A Simple Algorithm for Approximating the Text-To-Pattern Hamming Distance*

Tsvi Kopelowitz[1] and Ely Porat[2]

1 **University of Waterloo, Waterloo, Canada**
kopelot@gmail.com
2 **University of Bar-Ilan, Ramat Gan, Israel**
porately@cs.biu.ac.il

Abstract

The algorithmic task of computing the *Hamming distance* between a given pattern of length m and each location in a text of length n, both over a general alphabet Σ, is one of the most fundamental algorithmic tasks in string algorithms. The fastest known runtime for exact computation is $\tilde{O}(n\sqrt{m})$. We recently introduced a complicated randomized algorithm for obtaining a $1 \pm \epsilon$ approximation for each location in the text in $O(\frac{n}{\epsilon}\log\frac{1}{\epsilon}\log n\log m\log|\Sigma|)$ total time, breaking a barrier that stood for 22 years. In this paper, we introduce an elementary and simple randomized algorithm that takes $O(\frac{n}{\epsilon}\log n\log m)$ time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Pattern Matching, Hamming Distance, Approximation Algorithms

Digital Object Identifier 10.4230/OASIcs.SOSA.2018.10

1 Introduction

One of the most fundamental family of problems in string algorithms is to compute the distance between a given pattern P of length m and each location in a given larger text T of length n, both over alphabet Σ, under some string distance metric (See [24, 20, 2, 25, 8, 6, 3, 7, 29, 12, 28, 26, 9, 11, 31, 27, 19, 10, 15, 18, 17, 16, 5, 4, 30]). One of the most useful distance metrics in this setting is the *Hamming Distance* of two strings, which is the number of aligned character mismatches between the strings. Let $\texttt{HAM}(X,Y)$ denote the Hamming distance of two strings X and Y. Abrahamson [1] showed an algorithm whose runtime is $O(n\sqrt{m\log m})$. The task of obtaining a faster upper bound seems to be very challenging, and indeed there is a folklore matching conditional lower bound for combinatorial algorithms based on the hardness of combinatorial Boolean matrix multiplication (see [14]). However, for constant sized alphabets the runtime is solvable in $O(n\log m)$ using a constant number of convolution computations (which are implemented via the FFT algorithm) [20].

The challenge in beating Abrahamson's algorithm naturally lead to approximation algorithms for computing the Hamming distance in this setting, which is the problem that we consider here and is defined as follows. Denote $T_j = T[j,\dots,j+m-1]$. In the *pattern-to-text approximate Hamming distance problem* the input is a parameter $\epsilon > 0$, T, and P. The goal is to compute for all locations $i \in [1, n-m+1]$ a value δ_i such that $(1-\epsilon)\texttt{HAM}(T_i,P) \le \delta_i \le (1+\epsilon)\texttt{HAM}(T_i,P)$. For simplicity we assume without loss of generality that Σ is the set of integers $\{1,2,\dots,|\Sigma|\}$.

* This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 683064).

1st Symposium on Simplicity in Algorithms (SOSA 2018).
Editor: Raimund Seidel; Article No. 10; pp. 10:1–10:5
Open Access Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Karloff in [22] utilized the efficiency of the algorithm for constant sized alphabets to introduce a beautiful randomized algorithm for solving the pattern-to-text approximate Hamming distance problem, by utilizing projections of Σ to binary alphabets. Karloff's algorithm runs in $\tilde{O}(\frac{n}{\epsilon^2})$ time, and is correct with high probability.

Communication complexity lower bounds

One of the downsides of Karloff's algorithm is the dependence on $\frac{1}{\epsilon^2}$. In particular, if one is interested in a one percent approximation guarantee, then this term becomes 10000, which is extremely large for many applications. Remarkably, many believed that beating the runtime of Karloff's algorithm is not possible, mainly since there exist qualitatively related lower bounds for estimating the Hamming distance of two equal length strings (for a single alignment). In particular, Woodruff [32] and later Jayram, Kumar and Sivakumar [21] showed that obtaining a $(1 \pm \epsilon)$ approximation for two strings in the one-way communication complexity model requires sending $\Omega(1/\epsilon^2)$ bits of information. The lower bounds were extended to the two-way communication complexity model as well [13].

In [23] we showed that **this belief was flawed**, by introducing an $\tilde{O}(\frac{n}{\epsilon})$ time algorithm that succeeds with high probability. In particular, we proved the following.

▶ **Theorem 1** ([23]). *There exists an algorithm that with high probability solves the pattern-to-text approximate Hamming distance problem and runs in* $O(\frac{n}{\epsilon} \log \frac{1}{\epsilon} \log n \log m \log |\Sigma|)$ *time.*

Our algorithm in [23] turned out to be rather complicated and borrows ideas from sparse recovery and constructions of specialized families of hash functions.

A simpler algorithm

In this paper we show how to solve the pattern-to-text approximate Hamming distance problem faster (in terms of logarithmic factors) and simpler. The rest of this paper is devoted to proving the following theorem.

▶ **Theorem 2** ([23]). *There exists an algorithm that with high probability solves the pattern-to-text approximate Hamming distance problem and runs in* $O(\frac{n}{\epsilon} \log n \log m)$ *time.*

2 The Algorithm

For a function $h : \Sigma \to \Sigma'$ and for any string $S = s_1 s_2 \ldots s_k$, let $h(S) = h(s_1)h(s_2)\ldots h(s_k)$.

Local versus global operations

The operations that our algorithm performs during the approximation of the Hamming distance at some location j are partitioned into two types. The first type is *local* operations which are independent of the computations performed for other locations in T. The second type is *global* operations, which are operations that for efficiency purposes are computed as a batch for all of the alignments in T. In particular, all of the global operations in our algorithm are to compute $HAM(h(T_j), h(P))$ where $h : \Sigma \to \left[\frac{2}{\epsilon}\right]$. Such a computation will make use of the following Theorem (which uses the FFT algorithm; see [20]), and is summarized in Corollary 4.

Algorithm 1 The new algorithm.

```
ApproxHAM(T_j, P, ε)
1  for i = 1 to c log n
2        do Pick a random h : Σ → {1, 2, ..., 2/ε}.
3           compute x_i = HAM(h(T_j), h(P))
4  return max_{1≤i≤c log n} {x_i}
```

▶ **Theorem 3.** *Given a binary text T of size n and a binary pattern P of size m, there exists an $O(n \log m)$ time algorithm that computes for all locations j in T the number of times that a 1 in T_j is aligned with a 1 in P.*

▶ **Corollary 4.** *Given a text T of size n and a pattern P of size m both over alphabet Σ, there exists an $O(|\Sigma| \cdot n \log m)$ time algorithm that computes $HAM(T_j, P)$ for all locations j in T.*

The algorithm for Corollary 4 is implemented by considering a separate binary text and binary pattern for each character $\sigma \in \Sigma$. For character σ in this set, occurrences of σ in T are assigned to 1, while occurrences of other characters are assigned to 0. On the other hand, occurrences of σ in P are assigned to 0, while occurrences of other characters are assigned to 1. Applying Theorem 3 on the binary text and pattern defined by σ enables computing for every location j the number of times character σ in T_j contributes to $\mathtt{HAM}(T_j, P)$. A summation over all the mismatches for all the characters in Σ completes the computation of $\mathtt{HAM}(T_j, P)$. Since Theorem 3 is applied $|\Sigma|$ times, the Corollary follows. Notice that when the algorithm of Corollary 4 is applied, then each location in the text is charged an $O(|\Sigma| \log m)$ (global) time cost.

The algorithm

With the goal of easing the presentation of our algorithm, we focus on estimating the Hamming distance between T_j and P, and count the cost of global and local operations for this location. Since we are interested in algorithms that succeed with high probability (at least $1 - \frac{1}{n^{\Theta(1)}}$) then it suffices to show that with high probability the algorithm succeeds at location j. The pseudo-code for the algorithm is given in Algorithm 1.

Time complexity

Computing the Hamming distance between the projected text and projected pattern in Line 3 takes place by applying the algorithm from Corollary 4 where the alphabet is $\left[\frac{2}{\epsilon}\right]$. Thus, the total time cost for location j is $O(\frac{1}{\epsilon} \log n \log m)$, and so the overall time cost for all locations is $O(\frac{n}{\epsilon} \log n \log m)$.

Correctness

Let $d = HAM(T_j, P)$. The goal of the algorithm is to approximate d. Notice that the expected value of x_i is $E[x_i] = (1 - \frac{\epsilon}{2})d$, since each mismatch in the original text and pattern remains a mismatch after the projection obtained by applying h with probability $1 - \frac{\epsilon}{2}$. Thus,

$E[d - x_i] = \frac{\epsilon d}{2}$, and so by the Markov inequality,

$$\Pr[x_i < (1-\epsilon)d] = \Pr[d - x_i > \epsilon d] \leq \frac{E[d - x_i]}{\epsilon d} = \frac{1}{2}.$$

Since the algorithm returns the largest x_i value, the only way in which the algorithm fails is if all of the x_i values are less than $(1-\epsilon)d$. Since the choices of the projections is independent, this happens with probability at most n^{-c}.

References

1 K. Abrahamson. Generalized string matching. In *SIAM J. Computing 16 (6)*, page 1039–1051, 1987.

2 A. Amir, O. Lipsky, E. Porat, and J. Umanski. Approximate matching in the l1 metric. In *CPM*, pages 91–103, 2005.

3 Amihood Amir, Yonatan Aumann, Gary Benson, Avivit Levy, Ohad Lipsky, Ely Porat, Steven Skiena, and Uzi Vishne. Pattern matching with address errors: rearrangement distances. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1221–1229, 2006.

4 Amihood Amir, Yonatan Aumann, Piotr Indyk, Avivit Levy, and Ely Porat. Efficient computations of l_1 and l_{infinity} rearrangement distances. In *String Processing and Information Retrieval, 14th International Symposium, SPIRE 2007, Santiago, Chile, October 29-31, 2007, Proceedings*, pages 39–49, 2007.

5 Amihood Amir, Yonatan Aumann, Oren Kapah, Avivit Levy, and Ely Porat. Approximate string matching with address bit errors. In *Combinatorial Pattern Matching, 19th Annual Symposium, CPM 2008, Pisa, Italy, June 18-20, 2008, Proceedings*, pages 118–129, 2008.

6 Amihood Amir, Estrella Eisenberg, and Ely Porat. Swap and mismatch edit distance. In *Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings*, pages 16–27, 2004.

7 Amihood Amir, Tzvika Hartman, Oren Kapah, Avivit Levy, and Ely Porat. On the cost of interchange rearrangement in strings. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 99–110, 2007.

8 Amihood Amir, Moshe Lewenstein, and Ely Porat. Approximate swapped matching. In *Foundations of Software Technology and Theoretical Computer Science, 20th Conference, FST TCS 2000 New Delhi, India, December 13-15, 2000, Proceedings.*, pages 302–311, 2000.

9 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 377–386, 2010.

10 A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Accepted to 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

11 Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 550–559, 2004.

12 Ayelet Butman, Noa Lewenstein, Benny Porat, and Ely Porat. Jump-matching with errors. In *String Processing and Information Retrieval, 14th International Symposium, SPIRE 2007, Santiago, Chile, October 29-31, 2007, Proceedings*, pages 98–106, 2007.

13 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012.

14 Raphael Clifford. Matrix multiplication and pattern matching under hamming norm. `http://www.cs.bris.ac.uk/Research/Algorithms/events/BAD09/BAD09/Talks/BAD09-Hammingnotes.pdf`. Retrieved August 2015.

15 Raphaël Clifford, Klim Efremenko, Benny Porat, Ely Porat, and Amir Rothschild. Mismatch sampling. *Information and Computation*, 214:112–118, 2012.

16 Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. From coding theory to efficient pattern matching. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 778–784, 2009. URL: `http://dl.acm.org/citation.cfm?id=1496770.1496855`.

17 Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. Pattern matching with don't cares and few errors. *Journal of Computer System Science*, 76(2):115–124, 2010.

18 Raphaël Clifford and Ely Porat. A filtering algorithm for k-mismatch with don't cares. *Inf. Process. Lett.*, 110(22):1021–1025, 2010. `doi:10.1016/j.ipl.2010.08.012`.

19 Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 667–676, 2002.

20 M.J. Fischer and M.S. Paterson. String matching and other products. r.m. karp (ed.), complexity of computation. In *SIAM–AMS Proceedings, vol. 7,*, page 113–125, 1974.

21 T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

22 H. Karloff. Fast algorithms for approximately counting mismatches. In *Inf. Process. Lett. 48 (2)*, pages 53–60, 1993.

23 Tsvi Kopelowitz and Ely Porat. Breaking the variance: Approximating the hamming distance in $1/\epsilon$ time per alignment. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pages 601–613, 2015.

24 Vladimir Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. In *Probl. Inf. Transmission 1*, page 8–17, 1965.

25 O. Lipsky and E. Porat. Approximated pattern matching with the l1, l2 and linfinit metrics. In *SPIRE*, pages 212–223, 2008.

26 R. Lowrance and R. A. Wagner. An extension of the string-to-string correction problem. *J. of the ACM*, pages 177–183, 1975.

27 Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 315–323, 2009.

28 Benny Porat, Ely Porat, and Asaf Zur. Pattern matching with pair correlation distance. In *String Processing and Information Retrieval, 15th International Symposium, SPIRE 2008, Melbourne, Australia, November 10-12, 2008. Proceedings*, pages 249–256, 2008.

29 Ely Porat and Klim Efremenko. Approximating general metric distances between a pattern and a text. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 419–427, 2008.

30 Ely Porat and Ohad Lipsky. Improved sketching of hamming distance with error correcting. In *Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007, London, Canada, July 9-11, 2007, Proceedings*, pages 173–182, 2007.

31 Ariel Shiftan and Ely Porat. Set intersection and sequence matching. In *String Processing and Information Retrieval, 16th International Symposium, SPIRE 2009, Saariselkä, Finland, August 25-27, 2009, Proceedings*, pages 285–294, 2009.

32 David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 167–175, 2004.