Theoretical Computer Science 407 (2008) 587–590

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Note

Pattern matching with pair correlation distance☆

Benny Porat, Ely Porat*, Asaf Zur

Department of Computer Science, Bar-Ilan University, 52900 Ramat-Gan, Israel

ARTICLE INFO

Article history:
Received 18 February 2008
Received in revised form 30 May 2008
Accepted 8 August 2008
Communicated by M. Crochemore

ABSTRACT

In pattern matching with the *pair correlation* distance problem, the goal is to find all occurrences of a pattern P of length m, in a text T of length n, where the distance between them is less than a threshold k. For each text location i, the distance is defined as the number of different kinds of mismatched pairs (α, β), between P and $T[i \dots i+m]$. We present an algorithm with running time of $O\left(\min\{|\Sigma_P|^2\, n \log m, n\,(m \log m)^{\frac{2}{3}}\}\right)$ for this problem. Another interesting problem is the *one-side pair correlation* distance where it is desired to find all occurrences of P where the number of mismatched characters in P is less than k. For this problem, we present an algorithm with running time of $O\left(\min\{|\Sigma_P|\, n \log m, n\sqrt{m \log m}\}\right)$.

1. Introduction

Approximate pattern matching requires finding all occurrences of a pattern P in a text T where a match is defined by a distance metric and a threshold. The simplest distance metric is the *Hamming distance*, where the distance in location i is the number of mismatches between the pattern and the sub-string $T[i \dots i+m]$. Landau and Vishkin [10] used suffix trees and LCA queries to solve this problem in $O(nk)$. Amir et al. [4] used the Landau and Vishkin method and combined it with filtering and verification to get an algorithm that runs in $O\left(n\sqrt{k \log k}\right)$ and solves the *Hamming distance* problem. A more generalized problem is the *edit distance problem*, which also captures insertion and deletion. It was presented by Levenshtein [9] and a dynamic programming algorithm was presented by Lowrance and Wagner [11,15]. Other distance metrics were defined in *parameterized matching* [5–7,2,8], *function matching* [1], and *swap matching* [3].

In this paper we present a new metric, called *pair correlation distance*, which counts the number of different kinds of mismatched pairs. A mismatched pair (α, β), $\alpha \in \Sigma_P$, $\beta \in \Sigma_T$, increases the distance only once, regardless of the number of times it occurs. The need for such a metric arises in computational biology. For example, if substance A is required for some reaction but it is missing, it will be replaced by some other substance B. This can be addressed by *pair correlation*. Such a case may occur in protein chain synthesis, when some amino-acid is replaced by another one due to a shortage or some malfunction (radiation that deteriorates the protein structure). The reason for the first substitute may cause other similar substitutes, hence it is very important to detect such connection between mismatches. Sometimes finding a connection between repeating mismatches can yield better explanations for experiments than the traditional *edit distance*.

We also define another problem called *one-side pair correlation* where mismatches are counted by pattern characters only. This metric is used when we want to know that substance A is missing, but we are not interested in which substance replaces it.

Pair correlation is well motivated by music retrieval [14], stock market analysis [12] and copy detection [13], where mismatches influence each other.

☆ Research supported in part by the Israel Science Foundation (ISF) and by the Binational Science Foundation (BSF).

* Corresponding address: Department of Computer Science, Bar-Ilan University, office: room 305, 52900 Ramat-Gan, Israel. Tel.: +972 3 531 8866; fax: +972 3 736 0498.
E-mail addresses: bennyporat@gmail.com (B. Porat), porately@cs.biu.ac.il (E. Porat), zurasa@cs.biu.ac.il (A. Zur).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.08.023

2. Problem definition

Following are some useful definitions that we use throughout this document.

Notation 1. *Let S_1 and S_2 be two equal length strings of size ℓ, over alphabets Σ_1 and Σ_2, respectively. The function $Occ(a, b)$ denotes the number of times the symbol $a \in \Sigma_1$ is aligned with the symbol $b \in \Sigma_2$. Formally, $Occ(a, b) = |\{1 \leq i \leq \ell : S_1[i] = a \wedge S_2[i] = b\}|$.*

Definition 1. Let S_1 and S_2 be two equal length strings, over alphabets Σ_1 and Σ_2, respectively. *Pair Correlation Distance* is $PC(S_1, S_2) = |\{(a, b) : Occ(a, b) \geq 1, a \in \Sigma_1, b \in \Sigma_2\}|$. In other words, $PC(S_1, S_2)$ counts the number of pairs (a, b), $a \in \Sigma_1$, $b \in \Sigma_2$ that are mismatched.

Definition 2. Let $T = t_1 \dots t_n$ be a *text*, and $P = p_1 \dots p_m$ be a *pattern* over alphabets Σ_T and Σ_P, respectively, and let $k \in \mathbf{N}$. The *Pair Correlation Distance problem* of P and T with threshold k, is that of finding all locations $i = 1, \dots, n$, where the *Pair Correlation Distance* of P and a prefix of $t_i \dots t_n$ is less than k, i.e. all locations where $PC(P, t_i \dots t_{i+m}) \leq k$.

Definition 3. Let S_1 and S_2 be two equal length strings, over alphabets Σ_1 and Σ_2, respectively. Denote $Occ(a, b)$ as in Definition 1. *One Side Pair Correlation Distance* is $PC1(S_1, S_2) = |\{a : \exists b \in \Sigma_2 Occ(a, b) \geq 1\}|$. In other words, this metric counts the number of pattern symbols that caused at least one mismatch.

Definition 4. Let $T = t_1 \dots t_n$ be a *text*, and $P = p_1 \dots p_m$ be a *pattern* over alphabets Σ_T and Σ_P, respectively, and let $k \in \mathbf{N}$. The *One-Side Pair Correlation Distance problem* of P and T with threshold k, is that of finding all locations $i = 1, \dots, n$, where the *One-Side Pair Correlation Distance* of P and a prefix of $t_i \dots t_n$ is less than k, i.e. all locations where $PC1(P, t_i \dots t_{i+m}) \leq k$.

For example, consider the following two (equal length) strings:

```
abcaabbcd
fbeffbbee
```

Using the traditional Hamming distance, the number of mismatches is 6. Applying the *two-side pair correlation distance*, the number of mismatches is only 3 because there are only three pairs that are mismatched (f, a), (e, c), (e, d). The *one-side pair correlation distance* gives only 2 mismatches because there are only two symbols that are mismatched: f and e. This example shows that the *pair correlation distance* metric succeeds in detecting the similarity between the two strings while under the traditional Hamming metric they do not resemble each other. The result of this comparison helps a researcher to find out the real reason for the difference between those strings.

Remark Algorithms that solve traditional pattern matching problems, usually find all text locations that match the pattern. The algorithms presented in this paper, report the number of mismatches in each text location as well as the matches.

2.1. Naive algorithms

The naive algorithm runs over all text locations and compares each of them with the whole pattern. Hence its running time is $O(nm)$.

2.2. Convolutions

In some cases we can improve the naive algorithm by using convolutions. The naive algorithm finds for each symbol in the pattern how many times it appears against each symbol in the text for each text location. For each symbol $a \in \Sigma_P$ in the pattern we make a convolution with every symbol $b \in \Sigma_T$ in the text (except the symbol a itself, since it is not a mismatch). These convolutions give all errors caused by each pattern symbol. Knowing how many mismatches each symbol causes it is possible to calculate the *Pair Correlation Distance* for each location. Basically, this algorithm does what the naive algorithm does, but by using convolutions it achieves better running time than the naive algorithm when the alphabets are small.

The number of convolutions made is $O(|\Sigma_T||\Sigma_P|)$, which gives a total running time of $O(|\Sigma_T||\Sigma_P| n \log m)$.

For *one-side pair correlation* it is required to count how many errors each pattern symbol causes, regardless of the text symbols it is aligned with. Hence for each pattern symbol we make only one convolution to check how many errors it contributes. Therefore, the number of convolutions made is only $O(|\Sigma_P|)$, which results in a running time of $O(|\Sigma_P| n \log m)$.

2.3. Filtering and verification

In some cases it is possible to utilize some properties of the pattern and to divide the algorithm into two stages:

1. **Filtering** — In this stage a quick scan of the text is made in order to eliminate a considerable number of text locations.
2. **Verification** — In this stage each text location that passed the filtering stage is checked to see whether it is a match or not. Due to the filtering the number of locations to be verified is much lower than the total number of text locations.

In [4] this method is used widely. However in our case, some changes are required, as described in the next section.

3. Algorithm for one-side

In this section we deal with *one-side pair correlation*. This is a simpler problem than *two-side pair correlation* since we are interested only in pattern symbols. For each pattern symbol we want to know whether it causes at least one mismatch or doesn't cause any. This can be achieved by using one convolution for each symbol, resulting in $|\Sigma_P|$ convolutions and a running time of $O(|\Sigma_P| n \log m)$, as described in Section 2.2. This algorithm has reasonable running time when $|\Sigma_P|$ is smaller than m. However, when $|\Sigma_P| = O(m)$ the running time becomes $O(nm \log m)$ which is worse than that of the naive algorithm.

To improve this we define x to be a threshold such that if $|\Sigma_P| < x$ we make $|\Sigma_P|$ convolutions to solve the problem. The exact value of x will be determined later. From now on, we deal only with the case where $|\Sigma_P| \geq x$.

Definition 5. A pattern symbol is called *frequent* if it appears more than $\frac{m}{x+1}$ times in the pattern. Otherwise it is called *rare*.

The number of frequent symbols is no more than x, hence making a convolution for each frequent symbol results in running time of $O(nx \log m)$.

For rare symbols we use the filtering and verification method. In the filtering stage we look at the first occurrence of each of the pattern symbols. Following is the filtering algorithm:

1. Let *Offset* be an array of all offsets of the first occurrence in the pattern of each symbol in Σ_P.
2. Let *Score* be an array of length $|T|$, initialized to 0.
3. For $i = 1$ to $|T|$
   (a) *Score[i-Offset[T[i]]++*

The filtering stage actually counts how many first occurrences of pattern symbols are aligned with each text location. Each text location that got less than $|\Sigma_P| - k$ scores is discarded, since there are more than k mismatches. The number of locations that passed the filtering stage is at most $O\left(\frac{n}{|\Sigma_P|-k}\right)$. For each text location that passed the filtering stage we have to check only rare symbols, since the frequent symbols were counted by convolutions. Each rare symbol appears no more than $\frac{m}{x+1}$ times and there are no more than $|\Sigma_P|$ rare symbols, so for each text location we have to check at most $O\left(\frac{m|\Sigma_P|}{x+1}\right)$ locations in the pattern. We assume that $k \leq \frac{|\Sigma_P|}{2}$. Since there are no more than $O\left(\frac{n}{|\Sigma_P|-k}\right)$ text locations to check, the total time for rare symbols is $O\left(\frac{nm}{x}\right)$. The total time for all symbols, rare and frequent, is $O\left(xn \log m + \frac{nm}{x}\right)$. Optimizing over x values we find that the minimum is when $x = \sqrt{\frac{m}{\log m}}$, yielding a running time of $O\left(n\sqrt{m \log m}\right)$.

Conclusion: Total running time for *one-side pair correlation* is $O\left(\min\left\{|\Sigma_P| n \log m, n\sqrt{m \log m}\right\}\right)$.

In the above analysis we assumed that $k \leq \frac{|\Sigma_P|}{2}$. This assumption was made to bound the number of text locations that may pass the filtering stage. However, bounding k by $\frac{\sqrt{m}}{2\sqrt{\log m}}$, gives the same running time, because the filtering stage is done only when $|\Sigma_P| \geq \sqrt{\frac{m}{\log m}}$, which bounds the number of text locations that passed the filtering by $O\left(\frac{n}{|\Sigma_P|}\right)$. Hence, the constraint on k is that it should be less than $O\left(\max\left\{\frac{|\Sigma_P|}{2}, \sqrt{\frac{m}{\log m}}\right\}\right)$.

4. Algorithm for two-side

The algorithm shown above, for *one-side pair correlation distance*, can be extended to solve the problem of *two-side pair correlation distance*. We use a common technique in pattern matching, and divide the text into $\frac{n}{m}$ overlapping segments of size $2m$. In each segment there is a sub-segment where its alphabet is bounded by $O(|\Sigma_P| + k)$, otherwise there is no match.

The reason for this is that each text location that has more than $|\Sigma_P| + k$ different symbols is a mismatch. Hence, if a match exists there is a segment of size m with alphabet of size $|\Sigma_P| + k$. Because each segment is of size $2m$ there are no more than $O(|\Sigma_P| + k)$ different text symbols in each text segment. We assume that $k \leq \frac{|\Sigma_P|}{2}$. Using this assumption, the number of convolutions made is no more than $O\left(|\Sigma_P|^2\right)$.

As in *one-side pair correlation* we set a parameter x such that if $|\Sigma_P| < x$ we make $O\left(|\Sigma_P|^2\right)$ convolutions, otherwise we do the following.

We define a frequent symbol as in Definition 5, namely, a symbol that appears more than $\frac{m}{x+1}$ times. In contrast to the *one-side pair correlation* where we have to deal with pattern symbols only, here we have to handle also text symbols. There are four groups of symbols we have to check:

1. Frequent pattern symbols with frequent text symbols
2. Frequent pattern symbols with rare text symbols
3. Rare pattern symbols with frequent text symbols
4. Rare pattern symbols with rare text symbols.

For the first group we use convolutions. There are no more than x frequent pattern symbols, and no more than $2x$ frequent text symbols, hence the total number of convolutions is $O(x^2)$. To check all other groups we use the filtering and verification method. We apply algorithm **I** to eliminate text locations that got fewer than $|\Sigma_P| - k$ scores. After the filtering stage there are no more than $O\left(\frac{n}{|\Sigma_P|-k}\right)$ locations to check. In the verification stage we check each rare pattern symbol naively. There are no more than $|\Sigma_P|$ rare symbols, and each one of them has no more than x occurrences, hence the running time for each text location to count rare pattern symbols is $O\left(\frac{|\Sigma_P|m}{x+1}\right)$. The number of locations we have to check is no more than $O\left(\frac{n}{|\Sigma_P|-k}\right)$ (due to the filtering stage), hence the total running time for rare pattern symbols is $O\left(\frac{nm}{x}\right)$. So far we have checked frequent pattern symbols with frequent text symbols, and rare pattern symbols with frequent and rare text symbols. All we have to check now is frequent pattern symbols with rare text symbols. This is done exactly as we checked the rare pattern symbols — we check naively each text symbol. However, now we handle rare pattern symbols and don't care to avoid counting them twice. The running time for frequent pattern symbols with rare text symbols is also $O\left(\frac{nm}{x}\right)$.

The total time for all symbols, rare and frequent is $O\left(x^2 n \log m + \frac{nm}{x}\right)$. Optimizing over x values we find the minimum is where $x = \sqrt[3]{\frac{m}{\log m}}$, yielding a running time of $O\left(n\,(m \log m)^{\frac{2}{3}}\right)$.

Conclusion: The total running time for the *pair correlation distance* is $O\left(\min\left\{|\Sigma_P|^2\, n \log m,\ n\,(m \log m)^{\frac{2}{3}}\right\}\right)$.

As in *one-side pair correlation* we assumed that $k \leq \frac{|\Sigma_P|}{2}$. Here, in *two-side pair correlation* there are two reasons for this assumption: to bound the number of different text symbols in each text segment when using convolutions (in the case $|\Sigma_P| \leq \sqrt[3]{\frac{m}{\log m}}$), and to bound the number of text locations that may pass the filtering stage (otherwise).

Bounding k by $\frac{\sqrt[3]{m}}{2\sqrt[3]{\log m}}$, gives the same running time. In the case $|\Sigma_P| \leq \sqrt[3]{\frac{m}{\log m}}$, the number of different text symbols is no more than $O\left(|\Sigma_P| + \frac{\sqrt[3]{m}}{2\sqrt[3]{\log m}}\right)$, hence the number of convolutions made is bounded by $O\left(\left(\frac{m}{\log m}\right)^{\frac{2}{3}}\right)$. In the other case, when $|\Sigma_P| > \sqrt[3]{\frac{m}{\log m}}$, the number of text locations that pass the filtering is bounded by $O\left(\frac{n}{|\Sigma_P|}\right)$ resulting in a running time of no more than $O\left(n\,(m \log m)^{\frac{2}{3}}\right)$. Hence, the constraint on k is that it should be less than $O\left(\max\left\{\frac{|\Sigma_P|}{2}, \sqrt[3]{\frac{m}{\log m}}\right\}\right)$.

5. Summary

In this paper we presented and defined the problems of *one-side pair correlation* and *two-side pair correlation*. For *one-side pair correlation* we presented an algorithm that runs in $O\left(\min\left\{|\Sigma_P|\, n \log m,\ n\sqrt{m \log m}\right\}\right)$, for any k that is bounded by $O\left(\max\left\{\frac{|\Sigma_P|}{2}, \sqrt{\frac{m}{\log m}}\right\}\right)$. This algorithm was extended to solve the problem of *two-side pair correlation* with a running time of $O\left(\min\left\{|\Sigma_P|^2\, n \log m,\ n\,(m \log m)^{\frac{2}{3}}\right\}\right)$, for any k that is bounded by $O\left(\max\left\{\frac{|\Sigma_P|}{2}, \sqrt[3]{\frac{m}{\log m}}\right\}\right)$.

References

[1] A. Amir, Y. Aumann, R. Cole, M. Lewenstein, E. Porat, Function matching: Algorithms, applications, and a lower bound, in: Proc. of the International Colloquium on Automata, Languages and Programming, ICALP, 2003, pp. 929–942.
[2] A. Amir, M. Farach, S. Muthukrishnan, Alphabet dependence in parameterized matching, Inform. Process. Lett. 49 (1994) 111–115.
[3] A. Amir, G.M. Landau, M. Lewenstein, N. Lewenstein, Efficient special cases of pattern matching with swaps, Inform. Process. Lett. 68 (3) (1998) 125–132.
[4] A. Amir, M. Lewenstein, E. Porat, Faster algorithms for string matching with k mismatches, in: Proc. 11th ACM-SIAM Symp. on Discrete Algorithms, SODA, 2000, pp. 794–803.
[5] B.S. Baker, A theory of parameterized pattern matching: Algorithms and applications, in: Proc. 25th Annual ACM Symposium on the Theory of Computation, 1993, pp. 71–80.
[6] B.S. Baker, Parameterized pattern matching: Algorithms and applications, J. Comput. System Sci. 52 (1) (1996) 28–42.
[7] B.S. Baker, Parameterized duplication in strings: Algorithms and an application to software maintenance, SIAM J. Comput. 26 (5) (1997) 1343–1362.
[8] C. Hazay, M. Lewenstein, D. Sokol, Approximate parameterized matching, in: Proc. of the 12th Annual European Symposium on Algorithms, ESA, 2004, pp. 414–425.
[9] V.I. Levenstein, Binary codes capable of correcting, deletions, insertions and reversals, Soviet Phys. Dokl. 10 (1966) 707–710.
[10] G.M. Landau, U. Vishkin, Introducing efficient parallelism into approximate string matching, in: Proc. 18th ACM Symposium on Theory of Computing, 1986, pp. 220–230.
[11] R. Lowrance, R.A. Wagner, An extension of the string-to-string correction problem, J. ACM 22 (2) (1975) 177–183.
[12] G.V. Nosovskij, Mathematical analysis of stock market movement, in: 3rd International Conference on Cyberworlds, CW, 2004, pp. 320–321.
[13] S. Schleimer, D.S. Wilkerson, A.A. Winnowing, Local algorithms for document fingerprinting, in: Proceedings of the International Conference on Management of Data, SIGMOD, 2003, pp. 76–85.
[14] I. Shmulevich, O. Yli-Harja, E. Coyle, D. Povel, K. Lemstrom, Perceptual issues in music pattern recognition — Complexity of rhythm and key finding, in: Proc. of AISB Symposium on Musical Creativity, 1999, pp. 64–69.
[15] R.A. Wagner, On the complexity of the extended string-to-string correction problem, in: Proc. of the 7th ACM Symposium on the Theory of Computing, STOC, 1975, pp. 218–223.