# 生物序列数据 K-mer 频次统计问题的算法①

张鑫鑫[1,2], 陈 波[1,2], 何继凌[1], 徐 云[1,2]

[1](中国科学技术大学计算机科学与技术学院, 合肥 230027)
[2](安徽省高性能计算重点实验室, 合肥 230027)

**摘 要**: 生物序列的 k-mer 频次统计是生物信息处理中一个非常基础且重要的问题. 本文针对多序列在对齐模式下, 不同偏移处一段长度范围内的 k-mer 频次统计问题进行了研究. 提出了一种逆向遍历 k-mer 计数算法 BTKC. 该算法能够充分利用长度的 k-mer 统计信息, 快速得到长度的 k-mer 统计信息, 从而避免了统计任意长度的 k-mer 频次信息时都需要对所有序列进行遍历. 算法的时间复杂度分析及实验结果表明, 相比于传统的前向遍历 FTKC 算法, BTKC 算法性能提升非常明显, 且其时间复杂度与 k-mer 长度的变化范围无关, 非常适合于在 k-mer 长度变化范围较大的情况下使用.

**关键词**: k-mer; k-mer 计数; 频次统计; 逆向遍历; 生物信息处理

## Algorithms for Biological Sequence K-mer Frequency Counting Problem

ZHANG Xin-Xin[1,2], CHEN Bo[1,2], HE Ji-Ling[1], XU Yun[1,2]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
[2](Key Laboratory of High Performance Computing of Anhui Province, Hefei 230027, China)

**Abstract**: K-mer counting of biological sequence is a fundamental and very important problem in biological information processing. This paper focuses on counting k-mers at each position of multiple sequences within aligned mode. We present a new backward traverse k-mer counting algorithm called BTKC. BTKC algorithm takes full advantage of the k+1-mer's statistic information to obtain k-mer's statistic information quickly. Thus, it's no need to traverse the whole sequences when counting each single k-mer. Both the algorithm's time complexity and experiment results show that BTKC gets an obvious improvement compared with forward traverse k-mer counting algorithm FTKC, and its time complexity was found not to be realted with the range of k-mer length.

**Key words**: k-mer; k-mer counting; frequency statistic; backward traverse; biological sequence processing

生物序列主要指 RNA,DNA 及蛋白质序列等. 自人类基因组计划实施以来, 特别是随着第二代测序技术大规模的使用与推广, 现今一次深度测序技术可以产生 TB 级的序列数据. 而在对这些海量数据的分析和应用中, k-mer 频次统计是一个非常基础且重要的问题.

k-mer 频次统计信息可以用来揭示生物序列中各种子序列的分布规律, 它是一种衡量序列相似性的重要工具. 因而其在单体分型, 模体发现, 物种识别, 宏基因组分类, 序列拼接, 多序列比对, 变异探测及序列纠错等众多的生物学问题上都有着重要且广泛的应用. 特别是在单体分型及模体发现等一些需要探究序列中块属性的问题上, 常常还需要对多序列中不同偏移处一段长度范围内的 k-mer 进行频次统计. 鉴于此, 本文针对多序列在对齐模式下, 不同偏移处一段长度范围内的 k-mer 频次统计问题进行了研究, 并提出一种逆向遍历 k-mer 计数算法 BTKC, 相比于传统的前向遍历算法算法 FTKC, 该算法显著降低了 k-mer 频次统计算法的时间复杂度.

①基金项目:国家自然科学基金(60970085)

收稿时间:2013-08-29;收到修改稿时间:2013-09-26

## 1　相关工作

郝柏林院士等在2000年提出了一种使用2D分形图像的可视化方法来反映序列的 k-mer 的频次分布，并在此基础上设计了基于B树的快速k-mer频次统计算法[1,2]. 王树林等在2007年设计并实现了k-长DNA子序列内部计数算法和外部计数算法，在该算法中，利用一个巧妙构造的hash函数把k-长DNA子序列映射为整数，从而把 k-长 DNA 子序列的频次统计问题转化为整数关键字的重复计数问题，使得能够利用经典 B 树算法来解决 k-长 DNA 子序列的出现频次计数问题[3]. Marcais 等人在 2011 年提出了并实现了并行的 Jellyfish 算法对 k-mer 的频次进行统计，该算法在多线程环境下利用了前缀数组和优化的无锁哈希表，该算法不仅效率高并且是内存有效的[4]. Melsted 等人则在 2011 年使用 Bloom filter(布隆过滤器，一种概率数据结构)来存储 k-mer 频次统计过程中出现的各种 k-mer 子串，然后再进行二次扫描获取具体的 k-mer 频次，试验数据表明，该算法相对于其他基于 hash 的算法，能够显著减小内存的使用[5]. 而在 2013 年，Rizk 等人又提出了一种分块的 k-mer 频次统计算法 DSK. 该算法综合利用了 hash 映射和分块缓存，确保在任意规模的数据集上，该算法均可以在内存和磁盘空间受限情况下正常运行[6]. 但对于单体分型及模体发现等一系列需要对序列中不同偏移位置处一段长度范围内的k-mer进行频次统计从而探究序列的块属性的问题，目前鲜有研究.

## 2　问题定义及描述

我们给出 k-mer 的定义如下：设 $s$ 为长为 $m$ 的生物序列，则 $s=q_1q_2\cdots q_m$，其中,$q_1\in\sum$ ($\sum$ 为生物序列的符号空间，比如对于基因序列，则有$\sum=\{A,T,G,C\}$). 一个长为 $k$ 的子串是指序列 $s$ 中从任意位置处开始的个连续符号，称之为 k-mer.

对于多序列在对齐模式下，不同偏移处一段长度范围内的k-mer频次统计问题，我们给出如下定义：记 $\Omega$ 为 $n$ 条长为 $m$ 的基因序列在对齐模式下构成的序列集合，则 $\Omega=\{s_1,s_2,\cdots,s_n\}$，对于任意的$1\le i,j\le n$，要求 $s_i$ 和 $s_j$ 的起始位置对齐. 对于给定的 k-mer 长度变化范围 $k_1$ 和 $k_2(k_1\le k_2)$，分别对于各个 k-mer 长度 $k(k_1\le k\le k_2)$ 在各个偏移位置 $l(0\le l\le m-k)$ 上，计算由 $n$ 条序列中每一条序列从偏移 $l$ 处取长为 $k$ 的 k-mer 所构成的序列块中不同 k-mer 子串的出现频次.

图 1 给出了一个简单的示例，所有序列的起始位置对齐，k-mer 长度的变化范围为 3~5，首先计算偏移为 0 时，长度为 3 的红色序列块中各个 k-mer 的出现次数，然后计算长度为 4 的黑色序列块中各个 k-mer 的出现次数，直到最大值 5. 然后偏移位置向右滑动一位变为 2，再依次统计各 k-mer 长度下各个子串的出现频次. 不断的重复上述过程，直至到达序列的尾部.

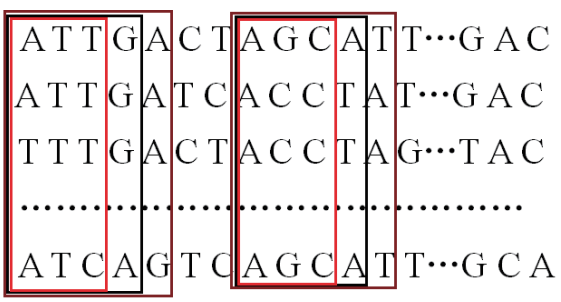


图 1　k-mer 示例图

## 3　算法描述及复杂度分析

对于 2 中描述的问题，目前的常规算法是前向遍历计数算法，我们简称其为 FTKC. 在 FTKC 算法中，对于每一个偏移位置 $l$，依次从 k-mer 长度变化范围的最小值 $k_1$ 遍历到最大值 $k_2$. 在每一次遍历时，均会扫描所有 $n$ 条序列，取每条序列当前偏移处，长度为当前 k-mer 长度的子串，使用 hash 表来来统计和存储各子串的出现频次.

对于 FTKC 算法，由算法的运行过程可知，在由 $n$ 条长度为 $m$ 的序列构成的序列空间中，对于任意 k-mer 长度 $k(k_1\le k\le k_2)$ 共构成了 $m-k+1$ 个从左向右滑动的块. 对每一个块进行 k-mer 频次统计时，均需要依次访问 $n$ 条序列，进行 $n$ 次 hash 操作，因而该算法的时间复杂度为 $O((k_2-k_1)mn)$.

注意到上述 FTKC 算法每次在块中进行 k-mer 统计时，均需要遍历所有 $n$ 条序列，而没有考虑充分利用已有的统计结果. 但我们仔细观察图 1 可以看出，对于同一偏移处，长度为 $k$ 的块中每一条序列均是长度为 $k+1$ 的块中每一条序列的长为 $k$ 的前缀字符串. 比如，对于偏移 0 处的第一条序列，ATT 为 ATTG 的前缀字符串. 更进一步考虑，比如对于 80 条序列构成的序列空间，如果我们已经得到了偏移为 0，k-mer 长度为 4 时的统计结果为{ATTA:12，ATTG:8，ATTC:9，

ATTT:10, ATCA:8, ATCG:7, ATCC:15, ATCT:11}, 那么下一步我们仅仅只需要对上述得到的 k-mer 频次结果集合进行遍历, 对提取的每一条记录取其长为 3 的子串并使用 hash 表进行计数, 则只需要 8 次遍历操作, 就可以得到偏移为 0, k-mer 长度为 3 时的结果为{ATT:39, ATC:41}, 然后再遍历 k-mer 长度为 3 时的结果集合, 得到 k-mer 长度为 2 时的结果为{AT:40}.

因此在我们提出的 BTKC 算法中, 对于每一偏移处, 首先遍历所有 $n$ 条序列, 取每条序列中当前偏移处长度为 $k_2$ (k-mer 长度最大值)的子串, 将该子串做为 key, 该子串的出现频次做为 value, 使用 hash 表进行统计和存储, 得到结果为 $H_{k_2}$. 下一步我们首先构建一个空的 hash 表 $H_{k_2-1}$, 然后对上一步得到的 $H_{k_2}$ 进行遍历, 对遍历得到的每个记录, 分别取其 key 的长为 $k_2-1$的前缀子串做为新的 key, 将该记录的 value 累加到$H_{k_2-1}$ 中该 key 对应的 value 上, 最终得到该偏移处, k-mer 长为 $k_2-1$时的频次统计结果. 不断重复上述过程, 通过遍历 k-mer 长度为 $k+1$时的已得结果, 得到 k-mer 长度为 $k$时的统计结果, 直至得到该偏移处 k-mer 长度为最小值 $k_1$时的结果 $H_{k_1}$. 然后再滑动到下一个偏移处, 重复上述过程, 直至滑动到最右偏移处, 最终得到所有的统计结果. 完整的算法如下所示:

输入: $n$ 条长为 $m$ 的序列集合 , 最小 k-mer 长度 $k_1$, 最大 k-mer 长度 $k_2$

输出: 各偏移处各个不同 k-mer 长度下的 k-mer 子串及其出现次数

```
Begin
初始化一个链表 T 用来存储各偏移处的结果
for 偏移 l 从 0 到 m − k2
  初始化一个链表 R, 用来存储偏移为 l 时的结果
  for k-mer 长度值 k 从 k2 到 k1
    初始化一个 hash 表 Hk
    If k 等于 k2
      for Ω 中的每一条序列
        取 s 从偏移 l 处开始长度为 k 的子串为 ss
        如果 ss 在 Hk 中, 则 Hk[ss] += 1
        如果 ss 不在 Hk 中, 则 Hk[ss] = 1
      end for
    else  // k 不等于 k2 时
      取 T 中的头节点获得已知的 Hk+1
      for Hk+1 中的每一条记录 hh
        设 K 为 hh 的 key 的长为 k 的前缀子串
        设 V 为 hh 的 value
        如果 K 在 Hk 中, 则 Hk[K] += V
        如果 K 不在 Hk 中, 则 Hk[K] = V
      end for
    将 Hk 添加到链表 R 的头部
  end for
  将 R 添加到链表 T 的头部
end for
End
```

对我们提出的逆向遍历算法 BTKC, 对于任意一偏移位置, 仅需在计算最大 k-mer 长度($k_2$)的频次结果时, 需要遍历所有 $n$ 条序列, 并假设此时得到的结果集合大小为 $n'(n'\le n)$. 对于计算其它长度为 $k(k_1\le k\le k_2)$ 的 k-mer 结果时, 我们仅需遍历上次已经获得的长度为 $k+1$ 的结果集合中每一条记录. 假设生物序列符号空间Σ 中每个符号在生物序列空间 Ω 中均匀分布, 则可知我们新获得的结果集合的大小是上次所得结果集合大小的 $1/|\Sigma|$, 重复上述过程, 直至到达 k-mer 长度的最小值($k_1$), 再转到下一偏移位置处, 继续从最大长度 k-mer 统计到最小长度. 因此可知整个算法的计算量为 $m(n+n'/|\Sigma|+n'/|\Sigma|^2\cdots+n'/|\Sigma|^{k_2-k_1})$, 由于 $n'\le n$ 且 $|\Sigma|\ge 2$, 所以整个算法的时间复杂度为 $O(mn)$, 且与 $k_1$ 和 $k_2$ 的取值无关. 与 FTKC 算法相比, BTKC 的时间复杂度仅为前者的 $1/(k_2-k_1)$. 特别是当 $k_2$ 和 $k_1$ 差值较大时候, BTKC 算法的性能提升更加明显.

## 4 测试结果及结论

实验中我们采用的是基因序列, 使用随机算法生成我们的模拟数据集. 测试平台为: Intel Core i5-2400 CPU (4 cores, 3.10GHz), DDR2 RAM (4.00GB), Win7 OS.

图 2 给出的了在相同序列长度(长为 100), 相同的 k-mer 变化范围(2~7), FTKC 和 BTKC 的运行时间与序列条数之间的关系. 从图中可以看出, 随着序列条数的增加, 两种算法的运行时间均呈线性增加. 该观察结果与我们在算法复杂度中分析的结果一致, 即两种算法时间复杂度与序列条数呈线性相关.

图 3 给出了在相同序列条数(5000 条), 相同的

k-mer 变化范围(2~7)时, FTKC 和 BTKC 的运行时间与序列长度之间的关系. 从图中可以看出, 随着序列长度的增加, 两种算法的运行时间也都呈线性增加. 该观测结果也与我们在算法复杂度中的分析结果一致, 即两种算法的时间复杂度与序列长度的呈线性相关.

图 4 中给出了在相同序列条数(10000 条), 相同序列长度(100), 不同k-mer长度变化范围(为了方便比较, k-mer 长度的最小值固定为 2, k-mer 长度的最大值变化)时, FTKC 和 BTKC 两种算法的运行时间与 k-mer 长度变化范围之间的关系. 从图中可以看出, FTKC 算法的运行时间随着 k-mer 长度变化范围的增加而呈线性增加, 而 BTKC 算法的运行时间在不同 k-mer 长度变化范围下几乎不变, 该结果表明, 相比较 FTKC 算法而言, 我们提出的 BTKC 算法的时间复杂度与 k-mer 长度变化范围无关, BTKC 算法的时间性能是 FTKC 的 $k_2 - k_1$ 倍, 特别是随着 k-mer 最大长度的增加, BTKC 算法的性能改进就越明显.

为了确定算法的有效性, 我们在真实数据集上进行了测试. 真实数据源自 D.ananassae 数据集中 Illumina RNA-Seq SRR332538 中前 10000 条长为 150 的 RNA 序列. 图 5 给出了在相同序列条数(10000 条), 相同序列长度(150), 不同 k-mer 长度变化范围(为了方便比较, k-mer 长度的最小值固定为 2, k-mer 长度的最大值变化)时, FTKC 和 BTKC 两种算法的运行时间与 k-mer 长度变化范围之间的关系. 从图中可以看出, 两种算法的运行时间变化趋势与我们在图 4 种观测的结果基本一致. 该结果表明, 相比较 FTKC 算法而言, BTKC 算法的时间性能在模拟数据集和真实数据集上都有着非常明显的提升, 这充分表明了 BTKC 算法的有效性.

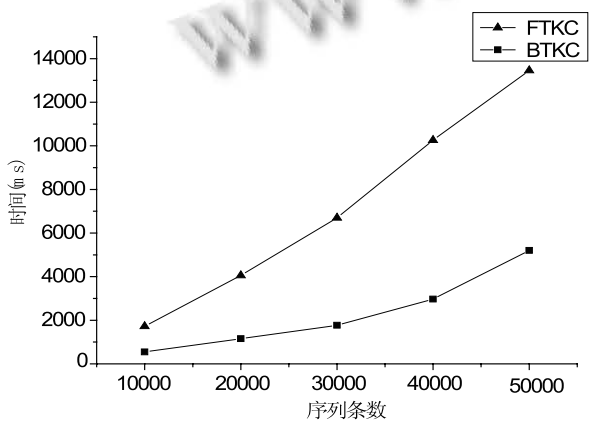


图 2　序列条数与运行时间关系

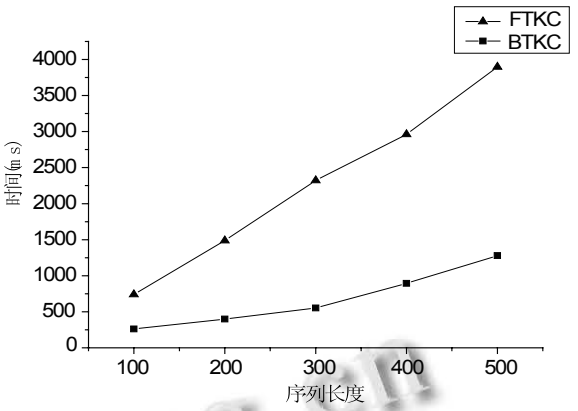


图 3　序列长度与运行时间关系

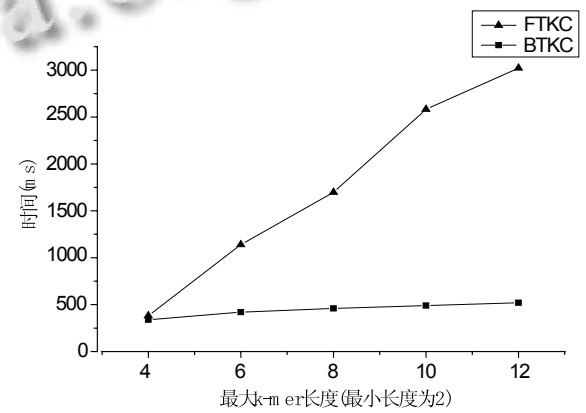


图 4　k-mer 长度变化范围与运行时间关系

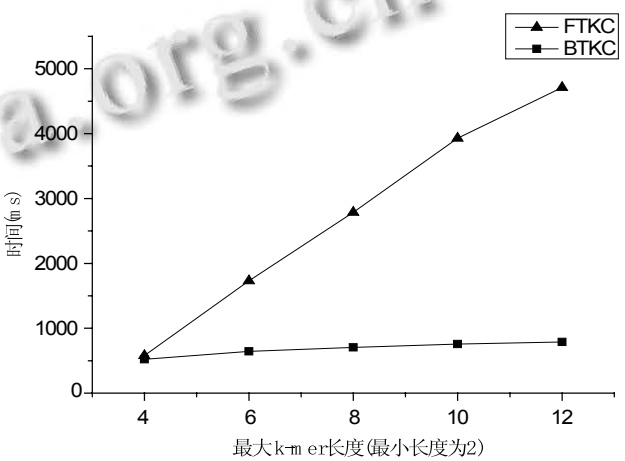


图 5 真实数据集(Illumina RNA-Seq SRR332538)上 k-mer 长度变化范围与运行时间关系

## 5　结语

本文针对多序列在对齐模式下, 不同偏移位置

(下转第 158 页)

## 3 结语

本文针对四带树木图像的偏色现象, 介绍了一种基于RGB色彩空间的分区域颜色校正方法. 这种方法根据四带树木图像在近红外光线影响下普遍偏品红的特征, 使用 RGB 模型中的色差 R-G 和 R 值对四带树木图像进行背景分离, 并利用中值滤波器和数学形态学对分割后的图像进行滤噪处理. 实验结果表明, 对光照比较弱、背景颜色与树木颜色相差较大的图像, 通过选择合适的阈值, 色差 R-G 和 R 值特征可以将大部分树木分割出来, 分区域颜色校正能够获得较好的校正效果. 但是, 对于背景颜色与四带树木图像颜色较接近或者光照较强时, 分割和校正效果都不太理想. 但本文的算法对这种图像的校正以及后续的四带树木图像的研究具有一定的参考价值.

## 参考文献


1 徐晓昭,沈兰荪,刘长江.颜色校正方法及其在图像处理中的应用.计算机应用研究,2008,25(8):2250–2254.

2 徐晓昭,蔡轶珩,刘晓民, 等.改进灰度世界颜色校正算法.光子学报,2010,39(3):559–564.

3 文帅,蔡轶珩,张新峰,等.基于色貌模型的图像颜色校正.测控技术,2010,29(5):19–22.

4 Naim M, Mat Isa NA. Pixel distribution shifting color correction for digital color images. Applied Soft Computing, 2012, 12(9): 2948 –2962.

5 Gijsenij A, Gevers T. Color constancy using natural image statistics and scene semantics. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(4): 687–698.

6 Navalpakkam V, Itti L. Modeling the influence of task on attention. Vision Research, 2005, 45(2):205–231

7 Finlayson GD, Drew MS. White-point preserving color correction. Proc. of the 5th Color Imaging Conference on Color, Science, Systems Applicaionts. Scottsdale, Arizoona. Society for Information Display. 1997. 258–261.

8 王易.基于 K 均值聚类分割彩色图像算法的改进.计算机应用与软件,2010,27(8):127–130.

9 胡国飞,傅健,彭群生.自适应颜色迁移.计算机学报,2004,27(9):1245–1249.

10 蔡世捷.基于Matlab的树木图像分割方法研究[学位论文].南京:南京林业大学,2005.

11 Yin J, Cooperstock JR. Color correction methods with applications to digital projection environments. Journal of WSCG, 2004, 12(1): 499–506.


(上接第 124 页)

处一段长度范围内的 k-mer 频次统计问题进行研究, 相比传统的基于正向遍历的k-mer计数算法FTKC, 提出并实现了一种逆向遍历 k-mer 计数算法 BTKC. 在每一偏移位置处, BTKC 算法首先遍历所有序列, 得到最长 k-mer 的频次统计信息, 在对其他长度的 k-mer 频次信息进行统计时, 均是利用已经获得长度的 k-mer 统计信息, 快速得到长度的 k-mer 统计信息, 从而避免了 FTKC 算法中, 统计每个长度的 k-mer 信息时均需要对所有序列进行遍历. 算法的时间复杂度分析及实验结果表明, 相比于 FTKC 算法, BTKC 算法性能提升明显,且其时间复杂度与 k-mer 长度的变化范围无关, 非常适合于在 k-mer 长度变化范围较大的情况下使用.

## 参考文献


1 Hao BL. Fractals from genomes–exact solutions of a biology-inspired problem. Physica A: Statistical Mechanics and its Applications, 2000, 282(1): 225–246.

2 Hao B, Lee HC, Zhang S. Fractals related to long DNA sequences and complete genomes. Chaos, Solitons & Fractals, 2000, 11(6): 825–836.

3 王树林,王戟,陈火旺,等.k-长 DNA 子序列计数算法研究.计算机工程,2007,9:14.

4 Marçais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics, 2011, 27(6): 764–770.

5 Melsted P, Pritchard JK. Efficient counting of k-mers in DNA sequences using a bloom filter. BMC bioinformatics, 2011, 12(1): 333.

6 Rizk G, Lavenier D, Chikhi R. DSK: k-mer counting with very low memory usage. Bioinformatics, 2013, 29(5): 652–653.