Fast algorithm for two-dimensional pattern matching with k mismatches

Jonas Ellert[1], Paweł Gawrychowski[2], Adam Górkiewicz[3], and Tatiana Starikovskaya[4]

[1]?
[2]?
[3]?
[4]?

Abstract

1 Introduction

We consider the one-dimensional all-substring Hamming distance problem (HD1D), where for a given text string T of length n and a string P of length m $(m < n)$, we want to calculate the Hamming distance between P and every fragment T of length m.

We consider the two-dimensional all-substring Hamming distance problem (HD2D), where for a given 2D string T of size $n \times n$ and a string P of size $m \times m$ $(m < n)$, we want to calculate the Hamming distance between P and every $m \times m$ fragment of T.

We also consider the bounded variants of HD1D and HD2D, where we are only required to calculate the distances which are not greater than k, for some parameter $k \in \mathbb{Z}^+$.

Fact 1. *Bounded* HD1D *can be solved in* $\tilde{\mathcal{O}}((m + k\sqrt{m})n/m)$ *time.*

Theorem 1 (Main result)**.** *Bounded* HD2D *can be solved in* $\tilde{\mathcal{O}}((m^2 + mk^{5/4})n^2/m^2)$ *time.*

2 Preliminaries

Something about the model of computation?

Definition 1 (Two-dimensional string)**.** We define a **string** S as an ordered pair $(S^{\mathbf{d}}, S^{\mathbf{f}})$ where $S^{\mathbf{d}} \subseteq \mathbb{Z}^2$ is a set of two-dimensional integer vectors and $S^{\mathbf{f}} : S^{\mathbf{d}} \to \Sigma$ is a function mapping the vectors to characters. For simplicity we will sometimes write $S(u)$ to denote $S^{\mathbf{f}}(u)$ for $u \in S^{\mathbf{d}}$. We will also sometimes write $u \in S$ to denote that $u \in S^{\mathbf{d}}$. We say that a string S is **partitioned** into strings $R_1, \ldots, R_\ell$ when the sets $R_1^{\mathbf{d}}, \ldots, R_\ell^{\mathbf{d}}$ partition $S^{\mathbf{d}}$. We call a string S **monochromatic** if $S^{\mathbf{f}}[S^{\mathbf{d}}] = \{\, \alpha \,\}$ for some $\alpha \in \Sigma$. We say that S is $n \times m$ for some integers $n, m > 0$ when $S^{\mathbf{d}} = \{\, 0, \ldots, n-1 \,\} \times \{\, 0, \ldots, m-1 \,\}$.

Remark. Note that we do not associate a two-dimensional string with a two-dimensional array of characters. Although we will occasionally use the array notation, we do it exclusively for $n \times m$ strings and the general definition is much broader.

Definition 2 (Shifting)**.** For a set of vectors V and a vector u, we denote $V + u$ as $\{\, v + u : v \in V \,\}$. For a string S and a vector u, we denote $S + u$ as a string R such that $R^{\mathbf{d}} = S^{\mathbf{d}} + u$ and $R^{\mathbf{f}}(v) = S^{\mathbf{f}}(v - u)$ for $v \in R^{\mathbf{d}}$. Intuitively, we shift the set of vectors while maintaining their character values.

Definition 3 (Hamming distance)**.** Consider two strings S, R. We define

$$\mathrm{Ham}(S, R) = |\{ u : u \in S, u \in R, S(u) \neq R(u) \}|.$$

Remark. Under such notation, the HD2D problem is equivalent to calculating the (bounded or unbounded) values of $\mathrm{Ham}(P + q, T)$ for all $q \in \mathbb{Z}^2$ such that $(P + q)^{\mathbf{d}} \subseteq T^{\mathbf{d}}$ (so for $q \in \{ 0, \ldots, n - m \}^2$).

Definition 4 (Dontcare symbol)**.** We define the **dontcare** symbol as a special character which matches with every character. We will denote it with ?. Unless stated otherwise, we assume it is not allowed in Σ and in both HD1D and HD2D every character present in T and P matches only with itself.

Definition 5 (Vector operators)**.** For a vector $u \in \mathbb{Z}^2$ we refer to its coordinates as $u.x, u.y$. For $u, v \in \mathbb{Z}^2$ we denote $u \cdot v = u.x \cdot v.x + u.y \cdot v.y$ and $u \times v = u.x \cdot v.y - u.y \cdot v.x$. Note that alternatively $u \cdot v = |u||v| \cos \alpha$ and $u \times v = |u||v| \sin \alpha$ where α is the angle between u and v.

Definition 6 (Quadrants)**.** We define the four **quadrants** as

$$\begin{aligned}
\mathcal{Q}_1 &= (0, +\infty) \times [0, +\infty),\\
\mathcal{Q}_2 &= (-\infty, 0] \times (0, +\infty),\\
\mathcal{Q}_3 &= (-\infty, 0) \times (-\infty, 0],\\
\mathcal{Q}_4 &= [0, +\infty) \times (-\infty, 0).
\end{aligned}$$

3 One-dimensional generalizations

In this section we explore some of the methods used for one-dimensional strings. Specifically, as our goal is to generalize the solution for pattern matching with k mismatches described in [2], we are especially interested in two-dimensional variants of the techniques that were used to solve the one-dimensional case.

Theorem 2 (Instancing)**.** *Consider an algorithm $\mathcal{A}$ which solves* HD2D *(bounded or unbounded), but only when $2|n$ and $n \leq \frac{3}{2}m$. If its running time is $\mathcal{T}(m)$, then there exists an algorithm which solves the general case in $\tilde{\mathcal{O}}(\mathcal{T}(m)n^2/m^2)$.*

Proof. Let $r = \lfloor m/2 \rfloor$ and let $n' = r + m - 1$ or $r + m$ if $r + m - 1$ is odd. For any query vector q consider a vector u such that $r|u.x, r|u.y$ and $q - u \in \{ 0, \ldots, r - 1 \}^2$. We have $\mathrm{Ham}(P + q, T) = \mathrm{Ham}(P + q - u, T_u)$ where $T_u^{\mathbf{d}} = \{ 0, \ldots, n' - 1 \}^2$, $T_u^{\mathbf{f}} = (T - u)^{\mathbf{f}}$. If $T_u^{\mathbf{f}}$ is not defined for some $v \in T_u^{\mathbf{d}}$, we can "pad" it with any symbol. We then have $(P + q - u)^{\mathbf{d}} \subseteq T_u^{\mathbf{d}}$. There are $\mathcal{O}(n^2/m^2)$ possible vectors u and we run $\mathcal{A}$ for every pair of T_u and P. □

Theorem 3 (Kangaroo jumps)**.** *Consider an $n \times n$ string T, $m \times m$ string P and set of vectors Q such that $(P + q)^{\mathbf{d}} \subseteq T^{\mathbf{d}}$ for every $q \in Q$. There exists an algorithm which calculates $d_q = \mathrm{Ham}(P + q, T)$ for every $q \in Q$ in total time $\tilde{\mathcal{O}}(n^2 + \sum_{q \in Q} d_q)$.*

Proof. For the sake of clarity of this proof, we will temporarily switch to the classical array notation for strings. Let $T_0, \ldots, T_{n-m}$ denote an array of two-dimensional strings (arrays) such that $T_k[0 \mathinner{..} n - 1, 0 \mathinner{..} m - 1] = T[0 \mathinner{..} n - 1, k \mathinner{..} k + m - 1]$. For every row $P[0], \ldots, P[m - 1]$ of P and every row $T_k[0], \ldots, T_k[n - 1]$ of every T_k we assign an integer identifier so that $\mathrm{Id}(P[i]) = \mathrm{Id}(T_k[j]) \Leftrightarrow P[i] = T_k[j]$ using KMR algorithm ([reference]) in $\tilde{\mathcal{O}}(n^2)$.

We use the approach described in [kangaroo reference]. There exists a data structure (suffix array) which for a given one-dimensional array S allows us to detect all mismatches between any given two of its subarrays of equal length. It can be built in $\tilde{\mathcal{O}}(|S|)$ and the query time is $\tilde{\mathcal{O}}(d)$ where d is the number of mismatches. We construct the suffix array for the concatenation of the following arrays:

- the rows $P[i]$ for every i,
- the rows $T[i]$ for every i,
- the array $\mathrm{Id}(P[0])\,\mathrm{Id}(P[1])\ldots\mathrm{Id}(P[m-1])$,
- the arrays $\mathrm{Id}(T_k[0])\,\mathrm{Id}(T_k[1])\ldots\mathrm{Id}(T_k[n-1])$ for every k,

the total length of which is $\mathcal{O}(n^2)$. Let us consider a problem of detecting mismatches between P and some $T' = T[j\,..\,j+m-1, k\,..\,k+m-1]$. We can firstly detect all such i for which $P[i] \neq T'[i]$ by querying the subarrays $\mathrm{Id}(P[0])\ldots\mathrm{Id}(P[m-1])$ and $\mathrm{Id}(T_k[j])\ldots\mathrm{Id}(T_k[j+m-1])$. For every such $P[i] \neq T'[i]$ we can then find all mismatches by querying $P[i]$ and $T[i+j][k\,..\,k+m-1]$. $\square$

Fact 2. *There exists an algorithm which solves* HD1D *in $\tilde{\mathcal{O}}(n|\Sigma|)$ time. It works by simply running $|\Sigma|$ instances of FFT. It allows dontcare symbols in T and P.*

Fact 3. *There exists a $(1+\varepsilon)$-approximate algorithm which solves* HD1D *in $\tilde{\mathcal{O}}(n)$ time. It was introduced in [3]. It allows dontcare symbols in T and P.*

Theorem 4. *There exists an algorithm which solves* HD2D *in $\tilde{\mathcal{O}}(n^2|\Sigma|)$ time. It allows dontcare symbols in T and P.*

Proof. We will again use the array notation. We construct one-dimensional strings $\bar{T}$ and $\bar{P}$ by concatenating subsequent rows $T[0], \ldots, T[n-1]$ of T and rows $P[0], \ldots, P[m-1]$ of P padded with some dontcare symbols:

$$\bar{T} = T[0]\ ?^m\ T[1]\ ?^m\ \ldots\ ?^m\ T[n-1]\ ?^m,$$
$$\bar{P} = P[0]\ ?^n\ P[1]\ ?^n\ \ldots\ ?^n\ P[m-1]\ ?^n.$$

We run the algorithm from Fact 2. The distance between $T[i\,..\,i+m-1, j\,..\,j+m-1]$ and P is equal to the distance between $\bar{T}[i(n+m)+j\,..\,(i+m)(n+m)+j-1]$ and $\bar{P}$ for every i, j. $\square$

Theorem 5. *There exists a $(1+\varepsilon)$-algorithm which solves* HD2D *in $\tilde{\mathcal{O}}(n^2)$ time. It allows dontcare symbols in T and P.*

Proof. Identical to Theorem 4., but we use the algorithm from Fact 3. instead of 2. $\square$

Remark. The same reduction as in Theorem 4. can be applied for every HD1D solution which allows dontcare symbols. Unfortunately, the most effective known algorithms for bounded HD1D rely on periodicity ([1], [2]) and inherently do not allow dontcare symbols, thus, they cannot be easily generalized.

Observation 1 (Dontcare padding)**.** *Every* HD2D *solution which allows dontcare symbols (eg. the algorithms from Theorem 4. and 5.) can be extended to also calculate the Hamming distance for occurrences of P which are not entirely contained in T. It can be done by padding the text with dontcare symbols and it does not change the complexity of the solution.*

4 Proof of Theorem 1.

We show an algorithm which works in time $\tilde{\mathcal{O}}(m^2 + mk^{5/4})$ assuming $2|n$ and $m < n \leq \frac{3}{2}m$. By Theorem 2., our thesis will follow.

We start by running the algorithm from Theorem 5. with $\varepsilon = 1$. We construct the set Q as the set of such vectors $q \in \mathbb{Z}^2$ for which the estimated value of $\mathrm{Ham}(P+q, T)$ is at most $2k$. For every $q \in \{0, \ldots, n-m\}^2 \setminus Q$ we say that $\mathrm{Ham}(P+q, T)$ equals ∞. The next step is to calculate the exact value of $\mathrm{Ham}(P+q, T)$ for every $q \in Q$.

Let us consider the case when $|Q| \le 2m + m^2/k$. We can run the algorithm from Theorem 3. and by the fact that $\mathrm{Ham}(P+q,T) \le 4k$ for every $q \in Q$, it will perform $\tilde{\mathcal{O}}(m^2+mk)$ operations. We are left with the case when $|Q| > 2m + m^2/k$, in which we take advantage of the fact that some strings $P+q$ for $q \in Q$ must have a large overlap and small Hamming distance from each other, and thus P must be periodic.

4.1 Two-dimensional periodicity

In this section we introduce a range of new tools related to two-dimensional periodicity. We then select some special periods of the pattern and show how to decompose it into some regularly structured monochromatic strings.

Definition 7 (Periodicity)**.** Consider any vector $\delta \in \mathbb{Z}^2$. We say that a string S has an ℓ-period δ when

$$\mathrm{Ham}(S+\delta, S) \le \ell.$$

Lemma 1. *For every $u, v \in Q$, the vector $u - v$ is an $8k$-period of P.*

Proof. $\mathrm{Ham}(P+u-v,P) = \mathrm{Ham}(P+u,P+v) \le \mathrm{Ham}(P+u,T)+\mathrm{Ham}(P+v,T) \le 4k+4k$. □

Theorem 6. *For an integer $\ell > 0$ and a set of vectors $U \subseteq \{0,\dots,\ell\}^2$ such that $|U| > 4\ell$ there exist $s,t,s',t' \in U$ such that $w = t - s$ and $w' = t' - s'$ hold the following conditions:*

- $0 < |w||w'| \le 22\frac{\ell^2}{|U|}$,
- $|\sin\alpha| \ge \frac{1}{2}$ *where α is the angle between w and w',*
- *$w, w', -w, -w'$ are all contained in different quadrants.*

There exists an algorithm which finds such w, w' in $\tilde{\mathcal{O}}(|U|)$ operations.

We run Algorithm 6. on the set Q (where $\ell = n - m \le m/2$, thus $|Q| > 2m + m^2/k \ge 4\ell$). We obtain vectors $\varphi \in \mathcal{Q}_4$ and $\psi \in \mathcal{Q}_1$ which by Lemma 1. are $\mathcal{O}(k)$-periods of P. We use them as constants throughout the rest of the description along with $p = \varphi \times \psi$. Note that because $|Q| > m + m^2/k$, we have $p \le |\varphi||\psi| = \mathcal{O}(\min\{m,k\})$.

Definition 8 (Lattice congruency)**.** We say that two vectors $u, v \in \mathbb{Z}^2$ are **lattice congruent** and denote $u \equiv v$ when there exist $s, t \in \mathbb{Z}$ such that $u - v = s\varphi + t\psi$.

Lemma 2 (Lattice base)**.** *There exists an array $\gamma_1,\dots,\gamma_p \in \mathbb{Z}^2$ such that $\gamma_i \not\equiv \gamma_j$ for $i \ne j$ and for every $u \in \mathbb{Z}^2$ there exists γ_i such that $u \equiv \gamma_i$.*

Definition 9 (Parquet)**.** Consider a set $U \subseteq \mathbb{Z}^2$. We call U a **parquet** if there exist some values (restrictions) $x_0, x_1, y_0, y_1, \varphi_0, \varphi_1, \psi_0, \psi_1 \in \mathbb{Z}$ such that

$$U = \{\, u : u \in \mathbb{Z}^2, x_0 < u.x \le x_1, y_0 < u.y \le y_1, \varphi_0 < \varphi \times u \le \varphi_1, \psi_0 < \psi \times u \le \psi_1 \,\}.$$

If some existing restrictions hold additional conditions, we classify U as a special case of parquet:

a) if $x_1 - x_0 \ge |\varphi.x| + |\psi.x|$ and $y_1 - y_0 \ge |\varphi.y| + |\psi.y|$, then we call U a **spacious** parquet,

b) if $x_0, y_0 = -\infty$ and $x_1, y_1 = +\infty$, then we call U a **simple** parquet,

c) if $x_0, y_0, \varphi_0, \psi_0 = -\infty$ and $x_1, y_1 = +\infty$, then we call U a **primitive** parquet.

Note that every primitive parquet is simple and every simple parquet is spacious.

Definition 10 (Subparquet)**.** Consider a set $V \subseteq \mathbb{Z}^2$. We call V a **subparquet** if there exist a parquet U and a vector $\gamma \in \mathbb{Z}^2$ such that

$$V = \{\, u : u \in U, u \equiv \gamma \,\}\,.$$

We call V a spacious/simple/primitive subparquet when there exists U which is (correspondingly) a spacious/simple/primitive parquet. We will abuse the notation and for non-empty V write $u \equiv V$ to (unambiguously) denote that $u \equiv \gamma$ for some vector $u \in \mathbb{Z}^2$.

Definition 11 (Parquet string)**.** For a string S, if $S^{\mathbf{d}}$ is a spacious/simple (sub-)parquet, then we call S a spacious/simple (sub-)parquet string.

Theorem 7 (Periodic parquet decomposition)**.** *Consider a spacious/simple parquet string R with $\mathcal{O}(k)$-periods φ and ψ. It can be partitioned into $\mathcal{O}(k)$ monochromatic spacious/simple subparquet strings, correspondingly. There exists an algorithm which finds this partitioning in $\tilde{\mathcal{O}}(|R^{\mathbf{d}}|)$ operations.*

Since $|\varphi.x|, |\varphi.y|, |\psi.x|, |\psi.y| \leq n - m \leq m/2$, the $m \times m$ string P is a spacious parquet string and satisfies the assumptions of Theorem 7. We partition P into a set of monochromatic spacious parquet strings $\mathcal{V}$, where $|\mathcal{V}| = \mathcal{O}(k)$. Note that because the text is not necessarily periodic, we unfortunately cannot use the same approach for T.

4.2 Text decomposition

In this section we show how to decompose the text using a similar but more nuanced approach. We then introduce an effective way to aggregate the contributions of every pair of strings that P and T are decomposed into.

Definition 12 (Active text)**.** Consider a set $U = \bigcup_{q \in Q} P^{\mathbf{d}} + q$. We define strings $T_a = (U, T^{\mathbf{f}})$ and $T_b = (T^{\mathbf{d}} \setminus U, T^{\mathbf{f}})$. We will call T_a the **active text** and T_b the **inactive text**. For every $u \in \mathbb{Z}^2$ we define its **border distance** as

$$\min\{\, \|u - v\|_\infty : v \in (\mathbb{Z}^2 \setminus U) \,\}\,.$$

Observation 2. $\mathrm{Ham}(P + q, T) = \mathrm{Ham}(P + q, T_a)$ *for every $q \in Q$.*

Theorem 8 (Active text decomposition)**.** *There exists an algorithm which for any $\ell = \mathcal{O}(m)$ partitions T_a into a set of monochromatic simple subparquet strings $\mathcal{U}$ and a string F, such that $|\mathcal{U}| = \mathcal{O}(\min\{\, m^2, \ell k \,\})$ and for every $u \in F$ its border distance is $\mathcal{O}(m/\ell)$. It does so in $\tilde{\mathcal{O}}(m^2)$ operations.*

Theorem 9 (Sparse Hamming)**.** *Consider a set of monochromatic simple subparquet strings $\mathcal{U}$, a set of monochromatic subparquet strings $\mathcal{V}$ and a set of vectors Q. There exists an algorithm which calculates*

$$\sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U + q, V)$$

for every $q \in Q$ in time $\tilde{\mathcal{O}}(\ell^2 + |\mathcal{U}||\mathcal{V}| + |Q|)$ assuming that all strings are defined for vectors with coordinate values from $\{\, 0, \ldots, \ell \,\}$ for some $\ell \in \mathbb{Z}^+$.

Theorem 10 (Dense Hamming)**.** *Consider a string F such that for every $u \in F$ its border distance is less than ℓ for some $\ell \in \mathbb{Z}^+$. There exists an algorithm which calculates* $\mathrm{Ham}(P + q, F)$ *for every $q \in Q$ in total time $\tilde{\mathcal{O}}(m^2 + m\ell k^{1/2})$.*

We partition T_a using the algorithm from Theorem 8. with $\ell = mk^{-3/4}$ into a set of simple subparquet strings $\mathcal{U}$ and a string F. For every $q \in Q$ we then have

$$\mathrm{Ham}(P + q, T_a) = \mathrm{Ham}(P + q, F) + \sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U - q, V)$$

which we calculate by summing the results of algorithms from Theorem 10. and Theorem 9.

4.3 Proof of Theorem 6.

Firstly, we find any closest pair of vectors $s, t \in U$ by running the standard $\tilde{\mathcal{O}}(|U|)$ time algorithm and denote $w = t - s$. We define a partial order $\leq_w$ where $v \leq_w u$ for some $u, v \in U$ when at least one condition holds:

(a) $u = v$,

(b) $u - v$ and w belong to the same quadrant,

(c) $\alpha \in (-\frac{\pi}{6}, \frac{\pi}{6})$ where α is the angle between w and $u - v$.

We find the longest chain C and the longest antichain A using dynamic programming in $\tilde{\mathcal{O}}(|U|)$ operations. We then find any closest pair of vectors $s', t' \in A$ and denote $w' = t' - s'$. We have the following inequalities:

(i) $|U| \leq |C||A|$ (by Dilworth's theorem),

(ii) $(|C| - 1)|w| \leq (1 + \sqrt{3})\ell$ (roughly by the fact that vectors in C must be increasing in a certain direction),

(iii) $(|A| - 1)|w'| \leq 2\ell$ (by using a similar argument for vectors in A).

By considering the assumption $|U| > 4\ell$ it can be proven that $|w||w'| \leq 22\frac{\ell^2}{|U|}$ and the other conditions also hold.

4.4 Proof of Theorem 7.

Definition 13 (Lattice graph)**.** For a set $U \subseteq \mathbb{Z}^2$ we define its **lattice graph** $G_U = (U, E_U)$ where

$$E_U = \left\{ \{ u, u + \delta \} : \delta \in \{ \varphi, \psi \}, u \in U, u + \delta \in U \right\}$$

so every vector is connected with its translations by $\varphi, \psi, -\varphi, -\psi$.

Lemma 3. *If U is a spacious subparquet, then G_U is connected.*

Firstly, we partition R into a set of subparquet strings $\mathcal{S}$. For every non-empty $S \in \mathcal{S}$ we consider a lattice graph $G_{S^{\mathbf{d}}}$. If S is not monochromatic, then since $G_{S^{\mathbf{d}}}$ is connected, there must exist a pair of neighboring vectors v, w such that $S(v) \neq S(w)$. We select any such pair and partition S into spacious (or simple if S is simple) subparquet strings S' and S'' such that $v \in S'$ and $w \in S''$. For example if $v = w + \varphi$, then $S' = \{ u : u \in S, \psi \times u \leq \psi \times v \}$ and $S'' = \{ u : u \in S, \psi \times u > \psi \times v \}$. In the cases when $v = w + \delta$ for $\delta \in \{ -\varphi, \psi, -\psi \}$ the construction in similar.

We can recursively partition S' and S'' further until we obtain monochromatic strings. Because R has $\mathcal{O}(k)$-periods φ and ψ, the total number of neighbor pairs v, w such that $S(v) \neq S(w)$ is $\mathcal{O}(k)$ throughout all $S \in \mathcal{S}$. Thus the total number of recursive calls is $\mathcal{O}(k)$ and because $|\mathcal{S}| = \mathcal{O}(k)$, the total number of constructed strings is $\mathcal{O}(k)$. The algorithm can be implemented to work in time $\tilde{\mathcal{O}}(|R^{\mathbf{d}}|)$.

4.5 Proof of Theorem 8.

We assume ℓ to be an even number smaller than $\frac{n}{4}$ (if it is not, we can find $\ell' = \Theta(\ell)$, which is). We start by partitioning $T^{\mathbf{d}}$ into **tiles**. We define $\varphi_{\min} = \min \{ \varphi \times u : u \in T^{\mathbf{d}} \}$, analogously $\varphi_{\max}, \psi_{\min}, \psi_{\max}$ and denote $\delta_\varphi = \frac{\varphi_{\max} - \varphi_{\min}}{\ell}$, $\delta_\psi = \frac{\psi_{\max} - \psi_{\min}}{\ell}$. We define a tile with integer coordinates (s, t) as a set of vectors $u \in \mathbb{Z}^2$ such that

$$\varphi_{\min} + s\delta_\varphi < \varphi \times u \leq \varphi_{\min} + (s + 1)\delta_\varphi,$$

$$\psi_{\min} + t\delta_\psi < \psi \times u \le \psi_{\min} + (t+1)\delta_\psi.$$

For a fixed tile U consider $x_{\min} = \min \{ u.x : u \in U \}$, analogously $x_{\max}, y_{\min}, y_{\max}$ and a set

$$R = \{ u : u \in \mathbb{Z}^2, x_{\min} \le u.x \le x_{\max}, y_{\min} \le u.y \le y_{\max} \} .$$

We classify U into one of three types:

a) if $U \cap T_a = \emptyset$ then U is an inactive tile,

b) if $U \cap T_a \neq \emptyset$, $R \not\subseteq T_a$ then U a border tile,

c) if $U \cap T_a \neq \emptyset$, $R \subseteq T_a$ then U is an active tile.

We define B as a set of all $u \in T_a$ contained in a border tile and construct $F = (B, T^{\mathbf{f}})$. Let us denote $z = \frac{n-1}{2}$. Consider a family of sets $\mathcal{R} = \{ R_i^1 \} \cup \{ R_i^2 \} \cup \{ R_i^3 \} \cup \{ R_i^4 \}$, where for every active tile U with coordinates (s, t) its members are placed into exactly one subset:

1) R_t^1 if $y_{\min} > z$, $x_{\max} \ge z$,

2) R_s^2 if $x_{\max} < z$, $y_{\max} \ge z$,

3) R_t^3 if $y_{\max} < z$, $x_{\min} \le z$,

4) R_s^4 if $x_{\min} > z$, $y_{\min} \le z$.

The number of non-empty sets $R \in \mathcal{R}$ is $\mathcal{O}(\ell)$. For each of them we consider $S = (R, T^{\mathbf{f}})$ which is a simple parquet string with $\mathcal{O}(k)$-periods φ and ψ, and we further partition it using algorithm from Theorem 7., thus constructing the set $\mathcal{U}$.

4.6 Proof of Theorem 9.

Firstly, we assume that all strings in $\mathcal{U}$ nor $\mathcal{V}$ are non-empty. For $U \in \mathcal{U}$, $V \in \mathcal{V}$, the value $\mathrm{Ham}(U + q, V)$ either equals $|(U^{\mathbf{d}} + q) \cap V^{\mathbf{d}}|$ if $U[U^{\mathbf{d}}] \neq V[V^{\mathbf{d}}]$ or 0 otherwise. We have

$$\sum_{U \in \mathcal{U}} \sum_{V \in \mathcal{V}} \mathrm{Ham}(U + q, V) = \sum_{(A,B) \in \mathcal{F}} |(A + q) \cap B|$$

where $\mathcal{F} = \{ (U^{\mathbf{d}}, V^{\mathbf{d}}) : U \in \mathcal{U}, V \in \mathcal{V}, U[U^{\mathbf{d}}] \neq V[V^{\mathbf{d}}] \}$. For every $(A, B) \in \mathcal{F}$ we can find primitive subparquets $A_1, \ldots, A_4$ such that for every q we have

$$|(A + q) \cap B| = |(A_1 + q) \cap B| - |(A_2 + q) \cap B| - |(A_3 + q) \cap B| + |(A_4 + q) \cap B|$$

thus we will consider four instances of a problem of calculating $\sum_{(A,B) \in \mathcal{F}'} |(A + q) \cap B|$ where A is a primitive subparquet and B is a subparquet for all $(A, B) \in \mathcal{F}'$.

We will write $u \le_{\varphi\psi} v$ to denote that $\varphi \times u \le \varphi \times v \wedge \psi \times u \le \psi \times v$ for some $u, v \in \mathbb{Z}^2$.

Theorem 11. *There exists a data structure which for a given set of vectors U and a set of parquets $\mathcal{S}$ calculates*

$$\sum_{v \in V} |\{ S : S \in \mathcal{S}, v \in S \}|$$

for a given query vector q where $V = \{ v : v \in U, v \le_{\varphi\psi} q \}$. It requires $\tilde{\mathcal{O}}(|U| + |\mathcal{S}|)$ preprocessing time and $\tilde{\mathcal{O}}(1)$ query time.

We consider the array $\gamma_1, \ldots, \gamma_p$ introduced in Lemma 2. We consider an array of data structures $J_1, \ldots, J_p$ described in Theorem 11. We construct J_i for a set of points $U_i = \{ u : u \in \mathbb{Z}^2, | u.x| \le \ell, |u.y| \le \ell, u \equiv \gamma_i \}$ and set of parquets $\mathcal{S}_i$. To construct $\mathcal{S}_i$ we consider every pair $(A, B) \in \mathcal{F}'$ and find a vector w and a parquet V such that

$$A = \{ u : u \le_{\varphi\psi} w, u \equiv w \},$$

$$B = \{ u : u \in (V + w), u \equiv B \}.$$

The set $\mathcal{S}_i$ contains the parquets V obtained for pairs (A, B) such that $B - w \equiv \gamma_i$.

For a query vector $q \in \mathbb{Z}^2$ we can obtain the value of $\sum_{(A,B)\in\mathcal{F}'} |(A + q) \cap B|$ by finding $i \in \{ 1, \ldots, p \}$ such that $\gamma_i \equiv q$ and making a query to J_i with vector q. For explanation, if $A + q \not\equiv B$, then $(A + q) \cap B = \emptyset$. Otherwise $q \equiv B - w \equiv \gamma_i$ and we have

$$(A + q) \cap B = \{ u : u \le_{\varphi\psi} w + q, u \in (V + w), u \equiv B \} = \{ v : v \le_{\varphi\psi} q, v \in V, v \equiv \gamma_i \}.$$

4.7 Proof of Theorem 10.

To be done.

References

[1] Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. *CoRR*, abs/1508.00731, 2015.

[2] Pawel Gawrychowski and Przemyslaw Uznanski. Optimal trade-offs for pattern matching with k mismatches. *CoRR*, abs/1704.01311, 2017.

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