doi:10.11887/j.cn.201602013 http://journal.nudt.edu.cn

利用局部评估的分布式图模式匹配算法*

张丽霞[1,2],王伟平[1],高建良[1],王建新[1]
(1.中南大学 信息科学与工程学院,湖南 长沙 410083;2.湖南师范大学 数学与计算机学院,湖南 长沙 410081)

摘 要:为了在分布式存储的大规模数据图上进行快速图模式匹配,提出利用局部评估的分布式图模式匹配算法。各计算节点并行地执行本地匹配;协调器节点收集局部匹配结果、计算边界点的匹配状态并发送给相应的计算节点;计算节点根据边界点的匹配状态确定与边界点相连的节点的匹配情况;协调器节点组合得出最大匹配集。实验结果表明:与已有的分布式图模式匹配算法相比,disGPM-PE算法都能够在不显著增加通信量的前提下避免数据片段间的依赖关系对执行时间的影响,从而减少图模式匹配的时间。

关键词:图模式匹配;分布式算法;局部评估
中图分类号:TP311 **文献标志码**:A **文章编号**:1001-2486(2016)02-075-07

A distributed graph pattern matching algorithm using partial evaluation

ZHANG Lixia[1,2], *WANG Weiping*[1], *GAO Jianliang*[1], *WANG Jianxin*[1]
(1. School of Information Science and Engineering, Central South University, Changsha 410083, China;
2. School of Mathematics and Computer Science, Hunan Normal University, Changsha 410081, China)

Abstract: In order to execute graph pattern matching quickly in distributed large-scale graphs, an effective distributed algorithm using partial evaluation, namely disGPM-PE was proposed. Firstly, partial matching was performed locally at each computer nodes in parallel. Secondly, a coordinator node assembled the partial matching results, evaluated and sent the matching conditions of boundary nodes to corresponding computer nodes. Thirdly, each computer nodes determines the matching conditions of the nodes connected to the boundary nodes. Finally, the maximum matching set was collected at the coordinator node. Experiment results show that the disGPM-PE algorithm can avoid the impact of the dependent relations between data fragments on the execution time. Compared with the previous distributed graph pattern matching algorithms, the disGPM-PE algorithm can reduce the execution time of graph pattern matching while do not increase the network traffic obviously.

Key words: graph pattern matching; distributed algorithms; partial evaluation

来自互联网及生活中的海量数据之间存在紧密的关联性,图作为一种被广泛应用的数据结构,非常适合刻画这种具有关联性的数据,图中的每个顶点代表现实世界中的实体对象,顶点之间的边表示实体之间的关系。图模式匹配是从一个数据图中找出与给定查询图(模式图)相同或相似的子图。图模式匹配通常定义为子图同构[1]。由于子图同构是NP完全问题(Non-determinsitic Polynomial Complete problem, NPC)并且在有些实际应用中发现有意义的匹配时过于严格,因此现实生活中图模式匹配通常定义为图模拟[2]。图模拟把图模式匹配定义为模式图中节点和数据图中节点之间的对应关系,要求数据图中节点保持模式图中对应节点的后继关系。图模式匹配应用十分广泛,例如在社交网络中用于社团发现,在万维网中用于Web文档分类,在软件工程领域用于软件代码剽窃检测[3]以及在生物领域用于蛋白质结构检测和功能预测等。

随着社交网络、生物网络和Web网络等的快速发展,互联网数据量急剧增长。已有的图模式匹配算法面临新的挑战:一方面,大规模的数据无法集中存储在一个数据中心上,只能分布存储在多个数据中心上,而且数据中心的地理位置通常较远;另一方面,数据规模的显著增大导致用于表示数据及数据间关系的图的规模显著增大,即使采用分布式算法进行图模式匹配也会面临通信量大、延迟长的问题,如何面向大规模图提高分布式算法的并行效率成为关键。文献[4]提出的分布

* **收稿日期**:2015-03-27
基金项目:国家自然科学基金资助项目(61232001,61173169);湖南省教育厅资助项目(15C0824)
作者简介:张丽霞(1979—),女,河南周口人,讲师,博士研究生,E-mail:smilingilsa6@163.com;
王伟平(通信作者),女,教授,博士,博士生导师,E-mail:wpwang@csu.edu.cn

式图模式匹配算法把存储在不同计算节点上的连通子图发送到一个计算节点上进行图模式匹配，缺点是计算节点间通信量较大。文献[5]提出的分布式算法只在计算节点间传送边界点的匹配状态，但需要每个计算节点建立本地的逻辑依赖图，对于存在多个节点上的连通子图，各节点需要依次等待消息才能进行下一轮的匹配，执行时间与计算节点上数据片段间的依赖关系相关。文献[6]提出了以顶点为中心的分布式图模式匹配方法，当图中节点个数多时，效率不高。另外，消息传递机制使得可以并行的操作序列化，从而降低了并发性。

为了在减少网络通信量的同时避免数据片段间的依赖关系对执行时间的影响，本文借鉴文献[7]针对分布式图上的节点间可达性查询问题的解决方法，通过对其思想进行改进和优化，提出了基于局部评估的分布式图模式匹配算法(distributed Graph Pattern Matching based on Partial Evaluation, disGPM-PE)。算法的基本过程如下：首先各计算节点并行地执行本地匹配，然后协调器节点收集局部匹配结果、计算边界点的匹配状态并发送给相应的计算节点，接着计算节点根据边界点的匹配状态确定与边界点相连的节点的匹配情况，最后协调器节点组合得出最大匹配集。与先前的算法相比，disGPM-PE 算法具有如下特点：①各计算节点并行进行本地匹配，能够避免某些分布式算法将分布在不同计算节点上的连通子图发送到一个计算节点上集中处理，造成较大通信量的问题；②所有计算节点进行局部评估时完全并行，收到协调器节点的求值结果后继续确定与边界点相连的节点的匹配状态时仍然是完全并发执行的；③协调器节点收集来自所有计算节点的匹配结果、边界点相关的布尔表达式，求值后将结果发给各计算节点。虽然在计算节点和协调器节点之间额外传送布尔表达式和求值结果在一定程度上增加了通信量，但是各计算节点的匹配次数固定为两次，匹配次数和并发度不受计算节点上的数据片段之间的依赖关系的限制。

1　图模式匹配相关定义

对于图模式匹配，模式图和数据图都是带标签的有向图，图中的每个节点有且仅有一个标签，该标签定义了节点的属性(如：关键词、技能、等级、姓名、公司等)，相关定义如下：

定义1(图)　$G=(V, E, L)$是一个图，V是节点集，$E\subseteq V\times V$是边集，L是一个标签函数，V中的每个节点都有一个标签att，即$L(v)=att$，att是v的属性。

定义2(图模式匹配)　假设有一个模式图$P=(V_p, E_p, L_p)$和一个数据图$G=(V, E, L)$，u和v分别是P和G中的节点。如果存在一个二元关系$R\subseteq V_p\times V$，满足：

1)如果$(u, v)\in R$，那么u和v有相同的标签即$L_p(u)=L(v)$；

2)对V_p中的任意一个节点u，V中都存在一个节点v，使得：$(u, v)\in R$；对E_p中的任意一条边(u, u')，在E中都存在一条边(v, v')使得$(u', v')\in R$；

则说G匹配P，标识为$P\lhd G$，R是一个匹配。对于任意P和G，如果G匹配P，一定存在一个最大的匹配关系。图模式匹配问题就是：如果$P\lhd G$，那么在G中为P找出一个最大匹配R。

定义3(分布式图)　$G=(V, E, L)$一个分布式图，$F=\{F_1, F_2, \cdots, F_k\}$称作$G$的一个划分，其中每个片段$F_i=\{V_i\cup F_i.O, E_i\cup CE_i, L_i\cup L_i.O\}$$(1\leqslant i\leqslant k)$且以下条件成立：

1)$V=\{V_1\cup V_2\cup\cdots\cup V_k\}$；

2)任一(V_i, E_i, L_i)是G的一个子图；

3)对V_i中的任一节点u，如果E中存在一条边(u, v)且v在另一个片段中，那么v称作是F_i的一个虚节点(virtual node)，$F_i.O$是F_i的虚节点的集合。相应地，如果存在一条从片段F_j中的节点v到片段F_i中的节点u的跨边(v,u)，那么u称作是F_i的一个入节点(in node)，我们用$F_i.I$表示F_i的入节点集合；

4)$CE_i=\{(u,v)\mid(u,v)\in E, u\in V_i, V\in F_i.O\}$是从$V_i$到其他片段中节点跨边的集合。

2　disGPM-PE 算法

2.1　算法主要思想及流程

在 disGPM-PE 算法中，每个计算节点处理的节点被分为以下几类：虚节点、内部节点、与虚节点相连的节点和入节点。虚节点是存储在别的数据片段和计算节点上、被复制一份存在本地的节点，本地仅知道其标签而不知其孩子节点，算法在进行本地处理时只能根据其标签把它加入对应节点的匹配集中而不能根据后继点对其进行筛选确定其最终匹配状态，这种匹配可能不是真的匹配、不会出现在最终的匹配集中，属于局部匹配；内部节点是指和虚节点没有关系的节点，算法仅根据本地数据就可以确定这些节点的匹配状态；与虚节点相连的节点是指有路径到达虚节点的节点，

如果本地不能确定其匹配状态,就把它们加入局部匹配集中;入节点是指存在一条来自于其他数据片段上边的节点,如果其匹配状态与虚节点匹配状态有关,则可用一个布尔表达式描述它们之间的关系。

假设数据图 G 的一个划分 $F=\{F_1, F_2, \cdots, F_k\}$,每一个片段 F_i 存放在一个计算节点 M_i 上,模式图及图模式匹配请求发送给一个协调器节点 M_c。disGPM-PE 算法流程如图 1 所示。

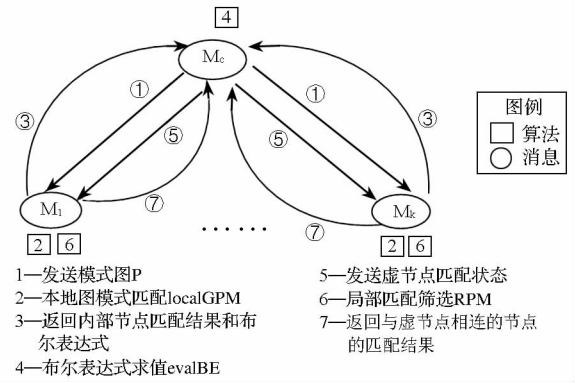


图 1　disGPM-PE 算法流程图

Fig. 1　Algorithm flow chart of disGPM-PE

算法具体操作过程如下:①协调器节点 M_c 把模式图发给每个计算节点;②每个计算节点收到模式图 P 后,并行地调用本地图模式匹配子算法(local Graph Pattern Matching,localGPM),根据本地数据执行图模式匹配,产生三个集合:内部节点匹配结果 $M_i.C$,与虚节点相连的节点的匹配结果 $M_i.U$ 和布尔表达式集合 $M_i.beset$;③每个计算节点把第二步产生的 $M_i.C$ 和 $M_i.beset$ 发送给协调器节点 M_c;④M_c 把从每个计算节点收集的 $M_i.beset$组合成一个布尔表达式系统,调用布尔表达式求值子算法(evaluate Boolean Expressions,evalBE)求得各虚节点的匹配状态;⑤M_c 把虚节点的匹配状态发给各计算节点;⑥各计算节点执行局部匹配筛选子算法(Refining Partial Matching,RPM)对局部匹配集 $M_i.U$ 进行筛选,确定与虚节点相连的节点的匹配状态,匹配结果存入最终匹配集 $M_i.F$ 中;⑦各计算节点把与虚节点相连的节点的匹配结果发到 M_c。最终,协调器节点 M_c 把从各计算节点收到的所有匹配结果组合在一起,就得到了最大匹配集 M。

2.2　本地图模式匹配子算法 localGPM

当每个计算节点 M_i 收到模式图 P 后,并行地调用 localGPM 根据本地数据 F_i 执行图模式匹配,输出三个集合:内部节点的匹配集 $M_i.C$、局部匹配集 $M_i.U$ 和布尔表达式集 $M_i.beset$。其执行过程如下:首先,仅根据标签产生模式图中的各个节点的候选匹配集;然后,对候选匹配集中的内部节点根据后继进行筛选,产生内部节点的匹配集 $M_i.C$;接着,对候选匹配集的虚节点 w,假设它属于模式图中节点 u 的候选匹配集且 u 在模式图中有后继,则将 w 加入局部匹配集 $M_i.U$ 中并在 $M_i.beset$中加入一个布尔变量 $X(u,w)$,表示需要从协调器节点获知 w 是否匹配 u,对于候选匹配集中与 w 相连的节点 w',如果其匹配状态取决于 w 的匹配状态,则将其加入局部匹配集 $M_i.U$;最后,对局部匹配集中 F_i 中的入节点 v,求出一个布尔表达式加入 $M_i.beset$,该表达式表示 v 匹配 u 需要 F_i 中的虚节点满足的条件。

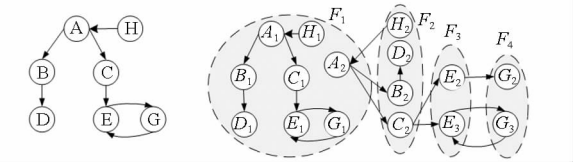
图 2 给出一个分布式数据图和需要查找的模式图的实例,图 2(a)为模式图,图 2(b)为数据图。

设 $F_1 \sim F_4$ 分别存放在计算节点 $M_1 \sim M_4$ 上。在 M_1 上除 A_2 为入节点且与虚节点相连外,$A_1 \sim H_1$ 均为内部节点,在执行 localGPM 时,所有内部节点都是和模式图中的点匹配的,因此 $M_1.C$ 包括所有的内部节点;B_2 和 C_2 是虚节点,将它们加入局部匹配集 $M_i.U$,并把 $X(B,B_2)$ 和 $X(C,C_2)$ 加入 $M_1.beset$,A_2 与虚节点 B_2 及 C_2 相连且其匹配状态不确定的,因此 A_2 也加入 $M_i.U$;由于 A_2 是入节点,因此将布尔表达式 $X(A,A_2)=X(B,B_2)\wedge X(C,C_2)$ 加入 $M_1.beset$。各计算节点执行 localGPM 算法后的结果如表 1 所示。

表 1　各计算节点执行 localGPM 后的结果

Tab. 1　Results of the computer nodes execute localGPM

M_i	$M_i.C$	$M_i.beset$	$M_i.U$
M_1	(A,A_1) (B,B_1) (C,C_1) (D,D_1) (E,E_1) (G,G_1) (H,H_1)	$X(A,A_2)=X(B,B_2)\wedge X(C,C_2)$ $X(B,B_2)$, $X(C,C_2)$	(A,A_2)
M_2	(B,B_2) (D,D_2)	$X(C,C_2)=X(E,E_2)\vee X(E,E_3)$, $X(E,E_2)$, $X(E,E_3)$, $X(A,A_2)$	(C,C_2)
M_3		$X(E,E_2)=X(G,G_2)$, $X(E,E_3)=X(G,G_3)$, $X(G,G_2)$, $X(G,G_3)$	(E,E_2) (E,E_3)
M_4		$X(G,G_3)=X(E,E_3)$, $X(E,E_3)$	(G,G_3)



(a)模式图 (b)数据图
(a) Pattern graph (b) Data graph

图 2 分布式数据图和模式图的一个实例
Fig. 2 An example of distributed data and pattern graph

2.3 布尔表达式求值子算法 evalBE

当各计算节点把本地匹配的结果 $M_i.C$ 和 $M_i.beset$ 发送给协调器节点 M_c 后，M_c 先组合这些结果得到 M 和布尔表达式集合(BEset)，然后执行布尔表达式求值子算法 evalBE 得到所有布尔表达式的值。

子算法 evalBE 首先根据 BEset 建立虚节点匹配状态逻辑关系图，然后对逻辑关系图中的各节点求值。建立虚节点匹配状态逻辑关系图的过程如下：*Beset* 中的每个布尔变量对应逻辑关系图中的一个节点；由于一个布尔等式左边的布尔变量的值和右边的布尔变量的值有关系，因此对于每个布尔等式，代表等式左边的布尔变量的节点和每个代表等式右边的变量的节点之间存在一条边，连接等式右边的变量之间的逻辑运算符作为代表等式左边布尔变量的节点的属性，当等式右边只有一个布尔变量时，代表等式左边布尔变量的节点的属性为空。

对于图 2 中的实例，建立的虚节点匹配状态逻辑关系图如图 3 所示。根据该逻辑关系图对图中各节点求值的过程如下：首先，由于叶节点(G, G_2)不属于 M，因此 $X(G,G_2)$ 为假；其次，因为节点 $X(E,E_3)$ 和节点 $X(G,G_3)$ 形成一个环，互为依赖条件，所以它们的值都为真；最后，按照广度优先搜索的逆序依次对逻辑关系图的剩余节点求值，能够得到逻辑关系图中所有节点的值。求值过程结束后，协调器节点把得到的布尔表达式的值发送给需要这些值的计算节点。

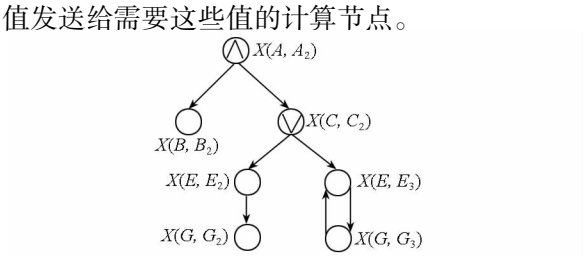


图 3 根据 BEset 建立的虚节点匹配状态逻辑关系图
Fig. 3 Constructed virtual node matching condition logic diagram according to BEset

2.4 局部匹配筛选子算法

各计算节点接收到布尔表达式的值后，调用局部匹配筛选子算法 RPM 对局部匹配集 $M_i.U$ 进行筛选，获得最终的匹配结果 $M_i.F$。算法的具体过程为：首先对每个收到的布尔等式进行处理，如果 $X(u,w)=1$，则表示 w 匹配 u，将 w 加入 u 的匹配集中；然后对 P 中节点 u 的局部匹配集中的每个节点 w 和 E_p 中的每条边 (u,u')，如果 u' 的局部匹配集中有 w 的孩子节点且至少有一个孩子节点在 u' 的匹配集中，那么可以确定 w 匹配 u。筛选操作完成后，各计算节点得到与虚节点相连的那些节点的最终匹配状态，并把匹配结果发送给协调器节点。

3 实验与结果分析

执行时间和通信量是考核图模式匹配算法的关键技术指标，本节首先比较 disGPM-PE 算法与已有算法的执行时间和通信量，然后考察模式图规模和数据图规模对 disGPM-PE 算法的执行时间和通信量的影响。实验在由 5 台机器组成的一个机器组上进行。每台机器配置均为 4 GB DDR3 内存和主频为 2.8 GHz 的 Intel CPU，机器之间由 1 Gbps 带宽的局域网络连接，其中一台机器充当协调器节点，其他机器是计算节点。每个实验重复 5 次，取平均结果为最终的实验结果。实验使用真实数据和合成数据两种不同类型的数据。在真实数据集中，使用亚马逊(Amazon)数据，数据图有 403 394 个节点和 3 387 388 条边，谷歌(Google)数据图有 875 713 个节点和 5 105 039 条边[8]。在合成数据集(Synthetic)中，产生随机数据时使用节点个数和图的边密度作为输入参数：对于数据图，l 是数据图中不同标签的个数，$|V|$ 为数据图的节点个数，α 是数据图的边密度，$|V|^{\alpha}$ 为数据图中边的条数；对于模式图，$|V_p|$ 是模式图的节点个数，α_p 是模式图的边密度。以下实验中，所有的数据集采用被大规模数据处理系统[9-10]普遍应用的哈希函数进行划分(节点 ID mod k)，分布在所有计算节点上。参数 l 默认为 200，α 和 α_p 均默认为 1.2，合成数据中数据图的节点个数默认为 1 000 000 个。

3.1 disGPM-PE 算法与已有分布式算法的比较

根据两个具有典型代表的分布式图匹配算法[4-5]的基本思想设计参考算法并与 disGPM-PE 算法进行比较。参考算法 1(refGPM－1)基于文献[5]算法基本思想，不同计算节点之间仅传送

虚节点的匹配信息,当位于不同计算节点上的数据片段之间具有依赖关系时,不作任何优化,多个计算节点根据数据片段之间的依赖关系依次进行图匹配。参考算法 2(refGPM – 2)基于文献[4]算法基本思想,首先每个计算节点并行地对本地数据进行图模式匹配,然后存储在不同计算节点上的连通子图被发送到一个计算节点上进行组合,最后再对组合后的子图进行图模式匹配。

首先使用默认实验环境配置,模式图中有 9 个节点,对 disGPM-PE 算法和两种参考算法进行对比实验。图 4 为各算法的通信量的比较结果。如图所示,无论是对于 Amazon,Google 还是合成数据集 Synthetic,refGPM – 1 算法具有最小的通信量,因为使用该算法时,不同计算节点之间仅需要传送边界点的匹配信息。而 refGPM – 2 算法的通信量最大,因为使用该算法时,需要根据数据图的划分情况,将存储在多个计算节点上的连通子图发送到一个计算节点上进行组合。所提出的 disGPM-PE 算法的通信量介于 refGPM – 1 和 refGPM – 2 算法的通信量之间,因为仅需要在计算节点之间传输布尔等式和虚节点的匹配状态信息,避免传输整个连通子图,因此通信量小于 refGPM – 2 算法的通信量,但是大于 refGPM – 1 算法仅传输边界点的匹配信息所需的通信量。

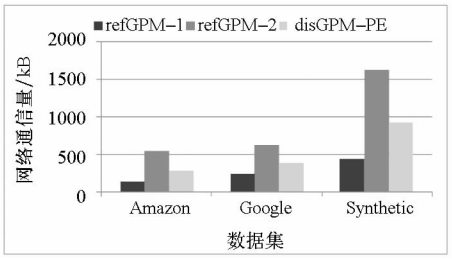


图 4　各算法通信量的比较结果

Fig. 4　Network traffic of different algorithms

图 5 为 disGPM-PE 与两种参考算法的执行时间的比较结果。如图 5 所示,对于三个数据集,refGPM – 1 算法的执行时间最长,原因是该算法执行过程中多个计算节点根据数据片段之间的依赖关系依次进行图匹配,具有依赖关系的数据片段所在的计算节点不能独立处理这些数据片段,依赖关系越多,并行性越差,执行时间越长。refGPM – 2 算法和 disGPM-PE 算法均能够对本地数据并发地进行图模式匹配,refGPM – 2 算法比 disGPM-PE 算法的执行时间稍长,因为其通信量较大造成通信时间较长。

分析图 4 和图 5 的实验结果发现:refGPM – 1 算法虽然具有最小的通信量,但是执行时间最长;

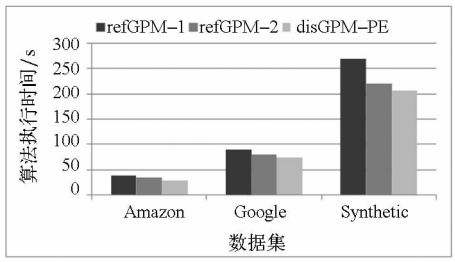


图 5　各算法执行时间的比较结果

Fig. 5　Execution time of different algorithms

refGPM – 2 算法虽然需要更多的通信量,但是却能够节省执行时间;所提出的 disGPM-PE 算法在具有比 refGPM – 2 算法执行时间稍短的情况下,具有较小的通信量。对于大规模分布式图来说,图匹配算法的执行时间非常重要,通信量也是必须控制的关键指标,所提出的 disGPM-PE 算法能够在降低执行时间的前提下控制节点间通信量的显著增长。

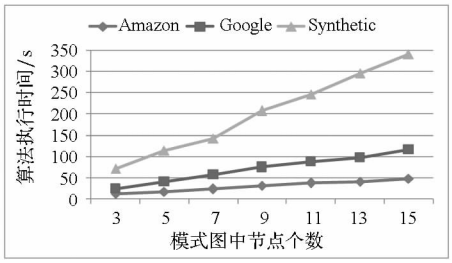
3.2　模式图大小对 disGPM-PE 执行时间的影响

算法执行时间包括本地匹配时间、通信时间、布尔表达式求值时间和局部匹配筛选时间。为了评估模式图对图匹配算法执行时间的影响,首先改变模式图的节点个数 $|V_p|$ 从 3 到 15,然后改变模式图的边密度 α_p 从 1.05 到 1.20,分别观察算法执行时间随着 $|V_p|$ 和 α_p 变化而变化的情况,实验结果如图 6(a) ~ (b)所示,对于所有的数据集,执行时间都分别随着模式图大小 $|V_p|$ 和边密度 α_p 的增加而增加,分析其原因如下:对于一个给定数据集,$|V_p|$ 增加意味着候选数据集个数增加,α_p 增加会造成每个候选匹配节点边过滤次数增加,因此本地匹配时间和局部匹配筛选时间都会随着 $|V_p|$ 和 α_p 的增加而增加,从而导致算法执行时间增加。

3.3　数据图大小对 disGPM-PE 执行时间的影响

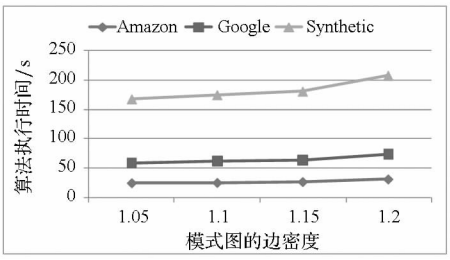
分析图 6(a)和图 6(b)还可以看出:在所有情况下,合成图的执行时间都比 Google 图的执行时间长,Google 图的执行时间比 Amazon 图的执行时间长,这些现象说明执行时间与数据图的大小有关,且随着数据图大小的增加而增加。为了进一步研究数据图大小对算法执行时间的影响,固定模式图,设置 $|V_p| = 9$ 且 $\alpha_p = 1.2$,在三个合成图(Synthetic_1 ~ 3)上进行实验,合成图 1 ~ 3 分别有 200 000,250 000 和 300 000 个节点。通过改变数据图的边密度 α 从 1.05 到 1.20,获得的实验结果如图 6(c)所示,disGPM-PE 算法执行时间随着 α 的增加而增加,且数据图规模越大,增

长速度越快。分析原因如下：当 α 增加时，图中边的条数增加，每个候选匹配节点的边筛选次数增加，从而导致算法执行时间增加。



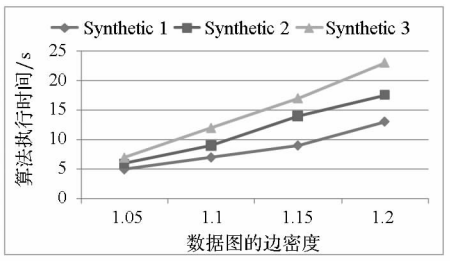
（a）模式图节点个数对算法执行时间的影响

（a）Execution time with different number of pattern graph nodes



（b）模式图的边密度对算法执行时间的影响

（b）Execution time with different edge density of pattern graph



（c）数据图规模对算法执行时间的影响

（c）Execution time with different size of data graph

图 6 模式图和数据图规模对算法执行时间的影响

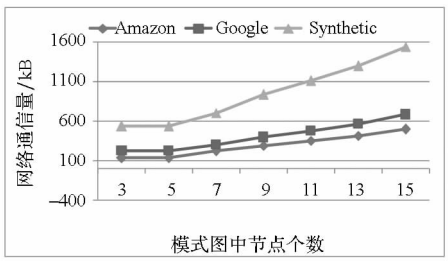
Fig. 6 Effects of the sizes of pattern graph and data graph on the execution time of disGPM-PE

3.4 模式图大小对 disGPM-PE 通信量的影响

网络通信量包括模式图、匹配结果、布尔表达式集合、虚节点的匹配状态。为了评估模式图对图匹配算法网络通信量的影响，先固定数据图的规模，然后分别改变模式图的节点个数 $|V_p|$ 从 3 ~ 15，改变模式图的边密度 α_p 从 1.05 ~ 1.20，观察网络通信量随 $|V_p|$ 和 α_p 变化而变化的情况，实验结果如图 7(a) 和图 7(b) 所示，网络通信量随着 $|V_p|$ 的增加而增加，但是对 α_p 的变化不敏感。这是因为：对一个给定数据集，$|V_p|$ 增加会造成匹配结果和布尔表达式集的增加，因此需要更多的通信量传输这些数据，而模式图中边的数量虽然会增加布尔表达式，但是同时会减少匹配结果，综合起来模式图的边密度 α_p 的变化对网络通信量造成的影响不大。

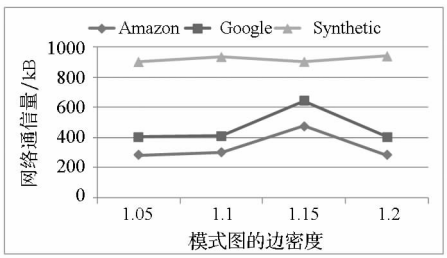
3.5 数据图大小对 disGPM-PE 通信量的影响

为了测试数据图对网络通信量的影响，同样采用固定模式图，设置 $|V_p|=9$ 且 $\alpha_p=1.2$，在与 3.3 节相同的三个合成图 Synthetic_1，Synthetic_2 和 Synthetic_3 上进行实验，实验结果如图 7(c) 所示。网络通信量随着数据图边密度 α 的增加而增加，且数据图规模越大增长速度越快。这是因为：较大的 α 意味着更多的跨边、更多的虚节点和更多的匹配结果，因此会造成更大的网络通信量。



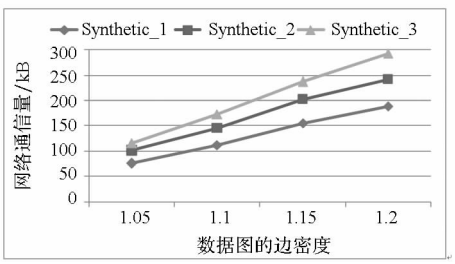
（a）模式图节点个数对网络通信量的影响

（a）Network traffic with different number of pattern graph nodes



（b）模式图的边密度对网络通信量的影响

（b）Network traffic with different edge density of pattern graph



（c）数据图规模对网络通信量的影响

（c）Network traffic with different size of data graph

图 7 模式图和数据图大小对算法网络通信量的影响

Fig. 7 Effects of the sizes of pattern graph and data graph on the network traffic of disGPM-PE

4 结论

针对分布式大图提出一种基于局部评估的分

布式图模式匹配算法 disGPM-PE,算法具有不需要明确知道数据是如何分布存储的特点,对图的划分和存储没有任何限制,因此具有更好的适用性。算法通过仅传输布尔等式和虚节点的匹配状态,能够避免部分已有算法由于需要传输连通子图而增加通信量,同时通过协调器节点统一计算虚节点匹配状态,能够在数据片段具有依赖关系时,避免串行进行图模式匹配,因此执行时间不会随依赖关系的增加而增加。实验结果表明,和已有算法相比, disGPM-PE 算法能够在降低执行时间的前提下有效控制节点间通信量的显著增长。

参考文献(References)

[1] Gallagher B. Matching structure and semantics: a survey on graph-based pattern matching [C]// Proceedings of AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection, Boston, USA, 2006: 45 - 53.

[2] Fan W F, Li J Z, Ma S, et al. Graph homomorphism revisited for graph matching [C]// Proceedings of the 36th International Conference on Very Large Data Bases, Singapore, 2010: 1161 - 1172.

[3] Liu C, Chen C, Han J W, et al. GPLAG: detection of software plagiarism by program dependence graph analysis [C]// Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA, 2006: 872 - 881.

[4] Ma S, Cao Y, Huai J P, et al. Distributed graph pattern matching[C]// Proceedings of the 21th International World Wide Web Conference, Lyon, France, 2012: 949 - 958.

[5] Fan W F, Wang X, Wu Y H, et al. Distributed graph simulation: impossibility and possibility [C]// Proceedings of the 40th International Conference on Very Large Data Bases, Hangzhou, China, 2014: 1083 - 1094.

[6] Frad A, Nisar M U, Ramaswamy L, et al. A distributed vertex-centric approach for pattern matching in massive graphs [C]// Proceedings of IEEE International Conference on Big Data, Santa Clara Marriott, CA, USA, 2013: 403 - 411.

[7] Fan W F, Wang X, Wu Y H. Performance guarantees for distributed reachability queries[C]// Proceedings of the 38th International Conference on Very Large Data Bases, Istanbul, Turkey, 2012: 1304 - 1315.

[8] Stanford University. Stanford large network dataset collection [EB/OL] (2014 - 01 - 15)[2014 - 04 - 10]. http://snap.stanford.edu/data/index.html.

[9] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters [C]// Proceedings of the 6th Symposium on Operating Systems Design and Implementation, San Francisco, USA, 2004.

[10] Malewicz G, Austern M H, BikA J C, et al. Pregel: a system for large-scale graph processing [C]// Proceedings of International Conference on Management of Data, Indianapolis, Indiana, 2010: 135 - 145.