的结点与邻居结点签名匹配列表 */

foreach v_i in Graph do

foreach <Sig[i]> in list[S]

/* 分别对结点与邻居结点的签名进行 hash 操作 */

hashtable1 = Minhash(Sig[i])

hashtable2 = Minhash(Sig[j])

$Count_{ij}$ = MatchTable(hashtable1, hashtable2)

/* 函数 MatchTable (xy) 返回 x 与 y 之间相同数量 */

$SortC_{ij}$ = sortCount ($Count_{ij}$)

/* 函数 sortCount((x) 返回降序排序的 x */

end foreach

end foreach

3.4 图聚类过程中的稀疏化处理计算

MR-LSH 算法的最后一个步骤主要是按照与 v_i 全部邻接边相互匹配的 $SortC_{ij}$, 采取 Map 操作存储其前 $d^e[i]$ 条结点邻接边数据信息。其中 $d[i]$ 说明 v_i 度数参数,e 小于 1 说明图聚类过程中稀疏化处理指数,通过 e 控制图数据信息的稀疏化。经过此部分相关操作,可知,e 与图模型的稀疏化呈现反比关系。详细操作流程如图 5 所示。

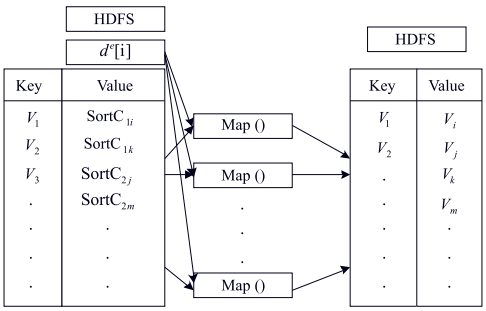

图 5　保留存储结点处理流程

在此部分中,Map 阶段对 MR-LSH 算法的第三步骤输出参数值进行了相关操作,并将 $d^e[i]$ 列入输入之中,组成了保留结点的整体过程,最终应用于 HDFS 平台中。该部分的形式化阐述如下:

Map:

<key1 = v_i, value1 = list[$SortC_{ij}$] >

→ <key2 = v_i, value2 = list[top] >

该部分具体处理流程如下:

输入:上一步输出的键值对数据信息 < v_i, list[$SortC_{ij}$] >, $d^e[i]$ 中 e 是用户定义的图稀疏化指数, $d[i]$ 是边数。

输出: < v_i, list[top] >,其中 value = list[top] 表示需要保留下来的结点的邻居结点。

list[top] ← φ / * 初始化结点需要保留下来的邻居结点 */

foreach v_i in Graph do

Top = ToSave(list[$SortC_{ij}$], $d^e[i]$)

/* 函数 ToSave (x, y) 返回 x 列表中前 y 条边,并且根据边找到其所含有的结点,并记录下来 */

end foreach

MR-LSH 算法的第四步骤进行之后,图模型中的每个结点均存储了 $e>1$ 的边的数量,保证了图模型在一个连通状态。

4 模拟实验

4.1 相关配置

采用开放式并行计算架构,即 MapReduce,应用于 Hadoop 分布式集群计算环境中。其计算环境由多台服务器与终端构成,现以六台为实例,一台主机(master),其余都是附属机(slave),计算环境下的各个结点 CPU 处理器工作频率保持在 3.20 GHz 条件下,采用因特尔双核处理芯片,内存大小保证至少 1 GB。Hadoop 分布式计算环境版本是 1.0.5,OS 是 Linux ubuntul 1.5 版本,使用计算机语言是 JAVA。模拟实验数据信息源是新浪微博社交虚拟网络的关联图模型。

模拟实验中使用 *Speedup* 参数值来表示 MR-LSH 算法性能指标参数变化:

$$S_{speedup} = T_i / T_1 \tag{4}$$

其中, T_i 表示的是第 i 个分布式集群计算环境下结点对图模型稀疏化分析与处理所花费时间值,而 T_1 表示单机情况下对图模型稀疏化分析与处理所花费时间值。

4.2 结果与分析

依据模拟实验中采用的图模型稀疏化处理机制的不同,其图模型稀疏化比率参数值 e 也随之改变,为了应对不同数据信息量与分类的图模型数据信息,其最佳 e 值也会有所不同,本实验初始化 $e=0.15$,然后进行相关操作。

为了显示出 MR-LSH 算法在超大规模、超大区域范围的分布式集群计算环境下的高效性能,模拟实验中采

取多种并行计算环境下的执行算法。MR-LSH 算法首先对 Map 任务阶段与 Reduce 任务阶段进行过程处理,其次对图模型数据信息进行分析,实现图聚类过程下稀疏化分析与处理机制。模拟实验相关数据信息如图 6 所示。

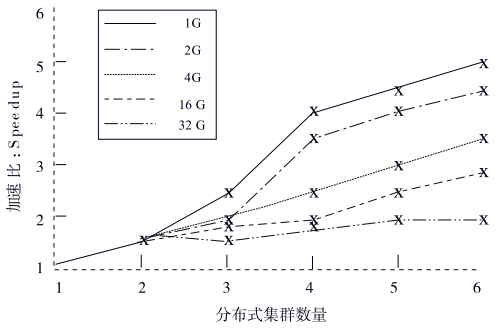

图 6 模拟实验分析结果

从模拟实验分析结果图 6 可知,对于超大规模、超大区域范围的分布式集群计算环境下,使用 Hadoop 并行计算平台能够有效降低时间损失,从而使得 *Speedup* 得到大幅度提高。依据并行计算平台理论架构体系的原理,图模型数据信息规模愈大时,其图聚类过程稀疏化比率参数值就增大,且呈现线性关系;然而随着分布式集群计算环境下各个结点的通信交互频繁,也会消耗一定数据信息性能,在图模型数据信息交互规模较小时,图聚类过程稀疏化分析与处理机制会降低,其 e 参数值同比减小。与此同时,当 *Speedup* 与分布式集群计算环境逐渐增加时,其图聚类过程稀疏化分析与处理机制会有所提高,其 e 参数值同比增加。

通过模拟实验可知,新型的 MR-LSH 算法适用于超大规模、超大区域范围的分布式集群计算环境下的图数据信息,由于在 MR-LSH 算法中添加了排序组合机制,使得结点与邻接结点之间的通信交互消耗得到降低,即图数据信息规模愈大,其 MR-LSH 算法效率性价比愈高。

5 结束语

针对超大规模、超大区域范围的分布式集群计算环境,文章以并行计算 MapReduce 架构理论为基础,对 Minhash 算法进行并行化分析处理与改进,设计出一种基于并行计算的高效稀疏化处理算法,即 MR-LSH 算法。该算法能够高效处理图聚类数据信息。通过模拟实验表明,此算法不仅可行,而且能够对图聚类数据信息进行快速稀疏化处理,具有一定的高效性。

参 考 文 献:

[1] Lin J, Schataz M. Design patterns for efficient graph algorithms in mapreduce [C]//Proceedings of the 8th Workshop on Mining and Learning with Graphs, Washington D.C., July 24-25, 2010:78-85.

[2] Luan T, Lu R X, Shen X M, et al. Social on the road: enabling secure and efficient social networking on highways [J]. IEEE Wireless Communications, 2015, 22(1):44-51.

[3] Yang H C, Dasdan A, Hsiao H L, et al. Map-Reduce-Merge: Simplified relational data processing on large clusters [C]//Proceeding of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, June 11-14, 2007:1029-1040.

[4] Vrba Z, Halvorsen P, Griwodz G, et al. Kahn process networks are a flexible alternative to mapreduce [C]//Proc of the 11th IEEE International Conference on High Performance Computing and Communications, Seoul, June 25-27, 2009:154-162.

[5] Sandholm T, Lai K. MapReduce optimization using regulated dynamic prioritization [J]. Performance Evaluation Review, 2009, 37(1):299-310.

[6] Liu Q, Todman T, Luk W. Combining optimizations in automated low power design [C]//Proc of Design, Automation and Test in Europe, Dresden, Germany, March 8-12, 2010:1791-1796.

[7] Chen Quan, Zhang Daqiang, Guo Mingyi, et al. SAMR: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment [C]//Proc of the 10th IEEE International Conference on Computer and Information Technology, Bradford, UK, June 29-July 1, 2010:2736-2743.

[8] Nicolas G P, Aida D H G. Scaling up data mining algorithms: Review and taxonomy [J]. Process in Artificial Intelligence, 2012, 1(1):71-87.

[9] 陈德华, 周 蒙, 孙 延 青, 等. MR-GSpar: 一 种 基 于 MapReduce的大图稀疏化算法[J]. 计算机科学, 2013, 40(10):190-194.

[10] 滕敏,卫文学,滕宁.K-最近邻分类算法应用研究[J].软件导刊,2015,14(3):44-46.

[11] Chen E.All-disturbances sketches revisited:HIP estimators for massive graphs analysis[J].IEEE Transactions on Knowledge and Data Engineering,2015(99):1.

[12] 温菊屏,钟勇.图聚类的算法及其在社会关系网络中的应用[J].计算机应用与软件,2012,29(2):161-178.

[13] Choi S S,Cha S H,Charles C T.A survey of binary similarity and distance measures[J].Systemics,Cybernetics and Informatics,2010,8(1):43-48.

An Efficient Sparsification Algorithm of Graph Based on Parallel Computing

LI Rong

(Wenzhou Radio & Television University, Wenzhou 325000, China)

Abstract: Aiming at the deficiencies existing in clustering analysis methods of graph, based on the research of MapReduce frame theory, MinHash algorithm and theories of data sampling and sparsification processing mechanism in graph clustering analysis, an efficient sparse processing algorithm of graph based on parallel computing is proposed. Based on the theory of MapReduce frame, this algorithm applies Minhash algorithm to analyze data information of graphs parallelly. Several tasks in the process of sparse processing of graphs clustering analysis are calculated, analyzed and processed efficiently by using MapReduce frame structure. Some simulation studies in Hadoop calculating environment have been conducted to test the performances of the proposed algorithm, and simulation results have demonstrated that the efficient sparse processing algorithm of graph based on parallel computing is feasible, and it can realize fast sparsification processing of clustering data of graphs.

Key words: MapReduce; Minhash; clustering analysis of graph; data sampling; parallel calculation