Approximating Approximate Pattern Matching

Jan Studený and Przemysław Uznański

ETH Zürich, Switzerland

Abstract

Given a text $T$ of length $n$ and a pattern $P$ of length $m$, the approximate pattern matching problem asks for computation of a particular distance function between $P$ and every $m$-substring of $T$. We consider a $(1 \pm \varepsilon)$ multiplicative approximation variant of this problem, for $\ell_p$ distance function. In this paper, we describe two $(1 + \varepsilon)$-approximate algorithms with a runtime of $\tilde{O}(\frac{n}{\varepsilon})$ for all (constant) non-negative values of $p$. For constant $p \geq 1$ we show a deterministic $(1 + \varepsilon)$-approximation algorithm. Previously, such run time was known only for the case of $\ell_1$ distance, by Gawrychowski and Uznański [ICALP 2018] and only with a randomized algorithm. For constant $0 \leq p \leq 1$ we show a randomized algorithm for the $\ell_p$, thereby providing a smooth tradeoff between algorithms of Kopelowitz and Porat [FOCS 2015, SOSA 2018] for Hamming distance (case of $p = 0$) and of Gawrychowski and Uznański for $\ell_1$ distance.

1 Introduction

Pattern matching is one of the core problems in text processing algorithms. Given a text $T$ of length $n$ and a pattern $P$ of length $m$, $m \leq n$, both over an alphabet $\Sigma$, one searches for occurrences of $P$ in $T$ as a substring. A generalization of a pattern matching is to find substrings of $T$ that are similar to $P$, where we consider a particular string distance and ask for all $m$-substrings of $T$ where the distance to $P$ does not exceed a particular threshold, or simply report distance from $P$ to every $m$-substring of $T$. Typical distance functions considered are Hamming distance, $\ell_1$ distance, or in general $\ell_p$ distances for some constant $p$, assuming input is over a numerical, e.g. integer, alphabet.

For reporting all Hamming distances, Abrahamson [Abr87] described an algorithm with the complexity of $O(n\sqrt{m \log m})$. Using a similar approach, the same complexity was obtained in [CCI05, ALPU05] for reporting all $\ell_1$ distances. It is a major open problem whether near-linear time algorithm, or even $O(n^{3/2-\varepsilon})$ time algorithms, are possible for such problems. A conditional lower bound [Cli09] was shown, via a reduction from matrix multiplication. This means that existence of combinatorial algorithm with runtime $O(n^{3/2-\varepsilon})$ solving the problem for Hamming distances implies combinatorial algorithms for boolean matrix multiplication with $O(n^{3-\delta})$ runtime, which existence is unlikely. If one is uncomfortable with badly defined notion of combinatorial algorithms, one can apply the reduction to obtain a lowerbound of $\Omega(n^{\omega/2})$ for Hamming distances pattern matching, where $2 \leq \omega < 2.373$ is a matrix multiplication exponent.\footnote{Although the issue is that we do not even know whether $\omega > 2$ or not.} Later, complexity of pattern matching under Hamming distance and under $\ell_1$ distance was proven to be identical (up to polylogarithmic terms) [GLU18, LP08].
The mentioned hardness results serve as a motivation for considering relaxation of the problems, with \((1 + \varepsilon)\) multiplicative approximation being the obvious candidate. For Hamming distance, Karloff [Kar93] was the first to propose an efficient approximation algorithm with a run time of \(\tilde{O}\left(\frac{n}{\varepsilon}\log^3 m\right)\).\(^2\) The \(\frac{1}{\varepsilon}\) dependency was believed to be inherent, as is the case for e.g. space complexity of sketching of Hamming distance, cf. [Woo04, JKS08, CR12]. However, for approximate pattern matching that was refuted by Kopelowitz and Porat [KP15, KP18], by providing randomized algorithms with complexity \(\tilde{O}\left(\frac{n}{\varepsilon}\log n \log m \log \frac{1}{\varepsilon} \log |\Sigma|\right)\) and \(\tilde{O}\left(\frac{n}{\varepsilon} \log n \log m\right)\) respectively. Moving to \(\ell_1\) distance, Lipsky and Porat [LP11] gave a deterministic algorithm with a run time of \(\tilde{O}\left(\frac{n}{\varepsilon}\log m \log U\right)\), while later Gawrychowski and Uznański [GU18] have improved the complexity to a (randomized) \(\tilde{O}\left(\frac{n}{\varepsilon} \log^2 n \log m \log U\right)\), where \(U\) is the maximal integer value on the input.

A folklore result (c.f. [LP11]) states that the randomized algorithm with a run time of \(\tilde{O}\left(\frac{n}{\varepsilon}\right)\) is in fact possible for any \(\ell_p\) distance, \(0 < p \leq 2\), with use of \(p\)-stable distributions and convolution. Such distributions exist only when \(p \leq 2\), which puts a limit on this approach. See [No03] for wider discussion on \(p\)-stable distributions. Porat and Efremenko [PE08] have shown how to approximate general distance functions between pattern and text in time \(\tilde{O}\left(\frac{n}{\varepsilon}\right)\). Their solution does not immediately translates to \(\ell_p\) distances, since it allows only for score functions of form \(\sum d(t_{i+j}, p_j)\) where \(d\) is arbitrary metric over \(\Sigma\). Authors state that their techniques generalize to computation of \(\ell_2\) distances, and in fact those generalize further to \(\ell_p\) distances as well, but the dependency \(\varepsilon^{-2}\) in their approach is unavoidable. [LP11] observe that \(\ell_2\) pattern matching can be in fact computed in \(\tilde{O}(n)\) time, by reducing it to a single convolution computation. This case and analogously case of \(p = 4, 6, \ldots\) are the only ones where fast and exact algorithm is known.

We want to point that for \(\ell_\infty\) pattern matching there is an approximation algorithm of complexity \(\tilde{O}\left(\frac{n}{\varepsilon}\right)\) by Lipsky and Porat [LP11]. Moving past pattern matching, we want to point that in a closely related problem of computing \((\min, +)\)-convolution there exists \(\tilde{O}\left(\frac{n}{\varepsilon}\right)\) time algorithm computing \((1 + \varepsilon)\) approximation, cf. Mucha et al. [MWW19].

Two questions follow naturally. First, is there a \(\tilde{O}\left(\frac{n}{\varepsilon}\log(\varepsilon)\right)\) algorithm for \(\ell_p\) norms pattern matching when \(p > 2\)? Second, is there anything special to \(p = 0\) and \(p = 1\) cases that allows for faster algorithms, or can we extend their complexities to other \(\ell_p\) norms? To motivate further those questions, observe that in the regime of maintaining \(\ell_p\) sketches in the *turnstile* streaming model (sequence of updates to vector coordinates), one needs small space of \(\Theta(\log n)\) bits when \(p \leq 2\) (cf. [KNW10]), while when \(p > 2\) one needs large space of \(\Theta(n^{1-2/p} \log n)\) bits (cf. [Gan15, LW13]) meaning there is a sharp transition in problem complexity at \(p = 2\). Similar phenomenon of transition at \(p = 2\) is observed for \(p\)-stable distributions, and one could expect such transition to happen in the pattern matching regime as well.

In this work we show that for any constant \(p \geq 0\) there is an algorithm of complexity \(\tilde{O}\left(\frac{n}{\varepsilon}\right)\), replicating the phenomenon of linear dependency on \(\varepsilon^{-1}\) from Hamming distance and \(\ell_1\) distance to all \(\ell_p\) norms. Additionally this provides evidence that no transition at \(p = 2\) happens, and so far to our understanding cases of \(p > 2\) and \(p < 2\) are of similar hardness.

### 1.1 Definitions and preliminaries.

Model. In most general setting, our inputs are strings taken from arbitrary alphabet \(\Sigma\). We use this notation only when structure of alphabet is irrelevant for the problem (e.g. Hamming distances). However, when considering \(\ell_p\) distances we focus our attention over an integer alphabet.

---

\(^2\)We use \(\tilde{O}\) notation to hide factors polylogarithmic in \(n, m, |\Sigma|, U\) and \(\varepsilon^{-1}\).
Distance between strings. Let $X = x_1x_2\ldots x_n$ and $Y = y_1y_2\ldots y_n$ be two strings. For any $p > 0$, we define their $\ell_p$ distance as

$$\ell_p(X, Y) = \left(\sum_i |x_i - y_i|^p\right)^{1/p}.$$ 

Particularly, $\ell_1$ distance is known as Manhattan distance, and $\ell_2$ distance is known as Euclidean distance. Observe that the $p$-th power of $\ell_p$ distance has particularly simpler form of $\ell_p(X, Y)^p = \sum_i |x_i - y_i|^p$.

The Hamming distance between two strings is defined as

$$\text{Ham}(X, Y) = |\{i : x_i \neq y_i\}|.$$ 

Adopting the convention that $0^0 = 0$ and $x^0 = 1$ for $x \neq 0$, we observe that $(\ell_p)^p$ approaches Hamming distance as $p \to 0$. Thus Hamming distance is usually denoted as $\ell_0$ (although $(\ell_0)^0$ is more precise notation).

Text-to-pattern distance. For text $T = t_1t_2\ldots t_n$ and pattern $P = p_1p_2\ldots p_m$, the text-to-pattern distance is defined as an array $S$ such that, for every $i$, $S[i] = d(T[i+1..i+m], P)$ for particular distance function $d$. Thus, for $\ell_p$ distance $S[i] = \left(\sum_{j=1}^{m}|t_{i+j} - p_j|^p\right)^{1/p}$, while for Hamming distance $S[i] = |\{j \in \{1, \ldots, m\} : t_{i+j} \neq p_j\}|$. Then $(1 + \varepsilon)$-approximate distance is defined as an array $S_\varepsilon$ such that, for every $i$, $(1 - \varepsilon) \cdot S[i] \leq S_\varepsilon[i] \leq (1 + \varepsilon) \cdot S[i]$.

Rounding and arithmetic operations. For any value $x$, we denote by $x^{(i)} = [x/2^i] \cdot 2^i$ the value with $i$ lowest bits rounded. However, with a little stretch of notation, we do not limit value of $i$ to be positive. We denote by $\|r\|_c$ the norm modulo $c$, that is $\|r\|_c = \min(r \text{ mod } c, c - (r \text{ mod } c))$.

1.2 Our results.

In this paper we answer favorably both questions by providing relevant algorithms. First, we show how to extend the deterministic $\ell_1$ distances algorithm into $\ell_p$ distances, when $p \geq 1$.

**Theorem 1.1.** For any $p \geq 1$ there is a deterministic algorithm computing $(1 + \varepsilon)$ approximation to pattern matching under $\ell_p$ distances in time $O\left(\frac{1}{\varepsilon} \log m \log U\right)$ (assuming $\varepsilon \leq 1/p$).

We then move to the case of $\ell_p$ distances when $p < 1$. We show that it is possible to construct a randomized algorithm with the desired complexity.
Theorem 1.2. For $0 < p < 1$, there is a randomized algorithm computing $(1 + \varepsilon)$-approximation to pattern matching under $\ell_p$ distances in time $O(p^{-1}e^{-1}n \log m \log^3 n \log n)$. The algorithm is correct with high probability.\footnote{Probability at least $1 - 1/n^c$ for arbitrarily large constant $c$.}

Finally, combining with existing $\ell_0$ algorithm from [KP18] we obtain as a corollary that for constant $p \geq 0$ approximation of pattern matching under $\ell_p$ distances can be computed in $\tilde{O}(\frac{n}{\varepsilon})$ time.

2 Approximation of $\ell_p$ distances

We start by showing how convolution finds its use in counting versions of pattern matching, either exact or approximation algorithms. Consider the case of pattern matching under $\ell_2$ distances. Observe that we are looking for $S$ such that $S[i]^2 = \sum_{j} (t_j - p_k)^2 = \sum_{j} t_j^2 + \sum_k p_k^2 - 2 \sum_{j} t_j p_k$. The last term is just a convolution of vectors in disguise and is equivalent to computing convolution of $T$ and reverse ordered $P$. Such approach can be applied to solving exact pattern matching via convolution (observing that $\ell_2$ distance is 0 iff there is an exact match).

We follow with a technique for computing exact text-to-pattern distance, for arbitrary distance functions, introduced by [LP11], which is a generalization of a technique used in [FP74]. We provide a short proof for completeness.

Theorem 2.1 ([LP11]). Text-to-pattern distance where strings are over arbitrary alphabet $\Sigma$ can be computed exactly in time $O(|\Sigma| \cdot n \log m)$.

Proof. For every letter $c \in \Sigma$, construct a new text $T^c$ by setting $T^c[i] = 1$ if $t_i = c$ and $T^c[i] = 0$ otherwise. A new pattern $P^c$ is constructed by setting $P^c[i] = d(c, p_i)$. Since $d(t_{i+j}, p_j) = \sum_{c \in \Sigma} T^c[i+j] \cdot P^c[j]$, it is enough to invoke $|\Sigma|$ times convolution.

Theorem 2.1 allows us to compute text-to-pattern distance exactly, but the time complexity $O(|\Sigma| n \log m)$ is prohibitive for large alphabets (when $|\Sigma| = o(\log n)$). However, it is enough to reduce the size of alphabet used in the problem (at the cost of reduced precision) to reach desired time complexity. While this might be hard, we proceed as follows: we decompose our weight function into a sum of components, each of which is approximated by a corresponding function on a reduced alphabet.

We say that a function $d$ is effectively over smaller alphabet $\Sigma'$ if it is represented as $d(x, y) = d'(\iota_1(x), \iota_2(y))$ for some $\iota_1, \iota_2 : \Sigma \rightarrow \Sigma'$ and $d'$. It follows from Theorem 2.1 that text-to-pattern under distance $d$ can be computed in time $\tilde{O}(|\Sigma'| n)$ (ignoring the cost of computing $\iota_1$ and $\iota_2$).

Decomposition. Let $D(x, y) = |x - y|^p$ be a function corresponding to $(\ell_p)^p$ distance, that is $\ell_p(X, Y)^p = \sum_{i} D(x_i, y_i)$. Our goal is to decompose $D(x, y) = \sum_{i} \alpha_i(x, y)$ into small (polylogarithmic) number of functions, such that each $\alpha_i(x, y)$ is approximated by $\beta_i(x, y)$ that is effectively over alphabet of $O(\frac{1}{\varepsilon})$ size (up to polylogarithmic factors). Now we can use Theorem 2.1 to compute contribution of each $\beta_i$. We then have that $G(x, y) = \sum_{i} \beta_i(x, y)$ approximates $F$, and text-to-pattern distance under $G$ can be computed in the desired $\tilde{O}(\frac{n}{\varepsilon})$ time. We present such decomposition, useful immediately in case of $p \geq 1$ and as we see in section 2.2 with a little bit of effort as well in case when $0 < p \leq 1$.\footnote{Probability at least $1 - 1/n^c$ for arbitrarily large constant $c$.}
**Useful estimations.** We use following estimations in our proofs.

For $p \geq 1$

\begin{align*}
(1 - \varepsilon)^p &\geq 1 - p\varepsilon, & &\text{for } 0 \leq \varepsilon \leq 1, \quad (1) \\
(1 + \varepsilon)^p &\geq 1 + p\varepsilon, & &\text{for } 0 \leq \varepsilon, \quad (2) \\
(1 - \varepsilon)^p &\leq 1 - p\varepsilon(1 - 1/p), & &\text{for } 0 \leq \varepsilon \leq 1/p. \quad (3)
\end{align*}

For $0 \leq p \leq 1$

\begin{align*}
(1 - \varepsilon)^p &\leq 1 - p\varepsilon, & &\text{for } 0 \leq \varepsilon \leq 1, \quad (4) \\
(1 - \varepsilon)^p &\geq 1 - 2p\varepsilon \ln 2, & &\text{for } 0 \leq \varepsilon \leq 1/2, \quad (5) \\
(1 + \varepsilon)^p &\geq 1 + p\varepsilon \ln 2, & &\text{for } 0 \leq \varepsilon \leq 1. \quad (6)
\end{align*}

2.1 Algorithm for $p \geq 1$

In this section we prove Theorem 1.1. We start by constructing a family of functions $F_i$, which are better refinements of $F$ as $i$ decreases.

**First step:** Let us denote

\[ F_i(x, y) = \left( \max(0, |x - y| - 2^i) \right)^p \quad \text{and} \quad f_i = F_i - F_{i+1}. \]

Observe that $F_u = 0$ (for $0 \leq x, y \leq U$). Moreover, there is a telescoping sum $F_i = \sum_{j=i}^u f_j$. To better see the telescopic sum, consider case $p = 1$. We then represent $F_u(x, y) = \sum_{i=-u}^u f_i(x, y) = (-2^{-u} + 2^{-u+1}) + (-2^{-u+1} + 2^{-u+2}) + \ldots + (-2^{i-1} + 2^i) + (|x - y| - 2^i) + 0 + \ldots + 0$. Such decomposition (for $p = 1$) was first considered, to our knowledge, in [LP11].

**Second step:** Instead of using $x$ and $y$ for evaluation of $F_i$, we evaluate $F_i$ using $x$ and $y$ with all bits younger than $i$-th one set to zero. Formally, define $x^{(i)} = \lfloor x/2^i \rfloor \cdot 2^i$, $y^{(i)} = \lfloor y/2^i \rfloor \cdot 2^i$. Now we denote

\[ G_i(x, y) = F_i(x^{(i)}, y^{(i)}). \]

Similarly as for $f_i$, define $g_i = G_i - G_{i+1}$. Using the same reasoning, we have $G_u = 0$. For integers $i \leq 0$ the functions $F_i$ and $G_i$ are the same (as we are not rounding) and therefore $F_{-u} = G_{-u} = \sum_{i=-u}^u g_i$. Intuitively, $g_i$ captures contribution of $i$-th bit of input to the output value (assuming all higher bits are set and known, and all lower bits are unknown).

**Third step:** Let $\eta$ be a value to be fixed later, depending on $\varepsilon$ and $p$. Assume w.l.o.g. that $\eta$ is such that $1/\eta$ is an integer. We now define $\tilde{g}_i$ as a refinement of $g_i$, by replacing $|x^{(i)} - y^{(i)}|$ with $\|x^{(i)} - y^{(i)}\|_{B_i}$ and $|x^{(i+1)} - y^{(i+1)}|$ with $\|x^{(i+1)} - y^{(i+1)}\|_{B_i}$, where $B_i = 2^i/\eta$, that is doing all the computation modulo $B_i$. To be precise, define
\[
\begin{align*}
\widetilde{G}_i(x, y) &= \left( \max(0, \|x^{(i)} - y^{(i)}\|_{B_i} - 2^i) \right)^p \\
\widetilde{G}_{i+1}(x, y) &= \left( \max(0, \|x^{(i+1)} - y^{(i+1)}\|_{B_i} - (2^{i+1}) \right)^p
\end{align*}
\]

and then \( \hat{g}_i = \widetilde{G}_i - \widetilde{G}_{i+1} \). Additionally, we denote for short \( \hat{G}_i = \sum_{j=i}^u \hat{g}_j \).

Intuitively, \( \hat{g}_i \) approximates \( g_i \) in the scenario of limited knowledge – it estimates contribution of \( i \)-th bit of input to the output, assuming knowledge of bits \( i + 1 \) to \( i + \log \eta^{-1} \) of input.

We are now ready to provide an approximation algorithm to \((\ell_p)^p\) text-to-pattern distances.

**Algorithm 2.2.**

**Input:**
- \( T \) is the text,
- \( P \) is the pattern,
- \( \eta \) controls the precision of the approximation.

**Steps:**

1. For each \( i \in \{-u, \ldots, u\} \) compute array \( S_i \) being the text-to-pattern distance between \( T \) and \( P \) using \( \hat{g}_i \) distance function (parametrized by \( \eta \)) using Theorem 2.1.

2. Output array \( S_\varepsilon[i] = \left( \sum_{j=-u}^{u} S_j[i] \right)^{1/p} \).

To get the \((1 + \varepsilon)\) approximation we need to run the Algorithm 2.2 with \( \eta = \frac{\varepsilon}{128} \).

Now, we need to show the running time and correctness of the result. Firstly, to prove the correctness, we divide summands \( \hat{g}_i \) into three groups and reason about them separately. As computing \( F_{-u}, G_{-u} \) (by summing \( f_i \)'s and \( g_i \)'s respectively) yields \((1 + \varepsilon)\) multiplicative error, we will show that the difference between computing \( g_i \) and \( \hat{g}_i \) brings only an additional \((1 + \varepsilon)\) multiplicative error.

**Lemma 2.3.** For \( i \) such that \(|x - y| \leq 2^i \) both \( g_i(x, y) = 0 \) and \( \hat{g}_i(x, y) = 0 \).

**Proof.** As both \( g_i, \hat{g}_i \) are symmetric functions, we can w.l.o.g. assume \( x \geq y \). \( \forall j \geq i: \)

\[
|x^{(j)} - y^{(j)}| = 2^j \left( \left| \frac{x}{2^j} \right| - \left| \frac{y}{2^j} \right| \right) \leq 2^j \left( \left| \frac{x}{2^j} \right| - \left| \frac{x - 2^i}{2^j} \right| \right) \leq 2^i.
\]

Therefore \( G_j = 0 \) from which \( g_i(x, y) = 0 \) follows. And because \( \|x^{(j)} - y^{(j)}\|_{B_j} \leq |x^{(j)} - y^{(j)}| \)
we have \( \hat{g}_i(x, y) = 0 \) as well. \( \square \)

**Lemma 2.4.** For \( i \) such that \(|x - y| > 2^i \geq 4\eta|x - y| \) we have \( g_i(x, y) = \hat{g}_i(x, y) \).
Theorem 2.7. \( \hat{g}_i(x, y) \) to hold, it is enough to show that both "norms" \(| \cdot |\) and \(\| \cdot \|_{B_i}\) are the same for \(x^{(i)} - y^{(i)}\) and \(x^{(i+1)} - y^{(i+1)}\). This happens if the absolute values of the respective inputs are smaller than \(B_i/2\). Let us bound both \(|x^{(i)} - y^{(i)}|\) and \(|x^{(i+1)} - y^{(i+1)}|\):

\[
\max(|x^{(i)} - y^{(i)}|, |x^{(i+1)} - y^{(i+1)}|) \leq |x - y| + 2^{i+1} \leq 2^{i+1}(1 + \frac{1}{8\eta}).
\]

We can w.l.o.g. assume \(\eta \leq 1/8\) in order to make \(\frac{1}{8\eta}\) a dominant term in the parentheses and reach:

\[
\max(|x^{(i)} - y^{(i)}|, |x^{(i+1)} - y^{(i+1)}|) \leq 2^{i+1}(1 + \frac{1}{8\eta}) \leq \frac{2^i}{2\eta} = B_i.
\]

Therefore \(\|x^{(i)} - y^{(i)}\|_{B_i} = |x^{(i)} - y^{(i)}|\) as well as \(\|x^{(i+1)} - y^{(i+1)}\|_{B_i} = |x^{(i+1)} - y^{(i+1)}|\) which completes the proof.

\[\square\]

Lemma 2.5. If \(p \geq 1\) then for \(i\) such that \(4\eta|x - y| > 2^i\) we have \(|g_i(x, y)| \leq 2p2^i \cdot |x - y|^{p-1}\).

Proof. For the sake of the proof, we will w.l.o.g. assume \(\eta \leq 1/8\). Denote \(A = |x^{(i)} - y^{(i)}|\), \(B = |x^{(i+1)} - y^{(i+1)}|\), \(A' = \max(0, A - 2^i)\) and \(B' = \max(0, B - 2^{i+1})\). Observe that \(|x - y| - 2^i \leq A \leq |x - y| + 2^i\) thus \(|x - y| - 2 \cdot 2^i \leq A' \leq |x - y|\), and similarly \(|x - y| - 2 \cdot 2^{i+1} \leq B' \leq |x - y|\) so \(|A' - B'| \leq 2 \cdot 2^i\). Assume w.l.o.g. that \(A' \geq B'\). We bound

\[
|g_i(x, y)| = ((A')^p - (B')^p) \leq p(A' - B')(A')^{p-1} \leq 2p2^i \cdot |x - y|^{p-1}
\]

\[\square\]

Lemma 2.6. If \(p \geq 1\) then for \(i\) such that \(4\eta|x - y| > 2^i\) we have \(\|\hat{g}_i(x, y)\| \leq 2p2^i \cdot |x - y|^{p-1}\).

Proof. Follows by the same proof strategy as in proof of Lemma 2.5, replacing \(| \cdot |\) with \(\| \cdot \|_{B_i}\).

\[\square\]

Theorem 2.7. \(\hat{G}_{-u} = \sum_{i \geq -u} \hat{g}_i\) approximates \(F_{-u}\) up to an additive \(32 \cdot p \cdot \eta \cdot |x - y|^p\) term.

Proof. We bound the difference between two terms:

\[
|F_{-u}(x, y) - \sum_{i = -u}^{u} \hat{g}_i(x, y)| \leq \sum_{i = -u}^{u} (|\hat{g}_i(x, y)| + |g_i(x, y)|) \\
\leq 2 \cdot \left(\sum_{i = -\infty}^{\log_2(4\eta|x - y|)} 2^i\right) \cdot 2 \cdot p \cdot |x - y|^{p-1} \\
\leq 32 \cdot \eta|x - y| \cdot p \cdot |x - y|^{p-1}
\]

where the bound follows from Lemma 2.3, 2.4, 2.5 and 2.6.

\[\square\]

We now show that \(F_{-u}\) is a close approximation of \(D\) \((D(x, y) = |x - y|^p)\).

Lemma 2.8. For integers \(x, y\) there is \(D(x, y) \cdot (1 - (2\ln 2)p/U) \leq F_{-u}(x, y) \leq D(x, y)\).
Proof. For $x = y$ the lemma trivially holds, so for the rest of the proof we will assume $x \neq y$. As $x, y$ are integers only, their smallest non-zero distance is 1. As $-u < 0$ the $|x - y| - 2^{-u} > 0$ and we bound $|x - y| \cdot (1 - 1/U) \leq \max(0, |x - y| - 2^{-u}) \leq |x - y|$. By (1) (when $p \geq 1$) or (5) (when $p \leq 1$) the claim follows.

By combining Theorem 2.7 with the Lemma 2.8 above we conclude that additive error of Algorithm 2.2 at each position is $(32p \cdot \eta + \frac{p}{U}) \cdot |x - y|^p = p(\varepsilon/4 + 1/U) \cdot |x - y|^p \leq p\varepsilon|x - y|^p$ (since w.l.o.g. $\varepsilon \geq 4/U$), thus the relative error is $(1 + p\varepsilon/2)$.

Finally, since $p \geq 1$ and w.l.o.g. $\varepsilon \leq 1/p$, by (2) and (3) $(1 + p\varepsilon/2)$ approximation of $\ell_p$ distances is enough to guarantee $(1 + \varepsilon)$ approximation of $\ell_p$ distances.

2.2 Algorithm for $0 < p \leq 1$

In this section we prove Theorem 1.2. We note that the algorithm presented in the previous section does not work, since in the proof of Lemma 2.5 and 2.6 we used the convexity of function $|t|^p$, which is no longer the case when $p < 1$.

However, we observe that Lemma 2.3 and 2.4 hold even when $0 < p \leq 1$. To combat the situation where adversarial input makes the estimates in Lemma 2.5 and 2.6 to grow too large, we use a very weak version of hashing. Specifically, we pick at random a linear function $\sigma(t) = r \cdot t$, where $r \in [1, 5]$ is a random independent variable. Such function applied to the input makes its bit sequences appear more "random" while preserving the inner structure of the problem.

Consider a following approach:

Algorithm 2.9.

1. Fix $\eta = \frac{\varepsilon p}{21606 \log^2 U \ln 2}$.
2. Pick $r \in [1, 5)$ uniformly at random.
3. Compute $T' = r \cdot T$ and $P' = r \cdot P$.
4. Use Algorithm 2.2 to compute $S'$, text-to-pattern distance between $T'$ and $P'$ using $\hat{G}_u$ distance function.
5. Output $S'' = S' \cdot r^{-1}$.

Now we analyze the expected error made by estimation from Algorithm 2.9. We denote the expected additive error of estimation of $(\ell_p)^p$ distances as

$$
err(x, y) \overset{\text{def}}{=} \mathbb{E}_{r \in [1, 5]} \left[ \left( \frac{1}{r} \right)^p \left| \hat{G}_u(rx, ry) - |rx - ry|^p \right| \right].
$$

Theorem 2.10. The procedure of Algorithm 2.9 has the expected additive error $err(x, y) \leq \frac{\varepsilon p}{3 \ln 2} |x - y|^p$.
Lemma 2.11. Let $\sum_{i=-u}^{k} g_i(rx, ry)$ be a sum of functions $g_i$ that are upper bounded by $32(\ln 2)\eta |x - y|^p$. Then:

$$\sum_{i=-u}^{k} |g_i(rx, ry)| \leq 32(\ln 2)\eta |x - y|^p,$$

Proof. Since $\eta \leq 1/32$, we have $2^{k+1} \leq 1/2 \cdot r|x - y|$:

$$\sum_{i=-u}^{k} g_i(rx, ry) \leq \left(\sum_{i=-\infty}^{k} g_i(rx, ry)\right) \leq \left|G_{k+1}(rx, ry) - D(rx, ry)\right| \leq \left((r|x - y|^p - (r|x - y| - 2^{k+1})^p\right) \leq r^p|x - y|^p \cdot 2p(\ln 2) \frac{2^{k+1}}{r|x - y|} \leq 32(\ln 2)\eta |x - y|^p.$$

Lemma 2.12. For $i \leq k = \log(8\eta|x - y|)$ we have $\sum_{i=-u}^{k} \tilde{g}_i(rx, ry) \leq (1728 + 72 \log U(\ln 2))\eta|x - y|^p$. Proof. Denote $A = \|\langle rx \rangle^{(i)} - (ry)^{i}\|_{B_i}$ and $B = \|\langle rx \rangle^{(i+1)} - (ry)^{(i+1)}\|_{B_i}$, and $A' = \max(0, A - 2^i)$, $B' = \max(0, B - 2^{i+1})$. Repeating reasoning from proof of Lemma 2.5, there is $|A' - B'| \leq 2 \cdot 2^i$, $\|\langle rx \rangle^{(i+1)}\|_{B_i} - 2 \cdot 2^i \leq A' \leq \|\langle rx \rangle^{(i+1)}\|_{B_i}$ and $\|\langle tx \rangle^{(i)}\|_{B_i} - 2 \cdot 2^{i+1} \leq B' \leq \|\langle tx \rangle^{(i)}\|_{B_i}$. We also bound $B_i = 2^i/\eta \leq 2^k/\eta = 8|x - y|$. Now let's bound $|\tilde{g}_i(rx, ry)|$. A simple bound applies

$$|\tilde{g}_i(rx, ry)| \leq \|\langle rx \rangle^{(i)}\|_{B_i}.$$

Proof. Assume that $x \neq y$, as otherwise the bound trivially follows. We bound the absolute error as follow, denoting $k = \log(8\eta|x - y|)$.

$$\text{err}(x, y) \leq \sum_{i=-u}^{k} \tilde{g}_i(rx, ry) \leq 32(\ln 2)\eta |x - y|^p.$$
If \( \|rx - ry\|_{B_i} \geq 6 \cdot 2^i \), then
\[
|\hat{g}(rx, ry)| = |(A')^p - (B')^p| = \max(A', B')^p \left( 1 - \left( 1 - \frac{|A' - B'|}{\max(A', B')} \right)^p \right)
\]
\[
\leq \|rx - ry\|_{B_i}^p \cdot \left( 1 - \left( 1 - \frac{2 \cdot 2^i}{\|rx - ry\|_{B_i} - 2 \cdot 2^i} \right)^p \right)
\]
\[
\leq \|rx - ry\|_{B_i}^p \cdot \left( 1 - \left( 1 - \frac{3 \cdot 2^i}{\|rx - ry\|_{B_i}} \right)^p \right)
\]
\[
\leq 6p(\ln 2)\|rx - ry\|_{B_i}^{p-1} \cdot 2^i
\]
(by (5)).

Consider function \( \tau(r) = \|rx - ry\|_{B_i} \) for \( 1 \leq r \leq 5 \). It is a continuous function of \( r \), taking values from \([0, B_i/2]\), and piecewise linear, increasing or decreasing with a slope \( r \), taking extreme values alternating 0 and \( B_i/2 \). Any non-extreme value is taken at least once, and for any two values number of times they are taken differs by at most 2. All in all, for any measurable subset \( X \subseteq [0, B_i/2] \), there is
\[
\Pr_{r \in [1,5]} \left[ \|rx - ry\|_{B_i} \in X \right] \leq \frac{3|X|}{B_i/2},
\]
with extreme case being: every value of \( x \in X \) has pre-image of size 3, while every other value has pre-image of size 1.

Combining both bounds:
\[
\mathbb{E}_{r \in [1,5]} |\hat{g}(rx, ry)| \leq \Pr_{r \in [1,5]} \left[ \|rx - ry\|_{B_i} \leq 6\eta B_i \right] \cdot (6 \cdot 2^i)^p + \sum_{\substack{\alpha \geq 12\eta \\ j=0,1,..}} \Pr_{r \in [1,5]} \left[ \|rx - ry\|_{B_i} \in [\alpha B_i/4, \alpha B_i/2] \right] \cdot (\alpha B_i/4)^{p-1} \cdot 2^i \cdot 6p(\ln 2).
\]

And using bounds on respective probabilities we reach:
\[
\mathbb{E}_{r \in [1,5]} |\hat{g}(rx, ry)| \leq 36\eta(6 \cdot 2^i)^p + 6p(\ln 2) \sum_{\substack{\alpha \geq 12\eta \\ j=0,1,..}} \frac{3}{2} \alpha \cdot (\alpha B_i/4)^{p-1} \cdot 2^i
\]
\[
\leq 36\eta(6 \cdot 2^i)^p + 9p(\ln 2) \sum_{\substack{\alpha \geq 12\eta \\ j=0,1,..}} \alpha \cdot \left( \frac{\alpha 2^i}{4\eta} \right)^{p-1} \cdot 2^i
\]
\[
\leq 36\eta(6 \cdot 2^i)^p + 9p(\ln 2) \sum_{\substack{\alpha \geq 12\eta \\ j=0,1,..}} \left( \frac{\alpha 2^i}{4\eta} \right)^p \cdot 4\eta
\]
\[
\leq 36\eta(48\eta|x - y|)^p + 9p(\ln 2) \sum_{\substack{\alpha \geq 12\eta \\ j=0,1,..}} (2|x - y|)^p \cdot 4\eta
\]
\[
\leq 1728\eta|x - y|^p + 72 \log U(\ln 2) \eta|x - y|^p
\]
\( (p \leq 1) \).
Finally, by combining both bounds, we get:

\[
\text{err}(x, y) \leq \frac{\varepsilon p}{6 \ln 2} |x - y|^p + \frac{\varepsilon p}{6 \ln 2} |x - y|^p
\]

\[
\leq \frac{\varepsilon p}{3 \ln 2} |x - y|^p
\]

(w.l.o.g. $\varepsilon \geq \frac{12(\ln 2)^2}{U}$)

To finish the proof of Theorem 1.2 we observe, that for any position $i$ of output, Algorithm 2.9 outputs $S''[i]$ such that $\mathbb{E}[(S''[i])^p - (S[i])^p] \leq \frac{\varepsilon p}{3 \ln 2} \cdot (S[i])^p$. By Markov’s inequality it means that with probability $2/3$ the relative error of $(\ell_p)^p$ approximation is at most $\frac{\varepsilon p}{m \cdot n} \cdot \varepsilon$. Thus, by (4) and (6) relative error of $\ell_p$ approximation is $\varepsilon$ with probability at least $2/3$. Now a standard amplification procedure follows: invoke Algorithm 2.9 independently $t$ times and take the median value from $S''[i], \ldots, S''[i]$ as the final estimate $S_\varepsilon[i]$. Taking $t = \Theta(\log n)$ to be large enough makes the final estimate good with high probability, and by the union bound whole $S_\varepsilon$ is a good estimate of $S$. The complexity of the whole procedure is thus $O(\log n \cdot \log U \cdot \eta^{-1} \cdot n \log m) = O(p^{-1} \varepsilon^{-1} n \log m \log^3 U \log n)$.

3 Hamming distances

As a final note we comment on a particularly simple form that Algorithm 2.9 takes for Hamming distances (limit case of $p = 0$).

\[
\tilde{g}_i(x, y) = \begin{cases} 
1 & \text{if } \|x(i) - y(i)\|_{B_i} = 1 \\
0 & \text{otherwise},
\end{cases}
\]

with Algorithm being simply: pick at random $r \in [1, 5]$, apply it multiplicatively to the input, compute text-to-pattern distance using $\sum_i \tilde{g}_i$ function.

Taking a limit of $p \to 0$ in proof of Theorem 1.2, we reach that bound from Lemma 2.12 becomes

\[
\mathbb{E}_{r \in [1, 5]} |\tilde{g}_i(rx, ry)| \leq 36\eta
\]

and since all other terms in error estimate have multiplicative term $p$ in front, we reach

\[
\text{err}(x, y) \leq 2 \log U \cdot \mathbb{E}_{r \in [1, 5]} |\tilde{g}_i(rx, ry)| \leq 72\eta \log U.
\]

We thus observe that expected relative error in estimation of Hamming distance is: $\mathbb{E}[S''[i] - S[i]] \leq 72\eta \log U \cdot S[i]$. With probability at least $2/3$ the relative error is at most $216\eta \log U$. Setting
\( \eta = \frac{1}{2^{10} \log^2} \) and repeating the randomized procedure \( \Theta(\log n) \) with taking median for concentration completes the algorithm. The total runtime is, by a standard trick of reducing alphabet size to \( 2m \), \( O(\frac{n}{\varepsilon} \log^2 m \log n) \), and while it compares unfavorably to algorithm from [KP18] (in terms of runtime), it gives another insight on why \( \tilde{O}(n/\varepsilon) \) time algorithm is possible for Hamming distance version of pattern matching.

References

[Abr87] Karl R. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.

[ALPU05] Amihood Amir, Ohad Lipsky, Ely Porat, and Julia Uman斯基. Approximate matching in the \( L_1 \) metric. In *CPM*, pages 91–103, 2005.

[CCI05] Peter Clifford, Raphaël Clifford, and Costas S. Iliopoulos. Faster algorithms for \( \delta, \gamma \)-matching and related problems. In *CPM*, pages 68–78, 2005.

[Cli09] Raphaël Clifford. Matrix multiplication and pattern matching under Hamming norm. [http://www.cs.bris.ac.uk/Research/Algorithms/events/BAD09/BAD09/Talks/BAD09-Hammingnotes.pdf](http://www.cs.bris.ac.uk/Research/Algorithms/events/BAD09/BAD09/Talks/BAD09-Hammingnotes.pdf), 2009. Retrieved March 2017.

[CR12] 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.

[FP74] M. J. Fischer and M. S. Paterson. String-matching and other products. Technical report, 1974.

[Gan15] Sumit Ganguly. Taylor polynomial estimator for estimating frequency moments. In *ICALP*, pages 542–553, 2015.

[GLU18] Daniel Graf, Karim Labib, and Przemyslaw Uznański. Brief announcement: Hamming distance completeness and sparse matrix multiplication. In *ICALP*, pages 109:1–109:4, 2018.

[GU18] Paweł Gawrychowski and Przemyslaw Uznański. Towards unified approximate pattern matching for hamming and \( L_1 \) distance. In *ICALP*, pages 62:1–62:13, 2018.

[JKS08] T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008.

[Kar93] Howard J. Karloff. Fast algorithms for approximately counting mismatches. *Inf. Process. Lett.*, 48(2):53–60, 1993.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA*, pages 1161–1178, 2010.

[KP15] Tsvi Kopelowitz and Ely Porat. Breaking the variance: Approximating the hamming distance in \( 1/\varepsilon \) time per alignment. In *FOCS*, pages 601–613, 2015.
[KP18] Tsivi Kopelowitz and Ely Porat. A simple algorithm for approximating the text-to-pattern hamming distance. In *SOSA*, pages 10:1–10:5, 2018.

[LP08] Ohad Lipsky and Ely Porat. $L_1$ pattern matching lower bound. *Inf. Process. Lett.*, 105(4):141–143, 2008.

[LP11] Ohad Lipsky and Ely Porat. Approximate pattern matching with the $L_1$, $L_2$ and $L_{\infty}$ metrics. *Algorithmica*, 60(2):335–348, 2011.

[LW13] Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *APPROX-RANDOM*, pages 623–638, 2013.

[MWW19] Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. Subquadratic approximation scheme for partition. In *SODA (to appear)*, 2019.

[Nol03] John Nolan. *Stable distributions: models for heavy-tailed data*. Birkhauser New York, 2003.

[PE08] Ely Porat and Klim Efremenko. Approximating general metric distances between a pattern and a text. In *SODA*, pages 419–427, 2008.

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *SODA*, pages 167–175, 2004.