软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(5):1096−1112 [doi: 10.13328/j.cnki.jos.004707]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一般间隙及一次性条件的严格模式匹配*

柴 欣[1], 贾晓菲[1], 武优西[1], 江 贺[2], 吴信东[3,4]

[1](河北工业大学 计算机科学与软件学院,天津 300401)

[2](大连理工大学 软件学院,辽宁 大连 116621)

[3](合肥工业大学 计算机科学与信息工程学院,安徽 合肥 230009)

[4](Department of Computer Science, University of Vermont, Burlington, USA)

通讯作者: 吴信东, E-mail: xwu@hfut.edu.cn

摘 要: 具有间隙约束的模式匹配是序列模式挖掘的关键问题之一.一次性条件约束是要求序列中每个位置的字符最多只能使用一次,在序列模式挖掘中采用一次性条件约束更加合理.但是目前,间隙约束多为非负间隙,非负间隙对字符串中每个字符的出现顺序具有严格的约束,一定程度上限定了匹配的灵活性.为此,提出了一般间隙及一次性条件的严格模式匹配问题;之后,理论证明了该问题的计算复杂性为 NP-Hard 问题.为了对该问题进行有效求解,在网树结构上构建了动态更新结点信息的启发式求解算法(dynamically changing node property,简称 DCNP).该算法动态地更新各个结点的树根路径数、叶子路径数和树根-叶子路径数等,进而每次可以获得一个较优的出现;之后,迭代这一过程.为了有效地提高 DCNP 算法速度,避免动态更新大量的结点信息,提出了 Checking 机制,使得 DCNP 算法仅在可能产生内部重复出现的时候才进行动态更新.理论分析了 DCNP 算法的时间复杂度和空间复杂度.大量实验结果验证了 DCNP 算法具有良好的求解性能.

关键词: 一般间隙;模式匹配;一次性条件;网树

中图法分类号: TP311

中文引用格式: 柴欣,贾晓菲,武优西,江贺,吴信东.一般间隙及一次性条件的严格模式匹配.软件学报,2015,26(5):1096−1112. http://www.jos.org.cn/1000-9825/4707.htm

英文引用格式: Chai X, Jia XF, Wu YX, Jiang H, Wu XD. Strict pattern matching with general gaps and one-off condition. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1096−1112 (in Chinese). http://www.jos.org.cn/1000-9825/4707.htm

Strict Pattern Matching with General Gaps and One-Off Condition

CHAI Xin[1], JIA Xiao-Fei[1], WU You-Xi[1], JIANG He[2], WU Xin-Dong[3,4]

[1](School of Computer Science and Software, Hebei University of Technology, Tianjin 300401, China)

[2](School of Software, Dalian University of Technology, Dalian 116621, China)

[3](School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China)

[4](Department of Computer Science, University of Vermont, Burlington, USA)

Abstract: Pattern matching with gap constraints is one of the key issues of sequential pattern mining. One-off condition which is always used in sequential pattern mining tasks means that the character of each position in the sequence can be used at most once for pattern matching. Recently, most researches focus on pattern matching with non-negative gaps, but the rule of non-negative gaps implies the character's order in the sequence may limit the flexibility of matching. For these reasons, this article proposes a strict pattern matching with general gaps and one-off condition and shows that this problem is NP-hard. To tackle this issue, a heuristic algorithm, named

* 基金项目: 国家自然科学基金(61229301, 61370144); 国家高技术研究发展计划(863)(2012AA011005); 教育部创新团队发展计划(IRT13059); 河北省自然科学基金(F2013202138); 河北省教育厅重点项目(ZH2012038)

收稿时间: 2014-05-22; 定稿时间: 2014-08-15; jos 在线出版时间: 2014-12-12

CNKI 网络优先出版: 2014-12-12 13:52, http://www.cnki.net/kcms/detail/11.2560.TP.20141212.1352.001.html

dynamically changing node property (DCNP), is designed based on nettree which dynamically updates the properties of each node such as the numbers of root paths, leaf paths and root-leaf paths, and thus can get a better occurrence. The above process is then iterated. To effectively improve the speed of DCNP and avoid dynamically updating information of nodes on a large scale, a checking mechanism is applied to allow DCNP update information of nodes only when the occurrence may have repetition. The space and time complexities of DCNP are also analyzed. Experimental results show that DCNP has better performance than other competitive algorithms.

Key words: general gap; pattern matching; one-off condition; nettree

模式匹配是许多问题的研究基础,如信息检索[1]、数据存储与管理[2]以及网络安全[3]等多方面.近年来,人们更加关注带通配符的模式匹配问题,特别是具有多个可变间隙的模式 P 可以描述为

$$p_0[a_0,b_0]p_1\cdots[a_{j-1},b_{j-1}]p_j\cdots[a_{m-1},b_{m-2}]p_{m-1},$$

其中,a_{j-1} 和 b_{j-1} 是指 p_{j-1} 和 p_j 之间可以通配的最小和最大间隙.由于允许用户指定不同长度的约束,具有更高的灵活性,这使得其在现实世界中有很多重要的应用.在生物信息学领域,TATA 通常出现在 CAATCT 之后,间隔 30~50 个字符[4],因此,利用带通配符的模式匹配技术可以在 DNA 序列中提取很多有价值的信息;在音乐信息检索方面,Crochemore 等人[5]提出了(δ,α)-近似匹配,并指出,这种匹配方法可以应用于音乐信息检索等领域;在序列模式挖掘中,具有间隙约束的模式匹配技术应用更加广泛.Zhang 等人[6]提出了周期间隙的序列模式挖掘问题,之后,文献[7]采用模式匹配技术对此问题进行了研究,提高了原有问题的挖掘速度;Ding 等人[8]采用 INSgrow 的模式匹配算法,对无重叠约束的序列模式挖掘问题进行了研究;鉴于在序列模式挖掘中,通常每个位置的字符只能使用一次,为此,Chen 等人[9]最早提出了具有一次性条件的模式匹配问题,文献[10]采用网树结构构造了启发式算法 SBO,提高了问题的求解质量;之后,Guo 等人[11]建立了更为有效的 WOW 算法.Wu 等人[12]对具有一次性条件的序列模式挖掘问题进行了研究,其核心技术依然是模式匹配技术.Lam 等人[13]也是在一次性条件下对压缩的序列模式的挖掘进行了研究.由于序列的有序性,因此,采用非负间隙的模式匹配技术无疑禁止了子模式串次序颠倒现象的发生.为此,Fredriksson 和 Grabowski[14,15]开展了间隙可以为负的一般间隙模式匹配研究,但是其研究仅仅关心在序列串中的何位置能够产生出现,而不关心具体如何产生的出现,因此,这种匹配方式被称为宽松匹配.文献[16]开展了一般间隙的严格匹配,在其研究中,具体衡量了每个模式子串是如何匹配序列中的字符的.但是在其研究中允许任意位置字符多次使用,其算法的计算复杂性为 P.若采用非负间隙进行模式挖掘,当序列数据库发生扰动时就会漏掉一些有价值的信息,因此,利用一般间隙进行模式挖掘更能够发现更多有用的信息.

由于模式匹配问题是序列模式挖掘的重要基础,因此,本文对具有一次性条件的一般间隙严格模式匹配问题(strict pattern matching with general gaps and one-off condition,简称 SPANGOO)进行研究.该问题具有如下 4 个特点:各个子模式串之间的间隙可以为负;序列串中任意位置的字符最多能使用一次;是一种严格的模式匹配;是一种精确的模式匹配.对上述特点举例进行说明.

例 1:给定字符串序列 $S=s_0s_1s_2s_3s_4s_5$=abaaba,模式串 $P=p_0[a_0,b_0]p_1[a_1,b_1]p_2$=a[0,1]b[0,1]a.

子模式 a[0,1]b 的含义是:在字符 a 和字符 b 之间可以通配 0 到 1 个字符,0 称为最小间隙约束,1 称为最大间隙约束.即,若 $p_0=s_i,p_1=s_j$,且 $0\leqslant j-i-1\leqslant 1$,则表示位置为 i 的字符 a 和位置为 j 的字符 b 满足间隙约束.

例 1 中的子模式串之间的最小间隙约束均大于等于 0,我们称这样的间隙约束为非负间隙约束;若最大间隙和最小间隙至少有一个小于 0,我们称这样的间隙约束为一般间隙约束.

例 1 中满足间隙约束的所有出现为{⟨0,1,2⟩,⟨0,1,3⟩,⟨2,4,5⟩,⟨3,4,5⟩}.若规定字符串中任意位置字符只能在所有出现中最多使用一次,则称为一次性条件约束[9].在一次性条件约束下,例 1 的最大出现数为 2,分别为{⟨0,1,2⟩,⟨3,4,5⟩}或者{⟨0,1,3⟩,⟨2,4,5⟩}.若允许任意位置字符多次出现,但是不允许两个出现中的相同位置字符一样,则称为无重叠约束,如出现⟨0,1,2⟩和出现⟨0,1,3⟩就不满足无重叠约束条件.⟨0,1,2⟩和⟨2,4,5⟩虽然两个出现中都有 2,但是出现在不同的位置,所以满足无重叠约束[8].

文献[16]最早对模式匹配的类型进行了分类,分为严格模式匹配[9−11,16]和宽松模式匹配[14,15].其中,严格模式匹配是指用一组值来描述一个出现,这组值由模式串中对应的字符在满足间隙约束的情况下在序列串中的

位置组成的.若按严格模式匹配,例 1 中有 4 个出现,分别是⟨0,1,2⟩,⟨0,1,3⟩,⟨2,4,5⟩,⟨3,4,5⟩.而宽松模式匹配是采用模式串中最后一个位置的字符在序列串中的位置来表示出现,若按照宽松模式匹配,例 1 中有 3 个出现,分别为 2,3,5.

在匹配类型方面,有精确模式匹配与近似模式匹配之分.精确模式匹配是指出现中 p_j 和 s_i 必须相同,如文献[9−11,16]都是对精确模式匹配进行的研究.与精确模式匹配相对应的是近似模式匹配,近似模式匹配按照距离计算公式的不同分为 Hamming 距离下的近似匹配[17]、编辑距离下的近似匹配[4]和 δ近似匹配[14,15]等.而本文的研究属于精确匹配.本文贡献如下:

(1) 提出了具有一般间隙的一次性条件下的严格模式匹配问题 SPANGOO.

在具有间隙约束的模式串中允许子模式串之间的间隙为负值,并且加入了一次性条件的约束,是一种序列串中任意位置字符最多使用一次的严格精确模式匹配.

(2) 提出了动态更新结点信息的启发式求解算法(dynamically changing node property,简称 DCNP).

DCNP 算法采用 DSGSP 策略和 SRMP-Gen 策略相结合的方法选择最优出现.在 DSGSP 策略中采用了动态更新结点属性的策略,通过动态更新共同祖先集的树根路径数、树叶路径数及位置相关数来消除内部重复现象.由于动态更新结点属性会大幅度提高算法的复杂性,因而提出 Checking 机制对模式串进行预处理,来判断出现内部是否会产生内部重复的现象.若不会产生重复,则不触发动态更新,从而提高了 DCNP 算法的运行效率.理论分析指出,DCNP 算法的空间复杂度和时间复杂度分别为 $O(m\times n\times MaxGap)$和 $O(n\times MaxGap\times(m^2+n))$.这里,$m$,$n$ 和 $MaxGap$ 分别是模式串长度、序列串长度以及模式串的最大间隙($MaxGap=\max(b_i-a_i-1)$).

(3) 对 DCNP 算法的求解性能进行了对比测试.

本文在大量真实生物数据上,与多种相关算法的改进算法进行了对比性实验,通过实验结果验证了本文算法的有效性,并对实验结果进行了分析.

本文第 1 节对相关工作进行总结.第 2 节给出 SPANGOO 问题的定义.第 3 节介绍网树的定义和性质,在此基础上给出 DCNP 算法,分析算法的空间复杂度和时间复杂度,并且通过实例来说明动态更新结点属性的过程.第 4 节通过大量对比性实验测试 DCNP 算法的性能.第 5 节得出本文的结论.

1 相关工作

目前,具有间隙约束的模式匹配问题可以从以下 6 个方面进行划分:是否具有多个可变间隙、一般间隙模式串与否、宽松模式匹配还是严格模式匹配、精确匹配还是近似匹配、是否具有一次性条件、是否具有长度约束.

Manber 等人[18]最早开展了具有可变间隙约束的模式匹配问题的研究,在其研究中,将通配符描述为[0,g]形式,其中,g 表示一个可变长度.但是,其研究的是单个可变间隙约束的模式匹配.伴随研究的不断深入发展,模式匹配技术由之前的单个可变间隙约束逐步发展为多个可变间隙约束,如 Navarro 和 Raffinot[19]在 2003 年运用具有多个可变间隙约束的模式匹配技术进行蛋白质查找.具有多个可变间隙的模式匹配技术具有更高的灵活性,在现实世界中更具实用意义.

Fredriksson[14]和 Grabowski[15]先后在 2006 年和 2008 年对一般间隙模式匹配进行了研究,并在音乐检索和序列蛋白质匹配等问题中进行了应用.文献[16]利用子网树对具有长度约束的一般间隙下的严格模式匹配进行了研究.一般间隙约束的模式匹配允许子模式串之间的间隙约束为负,应用在模式挖掘中能够发现更多有价值的信息.

Bille 等人[20]不但在 2010 年对具有多个可变非负间隙约束的宽松模式匹配问题进行了研究,而且在 2012 年分别对具有多个可变非负间隙约束的宽松匹配和严格匹配进行了研究[21].此外,在音乐信息检索中多使用宽松模式匹配[5],而在生物信息计算中多采用严格模式匹配[9].

由于在序列模式挖掘中很多情况下任意位置字符通常都使用 1 次,为此,Chen 等人[9]提出了具有一次性条件的模式匹配问题,利用一次性条件过滤掉大量冗余信息,提高匹配效率.Lam 等人[13]在一次性条件下对压缩的

序列模式挖掘问题进行了研究,表 1 给出了几种具有间隙约束的模式匹配对比.

Table 1 Comparison of pattern matching with gap constraints

表 1 几种具有间隙约束的模式匹配对比

相关工作	间隙个数及类型	间隙类型	宽松/严格匹配	匹配类型	出现约束	长度约束
Manber 和 Baeza-Yates[18]	一个可变间隙	非负间隙	严格	精确匹配	否	无
Bille 等人[20]	多个可变间隙	非负间隙	宽松	精确匹配	–②	–
Bille 等人[21]	多个可变间隙	非负间隙	宽松/严格①	精确匹配	–/无	–
Fredriksson 和 Grabowski[14,15]	多个可变间隙	一般间隙	宽松	δ近似	无	无
Guo 等人[11]	多个可变间隙	非负间隙	严格	精确匹配	一次性约束	有
He 等人[17]	多个可变间隙	非负间隙	严格	Hamming 近似	一次性约束	有
Ding 等人[8]	多个可变间隙	非负间隙	严格	精确匹配	无重叠约束	有
文献[16]	多个可变间隙	一般间隙	严格	精确匹配	无约束	有
本文	多个可变间隙	一般间隙	严格	精确匹配	一次性条件	无

注:① 该文献设计了两种算法,分别对宽松模式匹配和严格模式匹配进行了研究.

② 宽松模式匹配下通常不考虑一次性条件和长度约束问题,因此,本文以“–”表示不予考虑.

由表 1 可以看出:本文与文献[16]的主要区别在于,文献[16]研究不具有一次性条件的模式匹配问题,而本文是对具有一次性条件的模式匹配问题进行研究.文献[16]中,只需要考虑各个子模式间的间隙约束与长度约束即可,并且该问题计算复杂性为 P;而本文不仅要考虑子模式间的间隙约束,还要考虑出现是否满足一次性条件.本文理论证明了一般间隙及一次性条件下模式匹配问题的计算复杂性为 NP-Hard(证明详见第 2 节的定理 1),因此求解难度更大.为此,本文构建合理的策略,形成了启发式算法 DCNP,使得在满足一次性条件的情况下找出更多的出现.

本文的研究工作与文献[11]的区别在于:文献[11]是对非负间隙的严格模式匹配进行研究,而本文是对间隙约束可以为负的一般间隙模式匹配问题进行研究.在负间隙的作用下,匹配过程中会出现字符回溯的情况,使得子模式串 p_{j+1} 可能出现在子模式串 p_j 之前,从而使得一个出现中可能会出现位置重复的现象.因此,一般间隙下的一次性条件的模式匹配不仅要避免出现与出现之间的位置重复,而且还要避免出现内部存在重复的情况.在这个过程中,我们要判定内部重复在何处产生以及内部重复涉及的范围,并且构造合理的策略有效地消除内部重复.因而,本文研究的问题是比非负间隙下一次性条件的模式匹配更为复杂的研究.在应用方面,文献[12,22]都是在非负间隙下基于一次性条件的序列模式挖掘,但这样的序列模式挖掘潜在地约定了事件发生次序.尽管 Zhang 等人[23]的伴随关系挖掘可以解决事件发生的次序问题,但是这类研究不具有间隙约束.而在一般间隙下的基于一次性条件的序列模式挖掘将既可以解决事件发生次序问题,也同时具有了间隙约束.而模式匹配技术是序列模式挖掘中核心任务,因此,一般间隙的模式匹配要比非负间隙的模式匹配更具有实际意义.

2 SPANGOO 问题定义

2.1 问题定义

定义 1. 序列串 $S=s_0s_1\ldots s_i\ldots s_{n-1}$,这里,$n$ 表示 S 的长度,$s_i\in\Sigma$代表一种符号集.对于不同的应用,Σ可以是不同的符号集合,例如在 DNA 序列中,Σ是由{A,T,G,C}构成的.

定义 2. 具有间隙约束的模式串 P 可以表示为 $p_0[a_0,b_0]p_1\ldots[a_j,b_j]p_{j+1}\ldots[a_{m-2},b_{m-2}]p_{m-1}$,这里,$m$ 表示模式串 P 的长度,$p_j\in\Sigma$,a_j 和 b_j 是给定整数值,代表子模式 p_j 和 p_{j+1} 之间可以匹配通配符的最小和最大长度,这里,$a_j\leqslant b_j$.如果所有的 a_j 和 b_j 均大于等于 0,则称为非负间隙约束,间隙约束都为非负的模式串称为非负间隙模式串;若 a_j 和 b_j 至少有一个小于 0,则称为一般间隙约束,包含一般间隙约束的模式串称为一般间隙模式串.$MaxGap=\max(b_j-a_j+1)$称为模式串 P 的最大间距.

定义 3. 给定模式串 $P=p_0[a_0,b_0]p_1\ldots[a_j,b_j]p_{j+1}\ldots[a_{m-2},b_{m-2}]p_{m-1}$,若存在 $p_i=p_j$,其中 $i\neq j$,$0\leqslant i\leqslant m-1$,$0\leqslant j\leqslant m-1$,则称模式串为包含重复字符的模式串;否则称为不包含重复字符的模式串.

定义 4. 若一个位置索引序列 $I=\langle i_0,\ldots,i_j,\ldots,i_{m-1}\rangle$,其中,$0\leqslant i_j\leqslant n-1$,$0\leqslant j\leqslant m-1$,且 $i_j\neq i_{j-1}$,服从 $s_{i_j}=p_j$ 且

$$\begin{cases} a_{j-1} \leqslant i_j - i_{j-1} - 1 \leqslant b_{j-1}, & \text{if } i_{j-1} < i_j \\ a_{j-1} \leqslant i_j - i_{j-1} \leqslant b_{j-1}, & \text{if } i_{j-1} > i_j \end{cases},$$

则称 I 是模式串 P 在模式串 S 中满足间隙约束的一个出现.

由于一次性条件下一般间隙的模式匹配问题的特殊性,即,在一般间隙模式串中,子模式 p_{j+1} 可能出现在子模式 p_j 之前,如果存在 $p_{j+k+1}=p_j$(这里,$0<k\leqslant m-2$),就有可能产生 p_{j+k+1} 与 p_j 匹配相同位置字符的出现.显然,这样的出现并不是我们想要的.为了将这样的出现加以区分,我们给出如下定义:

定义 5. 给定一个出现 $I=\langle i_0,...,i_j,...,i_{m-1}\rangle$,如果存在 $i_k=i_g$ 的情况($k\neq g$,$0\leqslant k\leqslant m-1$,$0\leqslant g\leqslant m-1$),则称为内部有重复的出现;否则,称为内部无重复的出现.

例 2:给定序列串 $S=s_0s_1s_2s_3$=agag,模式串 $P=p_0[a_0,b_0]p_1[a_1,b_1]p_2$=g[−1,1]a[0,2]g.

由于 s_1=g,因此 p_0=g 可以与 s_1 匹配.在间隙[−1,1]的作用下,p_1=a 可以与 s_0 匹配;而在间隙[0,2]的作用下,p_2 不但可以与 s_3 匹配,而且可以与 s_1 匹配.若与 s_1 匹配就形成了出现⟨1,0,1⟩.这种情况就属于内部有重复的出现.

定义 6. 令集合 $OCC(S,P)$代表模式串 P 在序列串 S 中的所有出现,集合 $OCC(S,P)$的长度用$|OCC(S,P)|$来表示.

定义 7. 给定两个内部无重复的出现 $C=\langle c_0,...,c_k,...,c_{m-1}\rangle$和 $D=\langle d_0,...,d_j,...,d_{m-1}\rangle$,如果存在 $c_k=d_j$($0\leqslant k\leqslant m-1$ 且 $0\leqslant j\leqslant m-1$),则称出现 C 和出现 D 不满足一次性条件;否则,称出现 C 和出现 D 满足一次性条件.

定义 8. 若 $OCC(S,P)$的子集 $OCC_1(S,P)$中任何两个内部无重复的出现都满足一次性条件,则称子集 $OCC_1(S,P)$是具有一次性条件的一般间隙的模式匹配.具有一般间隙和一次性条件的严格模式匹配(SPANGOO)的判定问题是给定序列串 S 和模式串 P,判定模式 P 在序列 S 是否存在 t 个满足一次性条件的出现.而 SPANGOO 的优化问题在给定序列串 S 和模式串 P 中,最多存在多少个满足一次性条件的出现.本文的任务是求解 $OCC(S,P)$中任何两个内部无重复的出现都满足一次性条件的最大子集 $OCC_{MAX}(S,P)$.

例 3:给定序列串 $S=s_0s_1s_2s_3s_4s_5s_6s_7$=baaccbaa 和模式串 $P=p_0[a_0,b_0]p_1[a_1,b_1]p_2$=a[−1,2]b[−1,2]a.

根据给定条件可知,模式串 P 在序列串 S 中的所有出现 $OCC(S,P)$为{⟨1,0,1⟩,⟨1,0,2⟩,⟨2,5,7⟩,⟨2,5,6⟩,⟨6,5,6⟩,⟨6,5,7⟩},可知$|OCC(S,P)|$为 6,依据定义 7 和定义 8 可知,具有一次性条件的一般间隙的最大模式匹配问题的解 $OCC_{MAX}(S,P)$是{⟨1,0,2⟩,⟨6,5,7⟩}.

由例 3 可以看出,一般间隙严格模式匹配(SPANGOO)是比文献[10]的 MPMGOOC 问题更为复杂的问题.这是因为在一般间隙的作用下,会存在出现内部有重复的现象,如⟨1,0,1⟩和⟨6,5,6⟩这两个出现都是出现内部有重复的出现.而 MPMGOOC 问题中都是非负间隙,所以不会产生诸如⟨1,0,1⟩的出现.由于在 SPANGOO 中存在⟨1,0,1⟩的出现,在求解过程中不仅要增加一个是否有内部重复判定,更为重要的是:位置的重复使用会放大这些位置的实际作用,进而影响到问题的求解.为了消除这样的影响,在求解过程中需要注意如下 3 件事情:

(I) 如何通过模式串预先判定是否会出现内部重复,如果会出现内部重复,具体是哪个子模式产生的.在例 1 中是不会产生重复的,因为模式串是非负间隙的;但是即使是包含负间隙的模式串,也不一定会产生出现内部有重复的现象,例如模式串 a[−1,2]b[1,2]c 中,因为模式串是不包含重复字符的,所以不会产生内部重复;若模式串包含重复字符并且包含负间隙,也不一定会产生内部重复的现象,例如模式串 a[−1,2]b[1,2]a,尽管该模式中字符 a 有 2 个,假定 b 可以匹配的字符位置为 k,第 1 个 a 在子模式"a[−1,2]b"的作用下,a 可以匹配的字符位置最大为 k+1,但是第 2 个 a 在子模式"b[1,2]a"的作用下,a 可以匹配的字符位置最小为 k+2,这样就不会产生有重复的出现.因此,需要有效地依据模式串判定是否会产生内部重复以及内部重复产生的位置.

(II) 如果通过模式串预先判定会有出现内部产生重复的问题,需要在相应位置建立检查,以避免这样的出现.例如在例 3 中,在找到子出现⟨1,0⟩时,存在两种选择⟨1,0,1⟩和⟨1,0,2⟩.此时,需要建立检查机制,最终选择⟨1,0,2⟩.而在找到子出现⟨1⟩时,无需对子出现⟨1,0⟩开展检查机制,因为在子模式 p_1 位置处,就不会产生有重复的现象.因此需要有效的机制,仅在可能产生重复的位置建立检查.

(III) 如何判定重复可能产生的范围并有效地消除被放大的影响,这也是 SPANGOO 处理中最为复杂的过程.依然以例 3 为例进行说明,在选择了⟨1⟩以后,由于⟨1⟩存在作用被放大的现象,这需要消除⟨1⟩的影响,如果在整

个序列上消除,就会产生被扩大化的现象,进而导致算法性能的下降,因为⟨1⟩仅仅作用在局部位置,仅仅会在⟨1,0,1⟩和⟨1,0,2⟩两个出现中产生影响.那么,如何高效地找出⟨1⟩可能影响的区域,并在该区域内部消除影响呢?

2.2 计算复杂性分析

定义 9(迭代洗牌). 给定一个字符集Σ及两个均属于Σ^*且长度为 k 的字符串 $V=v_0v_1\ldots v_{k-1}$ 和 $W=w_0w_1\ldots w_{k-1}$,用"$V\Theta W$"表示洗牌 V 和 W,$V\Theta W$ 可以定义为:$V\Theta W=\{v_0w_0v_1w_1\ldots v_{k-1}w_{k-1}|$对于任意 $0\leqslant j<k$ 都有 $v_j\in\Sigma$且 $w_j\in\Sigma\}$.迭代洗牌 X 就是其可以表示为$\varepsilon\cup\{V\}\cup\{V\Theta V\}\cup\{V\Theta V\Theta V\}\cup\ldots$这里,$\varepsilon$表示空集.迭代洗牌的判定问题是:给定一个字符串 X,判定其是否由字符串 V 迭代洗牌构成的[24].

引理 1. 判定一个字符串 X 是否由字符 V 迭代洗牌构成的,其计算复杂性为 NP-Complete 问题.

证明:文献[24]给出了这个问题的计算复杂性证明. □

例 4:给定字符串 X=ataatt 和字符串 V=at.

X 是对字符串 V 迭代洗牌的结果,因为 $X\in\{V\Theta V\Theta V\}$;但是 X'=atatta 不是对 V 迭代洗牌的结果,因为子串"atta"不是对 V 迭代洗牌的结果.

引理 2. 具有非负间隙及一次性条件约束的模式匹配的判定问题计算复杂性为 NP-Complete 问题.

证明:显然,判定序列串 S 是由 t 次对模式串 P 迭代洗牌和判定模式串 P 在序列串 S 中出现次数是否为 t 个的计算复杂性是一致的.这里,模式串 P 是非负间隙的[9]. □

定理 1. 具有一般间隙及一次性条件约束的模式匹配的判定问题计算复杂性为 NP-Complete 问题.

证明:由于非负间隙模式可以看成是一般间隙模式的特殊形式,因此,一般间隙及一次性条件约束的模式匹配问题是比非负间隙及一次性条件约束的模式匹配问题更具一般性的问题.故,具有一般间隙及一次性条件约束的模式匹配的判定问题计算复杂性为 NP-Complete 问题. □

由于具有一般间隙及一次性条件约束的模式匹配的判定问题计算复杂性为 NP-Complete 问题,因此,具有一般间隙及一次性条件约束的模式匹配这个优化问题的计算复杂性为 NP-Hard 问题.

3 网树与 DCNP 算法

3.1 网树的概念

文献[25]最早提出了网树的概念,为了求解 SPANGOO 问题,本节在网树定义的基础上给出了网树的一些新的概念和性质,并对这些概念和性质进行了解释.

定义 10. 网树具有结点、树根、叶子、层、双亲、孩子等概念的树的拓展结构,其与树结构的区别在于:一棵网树可以有多个根结点;除根结点之外,网树的其他结点可以有多个双亲结点;网树的不同层上结点的标签可以相同,并用 n_j^i 表示第 j 层的结点 i.由于一个结点具有多个双亲,因此,一个结点到达根结点的路径数为多个.

定义 11. 文献[10]最早给出了树根路径数 RPN、树叶路径数 LPN 及树根-树叶路径数的定义 RLPN,这里再做一个简单的说明.从网树第 1 层所有的树根结点到结点 n_j^i 的路径数称为结点 n_j^i 的树根路径数 RPN(root path number),用 $N_r(n_j^i)$ 来表示,第 1 层结点的树根路径数为 1;从网树第 m 层所有叶子结点到结点 n_j^i 的路径数称为结点 n_j^i 的树叶路径数 LPN(leaf path number),用 $N_l(n_j^i)$ 来表示,第 m 层叶子结点的树叶路径数为 1,其余层叶子结点的树叶路径数为 0,这里,m 是网树的深度;从网树所有根结点到第 m 层所有叶子结点中的所有路径中,包含结点 n_j^i 的路径数称为该结点的树根-叶子路径数 RLPN(root-leaf path number),用 $N_f(n_j^i)$ 表示.

性质 1. 结点 n_j^i 的树根路径数是其所有双亲结点的树根路径数之和;结点 n_j^i 的叶子路径数是其所有孩子结点的叶子路径数之和;结点 n_j^i 的树根-叶子路径数是其树根路径数与其树叶路径数之积[10],即

$$N_f(n_j^i)=N_r(n_j^i)\times N_l(n_j^i).$$

性质 2. 位置 i 的位置相关数 $RP(i)$是网树中所有结点名称是 i 的结点的树根-叶子路径数之和[10],即

$$RP(i)=\sum_{j=1}^{m}N_f(n_j^i).$$

这里,m 是网树的深度.

定义 12. 结点 n_b^c 到达根结点的所有路径上的所有结点的集合称作结点 n_b^c 的祖先集,用 $A(n_b^c)$ 表示.祖先集中任何一个结点都称作结点 n_b^c 的祖先,结点 n_b^c 是自身的一个祖先结点.

定义 13. 给定一个结点集合 $D=\{d_0,d_1,\ldots,d_{l-1}\}$,集合 D 中所有元素的祖先集的交集称为集合 D 的共同祖先集,用 $C(D)$表示.位置 i 在集合 D 的共同祖先集中的所有树根路径数称为位置 i 在集合 D 的路径分支数,用 $pb(i,D)$表示[10].

定义 14. 一条从树根结点到达叶子结点的路径称为树根叶子路径.网树在移除一条树根叶子路径后,在新的网树上第 m 层所有叶子结点的树根路径数之和称为网树的剩余出现数.

定义 15. 我们将结点的双亲结点中位置相关数较小的结点作为该结点的最优双亲结点 OPP,若两个双亲结点的位置相关数相同,则选择在已有路径的共同祖先集中路径分支数最大的双亲结点作为当前结点的 OPP.根结点没有 OPP.

3.2 DCNP算法分析及设计

3.2.1 建立一般间隙网树的方法

由于 DCNP 算法是基于网树结构设计的,这里先给出建立一般间隙网树的方法.建立非负间隙网树的规则最早由文献[25]给出,在非负间隙条件下,只需要扫描一遍字符串就可以建立网树.但是在一般间隙下,在存在字符回溯的情况下,若仍然按照非负间隙的规则建立网树,就会漏掉一部分结点和父子关系,因此,一般间隙下应该逐层建立网树.这里重新给出一般间隙情况下建立网树的方法.首先从网树的第 1 层开始建立,逐个扫描字符串 S 中的字符 s_i,若 $p_0=s_j(0\leqslant j<n)$,则在第 1 层建立结点 n_1^j;建立第 $i(1<i\leqslant m)$层结点时,逐个扫描字符串 S 中的字符 s_i,若 $p_{i-1}=s_j(0\leqslant j<n)$,并且若这个结点 j 与第 $i-1$ 层某结点 n_{i-1}^e $(e\neq j)$满足局部间隙约束,则建立结点 n_i^j,并且在结点 n_{i-1}^e 与 n_i^j 间建立父子关系,再检查结点 n_i^j 与 $i-1$ 层其他结点是否满足间隙约束,若满足也建立父子关系.具体算法如下:

算法 1. CreNetTree.

输入:字符串 S,模式串 P.

输出:网树 NetTree.

```
1.  for i=1 to m step 1
2.      start=1;
3.      for j=0 to n–1 step 1
4.          if (p_{i–1}=s_j) then
5.              if (i=1) then
6.                  建立结点 n_1^j ;
7.              else
8.                  for k=start to  第 i–1 层的结点个数
9.                      if (j 与第 i–1 层的第 k 个结点的间隙>b_{i–2}) then
10.                         start++;
11.                         continue;
12.                     end if
13.                     if (j 与第 i–1 层的第 k 个结点的间隙<a_{i–2}) then break;
14.                     if (不存在结点 n_i^j ) then  建立结点 n_i^j ;
15.                     在结点 n_i^j 与第 i–1 层的第 k 个结点之间建立父子关系;
```

```
16.                start++;
17.              end for
18.            end if
19.          end if
20.      end for
21. end for
```

3.2.2 出现内部重复检测机制

引理 3. 在一般间隙模式串中,在模式串长度大于等于 3 的情况下,如果存在 $p_j=p_{j+k+1}$ 的情况,其中,$0\leqslant j\leqslant m-3$,$0<k\leqslant m-2$,并且满足 $\sum_{t=0}^{k}a_{j+t}+k\leqslant 0$,则可能会在 p_j 处产生内部有重复的出现;否则不会产生内部有重复的出现.

证明:给定模式串 $P=p_0[a_0,b_0]p_1\ldots[a_{j-1},b_{j-1}]p_j\ldots[a_{m-2},b_{m-2}]p_{m-1}$,设 p_j 的出现位置为 h,p_j 到 p_{j+k+1} 之间的间隙约束为 $[a_j,b_j]\ldots[a_{j+k},b_{j+k}]$,则 p_{j+k+1} 的最小位置为 $\sum_{t=0}^{k}a_{j+t}+k+h$($\sum_{t=0}^{k}a_{j+t}\leqslant 0$ 时)或 $\sum_{t=0}^{k}a_{j+t}+k+h+1$($\sum_{t=0}^{k}a_{j+t}>0$ 时).显然,$\sum_{t=0}^{k}a_{j+t}>0$ 时,p_{j+k+1} 只能出现在 p_j 之后,不会发生重复的可能;若 $\sum_{t=0}^{k}a_{j+t}\leqslant 0$,则 p_{j+k+1} 的最小位置为 $\sum_{t=0}^{k}a_{j+t}+k+h$;当 $\sum_{t=0}^{k}a_{j+t}+k\leqslant 0$ 成立时,则 p_{j+k+1} 的最小出现位置会小于等于 h,所以 p_j 会有和 p_{j+k+1} 重合的可能.因此,若 $p_j=p_{j+k+1}$ 并且满足 $\sum_{t=0}^{k}a_{j+t}+k<=0$,则在 p_j 处可能会产生内部有重复的出现.若 $\sum_{t=0}^{k}a_{j+t}+k>0$,则 p_{j+k+1} 只会出现在 p_j 之后,p_j 不可能会和 p_{j+k+1} 产生重复. □

因算法本身采用网树结构,因此查找出现时,是从最后一层叶子结点依次往前查找.因此在判断哪些子模式串会产生重复时,采用从后往前的查找方式.并且增加一个布尔型标志数组 *needrep* 来标示某子模式是否会和它之后的某个子模式存在重复的可能,*needrep* 的初始值为 false.当模式串为非负间隙模式或者为不包含重复字符时,不会出现内部重复的情况;否则,从 p_{m-3} 依次往前查找,检查子模式 p_{m-3} 是否会和它之后的模式发生重复.

若找到 $p_{m-3+k+1}=p_{m-3}$,并且满足 $\sum_{t=0}^{k}a_{j+t}+k\leqslant 0$,则将 p_{m-3} 的 *needrep* 设置为 true;若不存在这样的模式,则 p_{m-3} 的 *needrep* 值保持 false 不变,继续找 p_{m-2} 是否会和它之后的模式发生重复.依照上述过程检查,直到检查完 p_0 为止.具体算法给出如下:

算法 2. Checking.

输入:模式串 *P*.

输出:标志数组 *needrep*.

```
1.  初始化标志数组 needrep 使得其全为 false;
2.  if (模式串中间隙都为非负) then return needrep
3.  if (P 不包含重复字符) then return needrep
4.  else
5.     for i=m−3 downto 0 step−1
6.        for j=i+2 to m−1 step 1
7.           if (pi=p_j and  sum_{t=0}^{j−i−1} a_{j+t} +j−i−1<=0) then
8.              needrep[i]=true;
9.              break;
10.          end if
11.       end for
12.    end for
13. end if
14. return needrep
```

3.2.3 DSGSP 算法

DSGSP 算法与 MPMGOOC 问题[10]中的 SGSP 算法不同的是,在 SGSP 算法中,无需考虑出现内部重复的情况;但是在 DSGSP 算法中,由于负间隙的作用,我们要对可能产生重复的位置进行处理,并且要尽量缩小重复所影响的范围.我们通过动态更新已有路径中共同祖先集结点属性这一策略来解决这个问题,因此,DSGSP 算法的核心思想是:从第 m 叶子结点依次往上查找当前结点的 *OPP*,在查找之前先判定当前结点的上一层的 *needrep* 是否为 true:若为 true,则更新已有路径共同祖先集结点的属性;若为 false,则不用更新.然后,选择双亲结点中位置相关数较小的结点作为当前结点的 *OPP*,若位置相关数相同,则选择已有路径的共同祖先集中路径分支数较大的结点作为当前结点的 *OPP*,直到根结点.下面给出具体的 DSGSP 算法.

算法 3. DSGSP.

输入:标志数组 *needrep* 和叶子结点 *f*.

输出:出现 *A*.

```
1.  计算每个结点的树根路径数、树叶路径数、位置相关数;
2.  A[m−1]=f;
3.  for j=m−2 downto 0 step−1 do
4.    if (needrep[j]=true) then 更新已有路径共同祖先集结点的树根路径数、树叶路径数和位置相关数;
5.    计算每个结点 i 在已有路径 A 下的路径分支数 pb(i,A);
6.    n=A[j+1].number_of_parents;
7.    A[j]=A[j+1].parent[n−1];
8.    for k=n−2 downto 0 step−1 do
9.      if (RP(A[j]>RP(A[j+1].parent[k])) then A[j]=A[j+1].parent[k];
10.     if (RP(A[j])=RP(A[j+1].parent[k]) and (pb(A[j+1].parent[k],A)>=pb(A[j],A)))
        then A[j]=A[j+1].parent[k];
11.   end for
12. end for
13. return A
```

3.2.4 SRMP-Gen 算法

SRMP-Gen 选择当前结点的最右双亲结点作为 OPP,但是在一般间隙模式匹配中,SRMP-Gen 策略要避免⟨1,0,1⟩现象的出现.当选择第 i 层结点的 OPP 时,若 p_{i-2} 的 *needrep* 为 true,则需要判断当前最右结点是否和已有路径中的结点重复:若重复,则选择次右结点作为第 i 层的 OPP.若 p_{i-2} 的 *needrep* 为 false,则无需判断是否和已有路径中的结点重复.下面给出 SGMP-Gen 算法.

算法 4. SRMP-Gen.

输入:标志数组 *needrep* 和叶子结点 *f*.

输出:一个出现 *A*.

```
1.  A[m−1]=f;
2.  for j=m−2 downto 0 step−1 do
3.  n=A[j+1].number_of_parents;
4.    for k=n−1 downto 0 do
5.      if (used[A[j+1].parent[k]]=false) then
6.        if (needrep[j]=false) then   //不存在重复可能
7.          A[j]=A[j+1].parent[k];
8.          break;
9.        end if
```

```
10.          if (A[j+1].parent[k]存在已有路径之中) then   //存在重复可能,且存在已有路径中
11.             k−−;
12.             continue;
13.          end if
14.          A[j]=A[j+1].parent[k];   //虽然存在重复可能,但是不在已有路径中,赋值跳出
15.          break;
16.       end if
17.       k−−;   //标志位为 true,已经用过
18.    end for
19. end for
20. return A
```

3.2.5 DCNP 算法

将 DSGSP 算法和 SRMP-Gen 算法相结合,便得到了 DCNP 算法.DCNP 算法选择两者中剩余出现数大的出现作为最优出现.下面给出 DCNP 算法.

算法 5. DCNP.

输入:$P=p_0[a_0,b_0]p_1\ldots[a_{m-2},b_{m-2}]p_{m-1}$,$S=s_0s_1\ldots s_{n-1}$.

输出:解 C.

```
1.  调用算法 1,依据 P 和 S 建立一棵网树;
2.  调用算法 2,依据 P 建立标志数组 needrep;
3.  for k=第 m 层叶子结点数 downto 1 step−1
4.     if (used[第 k 个叶子结点]=true) then continue;
5.     B1=DSGSP(needrep,第 k 个叶子结点);   //算法 3
6.     y1=剩余出现数(B1);
7.     B2=SRMP-Gen(needrep,第 k 个叶子结点);   //算法 4
8.     y2=剩余出现数(B2);
9.     if (y1<y2) then B=B2 else B=B1;
10.    C=C∪B;
11. next k
12. return C
```

3.3 复杂度分析

通过以上算法分析可知,DCNP 算法的空间复杂度为 $O(m\times n\times MaxGap)$.这是因为网树最多有 m 层,每层最多有 n 个结点,每个结点最多有 $MaxGap$ 个双亲结点或者孩子结点,其中,m 为模式串的长度,n 为序列串长度,$MaxGap$ 为 P 的最大间隙.

首先分析 SRMP-Gen 算法的时间复杂度,可知 SRMP-Gen 算法的时间复杂度为 $O(m^2\times MaxGap)$;再分析 DSGSP 算法的时间复杂度,DSGSP 算法更新网树的树根路径数、树叶路径数的时间复杂度为 $O(m\times n\times MaxGap)$,更新位置相关数的时间复杂度为 $O(m\times n)$,DSGSP 更新祖先集的树根路径数、树叶路径数、位置相关数的时间复杂度为 $O(m^2\times MaxGap)$,计算 $pb(i,A)$的时间复杂度为 $O(m^2\times MaxGap)$,第 3 行~第 11 行的时间复杂度为 $O(m^3\times MaxGap)$,DSGSP 的时间复杂度为 $O(m\times MaxGap\times(m^2+n))$.

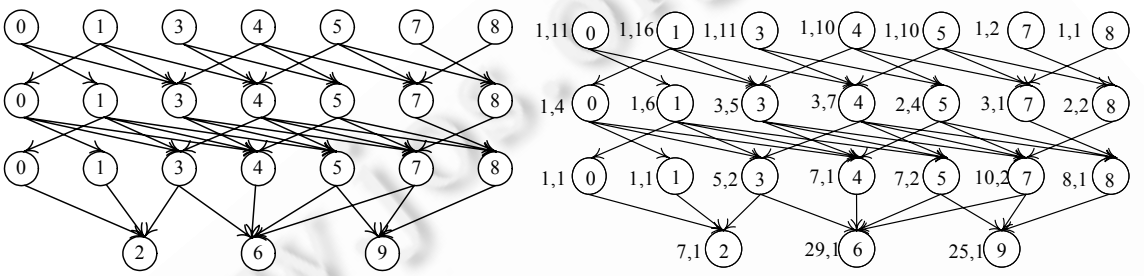
下面分析 DCNP 的时间复杂度.建立一棵网树的时间复杂度为 $O(m\times n\times MaxGap)$,DSGSP 算法的时间复杂度为 $O(m\times MaxGap\times(m^2+n))$,SRMP-Gen 的时间复杂度为 $O(m^2\times MaxGap)$,计算出现相关数的时间复杂度为 $O(m\times n\times MaxGap)$.当字符串长度为 n 模式串长度为 m 时,最多会有 n/m 个出现,因此,DCNP 算法的时间复杂度为 $O(n\times MaxGap\times(m^2+n))$.

3.4 运行实例

通过之前的 Checking 机制我们可以知道在哪个子模式处会产生内部重复,若已经知道某位置子模式存在重复的可能,即 p_i 的 *needrep* 标志位 true,在 DSGSP 算法中,为了消除重复并且缩小重复所影响的范围,我们更新已有路径的共同祖先集结点的树根路径数 RPN、树叶路径数 LPN 以及位置相关数 RP.将共同祖先集中存在于已有路径中结点的 RPN,LPN,RP 更新为 0,从而消除内部重复的情况.下面以实例来说明动态更新共同祖先集属性的过程,首先要将实例问题转化为一棵网树.

例 5:给定字符串序列 $S=s_0s_1s_2s_3s_4s_5s_6s_7s_8s_9$=aabaaabaab,模式串 $P=p_0[a_0,b_0]p_1[a_1,b_1]p_2[a_0,b_0]p_3$=a[−1,2]a[−1,3]a[−1,3]b,通过对模式串的预处理可知,p_0 和 *needrep* 标志位为 true,p_1,p_2 和 p_3 的 *needrep* 为 false.

(1) 依据一般间隙网树的建立规则,可将模式匹配问题转化为图 1(a),并且依据性质 1 计算树根路径数及树叶路径数如图 1(b)所示,树根路径数及树叶路径数在结点左方给出.



(a) 模式匹配问题转换的一棵等价网树图 (b) 网树及其 RPN,LPN

Fig.1 The nettree transformed from the pattern matching problem

图 1 模式匹配问题转化成网树

(2) 首先从最后一层的最后一个叶子结点开始往上查找双亲结点.选择结点 n_4^9 后,选择 n_4^9 的 OPP,由于 p_2 的 *needrep* 为 false,因此不对网树进行更新.结点 n_4^9 有 3 个双亲结点 n_3^5,n_3^7 和 n_3^8,根据图 2 计算得知:

$$RP(5)=N_f(n_3^5)+N_f(n_2^5)+N_f(n_1^5)=7\times2+2\times4+1\times10=32,$$

$$RP(7)=N_f(n_3^7)+N_f(n_2^7)+N_f(n_1^7)=10\times2+3\times1+1\times2=25,$$

$$RP(8)=N_f(n_3^8)+N_f(n_2^8)+N_f(n_1^8)=8\times1+2\times2+1\times1=13.$$

因此,选择位置相关数较小的结点 n_3^8 作为 OPP 结点.继续选择 n_3^8 的 OPP,由于 p_1 的 *needrep* 为 flase,因此不需要对已有路径的共同子集进行属性的更新;结点 n_3^8 有 3 个双亲结点 n_2^4,n_2^5 和 n_2^7,通过计算可知:

$$RP(4)=N_f(n_3^4)+N_f(n_2^4)+N_f(n_1^4)=7\times1+3\times7+1\times10=38,$$

$$RP(5)=32,$$

$$RP(7)=25.$$

因此,选择结点 n_2^7 作为 n_3^8 的 OPP.再找 n_2^7 的 OPP,因为 p_0 的 *needrep* 为 true,因此要更新已有路径 $\{n_2^7,n_3^8,n_4^9\}$ 的共同祖先集.

(3) 依据性质定义 12 和定义 13 可得已有路径 n_2^7,n_3^8 和 n_4^9 的共同祖先集如图 2(a)所示,共同祖先集的每层的开始位置和结束位置分别用 $a[i]$和 $b[i]$来表示,其中,i 表示所在层,curnode 表示当前结点.

(4) 更新共同祖先集中结点的属性.

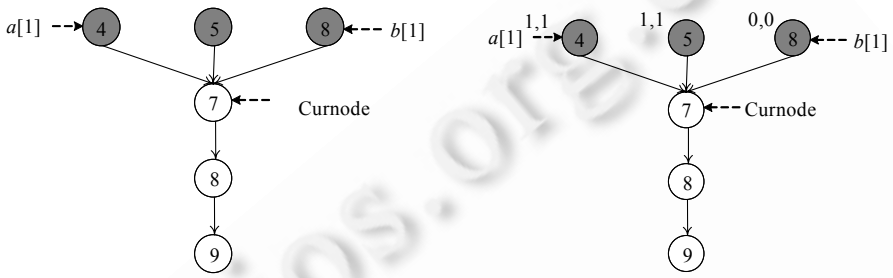
更新树根路径数从第 1 层开始,若该结点没有被使用,并且不存在已有路径中,则该结点树根路径数为 1,否则为 0;若不为第 1 层,该结点若没有被使用且不存在已有路径中,则该结点的树根路径数等于所有双亲结点的树根路径数之和,否则为 0.

更新树叶路径数从第 i−1 层开始,其中,i 为已有路径到达的层数.若该结点没有被使用并且不存在已有路径

中,则该结点的树叶路径数为 1,否则为 0;若不为第 i–1 层,该结点没有被使用且不存在已有路径中,则该结点的树叶路径数等于所有孩子结点的树叶路径数之和,否则为 0;

更新位置相关数,依据性质 2 计算共同祖先集中位置为 i 的位置相关数.

图 2(b)为更新共同祖先集属性后的树根路径数、树叶路径数的值.结点左方分别为为树根路径数和树叶路径数.因为位置 8 已经存在于已有路径中,所以更新后结点 n_1^8 的树叶路径数、树根路径数都为 0,以后进行选择都不会选择结点 8,从而有效地避免了内部重复情况的出现.综上所述,我们通过更新共同祖先集属性的方法可以有效地避免内部重复情况的发生,并且尽可能地缩小了更新结点属性的范围.



(a) 结点 n_2^7,n_3^8 和 n_4^9 的共同祖先集 (b) 共同祖先集更新后的树根路径数及树叶路径数

Fig.2 The common ancestor nodes of the current path

图 2 当前路径的共同祖先集

4 实验结果及分析

4.1 实验结果

本节采用真实生物数据来对比 SAIL-Gen,SGSP-Gen,SRMP-Gen,SBO-Gen 和 DCNP 算法的性能.其中,SAIL-Gen,SGSP-Gen,SRMP-Gen,SBO-Gen 分别为在一般间隙条件下对文献[9,10]的改进算法,这些算法的源程序可以在 http://wuc.scse.hebut.edu.cn/nettree/dcnp-2/处下载.实验运行的软、硬件环境为:酷睿双核 T4200,主频 2.0GHZ,内存 1.0GB,Windows 7 操作系统的笔记本.本文采用文献[10]的所使用的真实数据作为测试序列,这些真实 DNA 序列可以从美国国家生物计算信息中心的网站下载(http://www.ncbi.nlm.nih.gov/-genomes/FLU/SwineFlu.html).表 2 给出了的这 8 个序列串的特征.

Table 2 Sequences of real biological data

表 2 真实生物数据片段

序号	片段名称	位点	片段长度
*S*1	Segment 1	CY058563	2 286
*S*2	Segment 2	CY058562	2 299
*S*3	Segment 3	CY058561	2 169
*S*4	Segment 4	CY058556	1 720
*S*5	Segment 5	CY058559	1 516
*S*6	Segment 6	CY058558	1 418
*S*7	Segment 7	CY058557	982
*S*8	Segment 8	CY058560	844

为了测试在模式串中不存在重复字符和存在重复字符时算法的求解性能,我们选取了 *P*1~*P*4 模式为不存在重复字符的 4 种模式,*P*5~*P*9 模式为存在重复字符的 5 种模式,具体模式串在表 3 中给出.这 5 种算法在模式串中不存在重复字符的测试结果见表 4,在模式串中存在重复字符的测试结果见表 5,每种情况的最好解都用加粗方式显示.

Table 3 Patterns

表 3 模式串

序号	模式串
*P*1	a[−5,6]c[−4,7]g[−3,8]t
*P*2	c[−1,2]a[−2,3]t[−3,4]g
*P*3	g[1,2]t[0,3]c[−3,4]a
*P*4	t[−2,2]c[−2,2]a
*P*5	g[−1,5]t[0,6]a[−2,7]g[−3,9]t[−2,5]a[−4,9]g[−1,8]t[−2,9]a
*P*6	g[−1,9]t[−1,9]a[−1,9]g[−1,9]t[−1,9]a[−1,9]g[−1,9]t[−1,9]a[−1,9]g[−1,9]t
*P*7	g[−1,5]t[0,6]a[−2,7]g[−3,9]t[−2,5]a[−4,9]g[−1,8]t[−2,9]a[−1,9]g[−1,9]t
*P*8	g[−1,5]t[0,6]a[−2,7]g[−3,9]t[−2,5]a[−4,9]g[−1,8]t[−2,9]a[−1,9]g[−1,9]t
*P*9	t[−1,7]t[−1,7]a[−1,7]g[−1,7]t[−1,7]a[−1,7]g

Table 4 Number of occurrences of non-repetitive patterns in sequences *S*1~*S*8

表 4 无重复模式串在 *S*1~*S*8 序列串下的出现个数

模式	算法名称	解的数目/个							
		*S*1	*S*2	*S*3	*S*4	*S*5	*S*6	*S*7	*S*8
*P*1	SAIL-Gen	**299**	**302**	**306**	**223**	**199**	**198**	**150**	**118**
	SGSP-Gen	296	299	301	**223**	196	195	148	115
	SRMP-Gen	**299**	**302**	**306**	**223**	**199**	**198**	**150**	**118**
	SBO-Gen	**299**	300	**306**	**223**	**199**	**198**	149	117
	DCNP	**299**	300	**306**	**223**	**199**	**198**	149	117
*P*2	SAIL-Gen	**184**	**194**	**185**	**157**	**120**	**125**	**95**	**69**
	SGSP-Gen	180	192	184	153	119	124	**95**	**69**
	SRMP-Gen	**184**	**194**	**185**	**157**	**120**	**125**	**95**	**69**
	SBO-Gen	182	193	**185**	156	**120**	**125**	**95**	**69**
	DCNP	182	193	**185**	156	**120**	**125**	**95**	**69**
*P*3	SAIL-Gen	**97**	**117**	**103**	**79**	**72**	**73**	**58**	**36**
	SGSP-Gen	**97**	**117**	**103**	**79**	71	**73**	**58**	**36**
	SRMP-Gen	**97**	**117**	**103**	**79**	**72**	**73**	**58**	**36**
	SBO-Gen	**97**	**117**	**103**	**79**	**72**	**73**	**58**	**36**
	DCNP	**97**	**117**	**103**	**79**	**72**	**73**	**58**	**36**
*P*4	SAIL-Gen	**211**	**225**	**202**	**177**	**138**	**139**	**107**	**81**
	SGSP-Gen	209	223	201	**177**	**138**	**139**	**107**	**81**
	SRMP-Gen	**211**	**225**	**202**	**177**	**138**	**139**	**107**	**81**
	SBO-Gen	**211**	**225**	201	**177**	**138**	**139**	**107**	**81**
	DCNP	**211**	**225**	201	**177**	**138**	**139**	**107**	**81**

Table 5 Number of occurrences of repetitive patterns in sequences *S*1~*S*8

表 5 有重复模式在 *S*1~*S*8 序列串上的出现个数

模式	算法名称	解的数目/个							
		*S*1	*S*2	*S*3	*S*4	*S*5	*S*6	*S*7	*S*8
*P*5	SAIL-Gen	96	92	92	74	58	58	42	33
	SGSP-Gen	113	109	109	84	68	**73**	46	39
	SRMP-Gen	113	112	108	86	70	**73**	47	**45**
	SBO-Gen	114	**116**	**110**	87	71	71	**48**	44
	DCNP	**116**	115	**110**	**88**	**72**	72	**48**	44
*P*6	SAIL-Gen	76	76	76	60	49	49	32	31
	SGSP-Gen	79	82	80	64	53	52	34	32
	SRMP-Gen	83	82	81	64	53	54	34	31
	SBO-Gen	**86**	84	**84**	63	**54**	**55**	**35**	**34**
	DCNP	**86**	**86**	83	**65**	**54**	54	**35**	33
*P*7	SAIL-Gen	66	65	68	52	44	45	30	27
	SGSP-Gen	83	83	79	66	48	54	35	30
	SRMP-Gen	**88**	86	**83**	67	**54**	56	**37**	**34**
	SBO-Gen	86	**87**	82	**68**	53	**57**	**37**	33
	DCNP	86	**87**	**83**	67	**54**	**57**	**37**	33

Table 5 Number of occurrences of repetitive patterns in sequences *S*1~*S*8 (Contiuned)

表 5 有重复模式在 *S*1~*S*8 序列串上的出现个数(续)

模式	算法名称	解的数目/个							
		*S*1	*S*2	*S*3	*S*4	*S*5	*S*6	*S*7	*S*8
*P*8	SAIL-Gen	109	109	102	82	69	69	44	40
	SGSP-Gen	114	112	**110**	83	73	**74**	44	42
	SRMP-Gen	113	112	105	84	73	73	**46**	42
	SBO-Gen	**119**	113	107	**85**	74	**74**	45	**44**
	DCNP	117	**115**	108	83	**75**	**74**	**46**	**44**
*P*9	SAIL-Gen	127	119	118	90	81	75	59	47
	SGSP-Gen	137	127	121	90	84	**79**	**61**	**51**
	SRMP-Gen	135	126	127	**95**	85	78	57	50
	SBO-Gen	138	**128**	129	**95**	**87**	78	60	**51**
	DCNP	**139**	**128**	**130**	**95**	**87**	78	60	**51**

为了清晰地反映这 5 种算法在 *P*1~*P*9 模式在 8 种不同序列上的性能,表 6 给出了不同算法在不同模式下计算出现数的总和.

Table 6 Comparison of the number of occurrences of *P*1~*P*9 (No.)

表 6 5 种算法在 *P*1~*P*9 上的出现个数(个)

算法名称	模式串中不存在重复字符				模式串中存在重复字符				
	*P*1	*P*2	*P*3	*P*4	*P*5	*P*6	*P*7	*P*8	*P*9
SAIL-Gen	1 795	1 129	635	1 280	545	449	397	624	716
SGSP-Gen	1 773	1 116	634	1 275	641	476	478	652	750
SRMP-Gen	1 795	1 129	635	1 280	654	482	505	648	753
SBO-Gen	1 791	1 125	635	1 279	661	495	503	661	766
DCNP	1 791	1 125	635	1 279	665	496	504	662	768

为了清晰地反映这 5 种算法在 *P*1~*P*9 模式在 8 种不同序列上的时间性能,表 7 给出了不同算法在不同模式下的平均运行时间.

Table 7 Comparison of the running time on *P*1~*P*9 (ms)

表 7 5 种算法在 *P*1~*P*9 上的运行时间(ms)

算法名称	模式串中不存在重复字符				模式串中存在重复字符				
	*P*1	*P*2	*P*3	*P*4	*P*5	*P*6	*P*7	*P*8	*P*9
SAIL-Gen	335	215	121	211	172	213	191	189	179
SGSP-Gen	388	228	134	232	466	312	374	410	257
SRMP-Gen	357	232	129	244	210	236	246	219	203
SBO-Gen	413	240	135	243	527	363	433	468	285
DCNP	484	277	151	234	570	415	472	533	403

4.2 实验结果分析

(1) 在模式串中没有重复字符的情况下,SAIL-Gen 和 SRMP-Gen 属于完备性算法,尽管 DCNP 属于近似算法,但它的解极其接近完备解.

由表 3 可知,*P*1~*P*4 中各个字符均不同.而文献[9]证明了在此情况下,SAIL 算法是完备性算法.由于各个子模式彼此不同,因此即使在一般间隙下,也不会存在两个子模式匹配同一位置字符的现象.因此,SAIL-Gen 依然是完备性算法.SAIL-Gen 可以看成是从第 1 个叶子结点开始采用最左双亲策略进行求解,而 SRMP-Gen 是从最后一个叶子结点开始,采用最右双亲策略进行求解,因此二者算法原理相同,故而,SRMP-Gen 也可以在模式串中字符不同的情况下获得完备解.

通过表 4 和表 6 可以清晰地看到,DCNP 算法在 *P*1~*P*4 模式均可达到或接近完备解.通过表 6 可以看出:SAIL-Gen 算法在模式 *P*1 下 8 个序列上出现总个数为 1 795,而 DCNP 算法在 8 个序列上找到了 1 791 个出现.在这个实例上,DCNP 算法的近似度高达 99.7+%.同理,我们可以看到:DCNP 算法在模式 *P*2 和 8 个序列上近似度为 99.6+%,而在 *P*3 和 *P*4 的近似度则更高,充分地说明了当模式串中没有重复字符的情况下,DCNP 的解极其

接近完毕解.

(2) 在模式串中具有重复的字符情况下,SAIL-Gen 算法的性能最差,而 DCNP 算法的性能最好,SBO-Gen 算法的性能次之.

由于 *P*5~*P*9 中模式串中具有重复的字符,从表 5 可以清晰看到,SAIL-Gen 不但不能在任何实例上取得最好的结果;而且从表 6 可以看到,在 *P*5~*P*9 这 5 种模式在全部 8 个序列的出现数总和计算中,SAIL-Gen 算法均取得了最小值.更为重要的是,SAIL-Gen 算法的解性能相距 DCNP 算法解的差异显著.例如,SAIL-Gen 算法在模式 *P*5 和 8 个序列上仅仅找到了 545 个出现,而 DCNP 算法则找到了 665 个出现,SAIL-Gen 算法解性能在这组实例上仅仅达到 DCNP 算法的 81%.这些实验结果充分说明:在求解模式串中具有重复的字符情况下这种一般性问题时,不宜采用 SAIL-Gen 算法进行求解.

而通过表 5 可以清晰地看出,SBO-Gen 算法和 DCNP 算法在大多数情况下都取得了最好的结果;并且通过表 6 可以进一步发现,SBO-Gen 算法在 *P*5~*P*9 这 5 组模式上均取得了次好结果,而最好结果均由 DCNP 算法获得.充分地说明了 DCNP 算法的性能最好,SBO-Gen 算法的性能次之.产生这种现象的原因是:SBO-Gen 算法采用了 SGSP-Gen 与 SRMP-Gen 相结合的策略,每次选择出现相关数较少的策略,因而其可以很好地解决一次性条件下一般间隙的模式匹配问题.但是由于一般间隙下祖先结点的标号可能存在与子孙结点标号相同的情况,这需要动态更新各个结点的属性,而 DCNP 算法正是采用了这一原理,因而获得了比 SBO-Gen 算法更好的效果.

(3) 从表 7 的时间消耗上看,DCNP 算法最长,而 SAIL-Gen 算法最短.

SAIL 是文献[9]提出的一种在线算法,因此具有最好的时间性能来解决这个问题.SAIL-Gen 继承了这一特点,但是如前所述,当面对一般性问题时,SAIL-Gen 的求解性能很差,因此不宜采用 SAIL-Gen 算法进行求解.SBO- Gen 算法由于综合使用了 SGSP-Gen 算法和 SRMP-Gen 算法的核心策略,因此,SBO-Gen 算法求解时间比 SGSP- Gen 算法和 SRMP-Gen 算法的求解时间均长.DCNP 算法由于需要动态更新结点属性,因此其时间消耗比 SBO- Gen 算法略长.但是如前所述,DCNP 算法解的性能优于 SBO-Gen 算法.

综上所述,与其他 4 种算法相比,DCNP 算法的解的质量最好,但是时间消耗最长.

5 结 论

本文对目前具有间隙约束的模式匹配问题进行了分析和总结,针对目前的模式匹配大多只能按照模式串中规定的模式子串次序进行匹配问题进行研究,提出了一般间隙和一次性的模式匹配问题,而该问题的研究允许用户更加灵活地设定模式串.该问题具有如下一些特点:其间隙是可以为负值的一般间隙,序列中任何字符最多只能使用一次的限制,在匹配的过程中,仔细地考虑了各个模式子串所匹配的位置,因而是一种严格模式匹配,并且该问题是一种精确模式匹配.在理论证明了该问题的计算复杂性为 NP-Hard 的基础上,采用网树结构建立了求解算法 DCNP.该算法采用动态更新结点属性的方法,实现问题的求解.为了减少不必要的动态计算,本文提出了检测机制,仅对可能会造成出现内部产生重复的现象进行动态更新,以便提高 DCNP 算法的求解性能.之后,本文理论分析了 DCNP 算法的时间复杂度和空间复杂度.在大量真实数据集上验证了 DCNP 算法的性能.

模式匹配是序列模式挖掘的基础,本文仅对一般间隙和一次性条件的模式匹配问题进行了研究.下一步将对一般间隙的序列模式挖掘进行研究,该研究将有助于发现更多有价值的频繁模式.此外,在实际应用中,更多模式匹配是近似模式匹配,而本文所研究的问题依然属于精确模式匹配.而这些研究不但具有实际意义,而且研究难度也将更大,这些均属于未来研究的方向.

References:

[1] Chou C, Jea K, Liao H. A syntactic approach to twig-query matching on XML streams. Journal of Systems and Software, 2011, 84(6):993−1007. [doi: 10.1016/j.jss.2011.01.033]

[2] Wang M, Zhou JL, Le JJ. A data reusing strategy in column-store data warehouse. Chinese Journal of Computers, 2013,36(8): 1626−1635 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2013.01626]

[3] Song T, Li DN, Wang DS, Xue YB. Memory efficient algorithm and architecture for multi-pattern matching. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1650−1665 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4314.htm [doi: 10.3724/SP.J.1001.2013.04314]

[4] Akutsu T. Approximate string matching with variable length don't care characters. IEICE Trans. on Information and Systems, 1996, E79-D(9):1353−1354.

[5] Crochemore M, Iliopoulos C, Makris C, Rytter W, Tsakalidis A, Trichlas K. Approximate string matching with gaps. Nordic Journal of Computing, 2002,9(1):54−65.

[6] Zhang M, Kao B, Cheung D, Yip K. Mining periodic patterns with gap requirement from sequence. ACM Trans. on Knowledge Discovery from Data, 2007,1(2):Article 7. [doi: 10.1145/1267066.1267068]

[7] Wu Y, Wang L, Ren J, Ding W, Wu X. Mining sequential patterns with periodic wildcard gaps. Applied Intelligence, 2014,41(1): 99−116. [doi: 10.1007/s10489-013-0499-4]

[8] Ding B, Lo D, Han J, Khoo SC. Efficient mining of closed repetitive gapped subsequences from a sequence database. In: Ioannidis YE, Lee DL, Ng RT, eds. Proc. of the IEEE 25th Int'l Conf. on Data Engineering (ICDE). IEEE, 2009. 1024−1035. [doi: 10.1109/ICDE.2009.104]

[9] Chen G, Wu X, Zhu X, Arslan AN, He Y. Efficient string matching with wildcards and length constraints. Knowledge and Information Systems, 2006,10(4):399−419. [doi: 10.1007/s10115-006-0016-8]

[10] Wu YX, Wu XD, Jiang H, Min F. A heuristic algorithm for MPMGOOC. Chinese Journal of Computers, 2011,34(8):1452−1462 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01452]

[11] Guo D, Hu X, Xie F, Wu X. Pattern matching with wildcards and gap-length constraints based on a centrality-degree graph. Applied Intelligence, 2013,39(1):57−74. [doi: 10.1007/s10489-012-0394-4]

[12] Wu X, Zhu X, He Y, Araslan A. PMBC: Pattern mining from biological sequences with wildcard constraints. Computers in Biology and Medicine, 2013,43(5):481−492. [doi: 10.1016/j.compbiomed.2013.02.006]

[13] Lam H, Morchen F, Fradkin D, Calders T. Mining compressing sequential patterns. Statistical Analysis and Data Mining, 2012,7(1): 34−52. [doi: 10.1002/sam.11192]

[14] Fredriksson K, Grabowski S. Efficient algorithms for pattern matching with general gaps and character classed. In: Crestani F, Ferragina P, Sanderson M, eds. Proc. of the Int'l Conf. on String Processing and Information Retrieval. Springer-Verlag, 2006. 267−278. [doi: 10.1007/11880561_22]

[15] Fredriksson K, Grabowski S. Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. Information Retrieval, 2008,11(4):335−357. [doi: 10.1007/s10791-008-9054-z]

[16] Wu YX, Liu YW, Guo L, Wu XD. Subnettrees for strict pattern matching with general gaps and length constraints. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):915−932 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4381.htm [doi: 10.3724/SP.J.1001.2013.04381]

[17] He D, Wu X, Zhu X. SAIL-APPROX: An efficient on-line algorithm for approximate pattern matching with wildcards and length constraints. In: Proc. of the 2007 IEEE Int'l Conf. on Bioinformatics and Biomedicine (BIBM 2007). IEEE Computer Society, 2007. 151−158. [doi: 10.1109/BIBM.2007.48]

[18] Manber U, Baeza-Yates R. An algorithm for string matching with a sequence of don't cares. Information Processing Letters, 1991, 37(3):133−136. [doi: 10.1016/0020-0190(91)90032-D]

[19] Navarro G, Raffinot M. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. Journal of Computational Biology, 2003,10(6):903−923. [doi: 10.1089/106652703322756140]

[20] Bille P, Gørtz I, Vildhøj H, Wind D. String matching with variable length gaps. In: Chávez E, Lonardi S, eds. Proc. of the 17th Int'l Conf. on String Processing and Information Retrieval (SPIRE 2010). Mexico: Springer-Verlag, 2010. 385−394. [doi: 10.1007/978-3-642-16321-0_40]

[21] Bille P, Gørtz I, Vildhøj H, Wind D. String matching with variable length gaps. Theoretical Computer Science, 2012,443:25−34. [doi: 10.1016/j.tcs.2012.03.029]

[22] Wu XD, Xie F, Huang YM, Hu XG, Gao J. Mining sequential patterns with wildcards and the one-off condition. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1804−1815 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4422.htm [doi: 10.3724/SP.J.1001.2013.04422]

[23] Zhang S, Zhang J, Zhu X, Huang Z. Identifying follow-correlation itemset-pairs. In: Proc. of the Int'l Conf. on Data Mining (ICDM). IEEE Computer Society, 2006. 765−774. [doi: 10.1109/ICDM.2006.84]

[24] Warmuth M, Haussler D. On the complexity of iterated shuffle. Journal of Computer and System Sciences, 1984,28(3):345−358. [doi: 10.1016/0022-0000(84)90018-7]

[25] Wu Y, Wu X, Min F, Li Y. A nettree for pattern matching with flexible wildcard constraints. In: Proc. of the 2010 IEEE Int'l Conf. on Information Reuse and Integration (IRI 2010). IEEE Systems, Man, and Cybernetics Society, 2010. 109−114. [doi: 10.1109/IRI.2010.5558954]

附中文参考文献:

[2] 王梅,周娇玲,乐嘉锦.一种列存储数据仓库中的数据复用策略.计算机学报,2013,36(8):1626−1635. [doi: 10.3724/SP.J.1016.2013.01626]

[3] 嵩天,李冬妮,汪东升,薛一波.存储有效的多模式匹配算法和体系结构.软件学报,2013,24(7):1650−1665. http://www.jos.org.cn/1000-9825/4314.htm [doi: 10.3724/SP.J.1001.2013.04314]

[10] 武优西,吴信东,江贺,闵帆.一种求解 MPMGOOC 问题的启发式算法.计算机学报,2011,34(8):1452−1462. [doi: 10.3724/SP.J.1016.2011.01452]

[16] 武优西,刘亚伟,郭磊,吴信东.子网树求解一般间隙和长度约束严格模式匹配.软件学报,2013,24(5):915−932. http://www.jos.org.cn/1000-9825/4381.htm [doi: 10.3724/SP.J.1001.2013.04381]

[22] 吴信东,谢飞,黄咏明,胡学钢,高隽.带通配符和 one-off 条件的序列模式挖.软件学报,2013,24(8):1804−1815. http://www.jos.org.cn/1000-9825/4422.htm [doi: 10.3724/SP.J.1001.2013.04422]

柴欣(1962－),男,河南武陟人,教授,主要研究领域为智能计算,数据挖掘.

贾晓菲(1989－)女,硕士生,主要研究领域为模式匹配.

武优西(1974－),男,博士,教授,CCF 高级会员,主要研究领域为智能计算,数据挖掘.

江贺(1980－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,智能计算,数据挖掘.

吴信东(1963－),男,博士,教授,博士生导师,主要研究领域为数据挖掘,基于知识的系统,万维网信息探索.