Approximate String Searching with Fast Fourier Transforms and Simplexes

Daniel Liu[1]

[1]**Torrey Pines High School, San Diego, CA**

Corresponding author:
Daniel Liu[1]

Email address: daniel.liu02@gmail.com

ABSTRACT

Previous algorithms for solving the approximate string matching with Hamming distance problem with wildcard ("don't care") characters have been shown to take $O(|\Sigma|N\log M)$ time, where N is the length of the text, M is the length of the pattern, and $|\Sigma|$ is the size of the alphabet. They make use of the Fast Fourier Transform for efficiently calculating convolutions. We describe a novel approach of the problem, which makes use of special encoding schemes that depend on $(|\Sigma|-1)$-simplexes in $(|\Sigma|-1)$-dimensional space.

INTRODUCTION

Approximate string searching has been subject to rigorous research due their applications in bioinformatics and text retrieval. One such problem involves finding the hamming distance (*i.e.*, number of mismatches) at each location i in a text $\mathscr{T}$ of length N with a pattern $\mathscr{P}$ of length M, where $M \leq N$. Additionally, we have an alphabet Σ where $\forall i \in \{1 \ldots N\},\ \mathscr{T}_i \in \Sigma$ and $\forall j \in \{1 \ldots M\},\ \mathscr{P}_j \in \Sigma$. Also, we wish to allow wildcard (or "don't care") characters denoted by $\star$ to match any character in the alphabet Σ. More formally, we wish to find the following sum:

$$\mathscr{C}_i = \sum_{j=1}^{M} \delta(\mathscr{T}_{i+j}, \mathscr{P}_j), \quad \forall i \in \{1 \ldots N-M+1\} \tag{1}$$

where the δ is defined as

$$\delta(x,y) = \begin{cases} 1, & \text{if } x = y \text{ or } x = \star \text{ or } y = \star \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Fischer and Paterson (1974) first noticed that Fast Fourier Transforms can be used to solve the exact matching problem in $O(N\log M\log|\Sigma|)$. Their ideas can be extended to handle wildcards ("don't cares") and to calculate Hamming distances at each location in the text. Clifford and Clifford (2007) proposed a simple solution that lowered the time to $O(N\log M)$ for exact matching with wildcards. Clifford et al. (2010) discussed algorithms for handling matches with bounded number of mismatches and wildcards.

In terms of calculating Hamming distance at each location in the text, Linhart and Shamir (2009) proposed the use of *prime encodings* for a slight speedup over the straightforward extension of Fischer and Paterson (1974)'s ideas. Schoenmeyr and Zhang (2005) also achieved a 2 times speedup by mapping alphabet characters to roots of unity.

In this paper, we present two algorithms for approximate string searching with Hamming distance and wildcard characters. Though they do not represent significant speedups compared to previous results, they contain ideas that are of theoretical interest.

FAST FOURIER TRANSFORM

The Fast Fourier Transform $\mathscr{F}$ is an efficient method for transforming a discrete input signal (or any arbitrary sequence) into *frequency space* (the opposite is also possible, with the inverse transform $\mathscr{F}^{-1}$).

This allows the *convolution* of two arbitrary sequences a and b of complex values (*i.e.*, $\forall i \in \{1 \dots |a|\},\ a_i \in \mathbb{C}$ and $\forall j \in \{1 \dots |b|\},\ b_j \in \mathbb{C}$) to be efficiently calculated through the convolution theorem:

$$a * b = \mathscr{F}^{-1}\big(\mathscr{F}(a) \odot \mathscr{F}(b)\big) \tag{3}$$

where the $*$ operator indicates convolution and the $\odot$ operator indicates element-wise multiplication.

The time complexity of the FFT operation takes $O(|a| \log |a|)$ time for a sequence a of length $|a|$, but this can be reduced to $O(|a| \log |b|)$ by splitting a into smaller segments.

In this paper, we focus more on the string searching aspect of the problem, and less on further speedups for the FFT algorithm itself, as there already is a large body of work on improving the FFT algorithm.

BASIC ENCODING

First, we present an overview of basic extensions to Fischer and Paterson (1974)'s ideas to solve the matching with Hamming distance problem in $O(|\Sigma| N \log M)$ time. This method relies on the the so-called "one-hot encoding" of each letter $\in \Sigma$:

$$\begin{aligned}
e_1 &= \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \\
e_2 &= \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \end{pmatrix} \\
e_3 &= \begin{pmatrix} 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \\
&\vdots \\
e_{|\Sigma|} &= \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{|\Sigma| \text{ values}}
\end{aligned} \tag{4}$$

The alphabet encodings are used to create $\mathscr{T}'$ and $\mathscr{P}'$, which are of length $|\Sigma|N$ and $|\Sigma|M$, respectively. Both $\mathscr{T}'$ and $\mathscr{P}'$ are concatenations of the encodings: $\mathscr{T}' = e_{\mathscr{T}_1} e_{\mathscr{T}_2} e_{\mathscr{T}_3} \dots e_{\mathscr{T}_N}$ and $\mathscr{P}' = e_{\mathscr{P}_1} e_{\mathscr{P}_2} e_{\mathscr{P}_3} \dots e_{\mathscr{P}_M}$ (for simplicity, we consider the characters in the alphabet as numbers). To search for $\mathscr{P}$ in $\mathscr{T}$, we calculate the following sum (convolution):

$$\begin{aligned}
\mathscr{C}'_i &= \sum_{j=1}^{|\Sigma|M} \mathscr{T}'_{i+j} \mathscr{P}'_j, \quad \forall i \in \{1 \dots |\Sigma|N - |\Sigma|M + 1\} \\
&= \mathscr{T}' * \mathscr{P}'
\end{aligned} \tag{5}$$

We will refer to the sum $\mathscr{C}'_i$ as the score at each index i.

It is easy to see that for any two encodings that match, then the they contribute 1 to the score, and for any two encodings that do not match, they contribute 0 to the score. Therefore, the resulting Hamming distances for each match in location $\mathscr{T}$, $\mathscr{C}$, is

$$\mathscr{C}_i = |\mathscr{P}| - \mathscr{C}'_{|\Sigma|(i-1)+1}, \quad \forall i \in \{1 \dots N\} \tag{6}$$

To handle wildcard characters, we encode each wildcard character in $\mathscr{T}$ as

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \end{pmatrix}}_{|\Sigma| \text{ values}} \tag{7}$$

and each wildcard character in $\mathscr{P}$ as

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}}_{|\Sigma| \text{ values}} \tag{8}$$

The sum we calculate also needs to be changed to handle the wildcard characters:

$$\begin{aligned}
\mathscr{C}'_i &= \sum_{j=1}^{|\Sigma|M} \neg \mathscr{T}'_{i+j} \mathscr{P}'_j, \quad \forall i \in \{1 \dots |\Sigma|N - |\Sigma|M + 1\} \\
&= \sum_{j=1}^{|\Sigma|M} (1 - \mathscr{T}'_{i+j}) \mathscr{P}'_j \\
\mathscr{C}_i &= \mathscr{C}'_{|\Sigma|(i-1)+1}, \quad \forall i \in \{1 \dots N\}
\end{aligned} \tag{9}$$

Overall, wildcard characters are always encoded as a vector of zeros, with or without the negation. This allows wildcard characters to never contribute in increasing the the values in $\mathscr{C}$. Also, the negation of $\mathscr{T}$ allows matching encodings to contribute zero to the score, while disagreeing encodings increase the score by exactly one. Therefore, $\mathscr{C}$ represents the desired Hamming distances at each index.

Note that character sets in $\mathscr{T}$ can also be handled. A character set matches only a character out of a subset of Σ at a certain location in $\mathscr{T}$. For each character set C (where $C \subseteq \Sigma$), the encoding is the same as the wildcard characters in $\mathscr{T}$. The only difference is that the bit in the encoding that represents each character $c \in \Sigma, c \notin C$ is flipped from the original wildcard character encoding in $\mathscr{T}$ (*i.e.*, from 1 to 0). To handle character sets in the pattern, encodings for wildcard characters must all be flipped and $\mathscr{P}$ must be negated instead of $\mathscr{T}$ in the sum.

The overall time complexity of the aforementioned approaches are the same when no character sets are used:

$$\begin{aligned} &O\big(|\Sigma|N + |\Sigma|M + |\Sigma|N\log(|\Sigma|M)\big) \\ &= O\big(|\Sigma|N + |\Sigma|M + |\Sigma|N(\log|\Sigma| + \log M)\big) \\ &= O(|\Sigma|N\log M) \end{aligned} \tag{10}$$

since $M \leq N$, and if $|\Sigma| > M$, the extra characters in the alphabet that are not in $\mathscr{P}$ can be encoded into one "other" character that does not match the encoding of any existing character in $\mathscr{P}$, which ensures that $|\Sigma| = O(M)$. When character sets are used, the time complexity deteriorates very slightly.

SIMPLEX-BASED METHODS

One way to reduce the run time complexity of the basic encoding technique is to attempt to decrease the length of the encodings. First, each set of $|\Sigma|$ possible encodings for the characters in $|\Sigma|$ can be seen as vectors in $\mathbb{R}^k$ space, where k represents the number of dimensions, which is also the length the encoding. In the basic encoding technique, $k = |\Sigma|$. The goal is to minimize k while ensuring that the distance between character encoding matches and mismatches is distinguishable (and obviously, encoding matches and mismatches must be indicative of actual character matches and mismatches). The distance between two encoding vectors will be defined as the L_p norm between those two vectors (*i.e.*, $||v-u||_p$), for some chosen p.

Such a set of vectors can be represented using vertices on a *simplex*. In this case, we informally define a (regular) k-simplex Δ^k as a set of $k+1$ points in $\mathbb{R}^k$, where the distance (L_p norm) between each pair of points is the same, and the points lie on a k-dimensional unit ball defined by the L_p norm. Returning to the original problem, the set of $|\Sigma|$ points that define a $(|\Sigma|-1)$-simplex can be used as the encoding vectors. They satisfy the requirement where the distance for matches and mismatches must be distinguishable, as encodings that do not match will have the same L_p norm. Unfortunately, better compression using this technique cannot be achieved since it is known that no more than $k+1$ pairwise equidistant points can be present in $\mathbb{R}^k$ space.

Let each pair (k,p) represent the set of $k+1$ encodings that uses the L_p norm. We examine how $(|\Sigma|-1,0)$ (Hamming) encodings and $(|\Sigma|-1,2)$ (Euclidean) encodings can be applied to the string Hamming distance problem.

Hamming Encodings

A $(|\Sigma|-1,0)$ encoding set can be seen as a set of $|\Sigma|$ binary vectors that have the same pairwise L_0 distance. *I.e.*, $||v-u||_0 = d,\ \forall u,v \in (|\Sigma|-1,0)$ encoding set, where $u \neq v$ and d is a fixed constant distance. Note that the L_0 norm essentially counts the number of nonzero elements in a vector. Since the vectors are binary vectors, the distance operation can be seen as a bit count of the *XOR* of the bit vectors u and v, which can be represented using AND and OR operations: $u \oplus v = (\neg u \wedge v) \vee (u \wedge \neg v)$. Representing this as summation for over each encoding in $\mathscr{T}'$ and $\mathscr{P}'$, we get

$$\begin{aligned} \mathscr{C}'_i &= \sum_{j=1}^{(|\Sigma|-1)M} \neg\mathscr{T}'_{i+j}\mathscr{P}'_j + \mathscr{T}'_{i+j}\neg\mathscr{P}'_j, \quad \forall i \in \{1\ldots(|\Sigma|-1)N-(|\Sigma|-1)M+1\} \\ &= \sum_{j=1}^{(|\Sigma|-1)M} (1-\mathscr{T}'_{i+j})\mathscr{P}'_j + \mathscr{T}'_{i+j}(1-\mathscr{P}'_j), \quad \forall i \in \{1\ldots(|\Sigma|-1)N-(|\Sigma|-1)M+1\} \end{aligned} \tag{11}$$

which can be computed using FFTs in $O(|\Sigma|N\log M)$ time. Since each encoding mismatch contributes d to the score and each match contributes 0, the Hamming distance at each location in $\mathscr{T}$ is

$$\mathscr{C}_i = \frac{1}{d}\mathscr{C}'_{(|\Sigma|-1)(i-1)+1}, \quad \forall i \in \{1 \dots N\} \tag{12}$$

To handle wildcard characters, we create two new vectors $\mathscr{T}''$ and $\mathscr{P}''$ of length $(|\Sigma|-1)N$ and $(|\Sigma|-1)M$, respectively. They can be formed by mapping characters to vectors and concatenating those vectors. $\forall c \in \Sigma$, we map c to

$$\underbrace{\begin{pmatrix}1 & 1 & 1 & 1 & \cdots & 1\end{pmatrix}}_{|\Sigma|-1 \text{ values}} \tag{13}$$

and wildcards ($\star$) are mapped to

$$\underbrace{\begin{pmatrix}0 & 0 & 0 & 0 & \cdots & 0\end{pmatrix}}_{|\Sigma|-1 \text{ values}} \tag{14}$$

This mapping allows each position that match the wildcard characters to always result in $\Sigma - 1$ zeros in the summation due to the special $\mathscr{T}''$ and $\mathscr{P}''$ encodings:

$$\mathscr{C}'_i = \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j\left[(1-\mathscr{T}'_{i+j})\mathscr{P}'_j + \mathscr{T}'_{i+j}(1-\mathscr{P}'_j)\right], \quad \forall i \in \{1 \dots (|\Sigma|-1)N-(|\Sigma|-1)M+1\} \tag{15}$$

To implement this using FFTs, we must expand out the products:

$$\begin{aligned}
\mathscr{C}'_i &= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j(1-\mathscr{T}'_{i+j})\mathscr{P}'_j + \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j}(1-\mathscr{P}'_j) \\
&= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{P}'_j - \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j}\mathscr{P}'_j + \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j} - \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j}\mathscr{P}'_j \\
&= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{P}'_j + \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j} - 2\mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j}\mathscr{P}'_j
\end{aligned} \tag{16}$$

Since the indexes match for $\mathscr{T}'_{i+j}\mathscr{T}''_{i+j}$ and $\mathscr{P}'_i\mathscr{P}''_i$ multiplications, they can be precalculated before the FFTs. The overall time complexity of the algorithm is unchanged when wildcard characters are used.

We have shown that string searching with wildcards can be done with $(|\Sigma|-1,0)$ encodings. However, we have yet to discuss how to generate those encodings. One way to solve this problem involves $|\Sigma|-1$ graphs $G_1, G_2, G_3, \dots, G_{|\Sigma|-1}$, where each graph $G_i = (V_i, E_i), \forall i \in \{1 \dots |\Sigma|-1\}$. Each graph's set of vertices V_i contains all $2^{|\Sigma|-1}$ possible binary vectors of length $|\Sigma|-1$. In a graph G_i, two vertices u and v are connected by an edge iff $||v-u||_0 = i$. Then, the problem of finding pairwise equidistant points can be thought of as finding the *max clique* in each graph G_i. A clique is a subset of nodes, where each node is connected to all other nodes in the set, and we wish to find the largest node subset that forms a clique. A clique in graph G_i means that every node in that clique is the same distance i to all other nodes. Note that it may not always be possible to find a clique of size $|\Sigma|$ in any of the graphs.

As an example, for $|\Sigma| = 4$, the following binary vectors are possible $(|\Sigma|-1,0)$ encodings, where $d = 2$:

$$\begin{aligned}
e_1 &= \begin{pmatrix}0 & 0 & 0\end{pmatrix} \\
e_2 &= \begin{pmatrix}1 & 0 & 1\end{pmatrix} \\
e_3 &= \begin{pmatrix}1 & 1 & 0\end{pmatrix} \\
e_4 &= \begin{pmatrix}0 & 1 & 1\end{pmatrix}
\end{aligned} \tag{17}$$

Euclidean Encodings

Since it may not be always possible to generate $(|\Sigma|-1,0)$ encodings, which uses binary vectors, we can use *real* numbers and attempt to calculate the L_2 norm using FFTs. For two encoding vectors u and v, $||v-u||_2 = \sqrt{\sum_{i=1}^{|\Sigma|-1}(v_i-u_i)^2} = d$, where d is a constant distance. For simplicity, we consider d^2, which is also a constant, to get rid of the square root operation. Then, the sum of squared L_2 distances that we want can be expressed as

$$\begin{aligned}\mathscr{C}'_i &= \sum_{j=1}^{(|\Sigma|-1)M} (\mathscr{T}'_{i+j} - \mathscr{P}'_j)^2, \quad \forall i \in \{1\ldots(|\Sigma|-1)N-(|\Sigma|-1)M+1\} \\ &= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}'^2_{i+j} - 2\mathscr{T}'_{i+j}\mathscr{P}'_j + \mathscr{P}'^2_j\end{aligned} \tag{18}$$

This can be computed using FFTs. Also, though the overall equation is similar to that of Clifford and Clifford (2007), the underlying idea for their purpose is different. To convert squared distance to Hamming distance, we do the following:

$$\mathscr{C}_i = \frac{1}{d^2}\mathscr{C}'_{(|\Sigma|-1)(i-1)+1}, \quad \forall i \in \{1\ldots N\} \tag{19}$$

To handle wildcard characters, we can use a technique similar to handling wildcards with Hamming encodings. We create $\mathscr{T}''$ and $\mathscr{P}''$ as binary vectors in the same way as described in Hamming encodings, and multiply them to the distance calculation:

$$\begin{aligned}\mathscr{C}'_i &= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j\left[\mathscr{T}'^2_{i+j} - 2\mathscr{T}'_{i+j}\mathscr{P}'_j + \mathscr{P}'^2_j\right], \quad \forall i \in \{1\ldots(|\Sigma|-1)N-(|\Sigma|-1)M+1\} \\ &= \sum_{j=1}^{(|\Sigma|-1)M} \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'^2_{i+j} - 2\mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{T}'_{i+j}\mathscr{P}'_j + \mathscr{T}''_{i+j}\mathscr{P}''_j\mathscr{P}'^2_j\end{aligned} \tag{20}$$

Like with Hamming encodings, multiplications of vectors that have matching indexes can be precomputed before the FFTs. The $\mathscr{T}''$ and $\mathscr{P}''$ encodings allow the distance to be zero if the character in $\mathscr{T}$ or $\mathscr{P}$ are wildcards at a certain position. Note that the overall time complexity is still $O(|\Sigma|N\log M)$.

Constructing the $(|\Sigma|-1)$-simplexes with the L_2 norm for the encoding vectors can be easily done by first generating the $|\Sigma|-1$ one-hot vectors/points and adding an additional point that is equidistant to all of the $|\Sigma|-1$ points.

CONCLUSION

We discussed two new encoding methods and how it can be applied to finding the Hamming distance between a pattern string and a text string at each location in the text. Though it may not result in practical speedups, the theoretical ideas behind the approaches are novel. The main drawback is that they require multiple FFT calculations, which outweighs the benefits of the slightly shortened encoding vectors.

Pattern searching has many applications in bioinformatics and other fields. In particular, fast algorithms are necessary for searching DNA sequences due to the immense amount of DNA data present. Our work represents a step in constructing faster searching algorithms using the Fast Fourier Transform.

REFERENCES

Clifford, P. and Clifford, R. (2007). Simple deterministic wildcard matching. *Information Processing Letters*, 101(2):53–54.

Clifford, R., Efremenko, K., Porat, E., and Rothschild, A. (2010). Pattern matching with don't cares and few errors. *Journal of Computer and System Sciences*, 76(2):115–124.

Fischer, M. J. and Paterson, M. S. (1974). String-matching and other products. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Linhart, C. and Shamir, R. (2009). Faster pattern matching with character classes using prime number encoding. *Journal of Computer and System Sciences*, 75(3):155–162.

Schoenmeyr, T. and Zhang, D. Y. (2005). FFT-based algorithms for the string matching with mismatches problem. *Journal of Algorithms*, 57(2):130–139.