ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2011/05(09)-0769-12
DOI: 10.3778/j.issn.1673-9418.2011.09.001

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-51616056

# 支持局部最优化匹配的近似子串查询算法*

刘洪磊, 杨晓春+, 王 斌, 金 蓉
东北大学 信息科学与工程学院, 沈阳 110819

# Approximate Substring Query Algorithms Supporting Local Optimal Matching*

LIU Honglei, YANG Xiaochun+, WANG Bin, JIN Rong
College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
+ Corresponding author: E-mail: yangxc@mail.neu.edu.cn

**LIU Honglei, YANG Xiaochun, WANG Bin, et al. Approximate substring query algorithms supporting local optimal matching. Journal of Frontiers of Computer Science and Technology, 2011, 5(9): 769-780.**

**Abstract:** The algorithms of approximate string query based on edit distance have usually a given threshold *k*, and report those strings whose edit distances with the query are not bigger than *k*. However, when talking about approximate substring query, many answers got in this way have overlap, thus meaningless. So this paper proposes a concept of local optimal matching only computing those answers which are both not higher than *k* and local optimal, thus eliminating the overlapped answers and cutting the time cost. It also presents a definition of approximate substring query supporting local optimal matching along with a query algorithm based on gram index. Moreover, it analyzes the generality of the matching process, and studies the methods of filtering and limiting the boundary based on local optimal matching. Finally, it proposes an algorithm of local optimal approximate substring query with filtering, so that the efficiency of matching is promoted.
**Key words:** substring query; fuzzy query; gram index; edit distance

**摘 要:** 基于编辑距离的字符串近似查询算法一般是先给定阈值 *k*, 然后计算那些与查询串的编辑距离小于或等于 *k* 的结果。但是对于近似子串查询, 结果中有很多是交叠的, 并且是无意义的, 于是提出了一种局

*The National Natural Science Foundation of China under Grant No. 60973018, 60973020, 61173031 (国家自然科学基金); the Fundamental Research Funds for the Central Universities under Grant No. N090504004, N100704001, N090104001 (中央高校基本科研业务费专项资金).
Received 2011-04, Accepted 2011-06.

部最优化匹配的概念，只计算那些符合阈值条件，并且是局部最优的结果，这样不仅避免了结果的交叠，而且极大节省了时间开销。给出了支持局部最优化匹配的近似子串查询的定义，相应提出了一种基于 gram 索引的局部最优化近似子串查询算法，分析了子串近似匹配过程中的规律，研究了基于局部最优化匹配的边界限定和过滤策略，给出了一种过滤优化的局部最优化近似子串查询算法，提高了查询效率。

关键词：子串查询；模糊查询；gram 索引；编辑距离

文献标识码：A 中图分类号：TP311.131

## 1 引言

伴随着当代信息量的快速增长，如何从海量数据中提取人们需要的信息显得尤为重要。近似子串查询技术，即从长串中得到与查询相同或近似的子串，在 DNA 数据分析、文本信息的检索等领域都具有重要意义。例如对于从 Internet 中抽取的大量非结构化的文本数据，其中难免出现错误或不一致的数据，因此在查询时经常要采用近似查询的技术。

目前有多种不同的方法来评价两个字符串之间的差别大小。如 Levenshtein 在 1965 年提出的编辑距离(edit distance，也称为 Levenshtein distance)[1]，Sankoff 和 Kruskal 在 1983 年提出的 Hamming distance[2]，Das 等人在 1997 年提出的 Episode distance[3]，Needleman 和 Wunsch 在 1970 年提出的 longest common subsequence distance[4]。其中编辑距离得到了最广泛的应用，由于其广泛的适用性，目前许多算法都是基于它进行开发的。

基于编辑距离的算法一般都是先给定阈值 $k$，然后计算那些与查询串的编辑距离小于或等于 $k$ 的结果。但是对于近似子串查询，这样的结果中有很多是交叠的，并且是无意义的。如例 1 所示。

例 **1** 对于缺少空格的长串 $T$=“dynamicprogramming”和存在拼写错误的短串 $P$=“progrem”，若在 $T$ 中对 $P$ 进行一般的近似子串查询，设给定的阈值 $k$=2，那么 $T$ 的子串“progr”、“progra”、“program”、“programm”都能通过不超过两次的编辑操作转换成 $P$，即它们与 $P$ 的编辑距离都小于或等于 2，因此都会作为符合条件的结果返回。但是通过观察就会发现，这几个字符串之间有很大的交叠部分，甚至具有包含关系，这样的结果是完全没有必要的，只需将最接近的“program”返回即可。这就是本文提出的局部最优化匹配概念，就是在 $T$ 中近似查询 $P$ 时，能够把 $T$ 中与 $P$ 相近的并且是局部最优的子串作为结果返回。

本文主要解决以下问题：从给定的长串中，如人类基因数据链、从 Web 上提取的长字符串等，如何能够快速得到与给定的查询串满足阈值条件，并且是局部最优的子串。

本文的主要贡献如下：

(1) 考虑到目前近似子串查询中存在的问题，提出了一种局部最优化匹配的概念，给出了支持局部最优化匹配的近似子串查询的定义。

(2) 给出了一种基于 gram 索引的局部最优化近似子串查询算法。

(3) 分析了子串近似匹配过程中的规律，分别给出了针对局部最优化匹配的边界限定和过滤策略，提高了查询效率。

(4) 结合基于 gram 的查询算法与基于最优化匹配的边界限定和过滤策略，提出了一种过滤优化的局部最优化近似子串查询算法。

(5) 采用实际数据集对本文提出的算法进行了实验测试和分析，结果显示提出的算法具有很好的查询效率。

本文组织结构如下：第 2 章介绍了相关工作；第 3 章给出了一些背景知识和问题定义；第 4 章给出了基于 gram 索引的局部最优化近似子串查询算法的原理和流程；第 5 章分析了子串近似匹配过程中的规律，分别提出了基于局部最优化匹配的边界限定和过滤策略；第 6 章给出了过滤优化的局部最优化近似子串查询算法的整体流程；第 7 章通过实

际数据集对本文算法进行了实验测试和分析; 第 8 章总结全文, 并对未来工作进行愿望。

## 2 相关工作

子串查询技术被广泛应用, 因此一直备受关注, 并且已经有很多研究成果。例如 Knuth 等人在文献[5]中提出的 Knuth-Morris-Pratt 算法和 Boyer 与 Moore 在文献[6]中提出的 Boyer-Moore 算法都是子串精确查询的经典算法; 文献[7]采用 gram 索引进行子串的精确查询。对于近似子串的查询, 目前已经有很多在线的或基于索引的近似子串查询算法被提出来。文献[8]对其进行了很好的综述, 介绍了目前的研究现状和一些比较好的算法。

生物信息学中的局部比对技术与近似子串查询概念类似, 但其主要用来进行 DNA 的同源性分析。相似性的判断有很强的专业背景, 对其他领域并不适用。

本文采用 gram 倒排索引的技术加快查询过程。一个字符串的 gram 是指它特定长度的子串, 如“program”的 2-gram 集合为{pr,ro,og,gr,ra,am}。

文献[9–12]都对 gram 或倒排技术进行了详尽的研究。其中文献[10]对倒排索引技术进行了综述。文献[11–12]对 gram 分解长度的选取问题进行了研究, 提出了一种叫做 VGRAM(variable-length grams)的变长 gram 分割技术, 利用 VGRAM 建立的索引可以有效提高匹配效率。在利用 gram 索引进行匹配时, 通常会采用归并的方法进行频率统计和过滤工作, 文献[13]提出了多种有效的提高归并效率的方法。

匹配时倒排列表选取的顺序不同也会影响匹配时间, 文献[14]对这一问题进行了研究, 并提出了一种基于贪心算法的选取倒排列表归并顺序的方法以提高匹配效率。文献[7]对于子串精确匹配时 gram 组合选取问题进行了研究, 提出并比较了多种 gram 组合方式。

在生物信息学中得到广泛应用的局部比对技术与近似子串查询有些类似, 文献[15]提出的 BLAST(basic local alignment search tool)比对工具是目前在生物信息学中最流行的算法。

## 3 背景知识与问题定义

### 3.1 子串的编辑距离

编辑距离是用来评价两个字符串间近似程度的一种方式, 目前具有最广泛的应用。

定义 **1**(编辑距离[1]) 两个字串之间由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符, 插入一个字符, 删除一个字符。如例 2 所示。

例 **2** 字符串 $s_1$=“program”和字符串 $s_2$=“progrem”, 由 $s_2$ 变换成 $s_1$ 时需要将“e”替换成“a”, 因此 $ed(s_1,s_2)=1$。

计算编辑距离一般采用动态规划的方法。设两个字符串分别为 $x$ 和 $y$, $x[1,i]$和 $y[1,j]$分别代表 $x$ 和 $y$ 中长度为 $i$ 和 $j$ 的子串。建立大小为$|x|\times|y|$的矩阵 $\boldsymbol{C}$, $C_{i,j}$ 表示 $x[1,i]$和 $y[1,j]$的编辑距离。

由于 $x[1,i]$与空串的编辑距离是 $i$, 因此 $C_{i,0}=i$, 同理得 $C_{0,j}=j$。动态规划的迭代公式如下:

$$C_{i,j}=\begin{cases} C_{i-1,j-1} & x_i=y_j \\ 1+\min(C_{i-1,j},C_{i,j-1},C_{i-1,j-1}) & x_i\neq y_j \end{cases}$$

最终可得到 $ed(x,y)=C_{|x|,|y|}$。$C_{i,j}$ 的计算分为两种情况: 当 $x_i=y_j$, 不需要再进行编辑操作, 因此编辑距离等于 $C_{i-1,j-1}$; 若 $x_i\neq y_j$, 则从插入、删除和替换三种操作中选取一种编辑距离最小的。

子串的编辑距离与字符串间整体的编辑距离不同, 体现的是一个字符串与另一个字符串的所有子串之间的近似程度。

定义 **2**(子串的编辑距离) 一个字符串与另一个字符串中某一位置为结尾的所有子串中最小的编辑距离。如例 3 所示。

例 **3** 长串 $T$=“dynamicprogramming”, 短串 $P$=“progrem”, $P$ 与 $T$ 的编辑距离为 12, 但 $P$ 与 $T$ 中以第 14 个字符“m”为结尾的子串编辑距离为 1, 即 $P$ 与 $T$ 的子串“program”的编辑距离。

对以上的动态规划算法稍作改进, 即可得到子串的编辑距离计算方法。设 $x=P$ 为短串, $y=T$ 为长串, 与上面不同的是 $T$ 中的任意一个位置都可以作为一

次匹配的起始位置。$C_{i,j}$表示 $P[1,i]$与 $T$ 中任意以 $T_j$ 为结尾的所有子串中编辑距离的最小值，即

$$C_{i,j}=\min_{k\leqslant j} ed(P[1,i],T[k,j])$$

此时 $C_{0,j}$=0，因为空串是任意字符串的子串。图 1 为例 3 中计算子串编辑距离的例子。矩阵的最后一行代表 $T$ 中以每一个位置结尾的子串与 $P$ 的最小编辑距离。加粗位置代表 $P$ 与 $T$ 中的子串“program”的编辑距离为 1。

| | | d | y | n | a | m | i | c | p | r | o | g | r | a | m | m | i | n | g |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **0** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| r | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | **0** | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| o | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 1 | **0** | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| g | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 2 | 1 | **0** | 1 | 2 | 3 | 4 | 4 | 4 | 3 |
| r | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 3 | 2 | 1 | **0** | 1 | 2 | 3 | 4 | 5 | 4 |
| e | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 5 | 4 | 3 | 2 | 1 | **1** | 2 | 3 | 4 | 5 | 5 |
| m | 7 | 7 | 7 | 7 | 7 | 6 | 7 | 7 | 6 | 5 | 4 | 3 | 2 | 2 | **1** | 2 | 3 | 4 | 5 |

Fig.1 The matrix for calculating edit distance

图 **1** 计算子串编辑距离的示例矩阵

## 3.2 支持局部最优化匹配的近似子串查询

本文子串的相近程度用编辑距离来描述，用 $m_i$ 表示以 $T$ 中第 $i$ 个位置结尾的任意子串与 $P$ 的最小编辑距离，即图 1 中矩阵的最后一行。对于近似子串查询，若给定了阈值 $k$，直观的感觉应该是能够返回所有满足 $m_i\leqslant k$ 的子串。

但是通过观察就会发现，如例 1 和图 1 所示，若 $k$=2，那么 $T$ 的子串“progr”、“progra”、“program”、“programm”都会作为符合条件的结果返回。而这是完全没有必要的，只需将最接近的“program”返回即可。因此，本文提出了一种局部最优化匹配的概念。

定义 **3** (局部最优化匹配) 在长串 $T$ 中进行短串 $P$ 的模糊匹配时，满足下列条件的匹配结果 $R$ 被叫做局部最优化匹配: $R$ 是 $T$ 的子串，并且在 $T$ 中对 $R$ 进行扩展或缩短均会产生 $R$ 与 $P$ 的编辑距离增大的趋势，即 $R$ 是 $P$ 与 $T$ 计算子串编辑距离时所产生变化趋势的极值点。

如图 2 所示为图 1 中最后一行对应的曲线图，图中标示的两个点都为局部最优化匹配，因为它们都处在极值点的位置，即这两个点都具有局部最小的编辑距离。

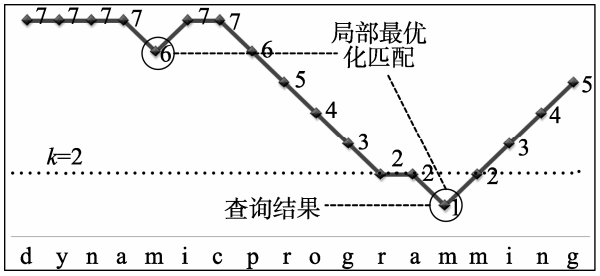



Fig.2 The result of approximate substring query supporting local optimal matching

图 **2** 支持局部最优化匹配的近似子串查询返回结果的示例

定义 **4**(支持局部最优化匹配的近似子串查询) 在长串 $T$ 中进行短串 $P$ 的近似子串查询，应能够返回结果 $R$，其中 $R$ 与 $P$ 的编辑距离小于或等于阈值 $k$，并且 $R$ 是一个局部最优化匹配。

例 **4** 如图 2 中，当 $k$=2 时，虽然此时有 4 个位置都符合阈值条件，即与虚线持平或在其下面的点，但是只返回图示的点，即处在极值点的子串“program”。这样做的好处是防止出现无意义的重叠结果，而且通过这种方式选取了局部区域中编辑距离的最小值，即最相近的子串。同时结合本文提出的过滤方法可以极大地节省时间开销。

## 4 基于 gram 索引的局部最优化近似子串查询算法

要解决定义 4 提出的问题，最基本的方法是使用 3.1 节提到的动态规划方法。首先通过动态规划方法计算得到 $T$ 中每一个位置对应的子串编辑距离。然后从左向右依次遍历每一个值，通过记录这些数值的变化趋势得到所有极值点，即局部最优化匹配。若这些局部最优化匹配满足相应的阈值条件，则找到了问题的解。

这种方法的时间复杂度为 $O(mn)$, $m$ 和 $n$ 分别是 $P$ 和 $T$ 的长度。显然这种方法是极其浪费时间和空

间的, 由于其想法的简单性, 将其称为 Naïve 算法。为了提高查询效率, 引入 gram 技术。

目前许多解决近似查询问题的算法, 如文献[9,16]都采用了 gram 倒排索引技术。但是目前 gram 技术通常采用的长度过滤和共同 gram 数量过滤等方法, 并不能直接应用在子串的近似查询上。下面对这些方法进行讨论和改进, 使之能够适应子串的近似查询。然后给出针对定义 4 的求解算法。

### 4.1 基于 gram 技术的近似查询的性质

一个 $q$-gram 就是一个字符串中长度为 $q$ 的子串, 用来作为一个长字串的标识。如果两个字符串的编辑距离满足一定的阈值 $k$, 那么它们必定至少有一定数量的共同的 gram, 即下界, 并且该下界是和 gram 长度 $q$ 和阈值 $k$ 相关的。例如“program”的 2-gram 集合为 {pr,ro,og,gr,ra,am}, 而“progrem”的 2-gram 集合为{pr,ro,og,gr,re,em}, 它们包含 4 个共同的 gram。

对于两个字符串 $s_1$ 和 $s_2$, 若 $ed(s_1,s_2)\leqslant k$, 那么字串长度$|s_1|$和$|s_2|$的最大差别不应该超过 $k$, 并且它们应该具有至少 $\max\{|s_1|,|s_2|\}+1-(k+1)\times q$ 个相同的 $q$-gram。

设查询结果为字符串集合 $R=\{r_1,r_2,\ldots,r_n\}$, 满足 $ed(P,r_i)\leqslant k$, 则查询结果应满足如下公认的性质。

性质**1** 查询串与结果串的最大长度差别不能大于 $k$, 即$|r_i|\leqslant|P|+k$。

证明 由于 $r_1$ 必须满足 $ed(P,r_i)\leqslant k$, 若 $r_1$ 与 $P$ 的长度差大于 $k$, 则至少需要大于 $k$ 次插入操作才能够将 $r_1$ 变换成 $P$, 即它们之间的编辑距离必然会大于 $k$, 所以查询串与结果串的最大长度差别不能大于 $k$。 □

性质**2** 结果串 $r_i$ 与查询串至少具有$|P|+1-(k+1)\times q$ 个共同 $q$-gram。

证明 对于长度为$|P|$的字符串, 其能分解的 $q$-gram 数量为$|P|+1-q$。若进行 $k$ 次编辑操作, 那么最多影响 $k\times q$ 个 gram, 因此查询串与结果串的编辑距离要想小于或等于 $k$, 至少要包含$|P|+1-(k+1)\times q$ 个共同的 $q$-gram。如例 5 所示。 □

例 **5** $T$=“dynamicprogramming”, $P$=“progrem”, 当 $k$=2, $q$=2 时, 根据定理 2, 结果串与 $P$ 至少要有 2 个共同的 2-gram。

目前使用 gram 技术的近似查询算法大都采用这两个性质进行过滤, 分别叫做长度过滤和共同 gram 数量过滤。但是由于近似子串查询中没有类似于“单词”和“记录”的概念, 这两个性质并不适用。要想将 gram 技术应用到子串的近似查询上, 首先要解决如下两个问题: 如何能够确定结果区域的长度, 如何统计共同 gram 的数量。

### 4.2 构造和筛选候选区域

为了得到最终结果, 如果遍历 $T$ 中以任意位置为结尾的子串进行验证, 必然可以得到结果, 但这样做代价太大。根据性质 1, 最终的结果串最大长度为$|P|+k$。因此, 将子串的结尾位置扩展成长度为 $|P|+k$ 的区域, 即为候选区域。

又根据性质 2, 结果串与 $P$ 必然包含一定数量的共同 gram, 如果能够统计候选区域中包含的共同 gram 的数量, 则可以筛选出最终的待验证区域进行验证。

如何构造及筛选候选区域要用到以下概念。

定义 **5**(gram 基准位置) $P$ 中的某一 gram 与 $T$ 中的某一 gram 匹配后, $P$ 的结尾位置在 $T$ 中的投影。

如图 3 所示, 设某一 gram 在 $P$ 中出现的位置为 $g_p$, 在 $T$ 中出现的位置为 $g_t$, 根据位置间的对应关系, gram 基准位置 $g_{\text{end}}$ 计算方法如下:

$$g_{\text{end}}=g_t+|P|-g_p$$

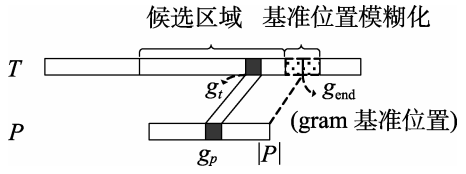



Fig.3 Gram base positions and candidate regions to verify

图 **3** **gram** 基准位置计算与候选区域构造

定义 **6** (gram 基准位置模糊化) 对 gram 基准位置进行向左向右的扩展, 保证以扩展位置为结尾的子串能够有效地包含对应的 gram, 这个扩展过程叫做 gram 基准位置模糊化。

如图3所示，因为当扩展距离超过$k$时，已经不满足阈值条件，同时若扩展位置已经小于 gram 的位置，则以其为结尾的子串必然不能包含该 gram。因此若设扩展后的位置为 $i$，则

$$\max(g_t+q-1, g_{\text{end}}-k) \leqslant i \leqslant g_{\text{end}}+k$$

**定理 1** 对 $P$ 分解的 gram 在 $T$ 中进行基准位置模糊化，只有当某一位置出现的次数大于或等于性质 2 中共同 gram 数量的下界时，以它为结尾，扩展长度为$|P|+k$的候选区域才有可能包含结果串。

证明 只有当某一位置出现的次数大于或等于共同 gram 下界时，才说明以其为结尾的子串与 $P$ 满足性质 2。又因为性质 1，结果串的长度不能大于$|P|+k$。 □

利用定理 1 可以构造候选区域，并对候选区域进行筛选，得到最终要验证的区域。

### 4.3 算法框架

根据定理 1，本文提出了一种基于 gram 索引的局部最优化近似子串查询算法(gram-based local optimal approximate substring query algorithm, GASQ)。GASQ 算法的整体流程：首先对 $P$ 进行 gram 分解，得到 gram 集合；然后通过查找 gram 索引得到每个 gram 在 $T$ 中出现的位置列表，将 gram 位置变换成 gram 基准位置，对 gram 基准位置进行归并；对 gram 基准位置模糊化后，统计每个扩展位置出现的频率，当频率大于下界时可将其作为候选位置，若候选位置通过第 5 章提出的过滤方法，则对其进行边界扩展，得到最终要进行验证的区域。

**算法 1** 基于 gram 索引的近似子串查询算法

输入：短串 $P$，长串 $T$，gram 索引 $G$，阈值 $k$。

输出：符合条件的子串位置。

```
1.  将 P 分解成 gram 集合
2.  查找 G 得到倒排列表 L
3.  将 L 变换成 gram 基准位置列表 L'
4.  while ( L' is not empty){
5.      smallest=MergHeap(L'); //得到一个最小位置
6.      FR=Fuzzy(smallest); //gram 基准位置模糊化
7.      for (each pos in FR){
8.          count=Count(pos); //统计 pos 出现的频率
9.          if (count is greater than lowerbound){
10.             region=[pos–|P|–k+1,pos]; //扩展区域
11.             Verify(region); //使用 Naïve 算法验证
12.         }//"if"
13.     }//"for"
14. }//"while"
```

**例 6** 如图 4 所示，短串 $P$="AGTC"，长串 $T$="GAAGGTCTCA"，阈值 $k$=1，不难计算此时共同 gram 数量的下界等于 1。首先对 $P$ 进行 gram 分割，查找索引，得到倒排列表；然后对 gram 的位置进行变换，得到 gram 基准位置，如"AG"在 $T$ 中出现的位置为 3，加上$|P|-1$ 后得到 gram 基准位置为 6；将 gram 基准位置进行归并，模糊化后统计每个扩展位置出现的频率，这个例子中共同 gram 数量的下界等于 1，因此 5、6、7、8、9、10 位置均符合条件，将其向前扩展 5 个位置得到最后要验证的区域。

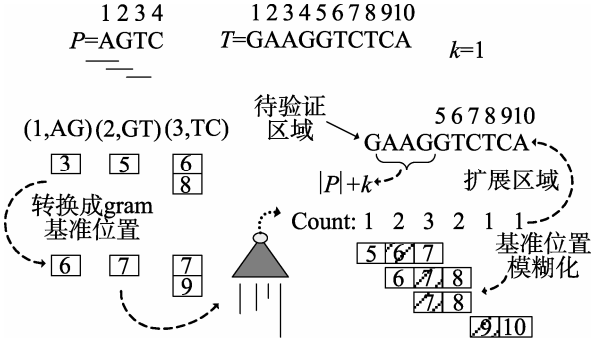



Fig.4 The flow of the algorithm

图 4 算法流程示例

## 5 基于局部最优化匹配的边界限定和过滤策略

观察例 6 不难发现，该例中对最后候选区域的筛选效果并不好，这是因为筛选过程只考虑了结果串与 $P$ 的编辑距离应小于或等于 $k$ 的条件。但是根据定义 4，结果串还应是局部最优化匹配，即处在极值点的位置上。因此本文分析了子串近似匹配时的规律，进而提出了基于局部最优化匹配的边界限

定策略 LOB(local optimal boundary constraint)和过滤策略 LOF(local optimal filtering), 可以在很大程度上提高查询效率。

### 5.1 基于局部最优化匹配的过滤策略

根据定义 4, 很容易得到如下定理。

**定理 2** 极值点必小于左端的值, 同时小于或等于右端的值。若设位置 $i$ 为一个极值点, 则有 $m_i < m_{i-1}$ 和 $m_i \leqslant m_{i+1}$。

若设某一极值点在 $T$ 中的位置为 $i$, 根据定理 2 可知, 极值点左端的编辑距离大于极值点处的编辑距离, 即从 $i-1$ 位置变换到 $i$ 位置时编辑距离发生了下降, 由此可以得出结论, $T_i$ 必然与 $P$ 中某一字符发生了匹配才会导致编辑距离的减少。又因为匹配的位置与 $P$ 结尾的偏差不能大于 $k$ 以满足阈值条件, 可得出以下定理。

**定理 3** 若 $T$ 中位置 $i$ 处为极值点, 则 $T$ 中第 $i$ 个字符 $T_i$ 满足如下性质:

$$T_i \in P[|P|-k, |P|]$$

由定理 2 可知, 极值点的右端大于或等于极值点处的编辑距离。下面研究什么样的匹配方式能够在已经满足定理 3 的情况下满足定理 2。

如图 5 为极值点的匹配示意图, 假设 $T$ 中位置 $i$ 为一极值点, 并且按照定理 3, $T_i$ 与 $P_j$ 进行匹配, 把图示的区域叫做 M 区域(mismatch zone), 在这个区域内两个字符串不能存在相同的字符。如果出现相同字符, 两个串的编辑距离必将会减少, 因为相同的字符减少了它们变换时的插入操作。当 M 区域内没有共同字符时, 区域内的编辑距离将与 $i$ 处的编辑距离相同。因此为了不让编辑距离下降, M 区域应具有如下性质。

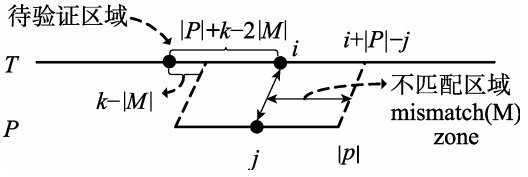



Fig.5 The matching process of extreme value point

图 5 极值点的匹配示意图

**定理 4** 若 $i$ 处为一极值点, 则图 5 中的 M 区域内 $T$ 与 $P$ 不能出现共同的字符, 即

$$T[i+1, i+|P|-j] \cap P[j+1, |P|] = \varnothing$$

利用定理 3 和定理 4 可以得出本文提出的基于局部最优化匹配的过滤方法。首先 $T$ 中极值点处的字符要在 $P$ 中最后 $k$ 长区域内出现过, 而且 M 区域内不能出现 $P$ 中对应区域出现过的字符。$P$ 中可能存在重复字符, 因此不能确定与哪个字符匹配, 取该字符出现的最后一个位置作为匹配位置计算 M 区域。

**例 7** 如例 6 中, 5、6、7、8、9、10 位置都满足阈值条件, 位置 5 对应的“G”由于没有在 $k$ 长区域内出现, 不符合定理 4 被舍弃, 位置 6 对应的“T”由于后边出现了“C”, 不符合定理 5 被舍弃。

下面简单介绍一下该过滤算法的实现技术。设字符集为$\Sigma$, 首先构建$|\Sigma|$大小的“不相容表”, 用来存储扫描 $P$ 的 $k$ 长区域后记录的每个字符后边不能出现的字符。再构建$|\Sigma|$大小的“排除表”, 用来记录扫描过程中遇到的所有字符的不相容字符集合。每返回一个符合下界限制的结尾位置时, 查找“排除表”, 若该字符在表中不存在, 则查找“不相容表”, 将该字符的不相容字符集合添加到“排除表”, 同时将该位置存到队列中; 若该字符在表中存在, 则表明队列中存在不可能是极值点的位置, 队列进行弹出操作, 同时将弹出的位置对应的不相容字符从“排除表”中删除, 直到当前字符不在“排除表”中出现为止, 同时将该位置存到队列中。当队列中的数据超过对应的 M 区域时, 说明找到了第一个可能是极值点的位置, 进行区域扩展形成验证区域并返回。

**算法 2** 基于局部最优化匹配的过滤算法

输入: 要验证的位置 $i$,不相容表 *MTable*。

输出: 待验证区域。

```
1. 构建大小为|Σ|的 ETable;
2. 构建空队列 Q;
3. while (T_i exists in ETable){
4.         Q.pop();
```

5. 从 *ETable* 中删除弹出的字符对应的不相容字符集合;
6. }//“while”
7. 从 *MTable* 中找出与 $T_i$ 不相容的字符集 *C*;
8. 将 *C* 存入 *ETable* 中;
9. *Q.push*($T_i$);
10. if (length of *Q* is greater than |*M*|){
11. *pos*=*Q.front*();
12. 扩展 *pos* 成为可验证的区域 *region*;
13. return *region*;
14. }//“if”

## 5.2 基于局部最优化匹配的边界限定策略

本文 4.2 节中构造候选区域时只是将区域设置为|*P*|+*k*长, 没有考虑定义4中局部最优化的条件, 从例 6 也可看出这种边界限定方法有时太松。这一节针对局部最优化匹配, 提出了改进的边界限定策略。

设结果串为 *R*, 由于从定理 3 已知 *R* 的结尾与 $P_j$ 匹配, 如图 5 所示, 则此时 $ed(R,P)\geqslant|M|$, 因为至少需要|*M*|个插入操作。又因为 $ed(R,P)\leqslant k$, 则有 $ed(R,P[1,j])\leqslant k-|M|$。根据性质 1 可知 *R* 的长度最长为|*P*|+*k*–2|*M*|, 由此可得如下定理。

**定理 5** 近似子串查询结果串的长度最长为|*P*|+*k*–2|*M*|, 因此当找到可能的极值位置 *i* 后, 只需向前扩展|*P*|+*k*–2|*M*|–1 个字符进行验证即可。

**例 8** 如例 6 中, 当找到位置 7 后, 因为对应的字符为“C”, 所以 *M*=0, 因此向前扩展|*P*|+*k*–1, 即 4 个字符形成验证区域。

## 6 过滤优化的局部最优化近似子串查询算法

将基于 gram 的查询算法与基于最优化匹配的边界限定和过滤策略相结合, 提出了过滤优化的局部最优化近似子串查询算法(gram-based local optimal approximate substring query algorithm with filtering, FGASQ )。FGASQ 算法是 GASQ 算法针对局部最优化匹配概念的改进, 引入了改进的边界限定和过滤策略, 提高了查询效率。

**算法 3** 过滤优化的局部最优化近似子串查询算法

输入: 短串 *P*, 长串 *T*, gram 索引 *G*, 阈值 *k*。
输出: 符合条件的子串位置。

1.
… 与 GASQ 算法相同
9.
10. *IsEx*=*Filter*(*pos*); //使用过滤策略验证位置
11. if (*IsEx* is true){
//使用改进的边界限定策略
12. *region*=[*pos*–|*P*|–*k*+2|*M*|+1,*pos*];
13. *Verify*(*region*);//使用 Naïve 算法验证
14. }//“if”
… 与 GASQ 算法相同
17.

**例 9** 使用 FGASQ 算法对例 6 进行计算, 计算过程与图 4 相似, 但是通过有效的边界限定和过滤策略缩短了最终的待验证区域, 如图 6 所示。综合例 7 和例 8, 5、6 位置被舍弃, 扩展 4 个字符形成待验证区域。

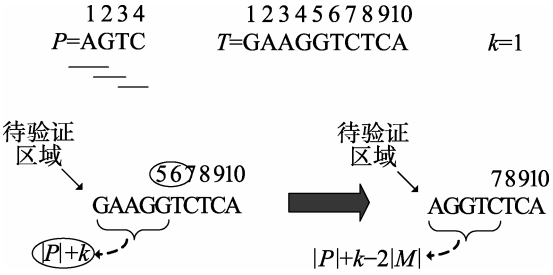



Fig.6 The difference between FGASQ and GASQ
**图 6 FGASQ 算法与 GASQ 算法的差别示例**

## 7 实验结果与分析

本文实验测试所使用的软硬件环境为:

(1) 操作系统是 Ubuntu(Linux) 10.04;

(2) 硬件环境是 Virtualbox 虚拟机, Pentium® Dual-Core CPU E5200@2.5GHz, 1 GB RAM。

测试使用的数据集为人类第一条染色体的基因数据。从该 DNA 数据中提取长度为 10 000 000(10M)的长串作为被查询的长串 *T*, 从 *T* 中随机抽取长度为 150 的短串作为查询短串 *P*。每次测试分别抽取 100 组 *P* 进行查询后取平均值。

**定义7**(验证区域比例) 最终通过 Naïve 算法进行验证的区域的大小占整个长串长度的比例, 计算公式为:

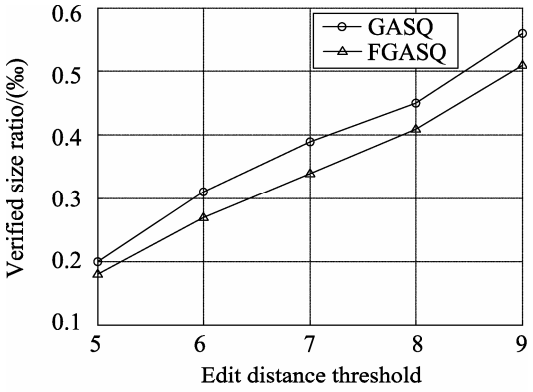
$$\text{验证区域比例}=\frac{\text{被验证的区域大小的总和}}{\text{被查询长串的长度}}$$

本文的测试部分使用验证区域比例和查询时间这两个指标来衡量算法的效率。验证区域比例和查询时间越小, 说明查询效率越高。

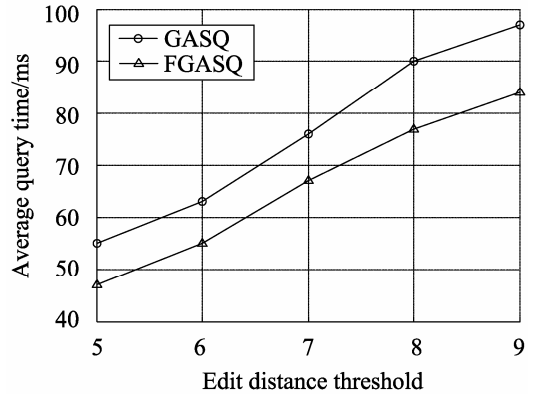
## 7.1 边界限定和过滤策略效果

分别对 GASQ 算法和 FGASQ 算法进行了测试对比, 通过比较可以看出边界限定和过滤策略对查询效率的影响。

从图 7 可以看出, 随着编辑距离阈值的增大, 查询时间也在增大, 这是因为查询得到的结果变多了。使用 FGASQ 算法的验证区域比例和查询时间比 GASQ 算法都有明显减少, 其中查询时间大概减少了 25%左右, 这说明边界限定和过滤策略明显提高了查询效率。




(a) Comparison of verified size ratio
(a) 验证区域比例比较

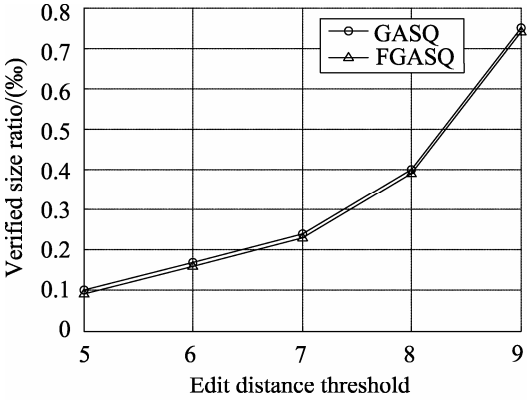



(b) Comparison of query time
(b) 查询时间比较

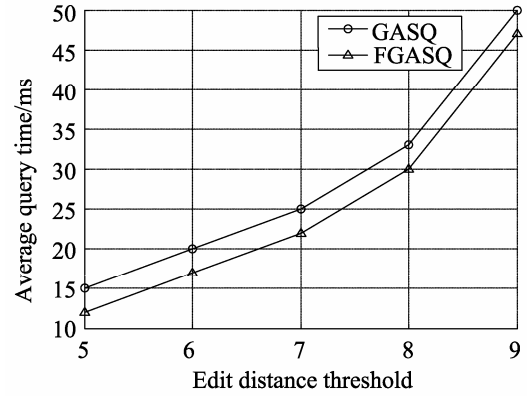
Fig.7 Comparison between GASQ and FGASQ when $|P|$=150 and $q$=9

**图 7 GASQ 和 FGASQ 在$|P|$=150 和 $q$=9 情况下的比较**

从图 8 与图 7 的比较可以看出, 增大了 gram 分解长度 $q$, 验证区域比例和查询时间都有明显改善, 这是因为 $q$ 增大, 减少了 gram 位置列表的数量。




(a) Comparison of verified size ratio
(a) 验证区域比例比较




(b) Comparison of query time
(b) 查询时间比较

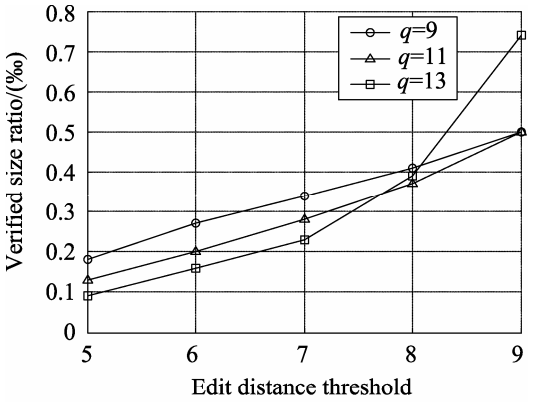
Fig.8 Comparison between GASQ and FGASQ when $|P|$=150 and $q$=13

**图 8 GASQ 和 FGASQ 在$|P|$=150 和 $q$=13 情况下的比较**

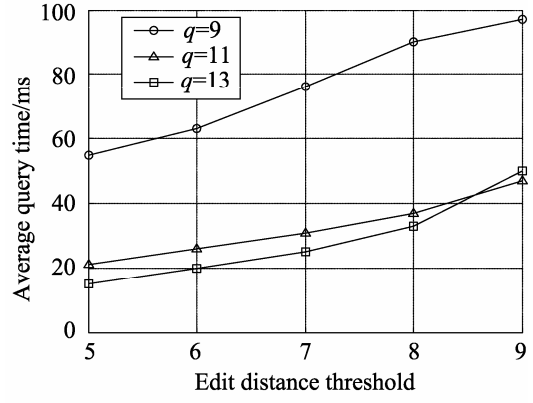
## 7.2 选取不同的 $q$ 长对查询效率的影响

选取不同的 gram 分解长度 $q$, 必然会对查询效率产生影响。这一节测试了在不同 $q$ 长下对 GASQ 和 FGASQ 算法产生的影响。

从图 9 中可以看出，基本的趋势为当 $q$ 增大时验证区域比例和查询时间都在减少，这是因为 $q$ 增大使 $P$ 分解得到的 gram 减少，减少了归并时间，增加了过滤效果，但是当 $q$ 增加到一定程度时，查询效率会下降。另外，虽然图9(a)中在 $k$=9 时，$q$=13 时的验证区域比例明显超过了另外两个，但是图 9(b)中却显示 $q$=13 时查询时间并不是最长的，这是因为 $q$=13 时虽然最后要验证的区域增大了，但是由于 gram 列表小，减少了归并时间。



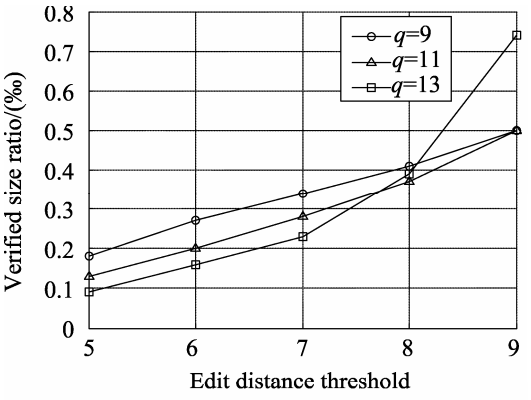

(a) Comparison of verified size ratio
(a) 验证区域比例比较



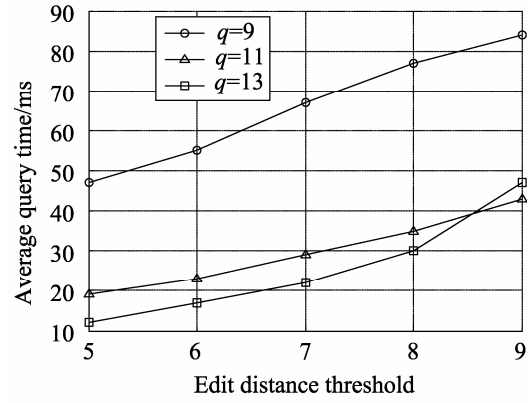

(b) Comparison of query time
(b) 查询时间比较

Fig.9 Effects of different $q$ on GASQ when $|P|$=150
图 **9** **GASQ** 在$|P|$**=150** 情况下选取不同 $q$ 的比较

图 10 中曲线与图 9 变化趋势相似，但是查询时间要比图 9 中的小，所以 FGASQ 与 GASQ 相比提高了查询效率。




(a) Comparison of verified size ratio
(a) 验证区域比例比较




(b) Comparison of query time
(b) 查询时间比较

Fig.10 Effects of different $q$ on FGASQ when $|P|$=150
图 **10** **FGASQ** 在$|P|$**=150** 情况下选取不同 $q$ 的比较

## 8 结论及未来工作

本文针对近似子串查询中存在的问题，提出了一种局部最优化匹配的概念，给出了支持局部最优化匹配的近似子串查询的定义。改进了现有的利用 gram 索引的近似查询算法，使之能够适应子串的近似查询，给出了一种基于 gram 索引的局部最优化近似子串查询算法。提出了基于局部最优化匹配的边界限定和过滤方法，并给出了一种过滤优化的局部最优化近似子串查询算法，通过测试表明其提高了查询效率。

下一步的工作将深入分析子串近似匹配过程中的规律，提高过滤效率。

## References:


[1] Levenshtein V I. Binary codes capable of correcting spurious insertions and deletions of ones[J]. Problems of Information Transmission, 1965, 1(1): 8–17.

[2] Sankoff D, Kruskal J. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison[M]. Reading, UK: Addison-Wesley Publication, 1983.

[3] Das G, Fleisher R, Gasieniek L, et al. Episode matching[C]// LNCS 1264: Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM ’97), Aarhus, Denmark, June 30-July 2, 1997. [S.l.]: Springer, 1997: 12–27.

[4] Needleman S, Wunsch C. A general method applicable to the search for similarities in the amino acid sequences of two proteins[J]. Journal of Molecular Biology, 1970, 48(3): 444–453.

[5] Knuth D E, Morris J H, Pratt V R. Fast pattern matching in strings[J]. SIAM Journal on Computing, 1977, 6(2): 323–350.

[6] Boyer R S, Moore J S. A fast string searching algorithm[J]. Communications of the ACM, 1977, 20(10): 762–772.

[7] Kim Y, Woo K G, Park H, et al. Efficient processing of substring match queries with inverted q-gram indexes[C]// Proceedings of the IEEE 26th International Conference on Data Engineering (ICDE ’10). Washington, DC, USA: IEEE Computer Society, 2010: 721–732.

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

[9] Kim M S, Whang K Y, Lee J G, et al. *n*-gram/2L: a space and time efficient two-level *n*-gram inverted index structure[C]//Proceedings of the 31st International Conference on Very Large Data Bases (VLDB ’05), 2005: 325–336.

[10] Zobel J, Moffat A. Inverted files for text search engines[J]. ACM Computing Surveys, 2006, 38(2).

[11] Yang Xiaochun, Wang Bin, Li Chen. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently[C]//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD ’08). New York, NY, USA: ACM, 2008: 353–364.

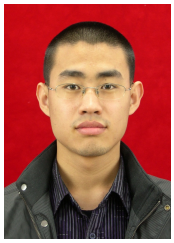
[12] Li Chen, Wang Bin, Yang Xiaochun. VGRAM: improving performance of approximate queries on string collections using variable length grams[C]//Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB ’07), 2007: 303–314.

[13] Li Chen, Lu Jiaheng, Lu Yiming. Efficient merging and filtering algorithms for approximate string searches[C]// Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE ’08). Washington, DC, USA: IEEE Computer Society, 2008: 257–266.

[14] Krauthgamer R, Mehta A, Raman V, et al. Greedy list intersection[C]//Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE ’08). Washington, DC, USA: IEEE Computer Society, 2008: 1033–1042.

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

[16] Behm A, Ji Shengyue, Li Chen, et al. Space-constrained gram-based indexing for efficient approximate string search[C]//Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE ’09). Washington, DC, USA: IEEE Computer Society, 2009: 604–615.


LIU Honglei was born in 1988. He is a master candidate at Northeastern University. His research interests include approximate string query and bioinformatics.

刘洪磊(1988—), 男, 河北任丘人, 东北大学硕士研究生, 主要研究领域为字符串近似查询, 生物信息学。

YANG Xiaochun was born in 1973. She received her Ph.D. degree in Computer Software and Theory from Northeastern University in 2001. Now she is a professor and doctoral supervisor at Northeastern University, and the senior member of CCF. Her research interests include theory and technology of database, data quality analysis and data privacy, etc.

杨晓春(1973—), 女, 辽宁沈阳人, 2001 年于东北大学计算机软件与理论专业获得博士学位, 现为东北大学教授、博士生导师, CCF 高级会员, 主要研究领域为数据库理论与技术, 数据质量分析, 数据隐私保护等。

WANG Bin was born in 1972. He is a lecturer at Northeastern University. His research interests include distributed database management and system structure.

王斌(1972—), 男, 辽宁沈阳人, 东北大学讲师, 主要研究领域为分布式数据管理, 体系结构。

JIN Rong was born in 1988. She is a master candidate at Northeastern University. Her research interest is approximate text search.

金蓉(1988—), 女, 湖北仙桃人, 东北大学硕士研究生, 主要研究领域为近似文本搜索。