文章编号：2095-2163(2020)03-0014-07 中图分类号：TP301.6 文献标志码：A

一种面向不确定极大团枚举的高效验证算法

杜 明，钟 鹏，周军锋
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：极大团作为稠密子图中具有代表性的一种，一直是数据挖掘领域关注的重点。极大团中蕴含的重要数据信息也被广泛应用于各种领域，例如社交网络中的社区发现等。本文研究在不确定图上枚举所有极大团的问题。现有方法基于“子图划分-求解-验证”的思想，可以有效利用极大团性质加速计算过程，但其问题在于验证算法 DPMC 的效率不稳定。当满足条件的极大团数量增多时，验证的效率会急速下降，严重影响系统的整体性能。本文提出一种高效的验证算法 FDPMU，通过构建映射表以及动态构建的倒排表，提高了算法的运行效率。最后，在多个真实数据集上进行比较，实验结果验证了 FDPMU 算法的高效性。

关键词：不确定图；不确定极大团；验证算法

An efficient verification algorithm to the maximal clique enumeration

DU Ming, ZHONG Peng, ZHOU Junfeng
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】As a representative type of dense subgraphs, the maximal clique has always been the focus of attention in the field of data mining. The important data information contained in maximal clique has also been widely used in various fields, such as community discovery in social networks, etc. This paper studies the problem of enumerating all maximal cliques on an uncertain graph. The existing method is based on the idea of " subgraph division-solving-verification" , which can effectively use the property of maximal clique to accelerate the computation. Here, the problem is that the efficiency of the verification algorithm DPMC is unstable. When the number of maximal cliques increases, the efficiency of verification will drop rapidly, which seriously affects the overall performance of the system. This paper proposes an efficient verification algorithm, FDPMU, which improves the operation efficiency of the algorithm by constructing a mapping table and a dynamically constructed inverted table. Finally, the comparison is performed on multiple real data sets, and the experimental results verify the efficiency of the FDPMU algorithm.

【Key words】uncertain graph; uncertain maximal clique; verification algorithm

0 引 言

因为现实生活中信息的繁杂多样，所以获得的数据往往有着不确定性，这些带有不确定性的数据通常用不确定图来存储表示，例如带概率信息的社交网络[1]、DNA 网络[2]、通信网络[3]等等。从不确定图中挖掘稠密子图，能够有助于更好地了解信息并解决实际生活中的问题[1,4-6]，例如对于社交网络，可以通过发现稠密子图了解到人们之间的好友信息进行好友推荐[8]。

极大团是一种典型的稠密子图。对于图中的任意顶点子集，如果其中任意两个顶点之间都有边相连，那么这个顶点子集就是一个团。如果不存在其它团包含该团，那么这个团就是一个极大团。给定不确定图及概率阈值 α，如果一个团的团概率大于等于 α，这个团就是一个 α-团。进一步，如果不存在一个更大的 α-团包含这个团，则该团就是一个 α-极大团。过去一段时间，α-极大团枚举问题得到了研究者的关注和深入研究[8-12]，然而，现有方法的效率仍然较低。

为了枚举 α-极大团，一种基本思路是通过选取不确定图中的任意顶点，然后以不断向外扩张的方式枚举所有的极大团[8-14]。其中典型的算法就是 MULE 算法[12]，MULE 算法按照每个顶点的编号顺序，递增地不断扩张，每一次扩张都选满足要求的顶点向里添加。以此来枚举出所有的 α-极大团，在递增过程中需要检测是否为 α-极大团，也就是极大性检测。该算法的时间复杂度是 $O(n \cdot 2^n)$，效率较低。

基金项目：国家重点研发计划(2017YFB0309800)；国家自然科学基金(61472339, 61572421, 61873337)。

作者简介：杜 明(1975-)，男，副教授，主要研究方向：自然语言处理、信息查询、数据分析；钟 鹏(1997-)，男，硕士研究生，主要研究方向：图的可达性查询、极大团枚举；周军锋(1977-)，男，教授，博士生导师，主要研究方向：大图数据的查询处理技术、推荐系统关键技术。

通讯作者：周军锋 Email：zhoujf@ dhu.edu.cn

收稿日期：2020-01-04

针对 MULE 算法存在的问题，文献[15]提出了一种基于子图划分的极大团枚举算法 EUMC+。EUMC+算法将枚举出所有 α-极大团的过程分解为“子图划分-求解-验证”三个阶段来高效地完成枚举过程。在子图划分阶段，将不确定图 G 作为确定图处理，将其划分成极大团子图。在求解阶段，对所有的极大团子图，调用 MULE 算法进行处理，得到所有的 α-团。最后，使用验证算法去除伪极大团，正确枚举出所有的 α-极大团。EUMC+中使用的验证算法 DPMC[15]，通过动态建立倒排表来完成去除伪极大团的工作。该算法在验证过程中，需要对每个 α-团中顶点的倒排表做交集，在 α-团数量非常多时，验证的效率会急速下降，严重影响系统的整体性能，不适宜处理稠密的不确定图。

针对以上问题，本文提出一种高效的验证算法 FDPMU，可以高效去除伪极大团，从而提升 α-极大团的枚举效率。本文后续内容安排如下：首先介绍问题定义及相关工作，然后提出新的高效验证算法 FDPMU，并详细描述算法的实现过程，接下来在 7 个真实数据集上运行算法并进行实验对比，最后总结全文。

1　相关工作

1.1　问题定义

给定不确定图 $G=(V,E,\beta)$，其中 V 表示 G 中顶点的集合，E 表示 G 中边的集合，β 表示图中的不确定性，指图中边的权值。在不确定图 $G=(V,E,\beta)$ 中，不同边的存在与否是相互独立的事件，边上的概率值并不相互影响[12]。对图 G 中的顶点 u，用 $L(u)$ 表示顶点 u 的所有邻居顶点。对于图中的顶点子集 C，$\max(C)$ 表示 C 中编号最大的顶点，$L(C)$ 表示 C 的所有邻居顶点。对于图 G 中的团 C，用 $clq(C,G)$ 表示 C 的团概率，并且规定 $clq(0,G)=1$。令 $n=|V|$，$m=|E|$，n 和 m 分别表示图 G 顶点的数量和边的数量。

以下介绍团、极大团等相关定义及问题的定义，定义 1 和定义 2 并不只适用于不确定图，也适用于确定图。

定义 1　团　在图 $G=(V,E)$ 中，如果顶点子集 C 中任意顶点之间都有边相连，则顶点子集 C 是一个团。

定义 2　极大团　对于图 $G=(V,E)$ 中的顶点子集 $M\subseteq V$，如果：(1) M 满足团的定义；(2) 不存在一个顶点 $v\in V\backslash M$，使得 $M\cup v$ 是一个团，则 M 是一个极大团。

定义 3　团概率　在不确定图 $G=(V,E,\beta)$ 中，对于任意的团 C，团概率 $clq(C,G)$ 为 C 中所有边的权值的乘积。

定义 4　α-团　在不确定图 $G=(V,E,\beta)$ 中，如果顶点子集 $C\subseteq V$，满足：(1) C 是一个团；(2) $clq(C,G)>=\alpha$，那么 C 是一个 α 团。

定义 5　α-极大团　在不确定图 $G=(V,E,\beta)$ 中，如果顶点子集 $C\subseteq V$，满足：(1) C 是一个 α 团；(2) 在图 G 中不存在一个顶点 $v\in V\backslash C$，使得 $C\cup v$ 是一个 α - 团，则 C 是图 G 中的一个 α - 极大团。

问题定义　给定不确定图 $G=(V,E,\beta)$ 和概率阈值 α，枚举图 G 中所有的 α - 极大团。

1.2　相关算法

1.2.1　MULE 算法

MULE(Maximal Uncertain cLique Enumeration)算法基于深度优先遍历(DFS)，采取顶点编号升序来处理不确定图中的顶点，并且通过维持集合 I 和集合 X 对搜索空间进行优化，以此高效地枚举不确定图中所有的 α - 极大团。

MULE 算法中用集合 C 来表示当前找到的 α - 团，当要去扩充 α - 团 C 时，只加入 C 的公共邻居顶点中编号大于 $\max(C)$ 的顶点。但是每次 C 中添加进新的顶点后，C 的团概率会降低，可能导致 $clq(C,G)<\alpha$，此时的集合 C 就不再是一个 α - 团。算法通过维持集合 I 来保证添加的顶点符合相邻以及概率的要求，集合 I 中存放的是数据对 (u,r)，u 表示顶点编号，且 $u>\max(C)$，r 表示将顶点 u 添加进入 α - 团 C 后，C 的团概率需要乘上的值。在不断的递归过程中，每次添加新的顶点进入 C 后，都需要更新集合 I，保证集合 I 中的顶点符合要求。

MULE 算法会维持另一个集合 X，确保团的极大性。X 集合中存放的也是数据对 (u,r)，含义和 I 集合中相同，只是 X 集合中存放的结点是已经在递归过程中处理过的顶点。当 I 集合为空时，只能说明此次递归过程结束，但是并不能证明 C 是一个 α - 极大团，可能 X 集合中依然有顶点能扩充 C，只有当 I 集合和 X 集合都为空时，C 才是一个符合条件的 α - 极大团。

MULE 算法降低了进行极大性检测的时间，但是 MULE 算法在处理大规模图时，因为待检测集合的增多，算法性能会急速下降。

1.2.2　EUMC+算法

EUMC+算法基于“子图划分-求解-验证”的思想枚举不确定图中所有的 α-极大团。在子图划分

阶段将不确定图当作确定图处理,调用 Degeneracy 算法[16]将不确定图划分为极大团子图。EUMC+算法可以在子图划分阶段将不可能成为极大团的顶点子集排除,同时确保不会造成 α-极大团的缺失。在求解阶段对所有的极大团子图调用 MULE 算法,枚举所有的 α-团。由于不同的极大团子图中有公共部分,所以在 MULE 算法枚举出的 α-团中,会存在某些团被其他 α-团包含的情况,这些被包含的团就是伪极大团。最后在验证过程中调用验证算法 DPMC 去除伪极大团。

在验证过程中,可以通过让所有的 α-团两两比较来去除伪极大团,但这样会导致算法效率低下。对于 2 个 α-团,只有结点个数较少的那个团才可能被包含,所以在将所有 α-团按照各团的结点个数降序排序后,只需要计算当前 α-团是否被序列前面的团包含,就可以判断该团是否为伪极大团。虽然在将 α-团排序后,加速了验证的过程,但 DPMC 算法依然存在许多的冗余计算,从而导致整个枚举极大团过程效率低下。

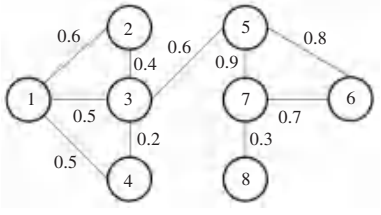

图 1 不确定图 *G*

Fig. 1 Uncertain graph *G*

2 高效的验证算法 FDPMU

2.1 FDPMU 算法的思想

在介绍本文方法之前,首先分析一下 EUMC+的验证算法 DPMC 中存在的冗余计算问题。以图 1 为例,给定概率阈值 0.1,在经过“子图划分-求解”过程后得到所有 α-团,再按各团的结点个数降序排序后得到 $A = \{\{1,2,3\}、\{5,6,7\}、\{1,3\}、\{1,4\}、\{3,4\}、\{3,5\}、\{7,8\}\}$。DPMC 算法通过动态构建倒排表来去除伪极大团,其中倒排表是 (*Key*, *Value*) 集合,*Key* 表示顶点编号,*Value* 是 *A* 集合中包含 *Key* 的 α - 团的下标。在验证过程中,首先根据 *A* 中的第一个 α - 团 {1,2,3} 建立倒排表,倒排表见表 1,其中第一列表示顶点编号,第二列是根据 {1,2,3} 建立的初始倒排表,其中存放的是 {1,2,3} 在 *A* 集合中的下标 0,第三列,第四列为后续验证过程中更新的倒排表。

遍历到 {5、6、7} 时,对倒排表中结点 5、6、7 对应的 *Value* 值求交集,交集为空集,说明 {5、6、7} 不被其他团所包含,即 {5、6、7} 是一个满足条件的 α - 极大团。更新倒排表,如表 1 中第三列“二次” 所示。在后续的验证过程中,每遍历到一个 α - 团,即对团中顶点的 *Value* 值求交集,以此判断是否为伪极大团。在验证过程中,不断更新倒排表,直到验证结束,最终倒排表见表 1 中第四列。

表 1 倒排表

Tab. 1 Inverted table

顶点编号	初始	二次	最终
1	0	0	0,2,3
2	0	0	0
3	0	0	0,2,3,4,5
4	Ø	Ø	1,5
5	Ø	1	1
6	Ø	1	1,6
7	Ø	1	2,4
8	Ø	Ø	1,6

DPMC 算法将遍历过程中获取的信息保存在倒排表中,提高了验证的效率,然而该算法依然存在着不足之处。对图 1,在子图划分阶段获得了 {5,6,7},{7,8} 等极大团子图,对子图调用 MULE 算法获得 {5,6,7},{7,8} 等 α-团,这些 α-团和极大团子图完全一致,不可能被其他 α-团包含,所以这些团一定是符合条件的 α-极大团,进行验证处理带来了冗余的计算。当这些团的数量增多时,验证的效率会急速下降,严重影响系统的整体性能,DPMC 算法还有待进一步优化。

定理 1 给定不确定图 *G*, 对于图中的所有 α-团,如果某个 α-团中存在着不属于其他团的顶点,则此 α-团是一个满足要求的 α-极大团。

证明 不确定图中的任意顶点一定会属于某个 α-极大团,因为无论怎么设定概率阈值,枚举的 α-极大团只能是图中的顶点子集,这些子集里一定包含了所有的结点。所以,如果一个极大团中存在着不属于其他团的顶点,则该团一定是一个符合条件的 α-极大团。

基于定理 1,本文提出一种高效的验证算法 FDPMU(Fast Delete Pseudo Maximal cliqUes)。其基本思想可以表述为:对于待处理的 α-团,如果其中存在着不属于其他团的顶点,则该团是一个满足要求的 α-极大团,不再进行验证。

2.2 FDPMU 算法

FDPMU 算法中,使用映射表 R 来记录顶点的出现次数,映射表是 (*Key*,*Value*) 集合,其中 *Key* 表

示顶点编号，*Value* 表示包含 *Key* 的 α－团个数，*Value* 初始值为 0。在“子图划分－求解－验证”的求解过程中，每枚举一个 α－团，就将 α－团中顶点对应的 *Value* 值加一，求解过程结束时映射表的建立也同时完成。此后，FDPMU 算法根据映射表以及动态构建的倒排表完成验证过程。FDPMU 算法流程具体如下。

算法 1　*FDPMU*( )

输入：所有的 α－团的集合 A，映射表 R

输出：所有的 α－极大团

```
1 L ← Generate L(A)
2 forall the C ∈ A do
3    if ∃v ∈ C and R[v] = 1 then
4          Output C as α-maximal clique
5    else
6          FindAndDelete(C,L)
```

FDPMU 算法中，第 1 行是根据 A 集合中的第一个元素建立倒排表的算法。*Generate* 算法流程具体如下。

算法 2　*Generate L*(*A*)

输入：所有的 α－团的集合 A

输出：倒排表 L

```
1    L← Ø
2 forall the vertex v ∈ A[0] do
3          L[v] ← 0
4 return L
```

FDPMU 算法中的第 2~6 行表示遍历 A 集合去除其中的伪极大团，第 3 行为通过映射表判断是否为满足要求的 α－极大团，若满足要求则进行输出。第 6 行表示在通过映射表无法识别伪极大团时，调用 *FindAndDelete*( ) 算法对 α－团进行验证处理，算法流程具体如下。

算法 3　*FindAndDelete*(*C*,*L*)

输入：α－团 C，倒排表 L

```
1 v0 ← the first vertex of C
2 X ← L[v0]
3 for all v ∈ C except v0 do
4    X ← X ∩ L[v]
5 if X = Ø then
6    Output C as α-maximal clique
7 forall v ∈ C do
8       L[v] ← the index of C in A
9 else
10      delete C form A
```

以图 1 为例，在完成“子图划分－求解”过程后，获得所有的 α－团，排序后表示为 A = { {1,2,3}、{5,6,7}、{1,3}、{1,4}、{3,4}、{3,5}、{7,8} }，同时完成映射表的建立，映射表见表 2，表 2 中第一行表示顶点编号，第二行表示包含该顶点的 α－团个数。

表 2　映射表

Tab. 2　Mapping table

顶点编号	1	2	3	4	5	6	7	8
团的个数	3	1	4	2	2	1	2	1

根据上述可知，FDPMU 算法利用映射表和倒排表来完成验证的过程，而且可知结果集 A 中的第一个 α－团{1,2,3}必定是一个满足条件的 α－极大团，所以首先根据{1,2,3}建立倒排表，倒排表见表 1，表 1 中各数据含义和 2.1 节中相同。

初始倒排表建立完成后，按照 A 集合中的顺序处理极大团，接下来处理 α－团{5,6,7}。首先根据映射表判断{5、6、7}中是否有不属于其他团的顶点，取得该团中顶点对应的 *Value* 值，判断是有顶点的 *Value* 值为 1，顶点 5、6、7 对应的 *Value* 值分别是 2、1、2，其中编号为 6 的顶点的 *Value* 值为 1，说明该顶点只属于{5,6,7} 这个 α－团，即该团是一个符合条件的 α－极大团，不再进行其他验证，直接更新倒排表即可，表 1 中第三列“二次”即是再次更新后的倒排表。

在后续的验证过程中，每遍历到下一个 α－团，先查找映射表，判断该团中是否存在 *Value* 值为 1 的顶点，若存在则该 α－团是一个满足要求的 α－极大团，若不存在则再对顶点对应的 *Value* 值求交集，来判断该团是否为伪极大团。在此过程中，不断更新倒排表，一直到验证结束，最终倒排表如表 1 中第四列所示。

FDPMU 算法通过映射表可以快速查找出满足条件的 α－极大团，减少了冗余的计算，提升了算法的性能。例如对于结果集 A = { {1,2,3}、{5,6,7}、{1,3}、{1,4}、{3,4}、{3,5}、{7,8} }，DPMC 算法会对 A 中每一个极大团都进行复杂的验证过程。但在 FDPMU 算法中，只通过映射表即可判断出{5,6,7}、{7,8}是满足条件的 α－极大团，不再需要额外验证。

2.3　算法分析

在顶点规模为 n 的图中，最多包含 $3^{n/3}$ 个极大团子图，对这些子图调用 MULE 算法后最少可以获得 $3^{n/3}$ 个 α－团，所以仅仅使用倒排表的 DPMC 验证

算法时间复杂度为 $O(n \cdot 3^{n/3})$。FDPMU 算法的最坏时间复杂度为 $O(n \cdot 3^{n/3})$，在最好的情况下 FDPMU 算法的时间复杂度仅为 $O(3^{n/3})$，在一般情况下，FDPMU 算法的性能也高于 DPMC 算法。

3 实验分析

3.1 实验环境

实验所使用的硬件平台是 Inter Core i5 主频为 2.60 GHz的 CPU，8 GB 的 RAM 内存，操作系统为 Windows10 64 位的系统；实验的运行环境为 Microsoft Visual Studio 2013；首先比较 FDPMU 算法和 DPMC 算法在验证过程中的性能。将 EUMC+算法中的 DPMC 算法替换为 FDPMU 算法后成为新的算法 EUML，最后比较 MULE 算法和 EUML 算法枚举所有 α-极大团的性能。以上算法均采用 C++语言实现。

3.2 数据集

本文所使用的数据由 7 个数据集组成，其中 Amaze(www.amaze.ulb.ac.be)和 Kegg(www.genome.ad.jp/kegg)数据集表示的是人的代谢网络；vchocyc、mtbrv、Anthra、ecoo、agrocyc 这些数据集来自 EcoCye(ecocyc.org)，描述的是生物中基因组之间的结构。这些数据源自生活中的不同领域，其中都蕴含着不确定信息，适合用不确定图存储表示。这些数据集的相关信息见表 3，其中符号 $|V|$ 和符号 $|E|$ 分别表示不确定图中的顶点数量和边的数量。

表 3 数据集

Tab. 3 Dataset

数据集	$\|V\|$	$\|E\|$
Amaze	3 342	7 200
Kegg	3 615	7 816
Vchocyc	9 491	20 286
Mtbrv	9 602	20 490
Anthra	12 495	26 208
Ecoo	12 620	26 700
Agrocyc	12 684	26 816

3.3 性能比较分析

3.3.1 验证算法性能比较

在现有的不确定图上，对于需要比较的验证算法 FDPMU 和 DPMC，给定同一 α 值，得到枚举所有 α-极大团的时间代价，最后通过时间的对比，验证了 FDPMU 算法的高效。

由实验结果可知，FDPMU 算法不仅可以正确去除伪极大团，而且在时间效率上比 DPMC 算法快了 4 倍左右。FDPMU 利用了伪极大团的性质，避免了冗余的验证计算，而且在 FDPMU 算法中寻找伪极大团以及删除伪极大团同时进行，提高了算法的效率。下面给定不同的概率阈值 α，在多个数据集上进行比较，并对实验结果进行分析。

给定概率阈值 α 为 0.1，DPMC 算法和 FDPMU 算法在 7 个不同数据集上的运行时间如图 2 所示。图 2 中，横坐标表示 7 个不同的数据集，纵坐标表示运行时间，单位是 ms。

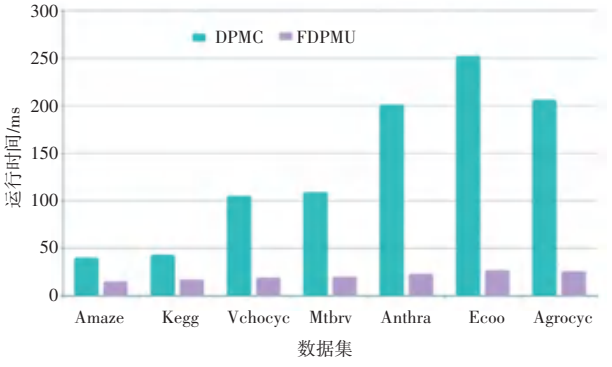

图 2 DPMC 算法和 FDPMU 算法在 7 个不同数据集上的运行时间对比

Fig. 2 Comparison of running time of DPMC algorithm and FDPMU algorithm on 7 different data sets

从图 2 可知，FDPMU 算法在性能上相较于 DPMC 算法有了较大的提高。尤其是，在 Ecoo 数据集和 Agrocyc 数据集中，FDPMU 算法比 DPMC 快了将近 12 倍左右。在 Amaze 数据集和 Kegg 数据集上，虽然 FDPMU 算法的提升并不显著，但也比 DPMC 快 2 倍左右。在其他数据集中，FDPMU 算法也有着较大的提升，图 2 可以很好地说明 FDPMU 算法的高效性。

给定概率阈值 α 为 0.2，分别在 7 个不同数据集上运行 DPMC 算法和 FDPMU 算法，运行时间对比如图 3 所示。

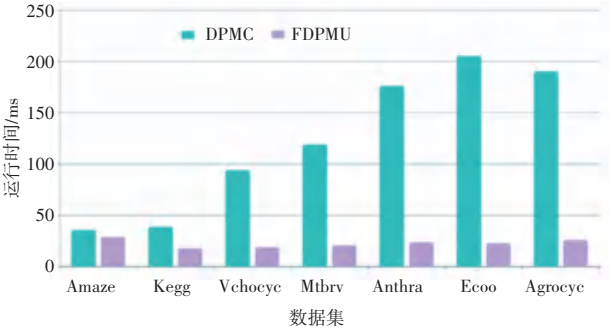

图 3 α=0.2 时，DPMC 算法和 FDPMU 算法在 7 个不同数据集上的运行时间对比

Fig. 3 When α = 0.2, the comparision of running time of DPMC algorithm and FDPMU algorithm on 7 different data sets

从图 3 可知，在 α 为 0.2 时，FDPMU 算法依然有着较大的提升，例如在 Anthra 数据集和 Ecoo 数据集上，FDPMU 算法相较于 DPMC 算法快了将近 10 倍。在有些数据集上也依然有着 3 倍的提升，这些数据都验证了 FDPMU 算法的高效性。可以看出在不同 α 值下 FDPMU 算法较 DPMC 算法都更高效。

为了更加准确地验证 FDPMU 算法的高效性，也记录了 α 为 0.3 和 α 为 0.4 情况下，2 个算法的运行时间比较，具体情况如图 4 和图 5 所示。

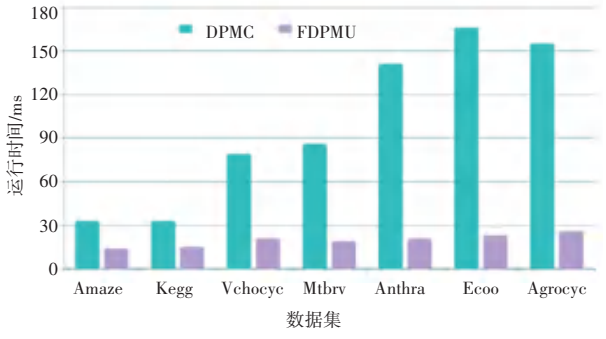

图 4　α=0.3 时，DPMC 算法和 FDPMU 算法在 7 个不同数据集上的运行时间对比

Fig. 4　When α = 0.3, the running time comparison of DPMC algorithm and FDPMU algorithm on 7 different data sets

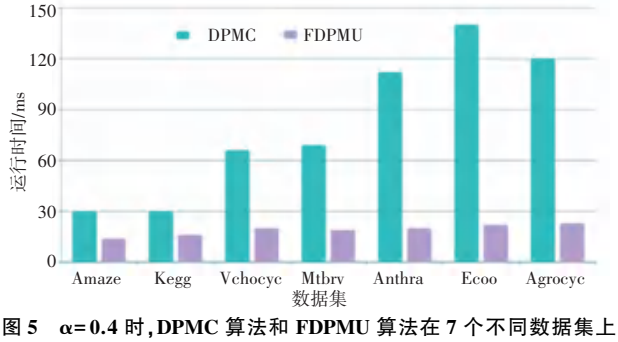

图 5　α=0.4 时，DPMC 算法和 FDPMU 算法在 7 个不同数据集上的运行时间

Fig. 5　When α = 0.4, the running time comparision of DPMC algorithm and FDPMU algorithm on 7 different data sets

从图 4 和图 5 可以看出，虽然随着 α 值的提升，FDPMU 算法相较于 DPMC 算法在时间上的提升倍数有所波动，但总体上依然快了 4～10 倍。图 4、图 5 进一步验证了 FDPMU 算法的高效性。

3.3.2　枚举极大团算法性能分析

对于 EUML 算法和 MULE 算法，在 7 个真实数据集上进行比较，记录运行时间并对算法性能进行分析比较。根据对实验结果的分析，EUML 算法相较于 MULE 算法，平均快了 6 倍左右。下面在多个数据集上进行比较，并对实验结果进行分析。

给定概率阈值 α 为 0.1，在 7 个真实数据集上运行 EUML 算法和 MULE 算法，记录时间并绘制成图，如图 6 所示。图 6 中，横坐标表示 7 个不同的数据集，纵坐标采用对数坐标表示运行时间，单位是 ms。

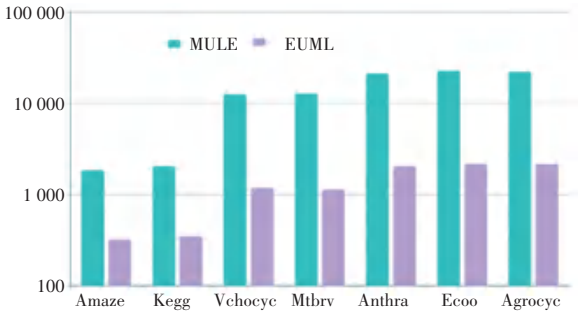

图 6　α=0.1，EUML 算法和 MULE 算法在 7 个真实数据集上的运行时间对比

Fig. 6　When α = 0.1, the running time comparison of EUML algorithm and MULE algorithm on 7 different data sets

从图 6 可知，EUML 算法相较于 MULE 算法在性能上有了较大的提高，例如在 vchocyc、ecoo、anthra 等等数据集上，EUML 比 MULE 算法快了将近 10 倍左右，在 amaze 和 kegg 数据集上，也快了 5 倍左右。图 6 很好地说明了 EUML 算法的高效性。

给定概率阈值 α 为 0.2 时，在 7 个数据集上运行算法并进行记录，所得如图 7 所示。

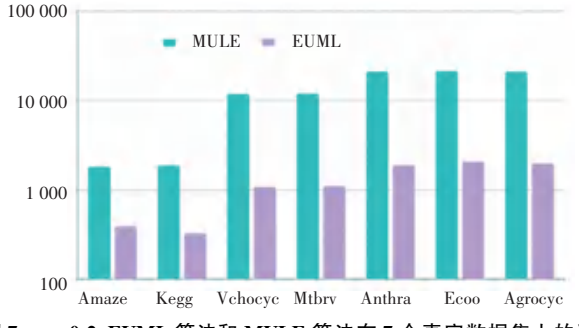

图 7　α=0.2，EUML 算法和 MULE 算法在 7 个真实数据集上的运行时间对比

Fig. 7　When α = 0.2, the running time comparison of EUML algorithm and MULE algorithm on 7 different data sets

从图 7 可知，在 α 为 0.2 时 EUML 算法的性能相较于 MULE 算法依然有着较大的提升。例如在 vchocyc、ecoo、anthra 等数据集上 EUML 算法比 MULE 算法快了 11 倍左右。从图 7 可知在 α 值为 0.1 和 0.2 的情况下，EUML 算法都有着高性能。

为了更好地比较算法的性能，记录了 2 个算法在 α 为 0.3 和 0.4 情况下的运行时间，如图 8 和图 9 所示。

从图 8 和图 9 可以看到虽然随着 α 值的提升，EUML 算法的提升倍数有所波动，不过 EUML 依然有着非常显著的优势。图 8、图 9 更好地验证了

EUML 的高效性。

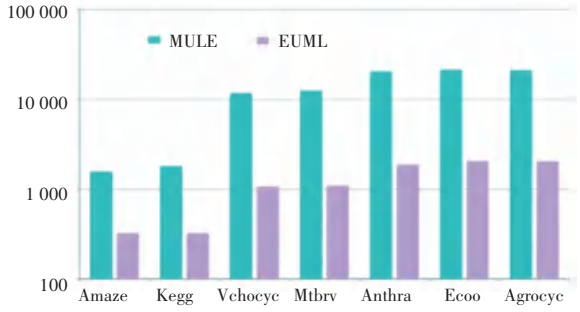

图 8 α=0.3,EUML 算法和 MULE 算法在 7 个真实数据集上的运行时间对比

Fig. 8 When α = 0.3, the running time comparison of EUML algorithm and MULE algorithm on 7 different data sets

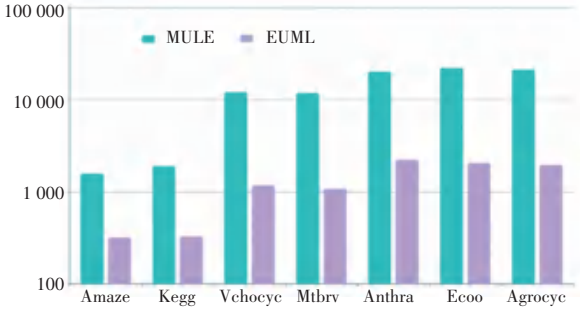

图 9 α=0.4,EUML 算法和 MULE 算法在 7 个真实数据集上的运行时间对比

Fig. 9 When α = 0.4, the running time comparison of EUML algorithm and MULE algorithm on 7 different data sets

4 结束语

针对已有概率极大团求解方法在验证阶段的低效性问题,本文提出了一种新的验证算法 FDPMU。FDPMU 算法通过构建映射表,提前筛选出符合条件的极大团,避免了冗余的验证计算,从而提高了算法效率。实验结果表明,FDPMU 算法比 DPMC 算法快 5 倍左右,而基于 FDPMU 的极大团枚举算法 EUML 比 MULE 算法快 5~10 倍。

参考文献

[1] RUAL J F, VENKATESAN K, HAO T, et al. Towards a proteome-scale map of the human protein-protein interaction network[J]. Nature, 2005, 437(7062): 1173.

[2] GHOSH J, NGO H Q, YOON S, et al. On a routing problem within probabilistic graphs and its application to intermittently connected networks[C]//Proceedings of the 26th IEEE International Conference on Computer Communications. Barcelona, Spain: IEEE, 2007: 1721.

[3] BISWAS S, MORRIS R. Exor: opportunistic multihop routing for wireless networks [J]. ACM SIGCOMM Computer Communication Review, 2005, 35(4): 133.

[4] DOURISBOURE Y, GERACI F, PELLEGRINI M. Extraction and classification of dense communities in the Web[C]//International Conference on World Wide Web. Alberta, Canada: ACM, 2007: 461.

[5] FRATKIN E, NAUGHTON B T, BRUTLAG D L, et al. MotifCut: Regulatory motifs finding with maximum density subgraphs[J]. Bioinformatics, 2006, 22(14): e150.

[6] GIBSON D, KUMAR R, TOMKINS A. Discovering large dense subgraphs in massive graphs[C]// International Conference on Very Large Data Bases. Trondheim, Norway: dblp, 2005: 721.

[7] CHEN W, WANG Y, YANG S. Efficient influence maximization in social etworks[C]// Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD '09). New York, NY, USA: ACM, 2009: 199.

[8] ROKHLENKO O, WEXLER Y, YAKHINI Z. Similarities and differences of gene expression in yeast stress conditions [J]. Bioinformatics, 2007, 23(2): 184.

[9] HARLEY E, BONNER A. Uniform integration of genome mapping data using intersection graphs[J]. Bioinformatics, 2001, 17(6): 487.

[10] JIN Ruoming, LIU Lin, AGGARWAL C C. Discovering highly reliable subgraphs in uncertain graphs[C]//Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, California: ACM, 2011: 992.

[11] KOCH I. Enumerating all connected maximal common subgraphs in two graphs[J]. Theoretical Computer Science, 2011, 250(1-2): 1.

[12] ARKO P M, PAN Xu, SRIKANTA T. Minig maximal cliques from an uncertain graph [C]//2015 IEEE 31st International Conference. Seoul, South Korea: IEEE, 2015: 243.

[13] ZOU Zhaonian, LI Jianzhong, GAO Hong, et al. Mining frequent subgraph patterns from uncertain graphs[J]. Journal of Software, 2009, 20(11): 2965.

[14] KHAN A, BONCHI F, GIONIS A, et al. Fast reliability search in uncertain graphs [C]//Proceedings of the 17th International Conference on Extending Database Technology. Athens, Greece: dblp, 2014: 535.

[15] 朱成名. 不确定图上极大团枚举算法研究[D]. 秦皇岛:燕山大学,2017.

[16] TOMITA E, TANAKA A, TAKAHASHI H. The worst-case time complexity for generating all maximal cliques and computational experiments[J]. Theoretical Computer Science, 2006, 361(1): 28.