ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.4, April 2010, pp.718−731
doi: 10.3724/SP.J.1001.2010.03610

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

基于参考集索引的高效序列相似性查找算法*

戴东波+, 熊 赟, 朱扬勇

(复旦大学 计算机科学技术学院,上海 200433)

Efficient Algorithm for Sequence Similarity Search Based on Reference Indexing

DAI Dong-Bo+, XIONG Yun, ZHU Yang-Yong

(School of Computer Science and Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: E-mail: daidongbo@fudan.edu.cn

Dai DB, Xiong Y, Zhu YY. Efficient algorithm for sequence similarity search based on reference indexing. *Journal of Software*, 2010,21(4):718−731. http://www.jos.org.cn/1000-9825/3610.htm

Abstract: Sequence data are ubiquitous in many domains such as text, Web access log and biological database. Similarity search in this kind of data is very important for knowledge acquisition and analysis. An indexing technique based on reference is an effective method for sequence similarity search, the main idea of which is to assign some sequences in database as reference sets, then filter out those sequences unrelated to query sequence and finally get the answer efficiently. This paper presents a similarity search algorithm IRI (improved reference indexing) which is based on current indexing technique using reference set and is more powerful in terms of filtration. First, previous query results are used to accelerate the current query. Then, the upper bound and lower bound based on sequence characteristic are proposed to make the bound tighter and improve the filtration capability. Finally, to avoid the time-consuming edit distance computing, only partial edit distance between prefix sequences need to compute, which makes the algorithm run faster. Real data including DNA and protein sequence data are used in the experiment. Comprehensive experimental results show that IRI is more efficient than the current reference-based indexing algorithm RI (reference indexing).

Key words: sequence similarity search; reference indexing; edit distance

摘 要: 序列数据在文本、Web 访问日志文件、生物数据库中普遍存在,对其进行相似性查找是一种重要的获取和分析知识的手段.基于参考集索引技术是一类解决序列相似性查找的有效方法,主要思想是找到序列数据库中的少数序列作为参考集,通过参考集过滤掉数据库中与查询序列不相关的数据,从而高效地回答查询.在现有基于参考集索引技术的基础上,提出一种过滤能力更强的序列相似性查询算法 IRI(improved reference indexing).首先,充分利用了先前的查询结果集来加速当前的查询,其次考虑了基于序列特征的上界和下界,使得应用参考集进行过滤的上下界更紧,过滤能力进一步加强.最后,为了避免候选集中费时的编辑距离计算,则只计算前缀序列间的编辑距离,从而进一步加速算法运行.实验采用真实的 DNA 序列和蛋白质序列数据,结果表明,算法 IRI 在查询性能上明显优于现有的基于参考集索引方法 RI(reference indexing).

* Supported by the National Natural Science Foundation of China under Grant No.0573093 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA02Z329 (国家高技术研究发展计划(863))

Received 2008-09-29; Revised 2008-12-09, 2009-01-20; Accepted 2009-03-31

关键词: 序列相似性查找;参考集索引;编辑距离
中图法分类号: TP311 文献标识码: A

序列数据是一种重要的数据类型,广泛存在于各个领域,如:商业领域的客户交易数据、媒体领域的文本和语音数据、Web 中用户访问序列以及生物信息中的 DNA 和蛋白质数据等[1−3].对序列数据库进行相似性查询是一种获取和分析知识的有效手段.如在基因数据库中进行相似性查找,可以根据查询序列和查询结果序列的结构相似性推测其功能的相似性,然后利用已知基因的功能预测新基因的功能.序列相似性查找在生物信息挖掘中是基本而又重要的任务之一.

常见的序列相似性查询有两类:范围查询和 k 近邻查询.k 近邻查询是在给定序列数据库中返回与查询序列最相似的前 k 条序列.由于 k 近邻查询可以利用范围查询完成[4],本文只着重考虑范围查询.范围查询的目的就是在序列数据库中返回与查询序列距离小于给定半径的所有序列数据,其定义如下:给定数据库 $S=\{S_1,S_2,\ldots,S_n\}$,查询半径 r 以及查询序列 q,求其查询结果 $R=\{S_i|S_i\in S,$且 $ED(S_i,q)\le r\}$,其中 $ED(S_1,S_2)$为编辑距离,是衡量序列 S_1 和 S_2 相似性的一个指标.两条序列 S_1 和 S_2 的编辑距离定义为将 S_1 映射到 S_2 所需要的最少操作步骤,其中的操作包括插入、删除和替代[5].

给定长度分别为 m 和 n 的序列 S_1 和 S_2, 其编辑距离 $ED(S_1,S_2)$的计算复杂度为 $O(m\times n)$.很多序列数据库中的序列平均长度一般都很长,所以计算序列间编辑距离的时间代价是很大的.如:把 DNA 序列看作是在字母表 $\Sigma=\{A,G,C,T\}$上的字符串,则在人类最短的 18 号染色体上的字母长度就到达 4.2×10^6 bp 左右(一个 bp 对应Σ上一个字母).在如此长的序列上即使是频繁计算其子序列和查询序列间的编辑距离,执行效率也是很低的.此外,随着各种序列数据的爆炸式增长,序列数据库的规模也在日益增大,这给高效的序列相似性查询带来了新的挑战.如随着生物信息学的快速发展和人们对各种生物体的广泛研究,导致存储生物基因序列(包括 DNA 和 RNA)的 Genbank 数据库[6]的数据量正以每 15 个月翻一倍的速度增长[7].因此,在海量的序列数据中快速找到所需信息是一项重要的研究工作.

现有的序列相似性查询技术分为非索引方法和基于索引的方法.非索引方法没有预先获取数据库中序列数据的信息,在进行查找时往往要遍历整个数据库,其性能不是很理想.基于索引的方法一般先基于数据库信息内容构建索引,在进行查询时,先由索引过滤掉与查询不相关的数据,然后在过滤后的候选子集中进一步查找相关数据,极大地提高了查询效率.但现有的基于索引技术仍存在一些不足之处:(1) 在新的查询请求到来时,从数据库索引直接从头搜索,没有充分利用信息丰富的先前查询结果.(2) 在由索引过滤不相关数据时,利用了查询序列和数据库中序列间编辑距离的上界和下界,这些上界和下界只是一般度量三角不等式的应用,未考虑基于序列本身特征的上界和下界.(3) 在过滤后的候选集中直接计算数据库中候选序列和查询序列的编辑距离,未考虑只进行部分编辑距离的计算来作进一步的优化.

针对上述问题,本文在现有基于参考索引方法的基础上,提出一种过滤能力更强的序列相似性查询算法 IRI(improved reference indexing),其主要贡献是:(1) 充分利用了先前查询结果来加速当前查询,并从理论上分析了先前查询结果大小必须满足的下界条件.(2) 在考虑序列特征的基础上,提出了编辑距离的上界和下界,使得过滤的上界和下界更紧,从而过滤能力进一步增强.(3) 根据给定的查询半径,对于候选集中序列和查询序列的编辑距离计算,不需要完全计算,只需计算它们前缀序列的编辑距离,从而使得算法效率进一步提高.

本文第 1 节介绍相关工作.第 2 节给出查询算法 IRI 的实现过程.第 3 节给出实验结果和分析.第 4 节对全文作总结并给出后续研究方向.

1 相关工作

序列的范围查询是一种近似查询,最直接的方法是在数据库中进行顺序扫描和直接编辑距离的计算,然而由于编辑距离计算的时间开销和空间开销都很大以及序列数据库规模日益增大,使得这种朴素的查找方法变得不可行.由此,很多研究人员提出了改进方法,主要分为非索引查询方法和索引查询方法.

(1) 非索引查询方法.Ukkonen 在文献[8]中提出两种近似编辑距离的方法 q-Gram 和 maximal matches.q-Gram 通过计算查询序列和数据库中序列之间共同的 q 长子序列来考察其相似程度,而 maximal matches 是找到它们之间的公共最长子序列来作为相似性度量,这些都是编辑距离的一个有界近似,且计算复杂度比求编辑距离要低,从而可以在小规模数据库中实现快速查找.对于给定半径 r 的范围查询,Ukkonen[9]使得两条长度为 n 的序列间编辑距离计算复杂度降为 $O(r\times n)$,然而当 r 为 $O(n)$时,其计算复杂度仍为 $O(n^2)$.Ricardo 等人采用基于 RAM(random access machine)机器模型上的位操作来模拟非确定性的有限自动机,从而实现了在线的序列近似匹配算法[10].文献[11]中应用基于位置敏感哈希函数(locality sensitive hashing,简称 LSH)的随机算法来加速序列间的比较,其基本思想是两条序列越相似,则它们的随机位置投影相等的可能性越大.非索引方法的特点是不对数据库作预处理,不使用索引结构,虽然无需预处理的时间开销,但一般需遍历整个数据库,因此,随着序列数据的不断增长,这些方法很难满足高效查询的需求.

(2) 索引查询方法.索引方法一般先对序列数据库进行预处理,建立合适的索引结构,并且这种索引结构相对原始数据库非常小,可以放入内存.用户提交一个查询和查询半径后,系统首先通过索引结构进行查询,过滤与查询序列不相关的记录,得到一个候选结果集后再进行后处理.著名的生物数据库搜索工具 FASTA[12]和 BLAST[13]是基于 q-Gram 索引机制进行相似性查询的.它们首先对数据库中的 q-长子序列进行哈希索引,然后利用索引对查询序列进行精确匹配,最后对匹配区域的左右两边进行允许有空位的扩展找到满足条件的查询结果.由于 FASTA 和 BLAST 是一种启发式的搜索工具,不能找到所有满足条件的匹配结果.PatterHunter[14]也是利用 q-Gram 索引进行查询,但是对不连续的 q-长子序列进行哈希索引,从而提高了查找的敏感度指标.在最近的序列相似性查找研究中,Chen 等人[15]通过序列数据库构造变长的 q-Gram 字典,然后把每条序列通过字典分离出各自的 q-Gram 集,并给出了相似序列必须共享的变长 q-Gram 下界来进行过滤.同样地,作者在随后的研究[16]中进一步给出了构造的变长 q-Gram 字典与查询性能的关系.对于采用 q-gram 倒排链表的索引方法,文献[17]提出几种有效合并倒排链表的方法,并与序列的各种过滤方法相结合,从而有效地加速了相似序列的查找性能.后缀树最早由 Weiner 提出[18],后缀数组[19]是后缀树的一个变种,但在具体实现上更节省存储空间.很多有效构建后缀树的方法以及利用后缀树和后缀数组进行相似性查找的方法也被提了出来[20−23].但基于后缀树的查询方法存在两个缺点:耗费太多存储空间和不能有效处理空位字符.文献[4]通过频率变换和小波变换把序列数据映射到一个向量空间,然后对向量空间数据点通过最小边界矩阵(minimum bounding rectangle,简称 MBR)划分为块,最后对这些块建立多分辨率的索引结构 MRS(multi resolution string).文献[24]采用一种对序列数据的二分频率变换进一步提高了 MRS 的过滤能力.VP(vantage point)树[25]和 MVP(multiple vantage points)[26]树是两种基于参考索引查询的数据结构.VP 树通过随机选择参考点,把数据空间划分为层次的球性区域而建立一棵平衡查找树.MVP 树是 VP 树的一个改进,在树的每层使用了多个参考点,从而使得过滤能力进一步增强.Buhler[27]通过空间嵌入的方法,将序列间的加权距离映射为向量空间的海明距离,从而使得序列间的相似性查找可以基于蛋白质的各种得分矩阵,更符合生物学意义.

文献[28]中实现的基于参考索引的查询算法是目前性能最好的系统之一,我们不妨称其为 RI(reference indexing)算法.RI 算法的基本思想是对于给定的查询集样本 Q,基于最大剪枝原则在序列数据库 S 找到一个大小为 m 的参考集 V 并预计算好参考集与数据库中元素之间的编辑距离,然后为每个数据库中元素分配一个参考子集.在查询 q 和查询半径 r 提交时,由 S 中每个元素 s_i 的参考子集可计算出 $ED(q,s_i)$的上界 UB 和下界 LB,即:$UB=\min(\bigcup_{v_j\in R_i}\left|ED(q,v_j)+ED(v_j,s_i)\right|)$,$LB=\max(\bigcup_{v_j\in R_i}\left|ED(q,v_j)-ED(v_j,s_i)\right|)$,其中 s_i 是查询数据库中的元素,R_i 为 s_i 的查询参考子集.利用 LB 和 UB,若 $r<LB$,则 s_i 被过滤掉;若 $r>UB$,则 s_i 加入查询结果集;若 $LB\le r\le UB$,则把 s_i 加入候选集作后处理.本文提出的 IRI 算法是在 RI 算法的基础上针对其不足而实现的.

2 IRI 算法实现过程

2.1 利用先前查询结果加速当前查询

若所有提交查询的查询半径都设置成相同数值,这在实际应用中并不适用,因为用户可能因为不同的查询序列所要求的相似性程度不一样,从而设置不同的查询半径.比如在生物数据库中进行查询时,远亲家族序列的查询相似性要求就比近亲家族的要求要低.所以本文考虑这种一般情况,认为不同查询有不同的查询半径.

RI 算法对先前提交的查询返回结果没有进行保存,下一个查询仍然从头开始计算.先前的查询结果含有丰富的信息,这些信息可以给当前查询提供帮助以加速其运行.先前的查询也可以看作是一个参考点,与 RI 算法中参考集提供查询上界和下界进行过滤一样,先前查询及其结果集也有可能提供部分过滤集和部分查询结果集.从直观上看,若当前查询与先前查询相似性很大,则先前查询过滤掉的集合很有可能也被当前查询过滤掉.反之,若当前查询与先前查询相似性很小,则先前查询结果集很有可能被当前查询过滤掉.给定一个先前的查询 q 和查询半径 r,记作 $Q(q,r)$,设其查询结果为 R,序列数据库中不在 R 中的元素集合设为 R',则对于 R 中任意元素 s,$ED(q,s)\le r$;对于 R' 中任意元素 s',$ED(q,s')>r$.设当前提交的查询为 $Q(q',r')$,其查询结果为 R'',则以下定理成立:

定理 1. 若 $ED(q,q')>r+r'$,则 $\forall s\in R, s\notin R''$ 成立;当 $r>r'$ 时,若 $ED(q,q')\le r-r'$,则 $\forall s'\in R', s'\notin R''$ 成立;当 $r\le r'$ 时,若 $ED(q,q')\le r'-r$,则 $\forall s\in R, s\in R''$,特别地,当 $r=r'$ 时,查询 q 就是 q',则 $R''=R$.

证明:由于编辑距离 ED 是一个度量,所以满足三角不等式.

1) 当 $ED(q,q')>r+r'$ 时,对于任意的 $s\in R$,由于 $ED(q,s)\le r$,则 $ED(q',s)\ge ED(q,q')-ED(q,s)>r+r'-r=r'$,所以 s 不在 q' 的查询结果集中,即 $s\notin R''$.

2) 当 $r>r'$ 时,若 $ED(q,q')\le r-r'$,对于任意的 $s'\in R'$,由 $ED(q,s')>r$,则 $ED(q',s')\ge ED(q,s')-ED(q,q')>r-(r-r')=r'$,$s'$ 不在 q' 的查询结果集中,即 $s'\notin R''$.

3) 当 $r\le r'$ 时,若 $ED(q,q')\le r'-r$,对于任意的 $s\in R$,由 $ED(q,s)\le r$,则 $ED(q',s)\le ED(q,q')+ED(q,s)\le r'-r+r=r'$,$s$ 在 q' 的查询结果集中,即 $s\in R''$.当 $r=r'$ 时,$ED(q,q')\le 0$,因为 ED 是一个度量,具有非负性质,所以 $q=q'$,即当前查询 q' 就是先前查询 q 的一个副本,此时两个相同查询且有相同查询半径,所以 $R''=R$. □

对于当前查询,定理 1 中证明的 3 种情况都属于一个先前查询结果能够过滤不相关数据或得到部分查询结果数据的情况,这对当前查询都是有利的,所以应尽量提高当前查询满足这 3 种条件之一的可能性.为此,我们保存多个先前查询结果,利用其综合过滤能力来提高查询效率.在定理 1 证明中的 3 种情况,第 2 种情况的过滤能力一般是最强的,即过滤与当前查询不相关数据是最多的.因为在实际应用中,给定的查询在大型序列数据库中获得的查询结果相对于数据库中所有数据来说,所占比例是很少的.如:若先前某个查询 q 的结果集中元素只占整个所查询的序列数据库的 1%,则在定理 1 中第 2 种情况满足的前提下,当前查询可以过滤掉 99%的不相关数据.因此,在保存的多个查询结果中,只要某个查询结果满足定理 1 中第 2 种情况,当前查询的后处理就只需集中在该先前查询结果集中.

设系统保存了 k 个先前的查询及其结果集:$(q_1,r_1,R_1),(q_2,r_2,R_2),\ldots,(q_k,r_k,R_k)$,其中 r_i,R_i 分别为先前查询 q_i 的查询半径和查询结果集,D 为查询数据库.因为定理 1 中第 2 种情况要求先前查询半径大于当前查询半径,为了尽可能先找到满足这种情况的先前查询结果而避免后续搜索,我们应先搜索查询半径较大的先前查询.不失一般性,我们设按照其查询半径大小降序排列的结果集顺序就是 $(q_1,r_1,R_1),(q_2,r_2,R_2),\ldots,(q_k,r_k,R_k)$,我们用以下算法 1 求得当前查询 q' 的过滤集 *Filter S* 和部分结果集 *Res*:

算法 1. Filter_By_PreResult.

输入:降序排列好的先前查询结果集:$(q_1,r_1,R_1),(q_2,r_2,R_2),\ldots,(q_k,r_k,R_k)$,当前查询 q' 和查询半径 r';

输出:查询 q' 的过滤集 *Filter S* 和部分结果集 *Res*.

$n\leftarrow 1; q\leftarrow q_1; r\leftarrow r_1; R\leftarrow R_1$; //$q$ 是正在搜索的先前查询;

Filter S 和 *Res* 置空;

```
While (q≠q_k)
    If (r>r′并且 ED(q,q′)≤r−r′) Filter S=(D−R);返回;          //定理 1 中证明的第 2 种情况
    else if (ED(q,q′)>r+r′)          Filter S=Filter S∪R;      //定理 1 中证明的第 1 种情况
        else if (r≤r′并且 ED(q,q′)≤r′−r)                       //定理 1 中证明的第 3 种情况
                if (r=r′)                                      //查询 q 和 q′相同,且查询半径相同
                    Res=R;  q′的全部查询结果←Res,返回;          //此时 q′的全部查询结果就为 Res
                else    Res=Res∪R;
    n←n+1;q←q_n;r←r_n;R←R_n;
end while;
```

算法 1 是通过搜索多个先前查询结果来求得过滤集和部分查询集的,最坏时间复杂度为 $O(kL^2)$,其中 L 为查询序列的平均长度,L^2 项是计算当前查询序列和先前查询序列间编辑距离的时间开销.一般来说,保存的先前查询结果越多,通过算法 1 获得的过滤集和部分查询集也越大,从而后面再通过参考集索引进行查询所涉及的数据库范围越小,使得后续查询速度提高.但存储越多的先前查询结果,其存储代价和的时间代价也越大,并且通过我们后面的实验发现,当存储的先前查询结果超过某值时,获得的部分查询结果集和过滤集大部分是重叠的.因此,不能存储所有先前的查询结果,我们只存储有限的 k 个先前查询结果.判断一个当前查询利用先前的 k 个查询结果是否提高查询效率的方法是看下面条件是否成立:

$$|Res|+|Filter\ S|>k \tag{1}$$

其中,$|Res|$和$|Filter\ S|$分别为当前查询利用 k 个先前查询结果集所获得的部分结果集大小和过滤集大小.公式(1)表明:通过定理 1 中 k 次编辑距离的计算,被过滤或作为部分结果的序列可避免在后选集中应用编辑距离进行计算,当避免的编辑距离计算次数大于已经进行的 k 次编辑距离计算时,应用定理 1 进行过滤才可以提高查询性能.

下面我们给出 k 的估计值.文献[29]提出,在很多实际的关于度量空间相似性查询应用中,任何数据点与其他数据点之间的距离分布 F 在很大程度上是相近的,且查询点与查询数据库之间的距离分布可以用 F 来近似.因此,我们对序列数据在编辑距离中形成的度量空间作以下假设:设任意两个随机序列间的编辑距离都是独立同分布的.此外,在诸如文本序列和蛋白质序列的许多实际应用中,给定查询半径的查询结果集大小是与查询半径多项式相关的[30,31],即:

$$N(r)=t\times r^c \tag{2}$$

其中,$N(r)$是查询半径为 r 的查询结果集中元素的个数,t 和 c 分别为常数.下面的命题 1 给出了在上述的假设条件下,k 在理论上有下界值.

命题 1. 设任意两个随机序列间的编辑距离都是独立同分布的,其概率分布函数为 $F(x)$.若 k 个先前查询结果集的查询半径都为 r,则对于任意给定当前查询 $Q(q',r')$和概率参数 Φ,为了使式(1)以概率 $1-\Phi$成立,则 k 值需满足 $k>\dfrac{2\times In1/\Phi}{p(1-1/(p\times t\times r^c))^2}$,其中, $p=1-\left[F(r+r')-F(|r-r'|)\right]$.

证明:当一个给定查询 $Q(q',r')$与某先前查询 $Q(q,r)$满足定理 1 证明中的任何 3 种情况之一时,则称 $Q(q',r')$命中 $Q(q,r)$.设 $Q(q',r')$命中 $Q(q,r)$的概率为 p,由定理 1 中所证明的 3 种情况的前提条件可知,当$|r-r'|<ED(q,q')\le r+r'$时,$Q(q',r')$不能命中 $Q(q,r)$,所以,

$$p=1-\left[F(r+r')-F(|r-r'|)\right] \tag{3}$$

由于所有 k 个先前查询结果集的查询半径都为 r,所以查询 $Q(q',r')$命中先前 k 个查询中的每一个概率都是相同的,即为式(3)中的 p.又因为 $ED(q,q')$是独立同分布的,所以查询 $Q(q',r')$是否命中先前 k 个查询可以看作是 k 重伯努利实验.用随机变量 x_i 来表征 $Q(q',r')$是否命中先前查询,当查询 $Q(q',r')$命中第 i 个先前查询时,$x_i=1$,否则 $x_i=0$.设 $X=\sum_i x_i$,则 X 为查询 $Q(q',r')$命中先前 k 个查询结果集的次数,X 的期望为 $E(X)=k\times p$.所以,若 $Q(q',r')$命中定理 1 证明中第 1 种或第 3 种情况,则由公式(2),有 $|Filter\ S|=t\times r^c$ 或 $|Res|=t\times r^c$, 若 $Q(q',r')$命中定理 1 证明

中第 2 种情况,则$|Filter\ S|=|D|-t\times r^c$, 其中,|D|是查询序列数据库 D 的大小.在一般实际应用中,查询结果集大小只占查询数据库的一小部分,即$|D|-t\times r^c>t\times r^c$,设 $Q(q',r')$至少命中 s 个先前查询结果,才能满足条件式(1),即:$\sum_{i=1}^{s}|Q_i|>k$, $|Q_i|$为第 i 个先前查询结果集大小.由$|Q_i|$的各种取值可知,$\sum_{i=1}^{s}|Q_i|>s\times t\times r^c$, 所以,当$s\times t\times r^c>k$, 即$s>\frac{k}{t\times r^c}$ 时,条件式(1)满足.为了保证条件式(1)以概率 1−Φ成立,需使下式成立:

$$\Pr\left(X>\frac{k}{t\times r^c}\right)>1-\Phi \tag{4}$$

我们先求 $\Pr\left(X<\frac{k}{t\times r^c}\right)$.因为 $E(X)=k\times p$,由 chernoff 不等式[32]:当 $p\times t\times r^c>1$ 时,有 $\Pr\left(X<\frac{k}{t\times r^c}\right)<e^{-k\times p\times\left(1-\frac{1}{p\times t\times r^c}\right)^2/2}$,从而有 $\Pr\left(X>\frac{k}{t\times r^c}\right)>1-e^{-k\times p\times\left(1-\frac{1}{p\times t\times r^c}\right)^2/2}$,当$1-e^{-k\times p\times\left(1-\frac{1}{p\times t\times r^c}\right)^2/2}>1-\Phi$ 时,即当

$$k>\frac{2\times In1/\Phi}{p(1-1/(p\times t\times r^c))^2} \tag{5}$$

时,式(4)成立. □

从命题 1 中的式(5)可以看出,若查询 $Q(q',r')$的命中率 p 和先前查询结果集大小 $t\times r^c$ 越大,则 k 的下界值越小;若参数Φ越小,则 k 的下界值越大.其中条件 $p\times t\times r^c>1$ 表明,查询 $Q(q',r')$在命中任何一个先前查询 $Q(q,r)$以后,获得的过滤集或部分结果集大小期望值至少要大于 1,即要大于查询$Q(q',r')$和$Q(q,r)$的一次距离计算,这也符合直观意义.为了使得 $p\times t\times r^c>1$ 条件成立,我们要尽量增大命中率 p 值和查询结果集大小,通过我们下面的启发式更新方法,在我们的实验中,通过设置各种参数,k 在内存允许的前提下,当取满足命题 1 中不等式(5)的较大值时是可以提高查询效率的.注意,在存先前查询结果时,存储的是序列的 ID 号而不是序列本身.

由于我们只存储先前的 k 个查询结果集 R,随着查询的不断进行,结果集 R 也要不断更新.当一个查询 q 获得查询结果返回时,q 是否要加入 R 以及替换掉 R 中哪个查询结果主要取决于 q 的加入能否使得后续查询利用更新后的 R 进行查询时的效率有所提高.我们利用以下两个观察来进行 R 启发式的更新:

观察 1. 在现实的大型数据库相似性查询应用中,由于是查找较为相似的序列数据,查询半径 r 不会太大,r 大到能够使所有序列满足查询的可能性较小.一般情况下,一个查询找到的相似性序列只占总数据库的一小部分,尽管增大先前查询半径会减少定理 1 证明中第 2 种情况的过滤集大小,但减少的过滤集数目相对总过滤数是较小的一部分,且此时会提高第 2 种情况被命中的可能性.如:先前查询的半径增大后,设其查询结果从 1%|D|增加到 5%|D|,第 2 种情况的命中率即使从 H 提高到(H+1.5%)(在我们的实验中,H 不超过 30%),情况 2 被触发后的过滤集也会从 H×99%|D|提高到(H+1.5%)×95%|D|.所以,在一般的现实应用中,增大先前查询半径后,即使第 2 种情况被命中的概率有较小增长,也会使情况 2 被触发后的过滤集增大.

观察 2. 一般情况下,若查询结果中的查询半径越大,则其查询结果集中的元素越多[30,31],从而有,若当前查询满足定理 1 中证明的第 1 种情况或第 3 种情况,则通过算法 1 获得的过滤集 $Filter\ S$ 或部分结果集 Res 中的元素也越多.

序列数据间的距离分布 F 也会影响定理 1 证明中 3 种过滤情况的总体命中率,从后面的实验可以看到,尽管增大先前查询半径,可以使总体命中率存在小幅下降,但由于定理 1 证明中第 2 种情况被命中的概率总在提高且产生的过滤集远大于第 1 种情况和第 3 种情况,所以总的过滤集仍在增大.由以上的观察 1 和观察 2 可知,在结果集 R 中保存查询半径较大的查询结果有利于后续查询,所以,我们采用以下启发式方法来对查询结果进行更新:若当前查询为 $Q(q',r')$,其查询结果为 R,设先前查询结果集某个查询为 $Q(q,r)$,满足 $r'>r$ 且 r 是先前查询结果集中查询半径最小的,则用查询 q'序列及其查询结果 R 替换查询 q 及其查询结果.

当用命题 1 中的公式(5)求 k 的下界值时用到一个假设:k 个先前查询结果集的查询半径都为 r.尽管查询的半径各不相同,由于我们启发式地更新先前查询结果的方法,总是把查询半径较大的查询结果插入到先前查询

结果集中,所以经过 n 次查询后,先前查询结果集中的查询半径是 n 个查询半径中最大的前 k 个.随着查询的不断进行,n 值不断增大,各查询的查询半径的方差可能会很大,但此时最大的前 k 个查询半径的方差相对要小得多,所以最大的前 k 个查询半径可近似看成是相同的.

2.2 基于序列特征的编辑距离上界和下界

RI 算法在利用参考集对当前查询 q(其查询半径为 r)过滤不相关数据时,主要是依据编辑距离的上界 UB 和下界 LB:$UB=\min(\bigcup_{v_j\in R_i}|ED(q,v_j)+ED(v_j,s_i)|)$, $LB=\max(\bigcup_{v_j\in R_i}|ED(q,v_j)-ED(v_j,s_i)|)$,其中 s_i 是查询数据库中的元素,R_i 为 s_i 的查询参考子集.若 $UB<r$,则把 s_i 加入 q 的结果集;若 $LB>r$,则过滤掉 s_i,否则,把 s_i 加入 q 的候选结果集作为后续进一步处理.上界 UB 和下界 LB 只是根据编辑距离是一个度量从而满足三角不等式这一性质而确定的,未考虑序列数据本身的特征,从而使得这两个界不是很紧,过滤能力有限.即若 s_i 落在查询 q 的 r 半径范围内且离 q 很近,但 $UB\geq r$;若 s_i 远远落在 q 的 r 半径范围外,却有 $LB\leq r$.图 1 所示例子说明了后面一种情况.s_i 不在查询 q 的结果集中,查询 q 利用两个参考序列 ref_1 和 ref_2 求得 $ED(q,s_i)$的编辑距离下界 $bound1$ 和 $bound2$,由于 $bound1$ 和 $bound2$ 都小于查询半径 r,所以查询 q 不能过滤掉 s_i.

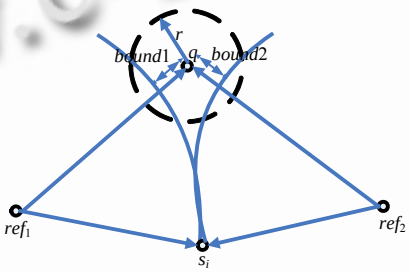

Fig.1 Query q can't filter s_i based on reference points ref_1 and ref_2

图 1 查询 q 基于参考点 ref_1 和 ref_2 不能过滤掉 s_i

鉴于上述不足,我们利用了已有的频率距离 FD(frequency distance)[4],并定义了一种不等长序列间的海明距离 HD(Hamming distance),这两种距离都是基于序列特征的.

定义 1[4]. 设序列 $S=s_1s_2\ldots s_n$ 是定义在基数为 T 的字母表$\Sigma=\{a_1,a_2,\ldots,a_T\}$上,$n_i$ 是字母 a_i 在 S 中的出现次数$(1\leq i\leq T)$,则称 $FV(S)=(n_1,n_2,\ldots,n_T)$是序列 S 的频率向量.若有序列 S_1 和 S_2,其各自的频率向量为 $FV(S_1)=(n_1^1,n_2^1,\ldots,n_T^1)$, $FV(S_2)=(n_1^2,n_2^2,\ldots,n_T^2)$,则称函数 $FD(S_1,S_2)=\max\left(\sum_{i=1}^{T}I_i^P(n_i^1-n_i^2),\sum_{j=1}^{T}I_j^N(n_j^2-n_j^1)\right)$为序列 S_1 和 S_2 的频率距离,其中 I_i^P 和 I_j^N 分别为示性函数,当 $n_i^1>n_i^2$ 时,$I_i^P=1$,否则 $I_i^P=0$;当 $n_j^2>n_j^1$ 时,$I_j^N=1$,否则 $I_j^N=0$.

定义 2. 设两条序列 $S_1=s_1s_2\ldots s_m$ 和 $S_2=p_1p_2\ldots p_n$ 的长度分别为 m 和 n,不妨设 $m\geq n$.设 $S_2=p_1p_2\ldots p_n$ 和等长的 S_1 前缀子序列 $S_1'=s_1s_2\ldots s_n$ 对应位置上不同元素个数为 k,则 $HD(S_1,S_2)=k+(m-n)$称之为序列 S_1 和 S_2 的海明距离.

对于两条长度分别为 m 和 n 的较长序列$(m\geq n)$,通过一遍扫描序列可以计算出其频率距离 FD 和海明距离 HD,时间复杂度都为 $O(m+n)$,远小于其编辑距离的计算复杂度 $O(m\times n)$.下面定理证明了两序列的频率距离和海明距离分别是其编辑距离的下界和上界.

定理 2. 给定两条序列 S_1 和 S_2,有 $FD(S_1,S_2)\leq ED(S_1,S_2)\leq HD(S_1,S_2)$成立.

证明:$FD(S_1,S_2)\leq ED(S_1,S_2)$的证明见文献[4].下面证明 $ED(S_1,S_2)\leq HD(S_1,S_2)$.

设 S_1 和 S_2 的长度为别为 m 和 n,且不失一般性,设 $m\geq n$,S_2 和等长的 S_1 前缀子序列 S_1'对应位置上不同元素个数为 k,则 $HD(S_1,S_2)=k+(m-n)$.S_2 可先经 k 次替换操作后变为 S_1',然后再由 $m-n$ 次插入操作就变为 S_1.由编辑距离的最少操作性质,有 $ED(S_1,S_2)\leq HD(S_1,S_2)$. □

RI 算法中利用参考集获得的上界 UB 和下界 LB 完全未考虑序列特征,所以,当查询 q 和数据库中元素 s 很相似时,由上界 UB 可能不会直接把 s 加入 q 的结果集;当查询 q 和 s 相距很远时,由 LB 也可能不会过滤掉 s.为

了提高直接把 s 加入结果集或过滤掉 s 的可能性,我们加进序列间的频率距离 FD 和海明距离 HD,这时上界和下界变为了 $UB'=\min(\cup_{v_j\in V}\left|ED(q,v_j)+ED(v_j,s_i)\right|\cup HD(q,s_i))$, $LB'=\max(\cup_{v_j\in V}\left|ED(q,v_j)-ED(v_j,s_i)\right|\cup FD(q,s_i))$,显然,$UB'\le UB$,$LB'\ge LB$,即 UB'和 LB'是更紧的上界和下界,过滤 s 或获取 s 部分结果集的能力进一步增强.

2.3 后处理中编辑距离的部分计算

在很多基于索引的序列相似性查询系统中,查询序列和过滤后的候选集中序列相似性一般是通过完全编辑距离的计算[4,28]来实现的,由于编辑距离计算复杂度较高,致使对候选集的处理效率较低.计算序列 S_1 和 S_2 编辑距离的基本方法是应用动态规划方法,将 S_1 和 S_2 的每个局部片段的编辑距离存储在一张最优比对表中,$ED(S_1,S_2)$就是表中的最后一个元素值[5].给定查询 q 及其查询半径 r,由于只需在过滤后的候选集中找到与 q 的编辑距离落在 r 范围内的序列,我们无需精确计算此编辑距离,有可能只要计算两条序列的前缀子序列间的编辑距离即可,即两条序列的最优比对表只需部分计算.

在对序列 $S=s_1s_2\ldots s_m$ 和 $P=p_1p_2\ldots p_n$ 进行比对求其编辑距离时,设其最优比对表为 $T(n+1$ 行和 $m+1$ 列),则有下面性质:

性质 1. 设两序列 S 和 P 的最优比对表为 T,$T(i,j)$表示 T 中第 i 行和第 j 列的值,$\Delta_i=0$ 或 1,$\Delta_j=0$ 或 1,且满足 $i-\Delta_i\ge 0, j-\Delta_j\ge 0$,其中 $0\le i\le n, 0\le j\le m$,则有 $\left|T(i,j)-T(i-\Delta_i,j-\Delta_j)\right|\le 1$ 成立.

证明:最优比对表的生成过程是:

$$T(0,j)=j, T(i,0)=i (0\le i\le n, 0\le j\le m)$$

和递归式

$$T(i,j)=\min\begin{cases}T(i-1,j-1)+P(s_i,p_j)\\T(i,j-1)+1\\T(i-1,j)+1\end{cases}, i\ge 1, 且 j\ge 1,$$

其中如果 $s_i=p_j$,则 $P(s_i,p_j)=0$,否则,$P(s_i,p_j)=1$.由归纳法易得证. □

性质 1 反映了在生成的最优比对表中,任何一个值与其左边、左上角和上边近邻的差值的绝对值不超过 1,表 1 所示的序列 *writers* 和 *vintner* 的最优比对表反映了性质 1.这个性质可以用来证明下面的定理 3.

Table 1 Optimal alignment table of sequence *writers* and *vintner*

表 1 序列 *writers* 和 *vintner* 的最优比对表

$T(i,j)$	–	*w*	*r*	*i*	*t*	*e*	*r*	*s*
–	0	1	2	3	4	5	6	7
v	1	1	2	3	4	5	6	7
i	2	2	2	2	3	4	5	6
n	3	3	3	3	3	4	5	6
t	4	4	4	4	3	4	5	6
n	5	5	5	5	4	4	5	6
e	6	6	6	6	5	4	5	6
r	7	7	6	7	6	5	4	5

定理 3. 设序列 $S=s_1s_2\ldots s_m$ 和 $P=p_1p_2\ldots p_n (m\ge n)$的最优比对表为 T,则当 $i\le j\le n$ 时,有 $T(i,i)\le T(j,j)$.

证明:我们证明 $T(i,i)\le T(i+1,i+1)$,即可证明原定理.由表 T 的生成过程:

(1) 若 $T(i+1,i+1)=T(i,i)+P(s_i,t_i)$,由 $P(s_i,t_i)=0$ 或 1,有 $T(i,i)\le T(i+1,i+1)$.

(2) 若 $T(i+1,i+1)=T(i+1,i)+1$,由性质 1,$|T(i+1,i)-T(i,i)|\le 1$,分 3 种情况:

① $T(i+1,i)-T(i,i)=1$,$T(i+1,i+1)=T(i,i)+2$,$T(i,i)<T(i+1,i+1)$.

② $T(i+1,i)-T(i,i)=0$,$T(i+1,i+1)=T(i,i)+1$,$T(i,i)<T(i+1,i+1)$.

③ $T(i+1,i)-T(i,i)=-1$,$T(i+1,i+1)=T(i,i)$.

(3) 若 $T(i+1,i+1)=T(i,i+1)+1$,证明与(2)类似,也有 $T(i,i)\le T(i+1,i+1)$.

综合上述(1)~(3),有 $T(i,i)\le T(i+1,i+1)$. □

定理 3 表明,在两序列的全局比对中,其局部等长前缀子序列的编辑距离具有非递减性质.如表 1 中两个不同阴影部分所示,有 $ED(wr,vi)\le ED(wri,vin)$.对于不同长度的序列,下面的定理 4 表明,在进行其编辑距离的比较时,编辑距离的计算只需部分求出即可.

定理 4. 设有序列 $S=s_1s_2\dots s_m$,$P=p_1p_2\dots p_n$($m\ge n$)和值 r,若存在某个 $k\le n$,使得 $ED(s_1s_2\dots s_k,p_1p_2\dots p_k)-(m-n)>r$ 成立,则有 $ED(S,P)>r$.

证明:设 S 和 P 的最优比对表为 T,由性质 1,$T(n,m)+1\ge T(n,m-1)$,$T(n,m-1)+1\ge T(n,m-2)$,…,$T(n,n+1)+1\ge T(n,n)$,即 $T(n,m)\ge T(n,n)-(m-n)$.而由定理 3 可知,当 $k\le n$ 时,$T(n,n)\ge T(k,k)$.所以当存在某个 $k\le n$,使得 $ED(s_1s_2\dots s_k,p_1p_2\dots p_k)-(m-n)>r$ 成立时,有 $ED(S,P)=T(n,m)\ge T(n,n)-(m-n)\ge T(k,k)-(m-n)>r$. □

若设查询序列为 S,P 为候选集中序列,定理 4 表明,只要部分计算 $ED(S,P)$,即可知 P 是否在查询 S 的结果集中.如表 1 所示,若 *writers* 为查询序列,其查询半径为 2,*vintner* 为候选集中序列,则只需计算 $ED(wri,vin)$(其值为 3),表中虚线部分的值无需计算就可推知 *vintner* 不在 *writers* 的查询结果中.

由第 2.1 节~第 2.3 节的 3 种优化策略可知,每种优化策略都能不同程度地为当前查询过滤一部分数据或得到部分的查询结果,所以这 3 种优化策略可综合在一起共同加速当前查询.由于我们在实验中发现,一般利用先前查询结果的过滤能力最强,所以在保存了满足条件的 k 个先前查询结果的前提下,我们先运行 Filter_By_PreResult 方法,若 k 个先前查询结果没有建立起来,可利用后两种优化策略以在线的方式先建立起先前查询结果集.综合 3 种优化策略的 IRI 整体算法描述如下:

算法 2. IRI.

输入:查询序列 q 和查询半径 r,查询数据库 D,先前查询结果 R 及需要保存的结果集大小 k(预计算好的);

输出:查询 q 的查询结果集 *q-Res* 以及更新后的先前查询结果集 R'.

1. 利用 RI 算法,在 D 中为每条序列分配参考子集 V;
2. *Filter S* 置空,*Res* 置空;$|R|\leftarrow 0$; //初始化 q 的过滤集和部分查询结果集;
3. If ($|R|$大小等于 k)
4. 调用 Filter_By_PreResult 获得 q 的过滤集 *Filter S* 和部分结果集 *Res*;
5. If (R 中有与 q 相同的查询及相同查询半径 r)
6. $q-Res=Res$;返回;
7. *Filter S*=*Filter S*∪(在($D-$(*Filter S*∪*Res*))中利用 V 所获得的下界和频率距离 FD 产生的过滤集);
8. *Res*=*Res*∪(在($D-$(*Filter S*∪*Res*))中利用 V 所获得的上界和海明距离 HD 产生的部分结果集);
9. $q-Res=Res$∪(在候选集($D-$(*Filter S*∪*Res*))中利用部分编辑距离计算求出 q 的部分结果集);
10. If ($|R|$的大小<k)
11. $R'\leftarrow R$∪(q 序列及其查询结果 $q-Res$);$|R|\leftarrow|R|+1$;
12. Else
13. $R'\leftarrow$利用 q 序列及其查询结果 $q-Res$ 更新 R; //先前查询结果集的启发式更新

算法 2 首先利用 RI 算法中最大剪枝策略在 S 中找到参考集 V(第 1 行),然后利用先前查询结果来加速 q 的查询,若有以前的查询副本,则可直接获得 q 的全部查询结果(第 3 行~第 6 行).接着,在利用 Filter_By_PreResult 算法获得的过滤后的数据集中,用基于序列特征更紧的上界和下界来进一步扩大 q 的过滤集和部分结果集(第 7 行~第 8 行).最后,在过滤后的候选集合 $D-$(*Filter S*∪*Res*),用部分编辑距离的计算来提高序列比较的效率,并用启发式方法更新先前查询结果集.

3 实验结果及其分析

所有实验在 2.0GHz 的 PC 上进行,内存为 1 GB,算法用 Java 语言实现.为了比较现有算法 RI 和文中提出的 3 种优化策略的改进算法,我们分别把利用先前查询结果集、基于序列特征的编辑距离上下界以及编辑距离部

分计算的 RI 算法称为 PreR_RI,TBound_RI 和 PED_RI,把全部利用 3 种优化策略的改进 RI 算法记作 IRI.

实验数据集来自真实的 DNA 数据和蛋白质数据.其中 DNA 数据取自 Genbank 数据库[6](ftp://ftp.ncbi.nih.gov/genbank/)中生物体 E.Coli(K-12MG1655),DNA 序列数据的字母表大小是 4(A,G,C,T).我们通过截取 E.Coli 数据库中不重叠序列来构造 4 个长度分别 25,50,100 和 200 的序列数据库 DNA25,DNA50,DNA100 和 DNA200,4 个数据库的序列条数都为 20 000.蛋白质数据来自著名的蛋白质数据库 SwissProt(ftp://ftp.ebi.ac.uk/pub/databases/swissprot/)中真核生物体内氨基酸序列,蛋白质序列的字母表大小是 20.我们从中随机取 10 000 条蛋白质序列构造两个蛋白质数据库 ProDB1 和 ProDB2,序列最大长度达到 500.其中 ProDB1 的大小为 7 000,ProDB2 的大小为 3 000.

对于上述构造的每个数据库,我们从中取出同物种的不同部分序列来作为查询序列,这些查询序列是不重叠的,查询序列条数为 100.此外,我们还构造两个 DNA 序列查询集,一个来自 Genbank 中的物种 D_rerio(斑马鱼),另一个来自生物体数据库的 M_musculus(小鼠).两个查询集都是长度为 100 的 100 条序列.除非特别说明,各查询的查询半径采取与文献[4]中类似的方法,在查询序列长度的 1%~10%之间取值.

3.1 查询性能

影响算法性能的关键是序列间两两编辑距离的完全计算,与文献[28]相同,所以,我们用序列间距离的完全计算次数来衡量算法的性能指标.实验结果都是由查询集上所获得的性能指标值取平均得到.

算法 PreR_RI 涉及到先前查询集大小 k 的选取.由第 2.1 节的命题 1 中式(5)可知,k 值的选取取决于当前查询命中先前某一查询的概率 p 和先前查询结果集大小参数 t 和 c.类似于文献[29]中的方法,p 可由实验数据集中两两序列间编辑距离的等距直方图模拟的距离分布得到,而参数 t 和 c 可由每个查询结果集大小与其半径的近似关系得出.当得到参数 p,t 和 c 后,代入第 2.1 节命题 1 中的式(5),取满足内存要求的尽可能大的 k 值.

因为 IRI 算法中基于序列特征的编辑距离上下界是基于 RI 算法的,且还保存了先前查询结果集,所以 IRI 建立索引的时间比 RI 要长.但建立索引的操作只执行一次,即当满足下界的 k 个查询结果集和每个数据点的参考子集建立起来之后,IRI 建立索引的时间被后面的查询所平摊后可以忽略不计.所以为了方便起见,下面的 IRI 算法和 PreR_RI 算法的性能计算都是基于索引建立以后进行的.

3 种优化算法与 RI 算法的性能比较如图 2 和图 3 所示.

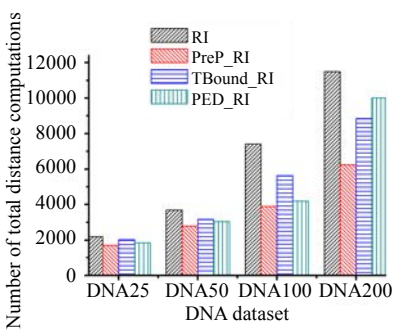

Fig.2 Performance comparison on DNA dataset
图 2 在 DNA 数据集上的性能比较

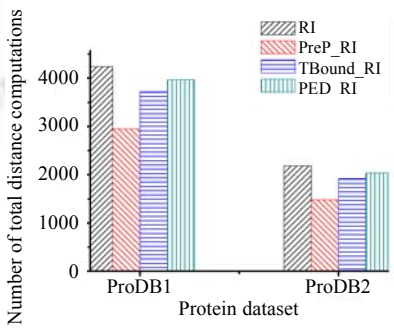

Fig.3 Performance comparison on Protein
图 3 在蛋白质数据集上的性能比较

从图中可以看出,3 种优化方法都能不同程度地提高查询性能.在图 2 中,由查询序列生成方法可知,随着 DNA 序列数据库长度的增加,一般查询半径也在增加,致使查询结果集扩大,从而有 4 种方法的距离计算次数也都不断增加.但当设置了合适的先前查询结果集大小 k 后, PreR_RI 的计算次数增长较 RI 要慢,这主要得益于 PreR_RI 采用的启发式更新查询结果的方法,随着查询半径的增大,PreR_RI 的过滤集或得到部分结果集也在增大,从而部分减少了计算次数,使得相对于算法 RI,PreR_RI 总的计算次数增长速度较慢.TBound_RI 的性能提高程度相对于 PreR_RI 较低,这是因为 RI 算法中利用三角不等式上下界的过滤集很大部分也在 TBound_RI 过滤

集中.PED_RI 算法受查询半径的影响较大,当查询半径较大时,满足计算部分编辑距离条件的可能性降低,因而性能提高有限,这在图 3 中也有体现,这是由于蛋白质数据库的平均长度较 4 个 DNA 数据库要长,因而查询半径也相对较大.

RI 算法与 3 种优化策略综合算法 IRI 的性能比较如图 4 和图 5 所示.IRI 在 DNA 和蛋白质数据库上的平均性能分别提高 45%和 52%,且利用先前查询结果策略是提高性能的最主要因素.

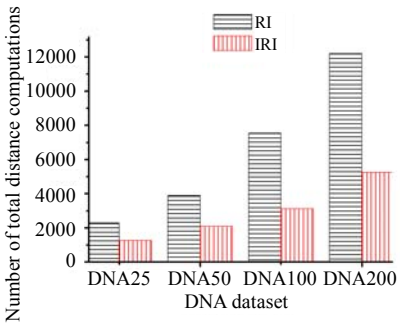

Fig.4 Performance comparison between RI and IRI on DNA dataset

图 4 RI 与 IRI 在 DNA 数据集上的性能比较

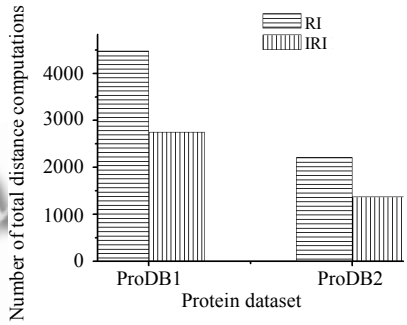

Fig.5 Performance comparison between RI and IRI on Protein dataset

图 5 RI 与 IRI 在蛋白质数据集上的性能比较

3.2 参数影响

考虑两个关键参数对查询性能的影响:先前查询结果集大小 k 和用户输入的查询半径参数 r.在数据集 DNA100 中计算得到的 k 值下界是 125.图 6 给出了不同 k 值对 PreR_RI 算法性能的影响情况.事实上,当 k 值大于 104 时,PreR_RI 一开始能够提高性能,在 k 取值 200 时,性能提高幅度最大,但在 k 再增大以后,性能提高不是很明显,实验结果表明,这是由于当保存过多的先前查询结果时,很多先前查询结果对当前查询所产生的过滤集或部分结果集大部分是重叠的.

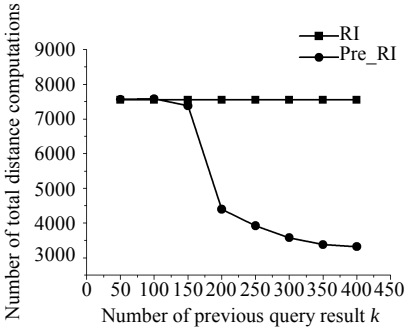

Fig.6 Impact of parameter k on performance of PreR_RI

图 6 参数 k 对 PreR_RI 的性能影响

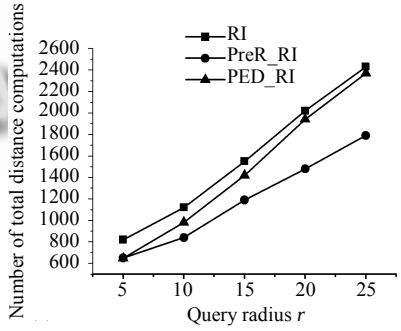

Fig.7 Impact of parameter r on performance of PreR_RI and PED_RI

图 7 参数 r 对 PreR_RI 和 PED_RI 的性能影响

此外,我们在实验中发现,当 k 取 100 时,完整距离计算次数反而比 k 取 50 时要多(分别为 7 568 和 7 584),其原因分析如下:在进行保存 50~100 个先前查询结果时,由于还未到达理论值 125,根据我们的算法 IRI,只是加进现有查询结果,而不是选择较大半径的查询结果进行更新.所以,尽管在先前 50 个查询结果的基础上又加进了 50 个查询结果集,但由于这 50 个新增查询的平均查询半径较小,使得一方面,这 50 个查询结果集较小(定理 1 中第 1 种和第 3 种情况命中时的过滤集和得到的部分结果集较小).另一方面,第 2 种情况的命中概率较低,所以新增的 50 个先前查询结果集对当前查询的性能提高有限,使得应用定理 1 进行过滤时计算的编辑距离次数大

于实际得到的过滤集或部分结果集大小.完整编辑距离计算次数反而比 k 取 50 时要多.但在 k 值为 150 以后,我们不断地选择半径较大的查询结果进行更新,使得平均的先前查询结果集较大(定理 1 中第 1 种和第 3 种情况命中时的过滤集和得到的部分结果集较大),且第 2 种情况的命中率也在上升,所以查询效率在稳定地提高.

我们在数据集 ProDB1 中选取一条长度为 220 的序列作为查询序列.因为查询半径对算法 TBound_RI 的影响规律不是很明显,我们只比较不同查询半径 r 对 PreR_RI 和 PED_RI 算法的影响,如图 7 所示.从图中可以看出,随着 r 的增大,3 种算法的完全距离计算次数都在增加,但由于 PreR_RI 的启发式更新先前查询结果集的方法,使得其获得过滤集和部分结果集的数量在增大,从而增长幅度较 RI 要慢.但是随着 r 的增加,PED_RI 中计算部分编辑距离的可能性减少,从而性能提高也在不断减少.我们取其他查询序列加以验证,也得到了类似的结论.

当先前查询结果集的平均查询半径在不断变化时,我们考察了当前查询在定理 1 中 3 种情况下分别被命中的次数变化情况.我们保存 100 个先前 DNA 序列查询结果集,查询集为 100 条 DNA 序列,来自生物体数据库 M_musculus(小鼠).对于每次的先前查询结果集,我们统计 100 次查询在定理 1 中 3 种情况下分别被命中的总次数.然后,每次对 100 个先前查询结果集中的查询半径增大其查询序列长度的 5%,并更新其查询结果集,再对同样的查询集统计在定理 1 中 3 种情况下分别被命中的总次数.图 8 给出了其变化情况.从图中可以看出,随着先前查询序列平均半径 $\bar{r}$ 的增大,定理 1 中第 1 种情况和第 3 种情况被命中的次数在逐步减少,且总体的数据间距离分布使得总体的命中率小幅下降,而第 2 种情况被命中次数在逐渐上升.如观察 1 所述,由于第 2 种情况的命中率不超过 30%,情况 2 被触发后的过滤集会增大.但从我们的实验中发现,100 次查询在不同 $\bar{r}$ 值下的过滤集和部分结果集之和的平均值仍是逐步增大的,这主要得益于第 2 种情况被命中的次数有所增加,而第 2 种情况被命中后的过滤集要远大于第 1 种和第 3 种情况,这也说明我们保存较大查询半径的先前查询结果集有利于提高查询效率.并且在 4 组不同实验中,每组中第 1 种情况和第 3 种情况分别是被命中次数最多和最少(第 3 种情况被命中的次数分别是 76,19,6 和 2)的,这表明,在一般的查询中,一个查询的查询结果相对于查询数据库来说只占很小一部分,从而一个先前查询结果被当前查询过滤掉的可能性较大,而成为当前查询部分结果的可能性较小.

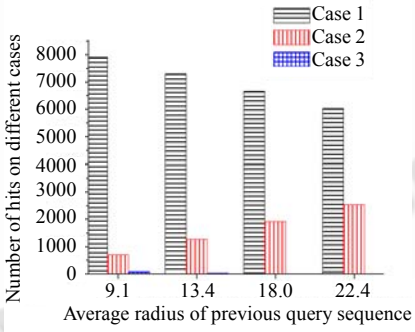

Fig.8 Number of hits on different cases vs. average radius of previous query sequences

图 8 各种情况被命中次数随先前查询序列平均半径的变化

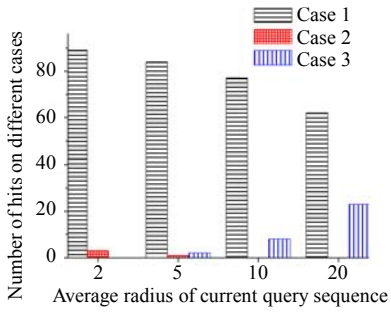

Fig.9 Number of hits on different cases vs. radius of current query sequences

图 9 各种情况被命中次数随查询序列半径的变化

我们又考察了定理 1 中 3 种情况被命中的次数随着当前查询序列半径而变化的情况.先前查询结果集仍使用图 8 中的 100 个先前 DNA 序列查询结果集并固定住不进行更新,查询序列在数据集 DNA100 中任取 1 条,分别变化其查询半径为序列长度的 2%,5%,10%,20%.然后重复 50 次取 3 种情况分别被命中次数的平均值.从图 9 中可以看出,第 1 种情况和第 2 种情况被命中的次数(第 2 种情况被命中的次数分别为 3,1,0,0)在减少,尽管第 3 种情况被命中的次数在增加,但从实验中发现,查询的过滤集和部分结果集之和整体上在逐步减少.这是由于随着查询半径增大的查询的不断进行,我们没有对先前查询结果集进行启发式更新,致使过滤集较大的第 2 种情况被命中的次数在减少,从而查询效率的提高程度也在降低.

4 结论与展望

本文在现有的基于参考集索引方法 RI 的基础上,提出了 3 种优化策略:利用先前查询结果来加速现有查询,设计基于序列特征的上下界使得过滤的上下界更紧,计算部分的序列编辑距离.通过理论分析和实验证明,这 3 种优化策略能够不同程度地提高查询性能,其中,利用先前查询结果加速现有查询的效果最为明显,且综合这 3 种优化策略的算法 IRI 在性能上较 RI 有较大的提高.今后我们的工作将对在查询数据库允许有插入和删除的动态环境下提高查询性能的问题上加以研究.

References:

[1] Dong GZ, Pei J. Sequence Data Mining. Heidelberg: Springer-Verlag, 2007.

[2] Sarawagi S. Advanced Methods for Knowledge Discovery from Complex Data. Heidelberg: Springer-Verlag, 2005. 153−187.

[3] Zhu YY, Xiong Y. DNA sequence data mining technique. Journal of Software, 2007,18(11):2766−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

[4] Kahveci T, Singh AK. An efficient index structure for string databases. In: Apers P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass R, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases (VLDB 2001). Rome: Morgan Kaufmann Publishers, 2001. 351−360.

[5] Gusfield D. Algorithms on Strings, Trees, and Sequences. New York: Cambridge Press, 1997.

[6] National Center for Biotechnology Information. Genbank database. 2008. http://www.ncbi.nlm.nih.gov/

[7] Benson DA, Karsh-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL. Genbank. Nucleic Acids Research, 2000,28(1):15−18. [doi: 10.1093/nar/28.1.15]

[8] Ukkonen E. Approximate string-matching with *q*-gram and maximal matches. Theoretical Computer Science, 1992,92(1):191−211. [doi: 10.1016/0304-3975(92)90143-4]

[9] Ukkonen E. Algorithms for approximate string matching. Information and Control, 1985,64(1-3):100−118. [doi: 10.1016/S0019-9958(85)80046-2]

[10] Baeza-Yates R, Navarro G. Faster approximate string matching. Algorithmica, 1999,23(2):127−158. [doi: 10.1007/PL00009253]

[11] Buhler J. Efficient large-scale sequence comparison by locality-sensitive hashing. Bioinformatics, 2001,17(5):419−428 [doi: 10.1093/bioinformatics/17.5.419]

[12] Pearson WR, Lipman DJ. Improved tools for biological sequence comparison. PNAS, 1998,85(8):2444−2448. [doi: 10.1073/pnas.85.8.2444]

[13] Altschul S, Gish W, Miller W, Meyers EW, Lipman DJ. Basic local alignment search tool. Journal of Molecular Biology, 1990, 215(3):403−410.

[14] Ma B, Tromp J, Li M. PatterHunter: Faster and more sensitive homology search. Bioinformatics, 2002,18(3):440−445. [doi: 10.1093/bioinformatics/18.3.440]

[15] Li C, Wang B, Yang XC. VGRAM: Improving performance of approximate queries on string collections using variable-length grams. In: Koch C, Gehrke J, Garofalakis NN, Srivastava D, Aberer K, Florescu D, Chan CY, Ganti V, Kanne CC Klas W, Neuhold EJ, eds. Proc. of the VLDB. Vienna: ACM Press, 2007. 303−314.

[16] Yang XC, Wang B, Li C. Cost-Based variable-length-gram selection for string collections to support approximate queries efficiently. In: Shasha D, Wang JTL, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Vancouver: ACM Press, 2008. 353−364.

[17] Li C, Lu JH, Lu YM. Efficient merging and filtering algorithms for approximate string searches. In: Proc. of the 24th Int'l Conf. on Data Engineering Cancun: IEEE Computer Society, 2008. 257−266. http://dx.doi.org/10.1109/ICDE.2008.4497434

[18] Weiner P. Linear pattern matching algorithms. In: Proc. of the 14th Annual Symp. on Foundations of Computer Science (FOCS). Iowa: IEEE Computer Society, 1973. 1−11. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4569722

[19] Manber U, Mysers E. Suffix arrays: A new method for on-line string searches. SIAM Journal on Computing, 1993,22(5):935−948. [doi: 10.1137/0222058]

[20] McCreight EM. A space-economical suffix tree construction algorithm. Journal of the ACM, 1976,23(2):262−272. [doi: 10.1145/321941.321946]

[21] Burkhardt S, Andreas C, Ferragina P, Peter Lenhof H, Rivals E, Vingron M. *q*-Gram based database searching using a suffix array (QUASAR). In: Proc. of the 3rd Annual Int'l Conf. on Computational Molecular Biology (RECOMB). Lyon, 1999. 77−83. http://portal.acm.org/citation.cfm?id=299460

[22] Muthukrishnan S, Sahinalp SC. Approximate nearest neighbors and sequence comparison with block operations. In: Proc. of the 32rd Annual ACM Symp. on Theory of Computing (STOC). Portland, 2000. 416−424. http://compbio.cs.sfu.ca/publications/nn.son.pdf

[23] Hunt E, Atkinson MP, Irving RW. A database index to large biological sequences. In: Appers P, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proc. of the VLDB. Morgan Kaufmann Publishers, 2001. 139−148.

[24] Wang GR, Ge J, Xu HY, Zheng RS. A sequence similarity query processing technique based on two-partitoning frequency transformation. Journal of Software, 2006,17(2):232−241 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/232.htm [doi: 10.1360/jos170232]

[25] Yianilos PN. Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proc. of the 4th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA). Austin, 1993. 311−321. http://portal.acm.org/citation.cfm?id=313789

[26] Bozkaya T, Ozsoyoglu M. Distance-Based indexing for high-dimensional metric spaces. In: Peckham J, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Tucson, 1997. 357−368. http://portal.acm.org/citation.cfm?id=253345

[27] Buhler J. Provably sensitive indexing strategies for biosequence similarity search. In: Proc. of the 6th Annual Int'l Conf. on Computational Molecular Biology (RECOMB). Washington, 2002. 90−99. http://portal.acm.org/citation.cfm?id=565208

[28] Venkateswaran J, Lachwani D, Kahveci T, Jermaine C. Reference-Based indexing of sequence databases. In: Dayal U, Whang KY, Lomet DB, Alonso G, Lohman GM, Kersten ML, Cha SK, Kim YK, eds. Proc. of the VLDB. Seoul: ACM, 2006. 906−917.

[29] Ciaccia P, Patella M, Zezula P. A cost model for similarity queries in metric spaces. In: Proc. of the 17th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS). Seattle: ACM Press, 1998. 59−68. http://portal.acm.org/citation.cfm?id=275495

[30] Sahinalp SC, Tasan M, Macker J, Ozsoyoglu ZM. Distance based indexing for string proximity search. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Computer Society, 2003. 125−136.

[31] Jr. Traina C, Traina AJM, Faloutsos C. Distance exponent: A new concept for selectivity estimation in metric trees. In: Proc. of the 16th Int'l Conf. on Data Engineering. San Diego: IEEE Computer Society, 2000. 195.

[32] Motwani R, Raghavan P. Randomized Algorithms. New York: Cambridge University Press, 1995.

附中文参考文献:

[3] 朱扬勇,熊赟.DNA 序列数据挖掘技术.软件学报,2007,18(11):2766−2781. http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

[24] 王国仁,葛健,徐恒宇,郑若石.基于二分频率变换的序列相似性查询处理技术.软件学报,2006,17(2):232−241.http://www.jos.org.cn/1000-9825/17/232.htm [doi: 10.1360/jos170232]

戴东波(1977−),男,湖南娄底人,博士,主要研究领域为数据挖掘,数据库,生物信息.

熊赟(1980−),女,博士,讲师,CCF 会员,主要研究领域为数据挖掘,数据库,生物信息.

朱扬勇(1963−),男,博士,教授,博士生导师,CCF 会员,主要研究领域为数据挖掘,数据库,生物信息.