PLOS ONE

RESEARCH ARTICLE

Fast sequence analysis based on diamond sampling

Liangxin Gao[1], Wenzhen Bao[1], Hongbo Zhang[1], Chang-An Yuan[2], De-Shuang Huang[1]*

1 Institute of Machine Learning and Systems Biology, School of Electronics and Information Engineering, Tongji University, Shanghai, China, **2** Science Computing and Intelligent Information Processing of GuangXi Higher Education Key Laboratory, Guangxi Teachers Education University, Nanning, Guangxi, China

* dshuang@tongji.edu.cn

Citation: Gao L, Bao W, Zhang H, Yuan C-A, Huang D-S (2018) Fast sequence analysis based on diamond sampling. PLoS ONE 13(6): e0198922. https://doi.org/10.1371/journal.pone.0198922

Editor: Fengfeng Zhou, Jilin University, CHINA

Received: January 27, 2018

Accepted: May 29, 2018

Published: June 28, 2018

Data Availability Statement: All data are available from the JASPAR database (JASPAR CORE REDUNANT 2016) and mouse and human genome. The particular data used in our study: http://jaspar2016.genereg.net/cgi-bin/jaspar_db.pl.

Funding: This work was supported by the grants of the National Science Foundation of China, Nos. 61732012, 61520106006, 31571364, 61672382, 61772370, U1611265, 61532008, 61472280, 61472173, 61702371, and 61772357, China Postdoctoral Science Foundation Grant, Nos. 2016M601646 and 2017M611619, and supported by "BAGUI Scholar" Program of Guangxi Province of China. De-Shuang Huang received the funding.

Abstract

Both in DNA and protein contexts, an important method for modelling motifs is to utilize position weight matrix (PWM) in biological sequences. With the development of genome sequencing technology, the quantity of the sequence data is increasing explosively, so the faster searching algorithms which have the ability to meet the increasingly need are desired to develop. In this paper, we proposed a method for speeding up the searching process of candidate transcription factor binding sites (TFBS), and the users can be allowed to specify *p* threshold to get the desired trade-off between speed and sensitivity for a particular sequence analysis. Moreover, the proposed method can also be generalized to large-scale annotation and sequence projects.

Introduction

Transcription factors (TFs) can suppress or activate gene expression by binding to specific DNA sites (TFBS),[1] and therefore play a central role in transcription regulation. Previous researches have concluded that TFs are inclined to bind to DNA sequences that follow specific patterns, called TF motifs.[2] Recently, the recognition of candidate TFBS on chromatin with a given TF motif has become a fundamental step to initiate the transcription of its target genes.

The existing candidate TFBS searching algorithms can be generally divided into the index-based algorithms and the online algorithms. The index-based algorithms commonly construct some special index structures, such as suffix trees[3, 4] or suffix arrays,[5] to accelerate the accessing of all the candidate's locations on the target sequence. However, although these index structures could improve the searching efficiency, their construction costs in the time and space are remarkably huge. On the other hand, the traditional online approaches commonly scan a target sequence from left to right with a sliding window whose width is the same as the given TF motif, then report its candidate TFBS. Although these methods avoid the costs of constructing index structures, their time complexities still reach O(mn), where m and n are the lengths of the target sequences and the TF motif respectively. Recently, many technologies have been integrated into the traditional online algorithms, including matrix partitioning,[6, 7] Fast Fourier Transform[8], data compression[9] and other similar approaches to reduce the time complexity. In those methods, the most representative algorithm is lookahead approach

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

[10] which resolves the problem of repeated matching by using partial thresholds that allow one to terminate the comparison of symbols when it is clear that no match will occur. However, it is unclear how efficient they can be in more challenging applications such as searching on high-throughput datasets or dealing with wider motif or less conservative TF motif. Additionally, in the TF motif modeling community, position weight matrix(PWM)[2], modeling the TF binding affinities with a matrix which describes the binding affinities as probability distributions over DNA alphabet, is one of the most commonly used TF models. And most of the candidate TFBS searching algorithms are based on this TF model, including naive algorithm, lookahead scoring algorithm, shift-add algorithm and so on.

Computationally, the given PWM can be regarded as the input features [11–13]. Motivated by this, the target sequence can be converted as a queried feature sets, if we split it into subsequences with a sliding window from left to right. Then the task of candidate TFBS searching can be abstracted as a classical k-Nearest Neighbor (KNN) problem, which is a widely studied formulation in the field of machine learning. With the formulation of candidate TFBS searching, there is no need to prepare index structures, in addition, it is anticipated that some advanced KNN technologies can be adopted to this formulation for obtaining high-quality solutions efficiently. In this paper, we will introduce a popular online search algorithm for quickly searching the significant matches. The inspiration come from the study of maximum all-pairs dot-product (MAD) searching problem which is the most complex part of the time complexity in the KNN technologies[14].

The key of KNN technologies is how to determine the similarity between samples which are represented by vectors with high-dimensional feature, i.e., $\mathbf{w} \in \mathrm{R}^d$, for some large d in the real world. Many studies of real world calculate dot product as a criterion for judging similarity, hence, $\mathbf{v} \cdot \mathbf{w}$ is a usually useful measure of distance between $\mathbf{v}$ and $\mathbf{w}$. Through MAD searching, the corresponding center point of every sample can be found, however, all-pairs dot-product takes a lot of time and in fact we only need to care about the large dot-product, not all-pairs dot-product. Diamond sampling algorithm[15] which is a new randomized approach is creatively promoted to solve the MAD problem by applying index sampling methods to avoid calculating all-pairs dot-product[16]. Designing a sampling procedure according to the relevant weights for the MAD problem is the key to the execution of diamond sampling algorithm.

The task of candidate TFBS searching is to find the max matching between position weight matrix and sequences. Using the idea of sampling according to the weights of diamond sampling algorithm, in the task of candidate TFBS searching, the original standard ordering which matches between position weight matrix and sequences in left-to-right order can be changed to a given permuted ordering[17]. The permuted ordering is given according to the matching failed expectation of every column in the position weight matrix and the corresponding expectation can be calculated through considering the distribution of background in the sequences, the information content of every column and so on.

In Results part, we demonstrated a series of experimental comparisons among the proposed algorithm and other current methods.

Methods

Notations and problem definition

The task of candidate TFBS searching is to find some appropriate subsequences for a given PWM in the sequences consisting of symbols in the four-nucleotide alphabet $\Re = \{\mathrm{A}, C, G, T\}$. In some literatures, the position weight matrix which is a real valued $n \times |\Re|$ matrix $\mathbf{M} = (\mathbf{M}(i,a))$ is also called position-specific scoring matrix, profile, and position weighted pattern. In this paper, we use PWM or pattern as for the abbreviation of position weight matrix for

convenience. In the PWM, the coefficient ($\mathbf{M}(i,a)$ represents the probability of alphabet a at each position i. The length and the alphabet of the M can be represented n and $\Re$, respectively. Table 1 is an example of pattern **M**.

The given pattern **M** can match any segments $\mathbf{Seq} = s_1 s_2 \ldots s_n$ where s_i represents a symbol in the alphabet $\Re$. The significant of match is calculated by the match score $G_M(s)$ of s with corresponding **M**. The match score is defined as

$$G_M(\mathbf{S}) = \sum_{i=1}^{n} \mathbf{M}(i, s_i). \quad (1)$$

Given some biological sequences $\mathbf{Seq} = s_1 s_2 \ldots s_m$ which consists of m symbols from the alphabet $\Re$. For any subsequences $s_i s_{i+1} \ldots s_{i+n-1}$ (call a k-mer) of length n in the sequence **Seq**, the significant of match can be obtained with given the pattern **M**. The significant of match at location i in the **M** can be denoted as

$$g_i = G_M(s_i s_{i+1} \ldots s_{i+n-1}) \quad (2)$$

In this paper, the problem of candidate TFBS searching can be described as follows: given a real-valued significance threshold k, the searching problem with threshold k is to find all location i in the sequence **Seq** such that $g_i \geq k$. In addition to getting all location i, the values g_i should also be provided for many applications. Note that the traditional studied problem of exact string matching is just a special case of this search problem. The problem of exact string matching can be simply described as finding all the positions i which are the start position of the expectant string $\mathbf{s} = s_1 s_2 \ldots s_n$ where all s_i is in the alphabet $\Re$. We can generate a weight matrix **M** meeting the following conditions: $\mathbf{M}(i,s_i) = 1$, and $\mathbf{M}(i,a) = 0$ if $a \neq s_i$, to solve easily the exact pattern search problem. The significant threshold k in this case can be explained for finding the exact positions where the alphabets of **s** match the corresponding alphabets of **Seq** in at least k positions. The problem of exact string matching can be solved easily when the significant threshold k be set to the same as the length of **s**.

Preprocessing of pattern and significant threshold

If the background sequence distributions is different, the conservativeness of the corresponding PWM is also different, even if the pattern is the same as each other[18]. In order to make effective use of this nature, the values $\mathbf{M}(i,a)$ are in fact the log-odds scores of a probabilistic model of a signal to be detected against the background, such as finding putative binding sites of transcription factors in DNA. The signal model can be normally represented by a $n \times |\Re|$ matrix **I** where $\mathbf{I}(i,a)$ is the probability of the alphabet symbol a occurring in the position i. There are many ways to obtain those probabilities such as from the corresponding empirically constructed count matrix and some of those probabilities may need to add pseudocounts to avoid logarithm with zero.

Table 1. A position weight matrix in the nucleotide sequence databases.

position	1	2	3	4	5	6
A	0.14	-4.16	1.03	-4.16	0.58	-0.36
C	0.17	-2.31	-4.16	-4.16	-2.31	-1.32
G	-1.06	1.64	-2.32	-0.85	-1.06	1.12
T	0.12	-4.16	-2.64	1.18	0.07	-0.77

The pattern was obtained from the count matrix GATA-3 of JASPAR which was transformed into log-odds matrix using background distribution $\mathbf{q}_A = 0.278$,$\mathbf{q}_C = 0.312$, $\mathbf{q}_G = 0.212$, $\mathbf{q}_T = 0.198$. A pseudocount $\mathbf{q}_s$ was first added to the counts for each alphabet symbol s.

https://doi.org/10.1371/journal.pone.0198922.t001

In order to facilitate the calculation, the background is usually constructed as a $n \times |\Re|$ matrix B and every row of matrix **B** has the same probability vector which equal to the probability of background probability distribution at corresponding the alphabet symbols. Since the probabilities of every row is the same, we use $\mathbf{q}_a$ to denote the background probability distribution for $a \in \Re$. Hence, $\mathbf{B}(i,\mathrm{a}) = \mathbf{q}_a$ for all i.

According to the log-odds score that consider the probability both in the signal model **M** and the background **B**, the match score between a subsequence and a matrix can be calculated as:

$$\begin{aligned} Score(\mathbf{S}) = \log\prod_{i=1}^{n}\frac{\mathbf{I}(i,\mathbf{S}_i)}{\mathbf{B}(i,\mathbf{S}_i)} &= \sum_{i=1}^{n}\log\frac{\mathbf{I}(i,\mathbf{S}_i)}{\mathbf{B}(i,\mathbf{S}_i)} \\ &= \sum_{i=1}^{n}\log\frac{\mathbf{I}(i,\mathbf{S}_i)}{\mathbf{q}_{\mathbf{S}_i}} \end{aligned} \tag{3}$$

Once the background **B** which is usually estimated from the sequence **S** is fixed, the model **I** can be translated into a position weighted matrix **M**:

$$\mathbf{M}(i,a) = \log\frac{\mathbf{I}(i,a)}{\mathbf{B}(i,a)} \tag{4}$$

Then, the score computed by (1) equals the above $Score(\mathbf{S})$.

The significance threshold k for the search is normally not easy to get, such that the standard statistical approach that use the p-values to control the confidence of the searching is more commonly used. When the p-value p is given, then the corresponding threshold is a value $k = k(p)$ and the probability of subsequence **s** is p such that $G_M(\mathrm{s}) \geq k$ in the background distribution. In order to convert p-value p to the significance threshold k more quickly, we use a well-known pseudopolynomial time dynamic programming algorithm [10, 19, 20] to evaluate the corresponding $k = k(p)$.

The faster lookahead scoring algorithm

In this section, some theoretical analysis is firstly used to compare our fast searching algorithm with some well-known searching algorithms.

When **S**, **M** and k are given, the TFBS searching problem can be solved by evaluating $\mathrm{w_i}$ using (1) and (2) for each $i = 1,2,\ldots,m\text{-}n\text{+}1$, then reporting all location i and corresponding w_i such that $w_i \geq k$. This primitive method is called as the naive algorithm (NA). For the time complexity of it, evaluating each w_i from (1) takes $O(n)$ such that the total searching time approximates to $O(mn)$, where n is width of the given pattern and m is the length of the given sequence **S**.

The main reason for which the low searching efficiency for the naïve algorithm is that the NA ignores the significant relationship between the score of segment w_i and the significance threshold k. When the NA algorithm process the subsequence, the NA algorithm does not end the matching process until it matches the end of the subsequence, even if it had been known some important information that the current subsequence cannot be matched successfully according to the currently matched information. In order to overcome the shortcoming of the NA algorithm, the lookahead scoring algorithm[10] is proposed as an improved algorithm of the NA algorithm, which utilize the significant relationship between $\mathrm{w_i}$ and the significance threshold k to speed up matching process. In order to guarantee each prefix of a candidate segment to decide quickly whether candidate segment match successfully, the lookahead scoring algorithm computes the intermediate score thresholds in advance. When the sequence **S**, the matrix **M** and the threshold k are given, although the score of the prefix of every candidate subsequences is fixed, the intermediate score thresholds cannot be directly computed by the score

of the prefix candidate segment. Through the evaluation for the maximal score of the suffix of a candidate subsequent, the intermediate score thresholds can be defined for $1 \leq t \leq n$ as

$$P_t = \sum_{i=t}^{n} \max_{\alpha \in \Re} \mathbf{M}(i, \alpha) \tag{5}$$

Obviously, $\mathbf{p}_t$ is the maximum score of all possible matches for the suffix which starts at position $t+1$ of any given candidate k-mer $\mathbf{S}_i \ldots \mathbf{S}_{i+n-1}$. The segment ending at position t of candidate k-mer $\mathbf{S}_i \ldots \mathbf{S}_{i+n-1}$ is the prefix and its match score can be computed as:

$$\mathbf{R}_t = G_M(\mathbf{s}_i \ldots \mathbf{s}_{i+t-1}) = \sum_{j=1}^{t} \mathbf{M}(j, \mathbf{s}_{i+j-1}) \tag{6}$$

When a segment is being matched at the position t, if $\mathbf{R}_t+\mathbf{P}_t$ is less than the matching significance threshold k, then we can draw a definite conclusion immediately that this k-mer cannot be matched successfully even if the matching score of suffix reach maximum. That reason of such decision can be interpreted as that the final match score is impossible for more than k for any possible k-mers which have the same prefix as the given sequence **Seq**.

When we have prior information about the possible match failure of currently matching segment, the matching process between sequences and model can be speeded up. In order to avoid the frequent calculation of intermediate score thresholds, all the intermediate thresholds can be calculated firstly as

$$\mathbf{T}_t = k - \mathbf{P}_t \tag{7}$$

And saved as an array for $t = 1,\ldots,n$. The actual matching subsequence is generated by the sliding window with the length n on the given sequence, and the starting position of the sliding window increment by 1 for next match, for example, the subsequence $\mathbf{S}_i \ldots \mathbf{S}_{i+n-1}$ is the current matching k-mer and the next match k-mer will be $\mathbf{S}_{i+1} \ldots \mathbf{S}_{i+n}$. For the strategy of match, the current matching score is initialized to zero before matching the next subsequence and the accumulated score will add the value $\mathbf{M}(t,\mathbf{S}_{i+t})$ of the matching alphabet at the position t only if the current accumulated score $\mathbf{R}_t$ is not less than $\mathbf{T}_t$. Once the above-mentioned conditions are not satisfied, the process of subsequence searching can be stopped at the current i and resumed next match at position $i+1$.

In the first section, we introduced that the task of candidate TFBS searching can be abstracted as a classical k-Nearest Neighbor (KNN) problem and the KNN problem can be viewed as a special case of maximum all-pairs dot-product (MAD) searching problem which can be solved by the diamond sampling algorithm. However, the diamond sampling algorithm provides an approximate result with the maximum probability rather than the precise result that we need in the task of candidate TFBS searching. Although the diamond sampling cannot be used directly to solve searching problem in the task of candidate TFBS searching, we can speed up the process of candidate TFBS searching with the weight sampling part of the diamond sampling algorithm.

Actually, the matching failed expectations at each position in the model is different from the task of candidate TFBS searching and those matching failed expectations can be viewed as some special weights to decide the order of the match. To speed up the task of candidate TFBS searching, we expect that the matching failed subsequence can be detected as early as possible by utilizing these special weights while the lookahead scoring algorithm ignores the information. Although the matching failed probability at each position can help us to speed up searching process indeed, it takes a very high cost to get the precise matching failed expectations. In this paper, we propose a further improved method based on the lookahead scoring algorithm,

called the faster lookahead scoring algorithm (FLS), which utilize an approximate expectation to replace the precise matching failed expectations and the position with higher matching failed expectation should be matched firstly. The matching failed expectation at each position is influenced by many factors including the information content of each position in the PWM, the background distribution for the given sequences and so on. The approximate matching failed expectation at position i can be defined as

$$\mathbf{E}_i = |\min_{\alpha\in\Re} \mathbf{M}(i,a)| + \max_{\alpha\in\Re} \mathbf{M}(i,a) - \sum_{\alpha\in\Re} \mathbf{q}_a \mathbf{M}(i,a) \tag{8}$$

Where $\mathbf{q}$ is the background distribution and $\mathbf{q}_a$ is the background probability of the alphabet a. The matching order can be defined as the decreasing order of the approximate matching failed expectation $\mathbf{E}_i$ for $1 \leq i \leq n$ in the matrix.

Due to the effect of permuted order for matching, the gap of maximum possible score and the actual score at position i will be widened by $\mathbf{E}_i$. So the partial score of matching subsequence, when evaluated in permuted order, will drop below the intermediate threshold $\mathbf{T}_i$ as much as possible on average. As a result, the failed matches in all subsequences will be detected as early as possible and the task of candidate TFBS searching will also take the minimal average time. Before the task of candidate TFBS searching, the intermediate score thresholds should be recalculated by the permuted order rather than by the original left-to-right order, and the permuted order is determined by decreasing all approximate matching failed expectations.

Implementation

Computing significance threshold

In our method, the significance threshold is not given directly but transformed from a specified probability threshold, called p-value, using quantile function. Its advantage is that the user can give a p-value to trade off the speed and sensitivity of the task TFBS candidate searching explicitly. Thus, the researcher can utilize the specified threshold to adjust the level of statistical significance for a particular analysis. A low p-value will reduce hits that are statistically very significant and lead to a high specific analysis. The advantage of this analysis is that the result can be relatively low false positives, but its disadvantage is that the result may miss much more distant sequence relationships. On the other hand, a high p-value is specified by user can analyze the particular sequence with high sensitivity. The calculation of significance threshold is difficult and a specific analysis usually requires a trade-off between relatively high speed and high sensitivity. Therefore, we utilize p-value to give a significance threshold indirectly in our method.

In order to calculate the quantile function more easily, some assumptions are declared as: the alphabet at each position in a random k-mer is independent and the background distribution is identical. The quantile function is not calculated directly, but by its inverse function, called the complementary cumulative distribution function (complementary CDF). On the other hand, the complementary CDF can be calculated by performing a summation over the probability mass function (PMF).[10]

We defines the segmental score as the random variable $\mathbf{X}$, and the PMF of $\mathbf{X}$ by $f(\mathbf{x})$. Then, if we give a particular segmental score γ, the p value can be defined as

$$\mathrm{G}(\gamma) = \Pr\{\mathbf{X} \geq \gamma\} = \sum_{x=\gamma}^{\infty} f(x)dx \tag{9}$$

Obviously, while the subsequence is assumed to be random in the task of TFBS candidate searching, the value of G(γ) is the probability of a score of k-mer, which the matching score is not less than γ in the given sequence.

For calculating the PMF, several methods have been proposed. In this paper, the adopted method is to compute the probability recursively with each column of the scoring matrix.[19, 21, 22] We defined a background frequency vector as

$$\mathbf{q} = \langle q(1), \ldots q(|\Re|)\rangle \tag{10}$$

Where $\Re$ is a set of possible alphabets. In our method, each position in a random k-mer is represented by a background frequency vector **q**. The background frequencies can be obtained by scanning the given sequence and counting number of various alphabets. Then, the PMF can be computed recursively, column by column:

$$\begin{aligned} f^{(0)}(x) &= \delta(x) \\ f^{(i)}(x) &= \sum_{\alpha\in\Re} q(\alpha) f^{(i-1)}(x - \mathbf{M}(i,\alpha)) \\ & \qquad\qquad i = 1, \ldots, m \\ f(x) &= f^{(m)}(x) \end{aligned} \tag{11}$$

Where the function $\delta(x)$ is initialized to 0 for $x \neq 0$ and 1.0 otherwise. However, the entire probability mass is usually initialized by score $x = 0$. Then, the revised version of the PMF, $f^{(i)}(x)$, is based on the previous PMF at each column i. In the process of computing PMF, simply (x–**M**(i,a)) is the score for all possible alphabet a. The desired PMF $f(x)$ is generated at the final iteration.

Once we have the PMF $f(x)$, the complementary CDF G(γ) can be computed by performing a summation over the PMF. Thus, the inverse function, $G^{-1}(\gamma)$, is expected as the quantile function. The quantile function generates a significance threshold $T^* = \lceil G^{-1}(\gamma^*)\rceil$ after given a γ threshold γ^*. The computed G(γ) contain all possible k-mers score, such that the significance threshold can be calculated readily for any given thresholds.

The overview of searching strategy

The workflow of the proposed algorithm is presented in Fig 1. Since the length of a given sequence usually stand at hundreds of millions, as a saved-time technique, the corresponding background distribution can be calculated at the same time when the sequence is streaming to memory. In addition, the process of log-odds of matrix will produce error message when the matrix of model contain zeros. Thus, the probabilistic matrix needs to add a pseudocounts firstly at the position whose corresponding probability is zero before log-odds of matrix.

The process of matching between the given sequence and the model can be proceed after the significance threshold, matching order and the array of intermediate thresholds are all computed. In addition, the order of match is not left-to-right but following the matching order in (8).

Results

In order to demonstrate the performance of our FLS (faster lookahead scoring algorithm), we implemented LS (lookahead scoring algorithm) and the well-known base-line algorithms NA (naive algorithm) as our contrast experiments from the original paper. The code of all experiments is written in C/C++ and the all tasks of candidate TFBS searching run for single-threaded. The running environment includes 3.2GHz Intel® Core™ i7-2600 processor with 3

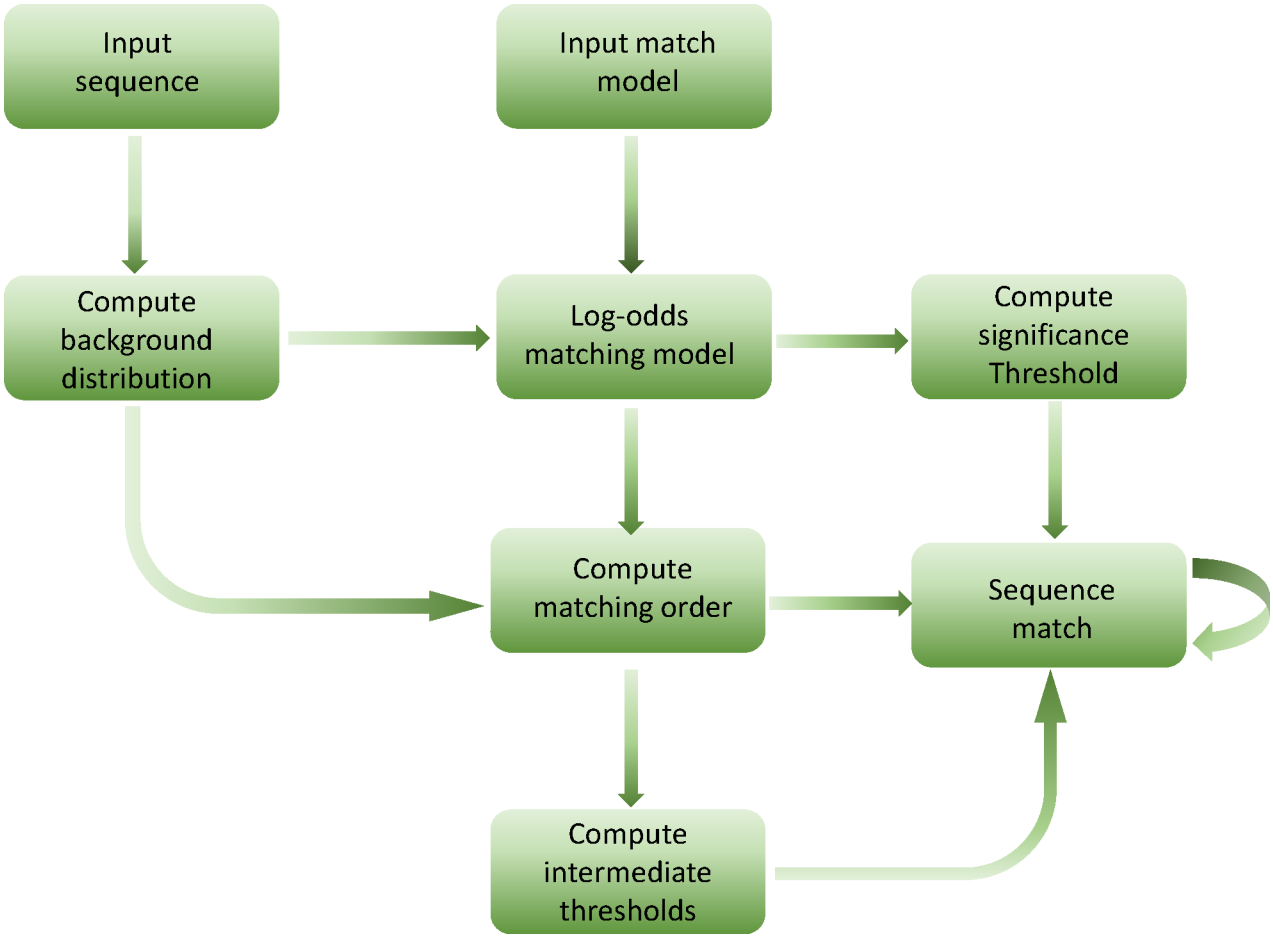

Fig 1. Overview of faster lookahead scoring algorithm with main steps.

https://doi.org/10.1371/journal.pone.0198922.g001

gigabytes of main memory, running under Ubuntu 16.04, and gcc is the only compiler used in the experiments. We also repeat same experiments with 2.5GHz Intel® Xeon® CPU e5-2650 v2, and get essentially similar results which slight differences explained by different cache memory size.

Datasets

With the development of gene sequencing, the size of the sequence database is increased explosively.[13, 22–24] In order to analyze these large number of sequence data, the efficient mathematical methods and computer algorithms used in the task of candidate TFBS searching need simple, logical and self-consistent. Information theory[25–27] that was created by Shannon is such mathematic tool that meets those requirements, and related theory comes directly from the physics underlying molecular binding interactions. In many researches, information theory play an import role in quantifying the sequence conservation in protein sequences and nucleotide.[28–35] The information content (IC) which is an import concept in the information theory will be used in the task of candidate TFBS searching to provide to assist in quantitative analysis[36]. Since the total information content of a model represents the distinguishable capability of the given model between a binding site (represented by the matrix) and the background model, therefore the information content of the given model in gene database should

be fixed to a series of specified information content or fit some relationship in the experiment of quantitative analysis[37]. However, searching those models that meet those requirements is almost impossible in larger gene databases, so we generate a series of model meeting those requirements based on the theory of information content. We also make some contrast experiments on real-world datasets to show the universality of FLS algorithm. Related real-world and artificial datasets can be described as

- Datasets of Artificial Sequence SEQ1: the most sequences in the gene database usually are non-uniform background distribution. For contrast with sequences in gene databases, we generate some artificial sequences with approximate uniform background distribution to contrast the performance of three algorithms. The generated sequence is consisted of 55 megabases amino acid that every amino acid is randomly created and the corresponding background distribution is guaranteed to approximate to a uniform distribution.
- Datasets of Artificial Model MOD1: based on the theory of information content, Staden proposed an efficient method which can numerically estimate the probability-generating function for model with the given information content. When we get the probability-generating functions for each column of an alignment matrix, the probability-generating function for a multi-column alignment matrix[33] can be approximate replaced by the Staden's approach. In order to contrast the influence of the different information content, we generate 26 matrixes which the length of all matrixes is fixed as 22 and information content of all matrix are increased gradually from 5 to 30. The single performance test is influenced seriously by the random error, so we generate 100 sets of the models using the same way and the averaged run time is viewed as the real performance of corresponding the algorithm.
- Datasets of Artificial Model MOD2: to contrast the influence of the different length of models on the performance of the three algorithms and eliminate the influence of the different information content of each model, the information content value of each model is set as 70% of the maximal information content of the corresponding model, and the lengths of all model are increased gradually from 5 to 30. We also generate 100 sets of the models using the same way and the averaged run time is approximately equal to the real performance of the corresponding algorithm.
- Datasets of Real-world Sequence SEQ2: to contrast with the artificial sequence, we concatenate to a 55 megabases long DNA sequences that all subsequences of DNA sequence are collected from the mouse and human genome.
- Datasets of Real-world Model MOD3: we collect 368 models about DNA from the known JASPAR database (JASPAR CORE REDUNANT 2016).[38, 39] the lengths of all models are also increased gradually from 5 to 21 and their average length is 13.2. In the real dataset, the number of models which their length exceed this range is very rare, so we left out specific modes. These models are divided into 25 groups which the model lengths in each group are increased from 5 to 21. In some group, due to the amount of some models having the same length are less than 25, so some models need to be repeated and divided into multiple groups at the same time.

Time and accuracy performance

The average running time on different artificial models or real-world model can be summarized in Figs 2–4. In those experiments, the p-values of all significant thresholds are set to the same value 0.0001.

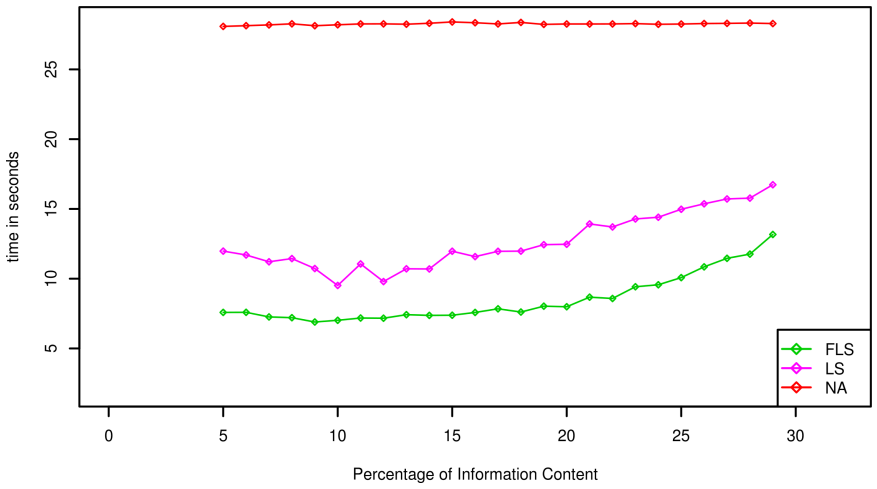

Fig 2. Average running times (in Seconds, Preprocessing excluded) of different algorithms for model MOD1 with p-value $\gamma = 0.0001$.

https://doi.org/10.1371/journal.pone.0198922.g002

The significant threshold k of each experiments is not given but is calculated indirectly through a given p-value. To evaluate performance of algorithms on varying significant thresholds, we have some experiments which the real-world models collect from JASPAR and the sequences are real-world SEQ2 with varying p-value. The average running times are summarized in Table 2. It's obvious that the faster lookahead scoring algorithm outperforms others and has better performance with the smaller γ, on the other hand, the average running time of the naive algorithm almost be keep at 33

The various information content of models will produce different influence on the searching speed, so that we evaluate three algorithms on the artificial datasets contained the models MOD1 and the sequence SEQ1. The p-value is set to 0.0001 and the average running times are depicted as Fig 2. Obviously, the faster lookahead scoring algorithm is always the fastest searching one among three algorithms when the information content is same. Otherwise, the

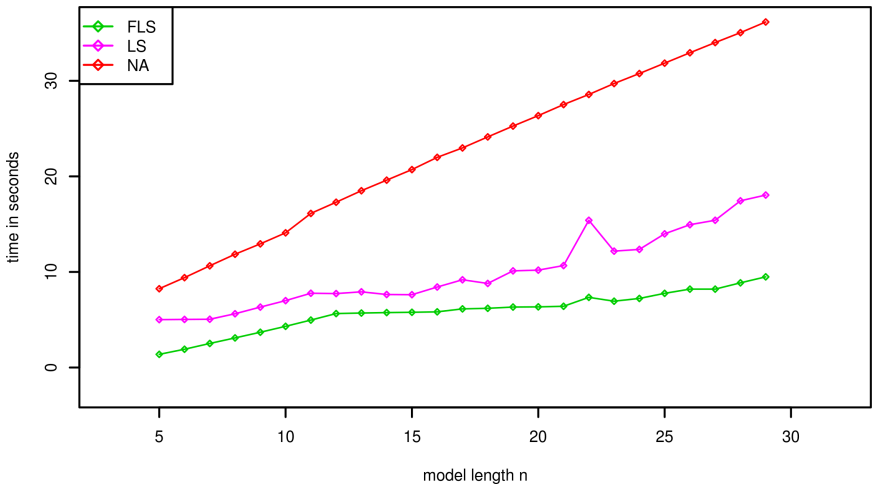

Fig 3. Average running times (in Seconds, Preprocessing excluded) of different algorithms for model MOD2 with p-value $\gamma = 0.0001$.

https://doi.org/10.1371/journal.pone.0198922.g003

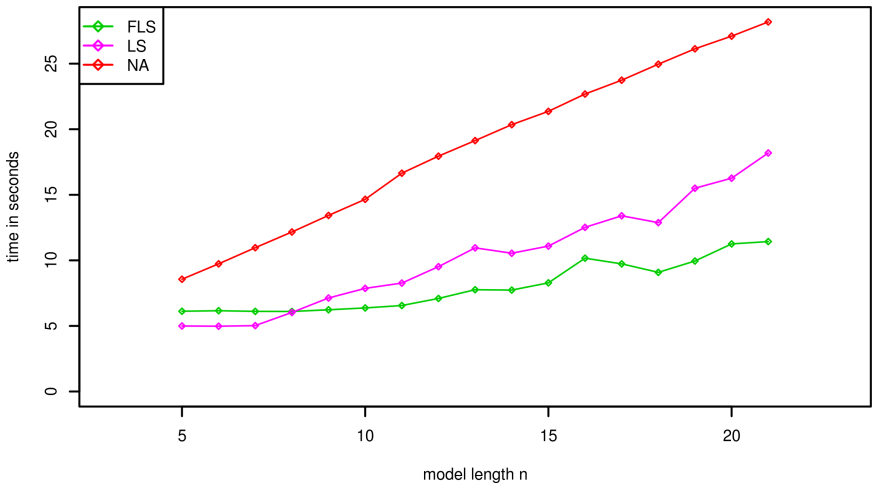

Fig 4. Average running times (in Seconds, Preprocessing excluded) of different algorithms for real-world model MOD3 and sequence SEQ2 with p-value $\gamma = 0.0001$.

https://doi.org/10.1371/journal.pone.0198922.g004

average running time also slow-growth when the information content of models increase gradually. There is a reasonable explanation for this phenomenon that the amount of the matching subsequence's prefixes which their matching score is above the corresponding intermediate threshold will increase as containing more information content, such that the corresponding running time will also deteriorate.

The various length of models will produce different influence on the searching speed, so that we evaluate three algorithms on the artificial datasets contained models MOD2 and sequence SEQ1. The p-value is set to 0.0001 and the average running times are depicted as Fig 3. The faster lookahead scoring algorithm is still the fastest searching one among three contrastive algorithms. Although the average running times of three algorithms increase at the same time with the longer length of models, the running time of the faster lookahead scoring algorithm increases more slowly compared with the other algorithms.

To evaluate the performance of three algorithm on the real-world databases, we experimented on the dataset contained the models MOD3 and the sequence SEQ2. The p-value is also set to 0.0001 and the average running times are depicted as Fig 4. The faster lookahead scoring algorithm is still the fastest one with the other ones. Note that the average running time of FLS is more than LS's one when the length of model is less than 8. The possible reason of this phenomenon can be described as that the advantage of permuted order of match will be disadvantage by adding additional operations compared with the LS algorithm when the length of model is too short.

Table 2. Average running times of various p-value.

$\gamma =$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
NA	34.83	32.53	32.48	32.99	33.85	34.02
LS	30.48	24.06	21.22	19.30	18.08	17.19
FLS	**22.39**	**16.43**	**13.26**	**11.67**	**9.92**	**9.32**

Average running times (in Seconds, Preprocessing excluded) of different algorithms for DNA pattern ($m = 21$) from JASPAR and varying p-Values, and each reported time is an average of 10 runs.

https://doi.org/10.1371/journal.pone.0198922.t002

Conclusions

It should be emphasized that the results of all algorithms in above experiments are same so that we ignore contrast of results and all algorithms in our experiments can find out the precise result. Moreover, the faster lookahead scoring algorithm has a clear speed-up advantage compared with the lookahead searching algorithm. Through above contrasting experiments, the searching performance of FLS is better for dealing with real-world datasets and artificial datasets. As the exponential growth of both DNA and protein sequence databases, the searching speed of the faster lookahead scoring algorithm will be more significance and more concerned.

Acknowledgments

This work was supported by the grants of the National Science Foundation of China, Nos. 61732012, 61520106006, 31571364, 61672382, 61772370, U1611265, 61532008, 61472280, 61472173, 61702371, and 61772357, China Postdoctoral Science Foundation Grant, Nos. 2016M601646 and 2017M611619, and supported by "BAGUI Scholar" Program of Guangxi Province of China.

Author Contributions

Conceptualization: Hongbo Zhang, De-Shuang Huang.

Methodology: Liangxin Gao.

Software: Liangxin Gao.

Writing – original draft: Liangxin Gao.

Writing – review & editing: Wenzhen Bao, Hongbo Zhang, Chang-An Yuan.

References

1. Matys V, Fricke E, Geffers R, Gößling E, Haubrock M, Hehl R, et al. TRANSFAC®: transcriptional regulation, from patterns to profiles. Nucleic acids research. 2003; 31(1):374–8. PMID: 12520026
2. Yu CP, Lin JJ, Li WH. Positional distribution of transcription factor binding sites in Arabidopsis thaliana. Scientific reports. 2016; 6:25164. https://doi.org/10.1038/srep25164 PMID: 27117388; PubMed Central PMCID: PMC4846880.
3. Dorohonceanu B, Nevill-Manning CG, editors. Accelerating protein classification using suffix trees. ISMB; 2000.
4. Schones DE, Smith AD, Zhang MQ. Statistical significance of cis-regulatory modules. BMC bioinformatics. 2007; 8(1):19.
5. Beckstette M, Strothmann D, Homann R, Giegerich R, Kurtz S, editors. PoSSuMsearch: Fast and Sensitive Matching of Position Specific Scoring Matrices using Enhanced Suffix Arrays. German Conference on Bioinformatics; 2004.
6. Beckstette M, Homann R, Giegerich R, Kurtz S. Fast index based algorithms and software for matching position specific scoring matrices. BMC bioinformatics. 2006; 7(1):389.
7. Liefooghe A, Touzet H, Varré J-S, editors. Large scale matching for position weight matrices. Annual Symposium on Combinatorial Pattern Matching; 2006: Springer.
8. Rajasekaran S, Jin X, Spouge JL. The efficient computation of position-specific match scores with the fast Fourier transform. Journal of Computational Biology. 2002; 9(1):23–33. https://doi.org/10.1089/10665270252833172 PMID: 11911793
9. Freschi V, Bogliolo A. Using sequence compression to speedup probabilistic profile matching. Bioinformatics. 2005; 21(10):2225–9. https://doi.org/10.1093/bioinformatics/bti323 PMID: 15713733
10. Wu TD, Nevill-Manning CG, Brutlag DL. Fast probabilistic analysis of sequence function using scoring matrices. Bioinformatics. 2000; 16(3):233–44. PMID: 10869016
11. Huang DS, Du JX. A Constructive Hybrid Structure Optimization Methodology for Radial Basis Probabilistic Neural Networks. IEEE Transactions on Neural Networks. 2008; 19(12):2099–115. https://doi.org/10.1109/TNN.2008.2004370 PMID: 19054734

12. Huang DS, Zheng CH. Independent component analysis-based penalized discriminant method for tumor classification using gene expression data. Bioinformatics. 2006; 22(15):1855–62. https://doi.org/10.1093/bioinformatics/btl190 PMID: 16709589
13. Yu H-J, Huang D-S. Normalized feature vectors: a novel alignment-free sequence comparison method based on the numbers of adjacent amino acids. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB). 2013; 10(2):457–67.
14. Deng SP, Huang DS, editors. SFAPS: an R package for structure/function analysis of protein sequences based on informational spectrum method. IEEE International Conference on Bioinformatics and Biomedicine; 2014.
15. Ballard G, Kolda TG, Pinar A, Seshadhri C, editors. Diamond sampling for approximate maximum all-pairs dot-product (MAD) search. Data Mining (ICDM), 2015 IEEE International Conference on; 2015: IEEE.
16. Deng SP, Zhu L, Huang DS. Predicting hub genes associated with cervical cancer through gene co-expression networks: IEEE Computer Society Press; 2016. 27–35 p.
17. Zhu L, Deng S-P, Huang D-S. A Two-Stage Geometric Method for Pruning Unreliable Links in Protein-Protein Networks. NanoBioscience, IEEE Transactions on. 2015; 14(5):528–34.
18. Deng SP, Zhu L, Huang DS. Mining the bladder cancer-associated genes by an integrated strategy for the construction and analysis of differential co-expression networks. Bmc Genomics. 2015; 16(S3):S4.
19. Staden R. Methods for calculating the probabilities of finding patterns in sequences. Bioinformatics. 1989; 5(2):89–96.
20. HUANG D-S. Radial basis probabilistic neural networks: model and application. International Journal of Pattern Recognition & Artificial Intelligence. 1999; 13(07):1083–101.
21. McLachlan A. Analysis of gene duplication repeats in the myosin rod. Journal of molecular biology. 1983; 169(1):15–30. PMID: 6620380
22. Zhu L, Guo W-L, Deng S-P, Huang D-S. ChIP-PIT: enhancing the analysis of ChIP-Seq data using convex-relaxed pair-wise interaction tensor decomposition. IEEE/ACM transactions on computational biology and bioinformatics. 2016; 13(1):55–63. https://doi.org/10.1109/TCBB.2015.2465893 PMID: 26886732
23. Zhu L, You Z-H, Huang D-S, Wang B. t-LSE: a novel robust geometric approach for modeling protein-protein interaction networks. PloS one. 2013; 8(4):e58368. https://doi.org/10.1371/journal.pone.0058368 PMID: 23560036
24. Huang DS, Jiang W. A general CPL-AdS methodology for fixing dynamic parameters in dual environments. IEEE Transactions on Systems Man & Cybernetics Part B. 2012; 42(5):1489–500.
25. Shannon CE. A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review. 2001; 5(1):3–55.
26. Pierce JR. An introduction to information theory: symbols, signals and noise: Courier Corporation; 2012.
27. Spellerberg IF, Fedor PJ. A tribute to Claude Shannon (1916–2001) and a plea for more rigorous use of species richness, species diversity and the 'Shannon–Wiener'Index. Global ecology and biogeography. 2003; 12(3):177–9.
28. Schneider TD, Stormo GD, Gold L, Ehrenfeucht A. Information content of binding sites on nucleotide sequences. Journal of molecular biology. 1986; 188(3):415–31. PMID: 3525846
29. Papp PP, Chattoraj DK, Schneider TD. Information analysis of sequences that bind the replication initiator RepA. Journal of molecular biology. 1993; 233(2):219–30. https://doi.org/10.1006/jmbi.1993.1501 PMID: 8377199
30. Crooks GE, Hon G, Chandonia J-M, Brenner SE. WebLogo: a sequence logo generator. Genome research. 2004; 14(6):1188–90. https://doi.org/10.1101/gr.849004 PMID: 15173120
31. Pietrokovski S. Searching databases of conserved sequence regions by aligning protein multiple-alignments. Nucleic acids research. 1996; 24(19):3836–45. PMID: 8871566
32. Blom N, Hansen J, Brunak S, Blaas D. Cleavage site analysis in picornaviral polyproteins: discovering cellular targets by neural networks. Protein Science. 1996; 5(11):2203–16. https://doi.org/10.1002/pro.5560051107 PMID: 8931139
33. Hertz GZ, Stormo GD. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics (Oxford, England). 1999; 15(7):563–77.
34. Wang B, Huang D-S, Jiang C. A new strategy for protein interface identification using manifold learning method. IEEE transactions on nanobioscience. 2014; 13(2):118–23. https://doi.org/10.1109/TNB.2014.2316997 PMID: 24771594

35. Huang D-S, Zhang L, Han K, Deng S, Yang K, Zhang H. Prediction of protein-protein interactions based on protein-protein correlation using least squares regression. Current Protein and Peptide Science. 2014; 15(6):553–60. PMID: 25059329

36. Huang DS, Zhang L, Han K, Deng S, Yang K, Zhang H. Prediction of protein-protein interactions based on protein-protein correlation using least squares regression. Curr Protein Pept Sci. 2014; 15(6):553–60. PMID: 25059329

37. Huang D-S. Systematic theory of neural networks for pattern recognition. Publishing House of Electronic Industry of China, Beijing. 1996; 201.

38. Henikoff JG, Greene EA, Pietrokovski S, Henikoff S. Increased coverage of protein families with the blocks database servers. Nucleic acids research. 2000; 28(1):228–30. PMID: 10592233

39. Hallikas O, Palin K, Sinjushina N, Rautiainen R, Partanen J, Ukkonen E, et al. Genome-wide prediction of mammalian enhancers based on analysis of transcription-factor binding affinity. Cell. 2006; 124 (1):47–59. https://doi.org/10.1016/j.cell.2005.10.042 PMID: 16413481