一种基于尾匹配 q-gram 的近似串匹配算法

孙德才[1] 王晓霞[2]
(渤海大学信息科学与技术学院 锦州 121013)[1] (渤海大学大学计算机教研部 锦州 121013)[2]

摘 要 近似串匹配是生物信息学、文本检索、信号处理等领域的一个基础问题,如何提高近似串匹配的速度一直都是研究的关键问题。提出一种新的在大文本库中快速查找近似匹配的无损过滤算法。为保证在大文本库中的匹配速度,本算法使用了查询速度较快的 q-gram 索引。为通过提高过滤算法的过滤效率达到提升算法整体性能的目的,详细分析了含有匹配串的文本区域,提取了一些基于尾匹配 q-gram 特征的新过滤条件,然后用这些特征优化了过滤算法的过滤标准。实验数据表明,新过滤条件有效地提高了算法的过滤效率,提升了算法的整体性能。结果显示新算法适合各种匹配错误率下的近似匹配,算法的通用性较强。

关键词 近似串匹配,过滤算法,q-gram 过滤,q 元语法
中图法分类号 TP391.3 **文献标识码** A

Approximate String Matching Using Tail Matched q-gram

SUN De-cai[1] WANG Xiao-xia[2]
(College of Information Science and Technology, Bohai University, Jinzhou 121013, China)[1]
(Teaching and Research Institute of College Computer, Bohai University, Jinzhou 121013, China)[2]

Abstract Approximate string matching is a basic problem in computational biology, text retrieval and signal processing, etc., and how to improve the matching speed is a key issue up to now. Here, a new lossless q-gram filter was proposed to enhance the performance of finding true matches in a large text database for a given query. Q-gram index is employed as the index structure of large text database for its fast searching speed. To improve new filter's performance by enhancing its filtration efficiency, some new features based on tail matched q-gram were extracted from text that includes true matches. In new filter, more irrelevant text is eliminated in filtration phase by using a filtration criterion which is enhanced by new extracted features, and the unfiltered text is verified by smith-waterman algorithm to locate the positions of all true matches. The experimental results demonstrate that the proposed filter's filtration efficiency and performance are both enhanced. As a result, new filter algorithm has strong commonality and is suitable for approximate string matching on condition of various matching error ratio.

Keywords Approximate string matching, Filter algorithm, q-gram filter, q-gram

1 引言

随着因特网的发展和普及,信息已实现了全球的共享与交互,信息的采集、传播和使用都变得非常便利。但巨大的信息量使得个人的接受能力严重超载,因此如何从这巨大的信息中迅速、准确地获取所需要的信息,一直是计算机科学研究中的基本问题。众多学者一直从事近似数据匹配问题的研究,如文献[1]等。目前,文本仍是使用最频繁、最广泛的信息载体。近似串匹配技术能从大文本库中快速地找出与查询相近的字符串,是解决问题最直接的方法。近似串匹配技术在众多领域都有广泛的应用,如生物信息学、文本检索、信号处理和模式识别等。

近似串匹配(Approximate String Matching, ASM)[2]是允许有错误发生的字符串匹配。近似串匹配的最基本的形式是在文本串 T 中查找所有与模式串 Q 之间错误数不大于一定阈值的所有匹配串。字符串间的错误数可采用编辑距离、汉明距离、最长公共子串距离和马尔可夫距离等表示,而编辑距离[3]是最常用的表示方式,一般用 $ED(A,B)$ 表示字符串 A 和 B 间的编辑距离。

近似串匹配算法根据匹配模式分两种[4]:On-line 和 Off-line。On-line 模式的匹配算法只允许对模式串进行预处理,而对文本串不能进行任何预处理操作。该模式的匹配算法已被广泛研究[2]。但当文本串非常长、且要频繁使用不同的模式串进行匹配时,即使采用目前最快的 On-line 模式也无法满足实际应用的需求,此时就需要 Off-line 算法。Off-line 算法允许对文本串进行预处理,预处理方式是为文本库建立适当

到稿日期:2013-08-19 返修日期:2013-10-14 本文受辽宁省社科联 2014 年度辽宁经济社会发展立项重点课题(2014lslktzdian-04),国家自然科学基金项目(61173142, 61202462),辽宁省教育厅一般项目(L2013422, L2012397),辽宁省"百千万人才工程"项目(2012921058)资助。
孙德才(1979—),男,博士,讲师,主要研究方向为近似串匹配、文本复制取证等,E-mail:sdecai@163.com;王晓霞(1977—),女,硕士,讲师,主要研究方向为近似串匹配、入侵检测等。

的索引结构，常用的索引结构有后缀树[5]、后缀数组[6]和 q-gram 索引[7,8]等。q-gram 索引[7,8]是一个索引项为 q-gram 的倒排索引。q-gram 索引因其占用内存少、容易管理等优点被广泛应用于生物信息学、文本检索等领域，如文献[8]基于二级 q-gram 索引的近似串匹配方法；文献[9]采用变长的 q-gram 索引处理数据库中字符串的快速匹配问题等。

近年来，过滤算法因采用过滤技术能快速抛弃大量无关文本片段，匹配速度非常快，而得到众多学者的广泛研究。过滤算法根据采用的匹配方法不同，其中绝大部分算法可以归为两类：

1)基于精确匹配子串的。文献[10,11]把模式串分成 $k+s$ 部分，然后验证在文本串中至少出现 s 部分且位置正确的文本区域，例如文献[10]中采用了 $s=1$，而文献[11]中采用了 $s>1$；文献[12-14]对文本串和模式串进行连续 q-gram 拆分，验证那些出现一定阈值以上 q-gram 的文本区域；文献[15]结合了文献[10,11]的分割方法和文献[12-14]的 q-gram 阈值方法，提出了一种混合过滤算法。文献[16-18]从文本串中选取一些子串，称为种子，出现一定数目种子的文本范围将会被扩展和验证；文献[19,20]报道了基于留空种子的算法及相关优化。

2)基于近似匹配子串的。文献[21]把模式串分成 j 部分，验证存在至少一部分错误不大于 $\lfloor k/j \rfloor$ 的文本区域；文献[22]把模式串分割成 q-sample，在索引中查找与模式串的 q-sample有一定数量错误的文本串子串的位置，并验证。

基于近似匹配子串的方法因匹配子串过程复杂，一般匹配效率不高。而精确匹配子串的方法匹配子串过程简单、速度快，著名的算法主要有 BLAST 家族[16-17]、PatternHunter[19]、QUASAR[12]、SWIFT[14]等。BLAST[16]算法是一个启发式算法，通过精确地匹配查询串的子串(称之为种子)，然后对匹配到的种子进行扩展来查找匹配串。但对那些不存在种子的匹配串则无法检出，即不是无损算法。PatternHunter[19]是一个基于留空种子的 q-gram 过滤算法，速度较快，但该算法的敏感性受种子的选择、模式串长度、文本近似程度等因素的限制，目前众多研究都集中在种子的优化上[20]。该类算法较适合固定长度、特定相似程度的匹配，算法的通用性较差。QUASAR[12]优化了 Jokinen 和 Ukkonen 的基本过滤定理[13]，算法的过滤过程简单，但过滤效率较低，使其整体的匹配速度较慢。SWIFT[14]采用了平行四边形的过滤区域，虽然过滤效率较高，但过滤过程复杂，使得其过滤时间也较长，影响了总体的匹配速度。

本文主要研究的是在大文本库 T 中快速查找与模式串 Q 间错误率不大于 e 的所有匹配串的问题，即 e 匹配问题。本文提出了一种基于尾匹配 q-gram 的近似串匹配无损过滤算法。该算法利用尾匹配 q-gram 的特征进行过滤，提高了过滤算法的过滤效率，减少了算法的验证时间，从整体上加快了近似串的匹配效率。第 2 节介绍了本文相关的基础知识；第 3 节给出了用于提取和描述新过滤条件的定义和定理；第 4 节详细地介绍了算法的匹配过程，并对算法进行了理论分析；第 5 节给出了实验结果和实验分析；最后给出了本文的结论。

2 基础知识

q-gram 是长度为 q 的字符串，其位置用其首字符在模式串或文本串中的偏移量表示。q-gram 索引主要包括两部分：词汇表和倒排列表。词汇表是 q-gram 项的集合，为能够快速定位任意 q-gram，词汇表经常采用哈希表或 B* 树。每个 q-gram 项后连有一个存储该索引项在文本中出现的地址集合，称为倒排列表。

编辑距离是指把一个字符串 A 经过编辑操作转变成字符串 B 所要进行的最小操作次数，这里的编辑操作包括插入、修改或删除 3 种。编辑距离计算的最基本方式是动态程序法，其主要过程是填充字符串 A 和 B 的编辑距离矩阵，然后在矩阵中回溯找出对应的编辑路径[2]。

最早提出基于 q-gram 的过滤定理的是 Jokinen 和 Ukkonen[13]，定理如下。

定理 1 如文本串 T 的子串 $T[j-|P|+1,\cdots,j]$ 内包含少于 $t(k)=|Q|+1-(k+1)q$ 个模式串 Q 的长度为 q 的子串(q-gram)，那么文本串 T 中一定不存在以 j 为结尾的子串与模式串 Q 的编辑距离不大于 k。

该定理描述了一个存在匹配串的文本区域具有的最基本的 q-gram 特征，即该区域与模式串间至少共享 $t(k)$ 个 q-gram。

错误率是字符串与模式串之间的编辑距离与模式串长度的比值，即 $e=k/|Q|$，它描述了一个字符串与模式串间的近似程度。匹配错误率(Matching Error Ratio)是近似串匹配中一个重要的匹配参数，即找出所有错误率不大于匹配错误率的字符串。基于 q-gram 的过滤算法匹配速度快，但当匹配错误率大于 $\frac{1}{q}-\frac{1}{|Q|}+\frac{1}{(|Q|q)}$ 时，算法将发生崩塌[23]。

过滤效率(Filtration efficiency)是描述算法过滤阶段抛弃无关片段的能力，定义为：

$$f_e=\frac{n-n_f}{n-n_t} \tag{1}$$

其中，n 表示整个文本库的长度，即 $n=|T|$，n_f 表示过滤算法在本次匹配中未过滤掉的文本区域的总长度，而 n_t 表示本次匹配文本库中真实存在的所有匹配串的长度总和。

3 新过滤特征提取

过滤算法的过滤效率代表算法抛弃无关文本区域的能力，而过滤特征的好坏又决定了算法过滤效率的高低。为提高过滤算法的过滤效率，本文通过详细分析含有匹配串的匹配区域，得出了一些新的特征。为更好地描述这些特征，这里将给出相关的定义和定理。

定义 1 设串 T' 是文本串 T 的一个子串，如串 T' 与模式串 Q 的编辑距离不大于 k，即 $ED(T',Q)\leqslant k$，则称 T' 是模式串 Q 的匹配串。设串 M 的长度为 q，即 M 是一个 q-gram 项，M 不仅是 T' 的子串，也是 Q 的子串，如 M 是 T' 与 Q 最小编辑距离对应的编辑路径上的最后一个(编辑前后位置对应) q-gram 项，则称 M 为 Q 与 T' 的尾匹配 q-gram。如图 1 所示，GC 是 A 和 B 串最小编辑距离对应的编辑路径上的尾匹配 2-gram，虽然二串间共享的 CA 也相对处于尾部，但在编辑路径上位置并不对应，所以 CA 不是尾匹配 2-gram。

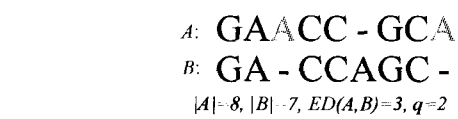

图 1 尾匹配 q-gram 示例

定理 2 设 T' 是文本串 T 的子串，如模式串 Q 与 T' 编辑

距离不大于 k，即 $ED(T',Q)\leqslant k$，且它们的尾匹配 q-gram 项 M 在 Q 中的位置为 p，在 T 中的位置为 t（位置都从 0 开始），则：

1）T' 串一定在 $T[t-|Q|+q-k, t+kq+q-1]$ 内。

2）尾匹配 q-gram 项 M 在 Q 中的最左位置 p 满足条件 $p\geqslant |Q|-(k+1)q$。

证明：

1）这里分两种情况讨论：

情况 a：T' 与 Q 间的编辑操作都集中在该尾匹配 q-gram 项 M 的左边。此时，M 是 Q 的最后一个 q-gram。根据尾匹配 q-gram 的定义可知，M 为 T' 的最后一个 q-gram，且位置为 t。由于所有的编辑操作都集中在 M 的左边，且替换操作不改变串长度，插入操作增长串长度，删除操作减少串长度，则当使用编辑操作把 Q 转化成 T' 的 k 个编辑操作都是插入时，T' 在 T 中位置最靠左，即 T' 首字符在 T 中的最左位置为 $t-|Q|+q-k$。如图 2 中 T_a' 串与模式串 Q 间的尾匹配串为 ATC。

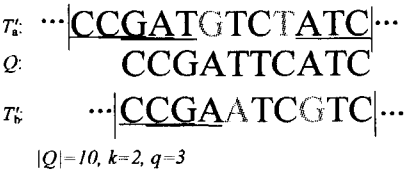

图 2　不同位置的尾匹配 q-gram

情况 b：T' 与 Q 间的编辑操作都集中在该尾匹配 q-gram 项 M 的右边。此时左边不存在编辑操作，即 M 前面的 q-gram 都一一对应匹配，根据定理 1 可知 T' 内至少存在 $|Q|+1-(k+1)q$ 个连续的 q-gram，总长度为 $|Q|-kq$，去除尾匹配 q-gram 剩余字符串的长度为 $|Q|-kq-q$。根据尾匹配 q-gram 的定义和 M 在 T' 中的位置为 t，可知 T' 在 T 中的首字符位置为 $t-|Q|+kq+q$。在原连续的 q-gram 上，删除和修改一个字符都最多影响 q 个 q-gram，插入一个字符最多影响 $q-1$ 个 q-gram。因为 M 后再无匹配的 q-gram，所以此时 T' 的长度最长不会超过 $|Q|$，所以 T' 串的尾字符的最右位置为 $t+kq+q-1$。如图 2 中 T_b' 串与模式串 Q 间的尾匹配串为 CGA。

综上所述，T' 一定在 $T[t-|Q|+q-k, t+kq+q-1]$ 范围内。

2）如 1）证明中当尾匹配 q-gram 项 M 左边无编辑操作时，M 的位置最靠左边，即对应情况 b。此时尾匹配 q-gram 项 M 前面至少存在 $|Q|+1-(k+1)q$ 个 q-gram（包括 M），因此其位置为 $|Q|-(k+1)q$，即 $p\geqslant |Q|-(k+1)q$。

定义 2　对模式串 Q 采用窗口长度为 q、滑动距离为 1 的 q-gram 项拆分，分得 q-gram 总数为 $|Q|-q+1$ 个，如两个 q-gram 相邻（即重叠 $q-1$ 个字符），则称这两个 q-gram 项为连续的，n 个连续的 q-gram 项称为 n 连续 q-gram，它们构成的子串称为 Q 的 n 连续 q-gram 串。如图 3 中 CCG 和 CGA 是 2 连续 q-gram，CCGA 是 2 连续 q-gram 串。

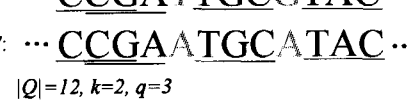

图 3　连续 q-gram 及连续 q-gram 子串

定理 3　如果文本串 T 中存在子串 T' 与模式串 Q 编辑距离不大于 k，即 $ED(T',Q)\leqslant k$，则串 T' 满足：

1）T' 与模式串 Q 间共享的 q-gram 数目不小于 $|Q|+1-(k+1)q$。

2）T' 中存在至多 $k+1$ 个模式串 Q 的连续 q-gram 串。

3）T' 中一定存在至少一个模式串 Q 的数目不少于 n 的连续 q-gram，且 $n\geqslant\lceil(|Q|+1-(k+1)q)/(k+1)\rceil$。

证明：

1）模式串 Q 分得的总 q-gram 数为 $|Q|-q+1$ 个，T' 与 Q 编辑距离不大于 k。如把串 Q 经过编辑操作转换成 T'，则一个插入操作最多影响串 Q 中 $q-1$ 个 q-gram，修改和删除操作最多影响 Q 中 q 个 q-gram，即一个编辑操作最多影响 q 个 q-gram，则 k 个编辑操作最多影响 kq 个 q-gram，因此 T' 中至少还存在 $|Q|+1-(k+1)q$ 个原来模式串 Q 的 q-gram 未被修改，即定理 1。

2）模式串 Q 原来是一个整体，这 k 个编辑操作最多把这些未改变的 q-gram 项分割成 $k+1$ 个子串，即 $k+1$ 个连续 q-gram 子串，如图 3 中 2 个编辑操作把原连续的 q-gram 分割为 3 个连续 q-gram 串。

3）因 T' 中至多存在 $k+1$ 个模式串 Q 的连续 q-gram 串，则在这 $k+1$ 个连续 q-gram 串中至少存在一个不小于 $\lceil(|Q|+1-(k+1)q)/(k+1)\rceil$ 的连续 q-gram，如图 3 中至少存在一个 2 连续 q-gram。

定理 4　如文本串 T 中存在 T' 与模式串 Q 编辑距离不大于 k，即 $ED(T',Q)\leqslant k$，设 Q 与 T' 的尾匹配 q-gram 项 M 在 T' 中的位置为 t，则：

1）设 t_1 为从尾匹配 q-gram 项 M 的位置 t 开始向左数第 $k+1$ 个连续 q-gram 串的首字符位置，设 $t_m=\max(t_1, t-|Q|+q-k)$，则 T' 一定属于 $T[t_m, t+kq+q-1]$。

2）Q 与 $T[t_m, t+kq+q-1]$ 至少共享 $|Q|+1-(k+1)q$ 个 q-gram。

3）$T[t_m, t+kq+q-1]$ 中至少存在一个模式串 Q 的数目不少于 $\lceil(|Q|+1-(k+1)q)/(k+1)\rceil$ 的连续 q-gram。

证明：

1）根据定理 3 的 2）可知，T' 中最多包含 $k+1$ 个连续 q-gram 子串，从 t_1 开始到位置 t，共有 $k+1$ 个连续 q-gram 子串，所以 t_1 位置之前的任何连续 q-gram 子串一定不属于 T'，即 T' 的首字符位置一定不小于 t_1。又根据定理 2 的 1），T' 一定在 $T[t-|Q|+q-k, t+kq+q-1]$ 中。根据以上两条件可知，T' 一定在 $T[t_m, t+kq+q-1]$ 范围内。

2）根据 1）可知，串 T' 一定属于 $T[t_m, t+kq+q-1]$，则又根据定理 1 可知，Q 与 $T[t_m, t+kq+q-1]$ 至少共享 $|Q|+1-(k+1)q$ 个 q-gram。

3）因 T' 在 $T[t_m, t+kq+q-1]$ 内，则根据定理 3 的 3）可知 $T[t_m, t+kq+q-1]$ 中至少存在一个不小于 $\lceil(|Q|+1-(k+1)q)/(k+1)\rceil$ 的连续 q-gram。

4　基于尾匹配 q-gram 的过滤算法

本文提出一种基于尾匹配 q-gram 的近似串全局匹配算法（TMQ，Tail Matched Q-gram），解决从一个大文本库中查找给定模式串 Q 的所有 e 匹配问题。

4.1　TMQ 算法的匹配流程

本文算法主要包含文本库预处理、输入、过滤、验证和输

出5部分，如图4所示。

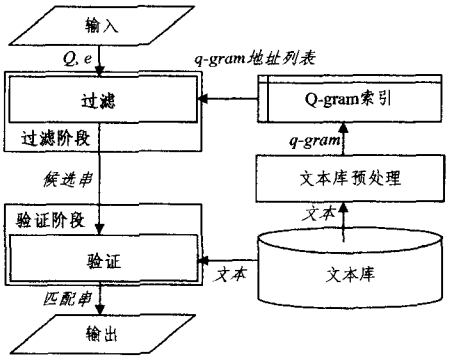

图4 TMQ算法流程图

4.1.1 文本库预处理

为进行近似匹配，首先需要为给定的文本库建立一个索引。因在同等内存环境下q-gram索引的查询速度要快于后缀数组[24]，因此本文预处理文本库采用了q-gram索引结构。建立q-gram索引时只需一次完整扫描文本库。首先文本库中所有的文件被看成一个在逻辑上首尾相连而成的长文本串T，并采用连续拆分方法从T中提取q-gram项信息。为提高q-gram的访问速度，这里采用哈希表作为词汇表。哈希表采用一个长度为σ^q的一位数组实现。哈希函数采用Karp-Rabin[25]函数。例如，$T[j,\cdots,j+q-1]$的哈希值采用式(2)计算。

$$KR(T[j,\cdots,j+q-1])=\sum_{i=j}^{j+q-1} I(T[i])\sigma^{j+q-1-i} \tag{2}$$

连续重叠拆分时，下一个q-gram的哈希值可由式(3)计算得出。

$$\begin{aligned}&KR(T[j+1,\cdots,j+q])\\&=(KR(T[j,\cdots,j+q-1])-I(T[j])\times\sigma^{q-1})\times\sigma+\\&\quad I(T[j+q])\end{aligned} \tag{3}$$

Q-gram索引的建立过程如下：

1)分配哈希表，长度为σ^q的一位数组，文本库中的所有文件按顺序编号，并按顺序访问，形成逻辑上的T。

2)读取一个文件内容，并存储该内容对应在T中的位置。然后依次提取该内容的q-gram项，并哈希q-gram项。最后根据q-gram项的哈希值找到在哈希表中的对应位置，并把该q-gram项的位置信息添加到对应的倒排列表中。

3)重复2)直到所有文件都处理完毕，q-gram索引建立完成。

q-gram索引的建立是按文件编号从小到大、文件内容从前到后的顺序处理的，所以索引中每个倒排列表都是升序序列。

4.1.2 输入

输入阶段需要用户输入要进行近似匹配的模式串Q，以及匹配错误率e。此时允许的最大编辑距离可通过$k=|Q|e$计算得出。

4.1.3 过滤

过滤阶段的主要任务是尽可能剔除那些不包含匹配串的文本区域，一个好的过滤器能够采用严格的过滤准则在非常短的过滤时间内抛弃大量的无关文本区域。过滤结束后将给出那些包含匹配串概率非常高的文本区域。

文本库已被视为长文本串T，为在T中找出模式串Q的所有匹配串，首先需要记录模式串Q中每个q-gram项在文本串T中的所有命中位置。这里使用一个长度为$|T|$的一位数组H来存储这些命中位置。如数组中$H[i]=1$则代表Q中存在q-gram项命中文本T的第i个位置，相反当值为0时代表无命中。因此，数组中每个元素的存储只需一个二进制位。本文算法中用定理2的条件2)确定尾匹配q-gram的位置范围，并根据定理4进行过滤，具体过滤过程如下：

1)分配长度为$|T|$的数组H，并全置0，即$H[i]=1,0\leqslant i<|T|$。设j为模式串Q中当前正处理的q-gram位置，初始值为$j=0$。

2)当前处理的模式串Q的第j个q-gram为$Q[j,\cdots,j+q-1]$，哈希该q-gram，并定位q-gram索引中的位置，然后按倒排列表顺序依次访问q-gram项地址，并置H数组。例如，t是倒排中一个q-gram地址，即q-gram项$Q[j,\cdots,j+q-1]$命中了文本T的位置t，即置$H[t]=1$。如此计算每一个地址，直到$Q[j,\cdots,j+q-1]$的地址列表处理完毕。接着$j=j+1$。

3)重复步骤2)直到$j=|Q|-(k+1)q$时，条件$j=|Q|-(k+1)q$是q-gram项可能成为尾匹配q-gram的最左位置(根据定理2的2))，转步骤4)。

4)提取$Q[j,\cdots,j+q-1]$的地址列表，并按顺序依次处理每一个地址。

5)提取一个地址，计算命中位置，置H数组。如地址t，则置$H[t]=1$，此时该地址可能是一个匹配串的尾匹配q-gram，因此进行候选串搜索。

6)从当前位置$H[t]$向前扫描H数组。统计q-gram命中总数h即数组值为1的数目；统计连续q-gram串的个数C即连续1的个数(包括1连续)，最长的连续q-gram数目c(连续1的个数)；直到找到$C=k+1$个连续q-gram串或到达$H[t-|Q|+q-k]$时停止。设t_1为$Q[j,\cdots,j+q-1]$的位置t开始向左数第$k+1$个连续q-gram串的首字符位置，设$t_m=\max(t_1,t-|Q|+q-k)$，即扫描到t_m为止。

7)此时，判断条件确定文本区域$T[t_m,t+kq+q-1]$是否为候选串。首先总命中数满足$h\geqslant|Q|+1-(k+1)q$；其次最长连续q-gram数满足$c\geqslant\lceil(|Q|+1-(k+1)q)/(k+1)\rceil$。如同时满足以上两条件，则区域$T[t_m,t+kq+q-1]$为一个可能包含匹配串的候选串，把串$T[t_m,t+kq+q-1]$加入到候选集合$V$中，以待验证。

8)重复步骤5)直到所有地址都处理完毕，$j=j+1$并转步骤9)。

9)重复步骤4)直到$j>|Q|-q$，模式串Q的所有q-gram都被处理完成，最终得到一个候选串集合V，该集合中包含所有通过了过滤但有待验证的候选串。

4.1.4 验证

验证阶段的主要任务是通过更精细的匹配算法计算通过过滤的候选串，从中找出所有匹配串。本文算法的过滤阶段采用了Smith-Waterman算法[26]，该算法能精确地找出候选串中所有的匹配串。

4.1.5 输出

输出阶段则以用户需要的形式显示匹配结果，需要显示的信息主要包括模式串和按近似度降序排序的匹配串信息。匹配串信息包括其在文本串中的位置、与模式串间的错误率、编辑距离以及对应的编辑路径等。

4.2 TMQ算法的退化

随着匹配错误率的逐渐增大，TMQ算法将不断发生退化：

1)当 $e<(|Q|-q)/(|Q|(q+1))$ 即 $\lceil(|Q|+1-(k+1)q)/(k+1)\rceil>1$ 时，最小连续q-gram的数目大于1，算法处于最优状态。在实际应用中除特殊情况外，TMQ算法都能工作在此状态。

2)当 $(|Q|-q)/(|Q|(q+1))\leqslant e<1/q-1/|Q|+1/(|Q|q)$ 即 $0<\lceil(|Q|+1-(k+1)q)/(k+1)\rceil\leqslant 1$ 时，最小连续q-gram数目为1，定理4的条件3)失效，TMQ算法匹配效率下降。

3)当 $e\geqslant 1/q-1/|Q|+1/(|Q|q)$ 即 $|Q|+1-(k+1)q<0$ 时，算法同其他q-gram过滤算法一样，将无法进行，发生崩塌[23]，TMQ算法退化为Smith-Waterman算法。

4.3 TMQ算法的时间复杂度和空间需求分析

为便于分析TMQ算法的时间复杂度和空间需求，这里设 $n=|T|$，$m=|Q|$。

首先分析TMQ算法匹配的时间复杂度。q-gram索引中倒排列表平均长度为 n/σ^q，查询串中共有 $m-q+1$ 个q-gram，且平均哈希一个q-gram并提取地址列表的时间为O(1)。这些q-gram中，前 $m-(k+1)q$ 个只需计算命中位置，并置命中数组，时间复杂度为 $O((m-(k+1)q)n/\sigma^q)$。后 $kq+1$ 个q-gram，除计算命中位置，置命中数组外，还要向前扫描候选串并过滤，平均扫描长度为 m，时间复杂度为 $O((kq+1)mn/\sigma^q)$。因此过滤阶段总时间复杂度为 $O(((kq+2)m-(k+1)q)n/\sigma^q)$，一般情况下 $(kq+2)m-(k+1)q\ll\sigma^q$，所以过滤阶段的时间复杂度为线性。设TMQ算法的过滤效率为 f，则参加验证的文本长度为 $n(1-f)$，进行验证所需时间为 $O(mn(1-f))$，当 $f\geqslant 1-1/m$ 时验证阶段也为线性，以下实验中各种匹配错误率下各个算法的过滤效率都满足该条件。综上所述，TMQ算法进行近似串全局匹配的总时间复杂度为 $O(((kq+2)m-(k+1)q)n/\sigma^q+nm(1-f))$，且一般为线性。

现在分析算法的内存空间需求。命中数组长度为 n，每个元素采用1个二进制位存储，空间需求为 $n/8$ 字节，过滤阶段空间需求 $O(n/8)$。算法采用Smith-Waterman算法进行验证，需要验证空间为 m^2。所以，TMQ算法匹配过程中最坏情况下空间需求为 $O(n/8+m^2)$，一般 $m\ll n$，所以空间需求也为线性。

4.4 TMQ算法的无损分析

若近似串匹配算法在任何情况下都不丢失匹配串，则称该算法是无损的。TMQ算法的无损性分析需对过滤阶段和验证分别进行分析。当 $e\geqslant 1/q-1/|Q|+1/(|Q|q)$ 时，算法发生崩塌，过滤阶段不会抛弃任何文本串。当 $e<1/q-1/|Q|+1/(|Q|q)$ 时，算法过滤阶段采用了定理4。定理4是任何一个含有匹配串的文本区域都具有的基本特征，即TMQ算法抛弃的都是一定不包含匹配串的文本区域。因此TMQ算法的过滤阶段是无损的。验证阶段的Smith-Waterman算法也是一个无损算法。因此，TMQ算法是一个无损算法。

5 实验

5.1 实验环境

近似串匹配技术在基因序列比对中应用较为广泛，为在同等应用环境下对比本文算法与各经典算法的性能差异，本文的实验数据来源于美国国家生物技术信息中心NCBI的人类不同完整基因序列(UniGene Build #223, Homo sapiens)[27]，约164MB基因序列文本，共123252个完整人类不同基因序列。

在基因序列比对中，近年来经典的近似串匹配算法主要有BLAST、PatternHunter、KS1、QUASAR和SWIFT等。经典的基因比对算法BLAST中 q 值设置为11，SWIFT，QUASAR中的 q 值也设置为11，这里不考虑 q 的取值问题，也采用 $q=11$，且基因序列映射函数中字符与数字的对应关系为 $\{0:A,1:C,2:G,3:T\}$。本文实验中实现了速度较快的KS1[10]、QUASAR[12]、SWIFT[14]和本文的TMQ算法。QUASAR算法中的块大小设置为 $2|Q|$，SWIFT算法中的Bin大小设置为 $k+2^x+1$，$x\in\mathbf{N}$，$2^x>k$。

为进行批量匹配实验，从基因库中随机选取了500个长度为2000的基因子序列，并对序列进行随机的编辑操作，保证与原序列之间的错误率小于5%。由这些序列构成了查询串集合。

所有的实验都在同一硬件和软件环境下进行。实验硬件环境是：处理器AMD X4 630和主存4GB；实验软件环境是：WINDOWS XP SP3和VISUAL C++ 6.0。

5.2 TMQ算法的性能表现

为分析对比各个算法在不同匹配错误率下的性能，实验中对查询串集的500个基因子序列分别采用KS1、QUASAR、SWIFT和TMQ算法进行批量匹配实验，匹配错误率分别采用0，0.005，0.01，0.02，0.03，0.04。实验中计算了不同匹配错误率下各算法使用查询串集进行匹配时的过滤时间、过滤效率、验证时间和匹配时间的平均值，对比结果如图5—图8所示。

图5对比了各算法的平均过滤时间。过滤时间是过滤阶段采用过滤条件抛弃无关文本片段所消耗的时间总和。从图中可以看出，TMQ算法在匹配错误率较低时消耗时间要少于SWIFT算法，而在匹配错误率较高时，因进行尾匹配q-gram搜索候选串并过滤的操作过多，使得消耗的时间要多于SWIFT算法。KS1算法的过滤时间最短，因其过滤过程只是简单的地址交运算，QUASAR算法次之，因其过滤过程简单。SWIFT算法因采用了较为复杂的平行四边形过滤区，所以消耗时间较多，尤其是匹配错误率较低时。

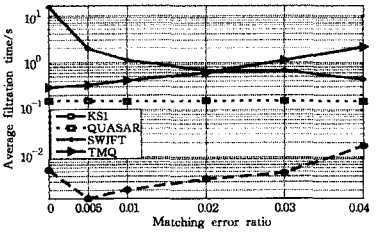

图5 平均过滤时间对比

图6对比了各算法的平均过滤效率，过滤效率计算方法见式(1)。由图可知，TMQ算法在匹配错误率较低时与SWIFT和KS1算法的过滤效率接近。当匹配错误率较高时，TMQ算法的过滤效率最高，这是因为其过滤条件苛刻，抛弃了更多的无关文本区域。SWIFT算法因其过滤标准严格，过滤效率一直较高。KS1算法在匹配错误率较低时，因分

块数目较少，获得了较高的过滤效率。但 QUASAR 算法因过滤过程简单，其过滤效率一直都比较低。

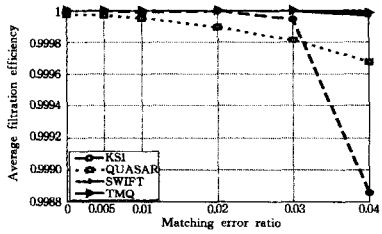

图 6　平均过滤效率对比

图 7 对比了各算法的平均验证时间。算法的过滤效率越高，则参加验证的候选串越少，验证时间越少。TMQ 算法因其过滤效率一直都较高，所以验证时间较短。

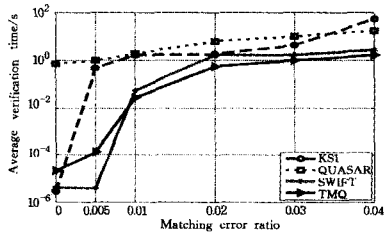

图 7　平均验证时间比较

图 8 对比了各算法的平均匹配时间。由图可知，TMQ 算法的匹配时间具有较高的稳定性，匹配时间一直都比较短。当匹配错误率为 0 时，KS1 算法因其地址交运算抛弃了所有非匹配串而获得了最快的匹配速度。当匹配错误率非常高时，SWIFT 过滤时间短，因而其匹配速度要稍快于 TMQ 算法。当匹配错误率处于中档时，TMQ 算法具有最快的匹配速度。QUASAR 算法的匹配时间一直都较长。

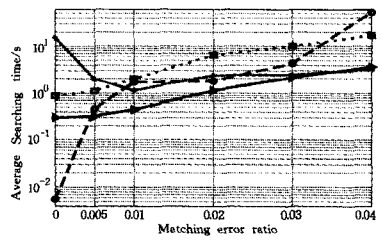

图 8　平均匹配时间比较

根据以上分析可得出如下结论：

1）TMQ 算法的整体性能较好，相对于其他算法其匹配速度一直都较快。

2）TMQ 算法对匹配错误率的改变不敏感，相对于其他算法更具通用性。

结束语　本文提出了一种在大文本库中进行快速近似匹配的无损过滤算法。文中首先通过分析含有匹配串的文本区域，提取了尾匹配 q-gram 特征，并使用这些特征设计了严格的过滤条件。然后给出了算法进行近似匹配的匹配过程，并分析了算法的复杂度、无损性和退化问题。最后通过实验说明新算法的尾匹配 q-gram 特征是有效的，提高了算法的过滤效率。理论分析和实验结果显示，新算法的整体性能较好，适合各种匹配错误率下的近似匹配，算法通用性强。

虽然新算法提高了算法的过滤效率，但当匹配错误率较高时，算法搜索候选串并过滤的操作次数增加，导致算法的过滤时间延长。本文下一步的工作是研究缩短过滤时间的相关方法和技术，并应用留空种子进一步优化本文算法。

参 考 文 献

[1] Dorneles C F, Goncalves R, Mello R D. Approximate data instance matching: a survey[J]. Knowledge and Information Systems, 2011, 27(1): 1-21

[2] Navarro G. A guided tour to approximate string matching[J]. ACM Computing Surveys, 2001, 33(1): 31-88

[3] Levenshtein V. Binary codes capable of correcting deletions, insertions, and reversals[J]. Soviet Physics Doklady, 1966, 10(8): 707-710

[4] Burkhardt S. Filter algorithms for approximate string matching [D]. Saarland: Department of Computer Science, Saarland University, 2002

[5] Tian Y, Tata S, Patel J M, et al. Practical methods for constructing suffix trees[J]. VLDB Journal, 2005, 14(3): 281-99

[6] Manber U, Myers G. Suffix arrays: A new method for on-line string searches[J]. SIAM Journal on Computing, 1993, 22(5): 935-948

[7] Navarro G, Baeza-yates R. A practical q-gram index for text retrieval allowing errors[J]. CLEI Electronic Journal, 1998, 1(2): 1-16

[8] Kim M S, Whang K Y, Lee J G. n-Gram/2L-approximation: A two-level n-gram inverted index structure for approximate string matching[J]. Computer Systems Science and Engineering, 2007 22(6): 365-379

[9] Kim Y, Park H, Shim K. Efficient processing of substring match queries with inverted variable-length gram indexes[J]. Information Sciences, 2013, 244(0): 119-141

[10] Wu S, Manber U. Fast text searching allowing errors[J]. Communications of the ACM, 1992, 35(10): 83-91

[11] Chang Y I, Chen J R, Hsu M T. A hash trie filter method for approximate string matching in genomic databases[J]. Applied Intelligence, 2010, 33(1): 21-38

[12] Burkhardt S, Crauser A, Ferragina P, et al. Q-gram based database searching using a suffix array[C]// Proceedings of the Annual International Conference on Computational Molecular Biology, RECOMB 99. New York, USA: ACM, 1999: 77-83

[13] Jokinen P, Ukkonen E. Two algorithms for approximate string matching in static texts [C] // 16th International Symposium Proceedings on Mathematical Foundations of Computer Science. Berlin, Germany: Springer-Verlag, 1991: 240-248

[14] Rasmussen K R, Stoye J, Myers E W. Efficient q-gram filters for finding all epsilon-matches over a given length[J]. Journal of Computational Biology, 2006, 13(2): 296-308

[15] Lu C W, Lu C L, Lee R C T. A new filtration method and a hybrid strategy for approximate string matching[J]. Theoretical Computer Science, 2013, 481: 9-17

[16] Altschul S F, Gish W, Miller W, et al. Basic local alignment search tool[J]. Journal of Molecular Biology, 1990, 215(3): 403-410

[17] Altschul S F, Madden T L, Alejandro A S, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs[J]. Nucleic Acids Research, 1997, 25(17): 3389-3402

[18] Sutinen E, Tarhio J. Filtration with q-samples in approximate string matching[C]// Proceedings of 7th Annual Symposium on

Combinatorial Pattern Matching, CPM 96. Berlin, Germany: Springer-Verlag, 1996: 50-63

[19] Ma B, Tromp J, Li M, et al. PatternHunter: faster and more sensitive homology search[J]. Bioinformatics, 2002, 18(3): 440-445

[20] Egidi L, Manzini G. Better spaced seeds using Quadratic Residues[J]. Journal of Computer and System Sciences, 2013, 79 (7): 1144-1155

[21] Baeza-Yates R, Navarro G. Faster approximate string matching [J]. Algorithmica, 1999, 23(2): 127-158

[22] Myers E W. A sublinear algorithm for approximate keyword searching[J]. Algorithmica, 1994, 12(4/5): 345-374

[23] Sutinen E, Szpankowski W. On the collapse of the q-gram filtration[C]// Proceedings of the International Conference on FUN with Algorithms. Waterloo, Canada: Carleton Scientific, 1998: 178-193

[24] Puglisil S J, Smyth W F, Turpin A. Inverted files versus suffix arrays for locating patterns in primary memory[C]// Proceedings of the 13th Symposium on String Processing and Information Retrieval. Berlin, Germany: Springer Verlag, 2006: 122-133

[25] Karp R M, Rabin M O. Efficient randomized pattern-matching algorithms[J]. IBM Journal of Research and Development, 1987, 31(2): 249-260

[26] Smith T F, Waterman M S. Identification of common molecular subsequences[J]. Journal of Molecular Biology, 1981, 147(1): 195-197

[27] NCBI. UniGene Build #223, Homo sapiens[DB/OL]. ftp. ncbi. nih. gov//repository/ UniGene/Homo_sapiens/Hs. seq. uniq. gz, 2010-04-28

(上接第 238 页)

[7] Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog-leaping algorithm[J]. Journal of Water Resources Planning and Management, 2003, 129(3): 210-225

[8] 贺毅朝, 曲文龙, 许冀伟. 一种改进的混合蛙跳算法及其收敛性分析[J]. 计算机工程与应用, 2011, 47(22): 37-40

[9] 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002(11): 32-38

[10] 李晓磊, 冯少辉, 钱积新, 等. 基于人工鱼群算法的鲁棒 PID 控制器参数整定方法研究[J]. 信息与控制, 2004, 33(1): 112-115

[11] Yang X S. Biology-derived algorithms in engineering optimization[M]// Olarius S, Zomaya A Y, eds. Handbook of Bioinspired Algorithms and Applications. Chapman & Hall /CRC, 2005

[12] Yang X S. Nature-Inspired Metaheuristic Algorithms[M]. UK: Luniver Press, 2008

[13] Engelbrecht A P. Computational intelligence: an introduction [M]. Wiley Publishing. Inc. , 2009

[14] Wolpert D H, Macready W G. No Free Lunch Theorems for Optimization [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82

[15] Pringle J W S. A physiological analysis of cicada song[J]. J. Exp. Biol. , 1954, 32: 525-560

[16] Simmons P J. Periodical cicada: sound production and hearing [J]. Science, 1971, 171: 212-213

[17] 刘勇, 康立山, 陈毓屏. 非数值并行算法——遗传算法[M]. 北京: 科学出版社, 2003: 22-86

[18] Iosifescu M. Finite Markov processes and their applications [M]. Chichester: Wiley, 1980

[19] 江瑞, 罗予频, 胡东成, 等. 一种协调勘探和开采的遗传算法: 收敛性及性能分析[J]. 计算机学报, 2001, 24(12): 1233-1241

[20] Yao Xin, Liu Yong, Lin Guang-ming. Evolutionary programming made faster[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82-102

[21] He S, Wu Q H, Saunders J R. Group search optimizer: An optimization algorithm inspired by animal searching behavior[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 973-990

(上接第 242 页)

[6] Rosenstein M T, Barto A G. Supervised Learning Combined with an Actor-Critic Architecture[J]. CMPSCI Technical Report 02-41. October 2002

[7] Peters J, Schaal S. Natural actor-critic [J]. Neurocomputing, 2008, 71(7-9): 1180-1190

[8] Bathnagar S, Sutton R S, Ghavamzadeh M, et al. Natural actor-critic algorithms[J]. Automatica, 2009, 45(11): 2471-2482

[9] Vamvoudakis K G, Lewis F L. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem[J]. Automatica, 2010, 46(5): 878-888

[10] Grondman I, Vaandrager M, Busoniu L, et al. Efficient Model Learning Methods for Actor-Critic Control[J]. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 2012, 42(3): 591-602

[11] Grondman I, Vaandrager M, Busoniu L, et al. Actor-Critic Control with Reference Model Learning[C]// Proceedings of the 18th IFAC World Congress. Milan, Italy, 2011: 14723-14728

[12] Kuvayev L, Sutton R. Model-Based Reinforcement Learning with an Approximate, Learned Model[C]// Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems. 1996: 101-105

[13] Goschin W S, Littman M. Integrating sample-based planning and model-based reinforcement learning[C]// Proc. Assoc. Adv. Artif. Intell. . Atlanta, GA, 2010: 612-617

[14] Santamaria J, Sutton R, Ram A. Experiments with reinforcement learning in problems with continuous state and action spaces [J]. Adaptive Bechavior, 1998, 6: 163-138

[15] Sherstov A A, Stone P. Function Approximation via Tile Coding: Automating parameter choice[C]// Zucker J-D, Saitta L, eds. SARA, volume 3607 of Lecture Notes in Computer Science. Springer, 2005: 194-205

[16] Lanzi P L, Loiacono D, Wilson S W, et al. Classifier Prediction based on Tile Coding[C]// Proceedings of the 2006 Geneticand Evolutionary Computation Conference Workshop Program (GECCO 2006). Seattle, Washington, 2006: 1497-1504