基于多重序列所有公共子序列的启发式算法度量多图的相似度

王 旭[1,2],欧阳继红[1,2],陈桂芬[3]

(1.吉林大学 计算机科学与技术学院,长春 130012;2.吉林大学 符号计算与知识工程教育部重点实验室,长春 130012;3.吉林农业大学 信息技术学院,长春 130118)

摘 要:提出了启发式A*算法度量任意多个图的相似度方法,该算法将多图表示多重序列,在多重序列的匹配点上计算多重序列的所有公共子序列数,得到的所有公共子序列数用来度量多图的相似度。该算法避免了在非匹配点上的冗余计算,最大化后缀序列的所有公共子序列数的启发函数值,将访问的节点限制在两个序列匹配的子集,减少了计算节点的个数。与现有度量图的相似度方法相比,该算法不仅可以度量任意多个图的相似度,而且计算过程简单,通过启发信息的引导能够快速地度量多图的相似度。

关键词:人工智能;多图相似度;启发式算法;所有公共子序列;多重序列;匹配

中图分类号:TP18 **文献标志码**:A **文章编号**:1671-5497(2018)02-0526-07

DOI:10.13229/j.cnki.jdxbgxb20170302

Heuristic algorithm of all common subsequences of multiple sequences for measuring multiple graphs similarity

WANG Xu[1,2],OUYANG Ji-hong[1,2],CHEN Gui-fen[3]

(1.*College of Computer Science and Technology, Jilin University, Changchun* 130012, *China*; 2.*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun* 130012, *China*; 3.*College of Information Technology, Jilin Agricultural University, Changchun* 130118, *China*)

Abstract: A heuristic algorithm A* for measuring the similarity of multiple graphs is proposed. In this algorithm, the multiple graphs are represented as multiple sequences, then the number of all common subsequences in the matches of the multiple graphs is calculated, and this number is used to measure the similarity of the multiple graphs. This algorithm avoids the redundant calculations in non-matches of multiple graphs, maximizes the heuristic function value of all common subsequences of the suffixes sequences, limits the search nodes to the subset of matches between two sequences, thus reducing the number of the calculation nodes. Compared with the existing graph similarity methods, the proposed algorithm can not only measure the similarity of multiple graphs, but also simplify the calculation process. The similarity of multiple graphs can be quickly measured using this algorithm with the boot of the heuristic information.

收稿日期:2017-03-30.

基金项目:国家自然科学基金项目(61472157,61602204,6152198).

作者简介:王旭(1982-),男,博士研究生. 研究方向:数据挖掘,图挖掘. E-mail: 75014557@qq.com

通信作者:欧阳继红(1964-),女,教授,博士生导师. 研究方向:数据挖掘,时空推理. E-mail:ouyj@jlu.edu.cn

Key words: artificial intelligence; multiple graph similarity; heuristic algorithm; all common subsequences; multiple sequences; matches

0　引　言

图是最重要和值得深入研究的数据结构之一,例如在生物信息学和化学领域中,DNAs 和分子都是以图的结构出现的[1,2]。在分析这种图数据类型时,度量图的相似度承担了一个重要角色[3,4],图的相似度度量方法能够处理这些领域的图数据,对 DNAs 和分子进行提取和分类,有效地解决分类问题。随着图数据日益增加及其复杂性,需要更加有效的方法处理任意多个图的相似度方法[5,6],而现有的大部分度量图的相似度方法都是度量两个图的相似度[7,8]。图相似度度量方法可将图表示为序列[9-11],用度量序列的方法度量图的相似度。度量序列相似度方法可以采用所有公共子序列方法(All common subsequences,ACS)[12,13]。ACS 方法通过计算所有公共子序列数度量序列的相似度,与最长公共子序列方法相比,ACS 不仅包含了最长的公共子序列,而且包含了第二长、第三长、……的公共子序列,最大化地获取公共子序列的信息,更能反映出序列间的相似程度。所有公共子序列数越大,序列相似度越大;反之,序列相似度越小。度量序列的相似度也可以采用启发式方法,在查找多重序列的匹配时,最大化启发估计值,以便最少地扩展节点,进而找到最优解。但现有的启发式算法提出的公共子序列只包含一个字符,不适应实际的应用[14],查找最长公共子序列的启发式算法[15],不如 ACS 方法更能反映出序列之间的相似程度。为度量多图的相似度,如何将图表示为包含更多原图信息的对应序列,并应用启发函数减少扩展节点的个数,成为度量多图相似度的重要问题。

本文提出了多重序列所有公共子序列的启发式算法度量多图的相似度,将多图表示多重序列,当多重序列的一个匹配出现时,该算法递归地计算在匹配点上的所有公共子序列数,通过下标比较查找多重序列的公共子序列、剔除重复的匹配,通过后缀序列的启发函数减小计算多重序列的所有公共子序列数的节点个数,有效地解决任意多个图的相似度问题。

1　图表示为序列

1.1　图表示序列方法

将图表示为序列,序列要保留原图的信息,这样通过度量序列的相似度度量图的相似度时,度量结果才能保证图的相似度的准确性。本文采用图的深度优先搜索方法,以任一顶点作为序列的初始节点,按照该顶点与其它顶点间的路径搜索,将图表示为顶点序列。顶点序列体现了图中顶点访问的层次顺序,反映了顶点间的路径信息以及连通性,较好地保留了图的信息。在搜索顶点时,若某一顶点已访问过,就不再从该顶点出发进行搜索,搜索图的过程实质上是对每个顶点查找其邻接顶点的过程。为保证度量的准确性,选择图中相同或相似位置的顶点作为序列的初始节点。应用图深度优先搜索方法可将 d 个图 $G_1, G_2, \cdots, G_d$ 表示为 d 个序列,这些 d 个序列组成集合 $S = \{s_1, s_2, \cdots, s_d\}$,$S$ 为字母表 Σ 上的多重序列(multiple sequences),$d \geqslant 2$。本文研究的图是顶点带标号的有向图。

以两个图 G_1 和 G_2(如图 1 所示)为例,选取具有相同结构的顶点(e 和 a)为序列的初始节点,应用图深度优先搜索方法将 G_1 和 G_2 表示为对应顶点序列 s_1 和 s_2,s_1 = expect,s_2 = accept。

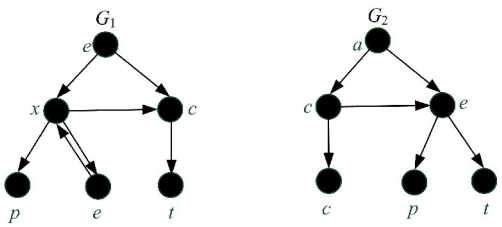

图 1　有向图 G_1 和 G_2

Fig.1　The directed graphs G_1 and G_2

1.2　所有公共子序列方法

所有公共子序列方法(ACS)可以度量序列的相似度,所有公共子序列数越大,序列的相似度越大;反之,序列的相似度越小。下面给出 d 个序列的所有公共子序列的定义,$d \geqslant 2$。

定义 1　从序列 s 中删除 0 个或更多的元素所获得的序列称为 s 的子序列;

(1) 若序列 x 为 s_i 的子序列,$1 \leqslant i \leqslant d$,则称 x 为多重序列 S 的公共子序列;

(2) 满足(1) 的情况下所有的 x 称为 S 的所有公共子序列。用 | MACS | 表示 S 的所有公共子序列数。

为度量多重序列 S 的相似度,应用所有公共子序列方法计算 | MACS |, | MACS | 越大,多重序列的相似度越大;反之,多重序列的相似度越小。以两个序列 s_1,s_2 为例,长度分别为 n_1,n_2,$ACS(x, y)$ 表示 s_1 和 s_2 所有公共子序列数,有下式成立。

$$ACS(x,y)=\begin{cases}1 & \text{if } x \text{ or } y=0\\ 2\times ACS(x-1,y-1) & \text{if } s_1[x]=s_2[y]\\ ACS(x-1,y)+ACS(x,y-1)- & \\ \quad ACS(x-1,y-1) & \text{if } s_1[x]\neq s_2[y]\end{cases} \tag{1}$$

式中:当 $x=0$ 或 $y=0$ 时,s_1 和 s_2 的公共子序列仅有空集∅,所以 $ACS(0, y)=1$,$ACS(x, 0)=1$ 和 $ACS(0, 0)=1$;当 $s_1[x]=s_2[y]$ 时,$s_1[x]$ 或 $s_2[y]$ 成为 s_1 和 s_2 新的公共子序列,$s_1[x]$ 或 $s_2[y]$ 与 $ACS(x-1, y-1)$ 已得到的公共子序列组成新的公共子序列,新的公共子序列数为 $ACS(x-1, y-1)$,在 $s_1[x]$ 或 $s_2[y]$ 点上的所有公共子序列数是 $ACS(x-1, y-1)$ 的2倍,所以 $ACS(x, y)=ACS(x-1, y-1)\times 2$,$0\leqslant x\leqslant n_1$,$0\leqslant y\leqslant n_2$。公式(1) 迭代地建立 $n_1\times n_2$ 矩阵 $\boldsymbol{M}$,计算 s_1 和 s_2 的所有公共子序列数,$ACS(x, y)$ 值越大,s_1 和 s_2 越相似。

对于图1中 G_1 和 G_2 的对应序列 s_1=expect 和 s_2=accept,由公式(1) 计算得到的所有公共子序列数如表1所示,s_1 和 s_2 的所有公共子序列为:{∅, e, p, c, t, ct, ep, et, pt, ept},s_1 和 s_2 所有公共子序列数 $ACS(s_1, s_2)=10$。

表1 s_1 和 s_2 的所有公共子序列数

Table 1 The number of all common subsequences between s_1 and s_2

		e	x	p	e	c	t
	1	1	1	1	1	1	1
a	1	1	1	1	1	1	1
c	1	1	1	1	1	2	2
c	1	1	1	1	1	2	2
e	1	2	2	2	2	3	3
p	1	2	2	4	4	5	5
t	1	2	2	4	4	5	10

2 ACS的启发式方法

多重序列表示为 $S=\{s_1, s_2, \cdots, s_d\}$,$d\geqslant 2$,对应的长度分别为 $n_1, n_2, \cdots, n_d$。S 中长度分别为 n_i,n_j 任意两个序列 s_i 和 s_j 的后缀序列为:$s_i[x+1, \cdots, n_i]$ 和 $s_j[y+1, \cdots, n_j]$,$0\leqslant x\leqslant n_i$,$0\leqslant y\leqslant n_j$,$0\leqslant i, j\leqslant d$,$ACS_{\text{suf}}(x, y)_{ij}$ 表示两个后缀序列的所有公共子序列数,有下式成立:

$$ACS_{\text{suf}}(x,y)_{ij}=\begin{cases}0 & \text{if } x=n_i \text{ or } y=n_j\\ 2\times ACS_{\text{suf}}(x+1,y+1)_{ij} & \text{if } s_i[x+1]=s_j[y+1]\\ ACS_{\text{suf}}(x,y+1)_{ij}+ACS_{\text{suf}}(x+1,y)_{ij}- & \\ \quad ACS_{\text{suf}}(x,y)_{ij} & \text{if } s_i[x+1]\neq s_j[y+1]\end{cases} \tag{2}$$

式中:当 $x=n_i$ 或 $y=n_j$,s_i 和 s_j 没有后缀序列,所以 $ACS_{\text{suf}}(n_i, y)_{ij}=0$,$ACS_{\text{suf}}(x, n_j)_{ij}=0$ 和 $ACS_{\text{suf}}(n_i, n_j)_{ij}=0$;当 $s_i[x+1]=s_j[y+1]$ 时,$s_i[x+1]$ 或 $s_j[y+1]$ 与 $ACS_{\text{suf}}(x+1, y+1)_{ij}$ 得到的公共子序列组成新的公共子序列,所以 $ACS_{\text{suf}}(x, y)_{ij}=ACS_{\text{suf}}(x+1, y+1)_{ij}\times 2$。公式(2) 迭代地建立 $(n_i-x)\times(n_j-y)$ 矩阵 M_{suf},计算 s_i 和 s_j 后缀序列的所有公共子序列数,$ACS_{\text{suf}}(x, y)_{ij}$ 值越大,s_i 和 s_j 后缀序列越相似。

p 表示多重序列 S 上的点,$p=(p_1, p_2, \cdots, p_i, \cdots, p_d)$,$1\leqslant i\leqslant d$,$1\leqslant p_i\leqslant n_i$,$p_i$ 表示序列 n_i 上的点,n_i 表示 n_i 的长度。对于点 p 的 d 个后缀序列 $s_i[p_i+1, \cdots, n_i]$,可以应用公式(2) 计算这些后缀序列的所有公共子序列数,$h^*(p)$ 表示任意两个后缀序列 $s_i[p_i+1, \cdots, n_i]$ 和 $s_j[p_j+1, \cdots, n_j]$ 的所有公共子序列数,$1\leqslant i, j\leqslant d$,有下式成立。

$$h^*(p)=\min_{1\leqslant i,j\leqslant d} ACS_{\text{suf}}(p_i,p_j) \tag{3}$$

定理1 若 $h(p)$ 表示 d 个后缀序列 $s_1[p_1+1, \cdots, n_1]$,$s_2[p_2+1, \cdots, n_2]$,$\cdots$,$s_i[p_i+1, \cdots, n_i]$,$\cdots$,$s_d[p_d+1, \cdots, n_d]$ 的所有公共子序列数,$1\leqslant i\leqslant d$,则 $h(p)\leqslant h^*(p)$。

证明:d 个后缀序列的所有公共子序列是 d 中任意两个序列最少的所有公共子序列的子集,d 个后缀序列的所有公共子序列数小于等于 d 中任意两个序列的所有公共子序列数的最小值,由公式(3) 可知,$h^*(p)$ 表示任意两个序列的所有

公共子序列数的最小值,$h^*(p)$ 肯定大于等于 d 个后缀序列的所有公共子序列数 $h(p)$,即 $h(p) \leqslant h^*(p)$,得证。

若 $s_1[p_1] = s_2[p_2] = \cdots = s_i[p_i] = \cdots = s_d[p_d]$,称点 $p = (p_1, p_2, \cdots, p_i, \cdots, p_d)$ 为多重序列 S 上的一个匹配。对于图 1 中 G_1 和 G_2 的对应序列 s_1 =expect 和 s_2 =accept,点(4,4)为 s_1 和 s_2 在字符 e 的匹配,(5,3)为 s_1 和 s_2 在 c 的匹配,s_1 和 s_2 的所有匹配出现的位置如表 2 所示。

表 2　s_1 和 s_2 匹配出现的位置

Table 2　The locations of the matches between s_1 and s_2

		e	x	p	e	c	t
a							
c						2	
c						②	
e		2			②		
p				4			
t							10

定义 2　对于多重序列 S 上的两个点 $p = (p_1, p_2, \cdots, p_i, \cdots, p_d)$,$q = (q_1, q_2, \cdots, q_i, \cdots, q_d)$,如果满足 $p_i < q_i$,$1 \leqslant i \leqslant d$,则称 p 的所有下标对应地小于 q 的所有下标,记 $Subscript(p) < Subscript(q)$。

若 p 和 q 为 S 上的两个匹配,$Combine(p, q)$ 表示 p 和 q 的合并后的序列:

$$Combine(p,q) = \begin{cases} pq & \text{if } Subscript(p) < Subscript(q) \\ qp & \text{if } Subscript(p) > Subscript(q) \end{cases} \quad (4)$$

对于图 1 中 G_1 和 G_2 的对应序列 s_1 和 s_2,(1, 4) 和 (3, 5) 分别为字符 e 和 p 的匹配,且 $Subscript(e) < Subscript(p)$,合并 (1, 4) 和 (3, 5) 得到的序列为:$Combine((1,4),(3,5)) = Combine(e,p) = (ep)$。

定理 2　若点 p 和 q 为 S 上的两个匹配,则 $Combine(p, q)$ 为 S 的公共子序列。

证明:根据定义 2 和公式(4),当 q 的所有下标对应地大于 p 的所有下标,即 $p_i < q_i$,或 q 的所有下标对应地小于 p 的所有下标,即 $q_i < p_i$,才可以合并 p 和 q。点 p 和 q 为 S 上的匹配,则 p 和 q 分别为 S 的公共子序列,对于 s_i 上的点 p_i 和 q_i,p_i 和 q_i 分别为 S 的公共子序列,当 $p_i < q_i$ 或 $q_i < p_i$,则序列 (p_iq_i) 或 (q_ip_i) 也是 S 的公共子序列,$1 \leqslant i \leqslant d$。所以合并 p 和 q 的序列 $Combine(p, q)$ 为 S 的公共子序列,得证。

合并两个匹配可以得到公共子序列,当新的匹配出现,如果新的匹配的下标大于或小于合并后序列的下标,就会产生新的公共子序列,所以需要标记合并后序列的下标。合并后的序列 $Combine(p, q)$ 的下标为 p 和 q 中最大的下标,有下式成立。

$$Subscript(Combine(p,q)) = \begin{cases} Subscript(q) & \text{if } Subscript(p) < Subscript(q) \\ Subscript(p) & \text{if } Subscript(p) > Subscript(q) \end{cases} \quad (5)$$

对于图 1 中 G_1 和 G_2 的对应序列 s_1 和 s_2,(1, 4) 为字符 e 的匹配,(3, 5) 为字符 p 的匹配,合并后的序列 (ep) 的下标 $Subscript(ep) = Subscript(p) = (3, 5)$。

由定理 2 可知,当 $Combine(p, q)$ 为 S 的公共子序列,$Combine(p, q)$ 可以看作 S 上新的匹配,$Combine(p, q)$ 下标为 p 和 q 中最大的下标,所以有:

推论 1　点 p, q, r 为 S 上的匹配,若 $Subscript(r) > Subscript(q) > Subscript(p)$,则 (pqr) 为 S 的公共子序列。

对于图 1 中的 s_1 和 s_2,(1, 4) 为字符 e 的匹配,(3, 5) 为字符 p 的匹配,(6, 6) 为字符 t 的匹配,合并后的序列 (ept) 为 S 的公共子序列。

定义 3　设 $f(n) = g(n) + h(n)$,$g(n)$ 表示从初始节点到节点 n 付出的实际代价,$h(n)$ 表示从节点 n 到目标节点的最优路径的估计代价,称使用 $f(n)$ 作为估价函数的 GRAPHSEARCH 算法为 A。若算法 A 中使用的启发函数 $h(n)$ 对任何节点都有 $h(n) \leqslant h^*(n)$,则称其为 A^* 算法。

$|MACS|$ 表示多重序列 $(s_1, s_2, \cdots, s_d)$ 的所有公共子序列数,$d \geqslant 2$,$f(p)$ 表示 d 个序列 $(s_1, s_2, \cdots, s_d)$ 的 $|MACS|$ 的估计函数,$g(p)$ 表示 d 个前缀序列 $s_1[1, \cdots, p_1]$,$s_2[1, \cdots, p_2]$,$\cdots$,$s_i[1, \cdots, p_i]$,$\cdots$,$s_d[1, \cdots, p_d]$ 的 $|MACS|$,$g(p)$ 可由公式(1)计算,$h(p)$ 表示 d 个后缀序列 $s_1[p_1+1, \cdots, n_1]$,$s_2[p_2+1, \cdots, n_2]$,$\cdots$,$s_i[p_i+1, \cdots, n_i]$,$\cdots$,$s_d[p_d+1, \cdots, n_d]$ 的 $|MACS|$,$1 \leqslant i \leqslant d$,$h(p)$ 可由公式(2)计算,所以 $|MACS|$ 的估计函数为:

$$f(p) = g(p) + h(p) \quad (6)$$

$h^*(p)$表示d中任意两个后缀序列$s_i[p_i+1,\cdots,n_i]$和$s_j[p_j+1,\cdots,n_j]$的|MACS|的估计函数,$1\leqslant i,j\leqslant d$,$h^*(p)$可由公式(3)计算。用$h^*(p)$代替$h(p)$计算后缀序列的|MACS|,|MACS|的估计函数变成:

$$f^*(p)=g^*(p)+h^*(p) \tag{7}$$

定理3 计算|*MACS*|的估计函数$f^*(p)$是A*算法。

证明:由公式(7)可知,$f^*(p)=g^*(p)+h^*(p)$,$f^*(p)$表示从初始节点出发经由节点p到达目标节点的所有公共子序列数的估计函数,$g^*(p)$表示从初始节点到节点p的所有公共子序列数,$h^*(p)$表示从节点p到目标节点的所有公共子序列数的估计函数。由定理1可知$h(p)\leqslant h^*(p)$,根据定义2,$f^*(p)$是A*算法,得证。

用MACS-A*表示计算|MACS|的估计函数$f^*(p)$,与A*算法查找最短搜索路径不同,MACS-A*是在d维矩阵里计算d个序列的所有公共子序列数。MACS-A*是A*算法的变形,在每次计算点p位置的所有公共子序列数时,最大化启发函数$h^*(p)$。$h^*(p)$越大,包含的启发信息越多,所需计算的节点越少,更快地找到目标节点,计算所有公共子序列数。若$h^*(p)=0$,点p到达目标节点,$f^*(p)=g^*(p)+h^*(p)=g^*(p)=|\text{MACS}|$,所以有:

推论2 若$h^*(p)=0$,则$g^*(p)=|\text{MACS}|$。

3 度量多图相似度算法MACS-A*

通过计算多重序列的所有公共子序列数可以度量多重序列的相似度,根据将多图表示为多重序列的图深度优先搜索方法,度量多重序列的相似度可以度量多图的相似度,所以,计算多重序列的所有公共子序列数可以度量多图的相似度,度量图的相似度问题简化为计算所有公共子序列数的问题。多重序列的所有公共子序列数越大,多图的相似度越大;反之,相似度越小。因此,本文提出了计算多重序列所有公共子序列数的启发式算法MACS-A*,通过计算d个序列的前缀序列和后缀序列的所有公共子序列数度量d个图的相似度,$d\geqslant 2$。

MACS-A*算法从初始点$p_0=(0,0,\cdots,0)$开始,计算d个序列的所有单个字符的匹配,放在集合Q中。从表1可以看出,当一个匹配出现时,该匹配为多重序列的公共子序列,且该匹配与之前的公共子序列组成新的公共子序列。但对于表2圆圈中标记的匹配位置,该匹配已经出现,与之前的公共子序列组成新的公共子序列也已经出现,而且公式(2)在该匹配计算的所有公共子序列数并没有变化。因为圆圈中匹配的下标不大于之前出现过的匹配的下标,所以当圆圈中的匹配出现时,不能组成新的公共子序列。对于圆圈中的匹配,在查找所有公共子序列和计算所有公共子序列数时,通过下标比较,不对该匹配进行合并序列、标记下标操作。

MACS-A*算法首先应用图深度优先搜索方法将d个图表示为d个序列;接着从Q中提取一个匹配p,T为d个序列的所有公共子序列的集合,比较p与Q、T中的序列的下标,合并序列后组成新的公共子序列,并对新的公共子序列标记下标,将不重复新的公共子序列放在T中;然后用公式(7)计算d个序列的所有公共子序列数,其中$f^*(p)$表示d个序列的所有公共子序列数估计函数,$g^*(p)$表示d个前缀序列的所有公共子序列数,$h^*(p)$表示d个后缀序列的所有公共子序列数的启发函数;最后返回d个序列的所有公共子序列和所有公共子序列数。

算法1:MACS-A*($s_1,s_2,\cdots,s_d$)算法

输入:图$G_1,G_2,\cdots,G_d$

输出:d个图的所有公共子序列和所有公共子序列数

步骤1:应用图深度优先搜索方法将d个图表示为d个序列($s_1,s_2,\cdots,s_d$),$d\geqslant 2$;

步骤2:初始$p_0=(0,0,\cdots,0)$,$g^*(p_0)=0$,$f^*(p_0)=h^*(p_0)$,集合T表示d个序列的所有公共子序列,初始$T=\{\varnothing\}$,空集的下标$Subscript(\varnothing)=(0,0,\cdots,0)$;

步骤3:计算d个序列的所有匹配,并放入Q中;

步骤4:|Q|表示d个序列的所有匹配数,从Q中取出一个匹配p,q表示Q中的任一匹配,t表示T中任一公共子序列,若|Q|>0,转到步骤5,否则,转到步骤10;

步骤5:比较p和q的下标,若$Subscript(p)>Subscript(q)$,否则,转到步骤6,合并$p$和$q$,$Combine(q,p)=(qp)$,并用$p$的下标标记$(qp)$的下标,$Subscript(qp)=Subscript(p)$,

若(qp)不在 T 中,$Label(qp) \notin Label(t)$,将$(qp)$放入 T 中,$T = T + \{qp\}$,转到步骤 7;

步骤 6:若 $Subscript(p) < Subscript(q)$,合并 p 和q,$Combine(q, p) = (pq)$,并用q的下标标记(pq)的下标,$Subscript(pq) = Subscript(q)$,若$(pq)$不在 T 中,$Label(pq) \notin Label(t)$,将$(pq)$放入 T 中,$T = T + \{pq\}$,转到步骤 7;

步骤 7:若 p 不在 T 中,$label(p) \notin label(t)$,将 p 并入 T 中,$T = T + \{p\}$,计算 d 个序列的所有公共子序列数,$f^*(p) = g^*(p) + h^*(p)$,转到步骤 8;

步骤 8:比较 p 和 t 的下标,若 $Subscript(p) > Subscript(t)$,将合并 p 和 t 的序列(tp)并入 T 中,$T = T + \{tp\}$,标记(tp)的下标,$Subscript(tp) = Subscript(p)$,转到步骤 9;

步骤 9:从 Q 中删除 p,$Q = Q - \{p\}$,转到步骤 4;

步骤 10:输出 T 和 $f^*(p)$,T 为 d 个序列的所有公共子序列集合,包括空集,$f^*(p)$ 为 d 个序列的所有公共子序列数。

算法 1 包括两部分:将 d 个图 $G_1, G_2, \cdots, G_d$ 表示为 d 个序列$(s_1, s_2, \cdots, s_d)$,$d \geqslant 2$;计算 d 个序列的所有公共子序列数,并输出所有公共子序列。为了方便分析算法 1 的时间复杂度,设 d 个图的顶点数均为n,则 d 个序列$(s_1, s_2, \cdots, s_d)$的长度均为 n。

算法 1 的步骤 1 应用图深度优先搜索方法将顶点带标号的有向图 G 表示为序列 s,时间复杂度为$O(n + |E|)$,其中 n 和 $|E|$ 分别表示 G 的顶点数和边数。图深度优先搜索方法将 d 个图表示为 d 个序列的时间复杂度为 $O(dn + d|E|)$。若 $|E| = n \times (n-1) = (n^2 - n)$,则 G 为完全有向图。最坏情况下,将 d 个图表示为 d 个序列的时间复杂度为 $O(dn + d(n^2 - n)) = O(dn^2)$。

算法 1 的步骤 3 为计算 d 个序列的所有匹配,从表 1 可以看出,建立 2 维矩阵 M_{ij} 需要 $O(n^2)$ 的时间复杂度,$1 \leqslant i, j \leqslant d$。从矩阵 M_{ij} 可看出,计算 2 个序列中的所有匹配,需要 $O(n^2)$ 的时间复杂度,计算 d 个序列的所有匹配,需要 $O(dn^2)$ 的时间复杂度。

在算法 1 的步骤 4 中,Q 为 d 个序列的所有单个字符匹配的集合,$|Q|$ 最大为 n,n 表示序列 s_i 的长度,$1 \leqslant i \leqslant d$。步骤 5、步骤 6 和步骤 8 为比较 p 和 Q 中匹配的下标、p 和 T 中的公共子序列的下标,时间复杂度为 $O(d)$。步骤 7 为避免重复序列出现在 T 中,比较新组成的公共子序列与 T 中序列的字符,判断新的序列是否存在 T 中,新的序列最长为 n,最多为 $|\Sigma|$ 个字符,Σ 表示字母表,比较序列字符的时间复杂度为 $O(n|\Sigma|)$。当新组成的公共子序列出现时,对 p 与 Q 中的每个匹配、T 中的公共子序列进行合并和标记下标,其操作的时间复杂度为 $O(1)$。所以,处理点 p 的时间复杂度总共为 $O(dn|\Sigma|)$。步骤 9 从 Q 中删除点 p,当执行完步骤 4,Q 为空,所需的时间复杂度为 $O(dn^2|\Sigma|)$。

点 p_0 从$(0, 0, \cdots, 0)$到$(n, n, \cdots, n)$,当处理 d 个序列的每个匹配时,用公式(1)和公式(3)递归地计算 d 个序列在点 p 的所有公共子序列数 $f^*(p) = g^*(p) + h^*(p)$,其时间复杂度也为 $O(dn^2|\Sigma|)$。结合步骤 3 的时间复杂度,用 MACS-A* 算法查找 d 个序列的所有公共子序列的时间复杂度为 $O(dn^2 + dn^2|\Sigma|)$,递归地计算 d 个序列的所有公共子序列数 $f^*(p)$ 的时间复杂度也为 $O(dn^2 + dn^2|\Sigma|)$。

所以,结合步骤 1 的时间复杂度,用 MACS-A* 算法度量 d 个图的相似度的时间复杂度为 $O(dn^2 + dn^2 + dn^2|\Sigma|)$。

4　结束语

本文提出的启发式算法 MACS-A* 通过计算多重序列的所有公共子序列数度量多图的相似度,由于所有公共子序列数的变化都是在多重序列的匹配出现之后,所以 MACS-A* 算法递归地计算在匹配点上的所有公共子序列数,不必计算所有点的公共子序列数,避免了在非匹配点上冗余计算。该算法在处理匹配的过程中最大化后缀序列的启发函数值 $h^*(p)$,将访问的匹配点 p 限制在矩阵 $\boldsymbol{M}$(公式(1))的匹配的子集,通过下标比较剔除不能组成新的公共子序列的匹配,进一步减少了计算节点的个数,能够快速地度量多图的相似度。

参考文献:

[1] Hattori M, Okuno Y, Goto S, et al. Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in

the metabolic pathways [J]. Journal of the American Chemical Society, 2003, 125(39): 11853-11865.

[2] Shanavas N, Wang H, Lin Z, et al. Supervised graph-based term weighting scheme for effective Text Classification [C] // Proceedings of the 22nd European Conference on Artificial Intelligence, Hague, Netherlands, 2016: 1710-1711.

[3] Elzinga C, Wang H. Kernels for acyclic digraphs [J]. Pattern Recognition Letters, 2012, 33(16): 2239-2244.

[4] Sugiyama M, Llinares F, Kasenburg N, et al. Significant subgraph mining with multiple testing correction [C] // Proceedings of the SIAM International Conference on Data Mining, Vancouver, Canada, 2015: 37-45.

[5] Li Tao, Dong Han, Shi Yong-tang, et al. A comparative analysis of new graph distance measures and graph edit distance [J]. Information Sciences, 2017, 403: 15-21.

[6] Zhu Gang-gao, Iglesias C. Computing semantic similarity of concepts in knowledge graphs [J]. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(1): 72-85.

[7] Sugiyama M, Borgwardt K. Halting in random walk Kernels [C]//28th International Conference on Neural Information Processing Systems, Montreal, Canada, 2015: 1639-1647.

[8] Borgwardt K, Kriegel H. Shortest-path Kernels on graphs [C]// Proceedings of IEEE International Conference on Data Mining, Houston, USA, 2005: 74-81.

[9] Szilagyi S, Szilagyi L. A fast hierarchical clustering algorithm for large-scale protein sequence data sets [J]. Computers in Biology and Medicine, 2014, 48: 94-101.

[10] Forster D, Bittner L, Karkar S, et al. Testing ecological theories with sequence similarity networks: marine ciliates exhibit similar geographic dispersal patterns as multicellular organisms [J]. BMC Biology, 2015,13(1):1-16.

[11] Yanardag P, Vishwanathan S. A structural smoothing framework for robust graph comparison [C]// Proceedings of Neural Information Processing Systems, Montreal, Canada, 2015: 2134-2142.

[12] Wang H. All common subsequences [C]// Proceeding 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 2007: 635-640.

[13] Lin Z, Wang H, McClean S. A multidimensional sequence approach to measuring tree similarity [J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(2): 197-208.

[14] Chin F, Poon C.Performance analysis of some simple heuristics for computing longest common subsequences [J]. Algorithmica, 1994, 12: 293 - 311.

[15] Wang Q, Korkin D, Shang Y. A fast multiple longest common subsequence (MLCS) algorithm [J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(3): 321-334.