Theoretical Computer Science 412 (2011) 6359–6370

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

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

On-line approximate string matching with bounded errors☆

Marcos Kiwi [a], Gonzalo Navarro [b], Claudio Telha [c,*]

[a] Departamento de Ingeniería Matemática & Centro de Modelamiento Matemático UMI 2807 CNRS-UChile, University of Chile, Chile

[b] Department of Computer Science, University of Chile, Chile

[c] Operations Research Center, MIT, United States

ARTICLE INFO

Article history:
Received 27 March 2010
Received in revised form 4 February 2011
Accepted 2 August 2011
Communicated by A. Apostolico

Keywords:
Algorithms
String-matching problem

ABSTRACT

We introduce a new dimension to the widely studied on-line approximate string matching problem, by introducing an *error threshold* parameter ϵ so that the algorithm is allowed to miss occurrences with probability ϵ. This is particularly appropriate for this problem, as approximate searching is used to model many cases where exact answers are not mandatory. We show that the relaxed version of the problem allows us breaking the average-case optimal lower bound of the classical problem, achieving average case $O(n \log_\sigma m/m)$ time with any $\epsilon = \text{poly}(k/m)$, where n is the text size, m the pattern length, k the number of differences for edit distance, and σ the alphabet size. Our experimental results show the practicality of this novel and promising research direction. Finally, we extend the proposed approach to the multiple approximate string matching setting, where the approximate occurrence of r patterns are simultaneously sought. Again, we can break the average-case optimal lower bound of the classical problem, achieving average case $O(n \log_\sigma (rm)/m)$ time with any $\epsilon = \text{poly}(k/m)$.

1. Introduction

In string matching, one is interested in determining the positions (sometimes just deciding the occurrence) of a given pattern P on a text T, where both pattern and text are strings over some fixed finite alphabet Σ of size σ. The lengths of P and T are typically denoted by m and n respectively. In approximate string matching there is also a notion of distance between strings, given say by $d : \Sigma^* \times \Sigma^* \to \mathbf{R}$. One is given an additional non-negative input parameter k and is interested in listing all initial or final positions (or just deciding the occurrence) of substrings S of T such that S and P are at distance at most k. In the "on-line" or "sequential" version of the problem, one is not allowed to preprocess the text.

Since the 60s several approaches were proposed for addressing the approximate matching problem, see for example the survey by Navarro [11]. Most of the work focused on the *edit* or *Levenshtein* distance d, which counts the number of *differences* between two strings, that is, the number of character insertions, deletions, and substitutions needed to make the strings equal. This distance turns out to be sufficiently powerful to model many relevant applications (e.g., text searching, information retrieval, computational biology, transmission over noisy channels, etc.), and at the same time sufficiently simple to admit efficient solutions (e.g., $O(mn)$ and even $O(kn)$ time).

A lower bound to the (worst-case) problem complexity is obviously $\Omega(n)$ for the meaningful cases, $k < m$. This bound can be reached by using automata, which introduce an extra additive term in the time complexity which is exponential

☆ A preliminary extended abstract of this paper appeared in Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching CPM'08 (Kiwi et al., 2008) [10].

* Corresponding author. Tel.: +1 6173886407.
E-mail addresses: gnavarro@dcc.uchile.cl (G. Navarro), ctelha@mit.edu (C. Telha).
URL: http://www.dim.uchile.cl/~mkiwi (M. Kiwi).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.08.005

in m or k. If one is restricted to polynomially-bounded time complexities on m and k, however, the worst-case problem complexity is unknown.

Interestingly, the average-case complexity of the problem is well understood. If the characters in P and T are chosen uniformly and independently, the average problem complexity is $\Theta(n(k + \log_\sigma m)/m)$ time. This was proved in 1994 by Chang and Marr [4], who gave an algorithm reaching the lower bound for $k/m < 1/3 - O(\sigma^{-1/2})$. In 2004, Fredriksson and Navarro [7] gave an improved algorithm achieving the lower bound for $k/m < 1/2 - O(\sigma^{-1/2})$. In addition to covering the range of interesting k values for virtually all applications, the algorithm was shown to be highly practical.

It would seem that, except for determining the worst-case problem complexity (which is mainly of theoretical interest), the on-line approximate string matching problem is closed. In this paper, however, we reopen the problem under a relaxed scenario that is still useful for most applications and admits solutions that beat the lower bound. More precisely, we relax the goal of listing *all* positions where pattern P occurs in the text T to that of listing each such position with probability $1-\epsilon$, where ϵ is a new input parameter.

There are several relevant scenarios where fast algorithms that err (with a user-controlled probability) are appropriate. Obvious cases are those where approximate string matching is used to increase recall when searching data that is intrinsically error-prone. Consider for example an optical character recognition application, where errors will inevitably arise from inaccurate scanning or printing imperfections, or a handwriting recognition application, or a search on a text with typos and misspells. In those cases, there is no hope to find exactly all the correct occurrences of a word. Here, uncertainty of the input translates into approximate pattern matching and approximate searching is used to increase the chance of finding relevant occurrences, hopefully without introducing too many false matches. As the output of the system is an approximation to the ideal answer using a correct approximate string matching technique is a perfectly tolerable second approximation, and even welcome if it allows for faster searches.

A less obvious application arises in contexts where we might have a priori knowledge that some pattern is either approximately present in the text many times, or does not even approximately occur. Some examples are genetic markers that might often appear or not at all, some typical pattern variants that might appear in the denomination of certain drugs, people names, or places, of which typically several instances occur in the same text. Further, we might only be interested in determining whether the pattern occurs or not. (A feature actually available in the well known `grep` string searching utility as options `-l` and `-L`, and also the approximate string searching utilities `agrep` and `ngrep`.) In this context, a text with N approximate pattern occurrences will be misclassified by the inexact algorithm with very low probability, ϵ^N (if failing to report different occurrences are independent events).

Another interesting scenario is that of processing data streams which flow so fast that there is no hope for scanning them exhaustively (e.g. radar derived meteorological data, browser clicks, user queries, IP traffic logs, peer-to-peer downloads, financial data, etc.). Hence even an exact approximate search over part of the data would give only partial results. A faster inexact algorithm could even give better quality answers as it could scan a larger portion of the data, even if making mistakes on it with controlled probability.

1.1. Related work

Approximate string matching is a central problem in many areas such as information retrieval, computational biology, transmission over noisy channels, etc. Each application uses a different model to define how different two strings are. This leads to the notion of "distance" between strings — the idea is that the distance should be small if one of the strings is likely to be an erroneous variant of the other one in the model under consideration. The algorithmic nature of the problem depends strongly on the distance function considered, and the solutions range from linear time to NP-hard. The scope of the area is broad, moreover its relevance is undisputed. It is thus not surprising that it has received much attention by researchers. In this short contribution we cannot make true justice to all who have contributed to the study of the area. We refer the interested reader to the extensive survey of the subject by Navarro [11].

The concrete proposals of d-Approximate String Matching algorithms with errors that we introduce in this work are based in the so called filtering algorithms [11, Section 8]. This latter technique was introduced in the 90s by Tarhio and Ukkonen [15]. It consists of algorithms that quickly discard areas of the text that cannot approximately match the pattern. Here quickly means carrying out $O(n)$ or even less comparisons on average. Filtering algorithms have been proposed with optimal average cost $O(n(k + \log_\sigma m)/m)$. In practice, filters are also the fastest approximate string matching algorithms. Since the main objective of this work is to introduce a new model for approximate string matching for which it is possible to beat the complexity of the fastest currently known algorithms for the problem, it is natural to focus attention and adapt ideas from filtering type algorithms to the new setting we introduce. Specifically, the algorithmic proposals we put forth are particularly inspired in the filtering algorithms of Tarhio and Ukkonen [15], Ukkonen [16], Chang and Lawler [2], and Chang and Marr [4].

The main motivation for the new framework proposed in this work comes from the so called testing and property testing literature, where the aim is to devise sublinear time algorithms obtained by avoiding having to read all of the input of a problem instance. These procedures typically use randomization, reading a very small fraction of the input while providing answers which in some sense are approximate, or wrong with some probability. The testing paradigm was pioneered by Blum and his collaborators as an alternative approach to classical program verification techniques. A seminal and influential paper in this area is due to Blum et al. [1]. The new framework quickly found many applications, both of a theoretical

and practical nature. The theory was later extended to what has become also known as sublinear time algorithms. The interested reader is referred to the excellent surveys concerning these topics of Fischer [6], Goldreich [8], Ron [13], and Rubinfeld [14].

1.2. Main contributions

The contributions of our work are both conceptual and technical in nature. Concerning the former aspect, we add a new dimension to an important category of information retrieval problems by relaxing the typical goal of finding every occurrence of the object sought, and instead allowing each occurrence to be missed with a small probability. We establish both the plausibility of our proposal as well as its practicality.

Concerning the technical aspects of our work, we show how to break the average-case optimal lower bound for the classical approximate string matching problem. In Section 3.1 we describe a procedure based on sampling q-grams motivated by the filtering algorithm of Ukkonen [16]. For a fixed constant $t > 0$ and $k < m/\log_\sigma m$, the derived algorithm has an average case complexity of $O(tn \log_\sigma m/m)$ and misses pattern occurrences with probability $\epsilon = O((k \log_\sigma m/m)^t)$. Note that the time equals Yao's lower bound for *exact* string matching ($k = 0$). In contrast, Ukkonen's original algorithm takes $O(n)$ time. In Section 3.2 we describe an algorithm based on Chang and Marr's [4] average-optimal algorithm. For fixed $t > 0$, we derive an $O(tn \log_\sigma m/m)$ average-time approximate matching algorithm with error $\epsilon = O((k/m)^t)$. Note that the latter achieves the same time complexity for a smaller error, and that it works for $k = O(m)$, whereas the former needs $k < m/\log_\sigma m$. The discrepancy between both algorithms inherits from that of the original classical algorithms they derive from, where the original differences in time complexities has now translated into their error probabilities. It is important to stress that both algorithms beat the average-complexity lower bound of the problem when errors are not allowed, $\Omega(n(k + \log_\sigma m)/m)$ time, as they remove the $\Omega(kn/m)$ term in the complexity (the k/m term now shows up in the error probability).

In Section 4 we present some experimental results that corroborate the theoretical results of Section 3 and give supporting evidence for the practicality of our proposals. In particular, the experiments favor the technique of Section 3.1 over that of Section 3.2, despite the theoretical superiority of the latter.

The derived average case complexity results are for random text, but hold even for fixed patterns. Our analyzes focus exclusively on Levenshtein distance d, but can probably be adapted to other scenarios. We give a minor hint in this direction by showing in Section 5 how easily one can adapt the arguments of Section 3 to address the problem of approximate multi-pattern string searching. In particular, we again can break the average-case optimal lower bound of the classical problem, achieving average case $O(n \log_\sigma (rm)/m)$ time with any $\epsilon = \text{poly}(k/m)$, where r denotes the number of patterns whose occurrences in the text are being sought.

In Section 6 we discuss some possible future research directions and extensions of our work.

2. Model for approximate searching allowing errors

In this section we formalize the main concepts concerning the notion of approximate matching algorithms with errors. We adopt the standard convention of denoting the substring $S_i \ldots S_j$ of $S = S_1 \ldots S_n$ by $S_{i..j}$ and refer to the number of characters of S by the *length of* S, which we also denote by $|S|$. We start by recalling the formal definition of the approximate string matching problem when the underlying distance function is d. Henceforth, we abbreviate d-Approximate String Matching as d-ASM.

Problem	d-Approximate String Matching
Input	Text $T \in \Sigma^*$, pattern $P \in \Sigma^*$ and parameter $k \in \mathbf{N}$.
Output	$S = S(T, P, k) \subseteq \{1, \ldots, n\}$ such that $j \in S$ if and only if there is an i such that $d(T_{i..j}, P) \leq k$.

When the text T and pattern P are both in Σ^*, and the parameter k is in $\mathbf{N}$ we say that (T, P, k) is *an instance of the* d-ASM *problem*, or simply *an instance* for short. We henceforth refer to $S(T, P, k)$ as the *solution set* of instance (T, P, k). We say that algorithm $\mathcal{A}$ solves the d-ASM problem if on instance (T, P, k) it outputs the solution set $S(T, P, k)$. Note that $\mathcal{A}$ might be a probabilistic algorithm, however its output is fully determined by (T, P, k).

We consider the possibility of missing some occurrences in $S(T, P, k)$. For a randomized algorithm $\mathcal{A}$ that takes as input an instance (T, P, k), let $\mathcal{A}(T, P, k)$ be the distribution over sets $S \subseteq \{1, \ldots, n\}$ that it returns.

Henceforth we denote by $X \leftarrow \mathcal{D}$ the fact that the random variable X is chosen according to distribution $\mathcal{D}$. For any set C, we denote $X \leftarrow C$ a random variable X that is uniform on C. In general, we denote the probability that $X \in C$ when X is chosen according to the distribution $\mathcal{D}$ by $\mathbf{Pr}[X \in C; X \leftarrow \mathcal{D}]$ or $\mathbf{Pr}_{X \leftarrow \mathcal{D}}[X \in C]$. Also, we might simply write $\mathbf{Pr}_X[X \in C]$ or $\mathbf{Pr}[X \in C]$ when it is clear from context that $X \leftarrow \mathcal{D}$. The notation generalizes in the obvious way to the case where X is a random vector, and/or when instead of a probability one is interested in taking expectation (denoted by $\mathbf{Ex}[\cdot]$).

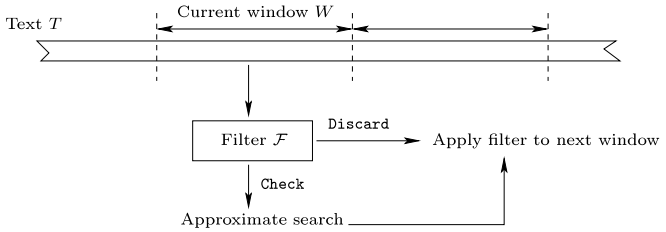

Fig. 1. Generic d-Approximate String Matching algorithm.

We say that randomized algorithm $\mathcal{A}$ solves the d-ASM problem with (ϵ, ϵ')-error provided that on any instance (T, P, k) the following holds:

Completeness: if $i \in S(T, P, k)$, then $\mathbf{Pr}\left[i \in S'; S' \leftarrow \mathcal{A}(T, P, k)\right] \geq 1 - \epsilon$,
Soundness: if $i \notin S(T, P, k)$, then $\mathbf{Pr}\left[i \in S'; S' \leftarrow \mathcal{A}(T, P, k)\right] \leq \epsilon'$,

where the two probabilities above are taken only over the source of randomness of $\mathcal{A}$.

When $\epsilon' = 0$ we say that $\mathcal{A}$ has *one-sided* ϵ*-error* or that it *is one-sided* for short. When $\epsilon = \epsilon' = 0$ we say that $\mathcal{A}$ is an *errorless* or *exact* algorithm.

Although a d-ASM algorithm with errors is guaranteed to output any index in the solution set $S(T, P, k)$ with probability at least $1 - \epsilon$ (and an index not in $S(T, P, k)$ with probability at most ϵ') there could potentially be correlations between the events $\{i \in \mathcal{A}(T, P, k)\}$ and $\{i' \in \mathcal{A}(T, P, k)\}$ for $i \neq i'$. Some sort of correlation is to be expected when i and i' are close to each other. However, in order to guarantee that it will be highly unlikely that a d-ASM algorithm will miss many sufficiently spread-out occurrences of the pattern, one desires that correlations are weak or non-existent when i and i' are far apart. In fact, one desires even more. Indeed, for any "sufficiently far apart" set of indices $L \subseteq \{1, \ldots, n\}$, one would like that the family of events $\{l \in \mathcal{A}(T, P, k)\}$ with $l \in L$ is independent. All of the d-ASM type algorithms we discuss in this paper do in fact have the aforementioned property when sufficiently far apart is interpreted as a collection of indices whose pairwise difference is at least $\Omega(m)$.

We say that randomized algorithm $\mathcal{F}$ is a d-ASM *probabilistic filter with* α*-error* or simply *is an* α*-filter* for short, provided that on any instance (W, P, k) the following holds: if $d(P_{i..j}, W) \leq k$ for some pattern substring $P_{i..j}$, then $\mathbf{Pr}\left[\mathcal{F}(W, P, k) = \texttt{Check}\right] \geq 1 - \alpha$, where the probability is taken over the source of randomness of $\mathcal{F}$. If a filter does not return `Check` we assume without loss of generality that it returns `Discard`.

The notion of an α-filter is crucial to the ensuing discussion. Roughly said, a filter $\mathcal{F}$ will allow us to process a text T by considering non-overlapping consecutive substrings W of T, running the filter on instance (W, P, k) and either: (1) in case the filter returns `Check`, perform a costly approximate string matching procedure to determine whether P approximately occurs in T in the surroundings of window W, or (2) in case the filter does not return `Check`, discard the current window from further consideration and move forward in the text and process the next text window. The previously outlined general mechanism is the basis of the generic algorithm we illustrate in Fig. 1 and describe below.

The attentive reader would have noticed that when defining probabilistic filters we substituted the notation T for texts by W. This is done in order to stress that the probabilistic filters that we will talk about access the text T by sequentially examining substrings of T which we will refer to as *windows*. These windows will typically have a length which is independent of n, more precisely they will be of length $O(m)$.

We now precisely describe the central role played by probabilistic filters in the design of d-ASM algorithms with errors. First, from now on, let w denote $\lfloor (m - k)/2 \rfloor$. Henceforth, let $W_1, \ldots, W_s$ be such that $T = W_1 \ldots W_s$ and $|W_p| = w$ (pad T with an additional character not in Σ as necessary). More precisely, let $s = \lceil n/w \rceil$ and $W_p = T_{(p-1)w+1..pw}$. Given any probabilistic filter $\mathcal{F}$ and an exact algorithm $\mathcal{E}$ we can devise a generic d-ASM algorithm with errors such as the one specified in Algorithm 1.[1]

We will shortly show that the generic algorithm $\mathcal{G}$ is correct. We also would like to analyze its complexity in terms of the efficiencies of both the probabilistic filter $\mathcal{F}$ and the exact algorithm $\mathcal{E}$. However, we first need to introduce the complexity measures that we will be looking at. Let $\mathbf{Time}_{\mathcal{A}}(T, P, k) \in \mathbf{N} \cup \{+\infty\}$ be the expected time complexity of $\mathcal{A}$ on the instance (T, P, k), where the expectation is taken over the random choices of $\mathcal{A}$. We also associate to $\mathcal{A}$ the following average time complexity measures:

$$\mathbf{Avg}_{\mathcal{A}}(n, P, k) = \mathbf{Ex}_{T \leftarrow \Sigma^n}\left[\mathbf{Time}_{\mathcal{A}}(T, P, k)\right],$$
$$\mathbf{Avg}_{\mathcal{A}}(n, m, k) = \mathbf{Ex}_{T \leftarrow \Sigma^n, P \leftarrow \Sigma^m}\left[\mathbf{Time}_{\mathcal{A}}(T, P, k)\right].$$

[1] For $A \subseteq \mathbf{Z}$ we use the standard convention of denoting $\{a + x : x \in A\}$ by $a + A$.

Algorithm 1 Generic d-approximate string matching with errors

1: **procedure** $\mathcal{G}(T, P, k)$ ▷ $T \in \Sigma^n, P \in \Sigma^m, k \in \mathbf{N}$
2: $S \leftarrow \emptyset$
3: $w \leftarrow \lfloor (m-k)/2 \rfloor$
4: $s \leftarrow \lceil n/w \rceil$
5: **for** $p \in \{1, \ldots, s\}$ **do**
6: **if** $\mathcal{F}(W_p, P, k) = \texttt{Check}$ **then** ▷ Where $W_p = T_{(p-1)w+1..pw}$
7: $S \leftarrow S \cup \left((pw - m - k) + \mathcal{E}(T_{pw-m-k+1..(p-1)w+m+k}, P, k)\right)$ ▷ See footnote[2]
8: **return** S

Let $\mathbf{Mem}_{\mathcal{A}}(T, P, k) \in \mathbf{N} \cup \{+\infty\}$ be the maximum amount of memory required by $\mathcal{A}$ on instance (T, P, k), where the maximum is taken over all possible sequences of random bits on which $\mathcal{A}$ may act, and let

$$\mathbf{Mem}_{\mathcal{A}}(n, P, k) = \max_{T \in \Sigma^n} \mathbf{Mem}_{\mathcal{A}}(T, P, k),$$
$$\mathbf{Mem}_{\mathcal{A}}(n, m, k) = \max_{T \in \Sigma^n, P \in \Sigma^m} \mathbf{Mem}_{\mathcal{A}}(T, P, k).$$

We similarly define $\mathbf{Rnd}_{\mathcal{A}}(T, P, k)$, $\mathbf{Rnd}_{\mathcal{A}}(n, P, k)$, and $\mathbf{Rnd}_{\mathcal{A}}(n, m, k)$, but with respect to the maximum number of random bits used by $\mathcal{A}$. Also, the same complexity measures can be defined for probabilistic filters and exact algorithms.

Theorem 1. *Suppose $m > k$. Let $\mathcal{F}$ be an α-filter and let $\mathcal{E}$ be the standard deterministic $O(kn)$ dynamic programming algorithm for the d-ASM problem. Let $\mathcal{W} \subseteq \Sigma^w$. Then, the generic algorithm $\mathcal{G}$ is a d-ASM algorithm with one-sided α-error such that*

$$\mathbf{Avg}_{\mathcal{G}}(n, P, k) \leq s \cdot \mathbf{Avg}_{\mathcal{F}}(w, P, k) + s \cdot O(mk) \cdot \left(\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] + \max_{W \notin \mathcal{W}} \mathbf{Pr}[\mathcal{F}(W, P, k) = \texttt{Check}]\right) + O(s).$$

Also, $\mathbf{Mem}_{\mathcal{G}}(n, P, k) = \mathbf{Mem}_{\mathcal{E}}(3w + 4k + 2, P, k) + \mathbf{Mem}_{\mathcal{F}}(w, P, k)$ (ignoring the space required to output the result), and $\mathbf{Rnd}_{\mathcal{G}}(n, P, k) = s \cdot \mathbf{Rnd}_{\mathcal{F}}(w, P, k)$.

Proof. First, let us establish completeness of $\mathcal{G}$. Assume $i \in S(T, P, k)$. Let $p + 1$ be the index of the window to which the character T_i belongs. As any occurrence of P in T has length at least $m - k$, W_p is completely contained in the occurrence finishing at i, and thus W_p must be at distance at most k of a substring of P. It follows that $\mathcal{F}(W_p, P, k) = \texttt{Check}$ with probability at least $1-\alpha$, in which case line 7 of the algorithm will run an exact verification with $\mathcal{E}$ over a text area comprising any substring of length $m+k$ that contains W_p. Since $m+k$ is the maximum length of an occurrence of P in T, it follows that i will be included in the output returned by $\mathcal{G}$. Hence, with probability at least $1 - \alpha$ we have that i is in the output of $\mathcal{G}$.

To establish soundness, assume $i \notin S(T, P, k)$. In this case, i will never be included in the output of $\mathcal{G}$ in line 7 of the algorithm.

We now determine $\mathcal{G}$'s complexity. By linearity of expectation and since $\mathbf{Time}_{\mathcal{E}}(O(m), m, k) = O(mk)$, we have

$$\mathbf{Avg}_{\mathcal{G}}(n, P, k) = \sum_{p=1}^{s} \left(\mathbf{Ex}_T\left[\mathbf{Time}_{\mathcal{F}}(W_p, P, k)\right] + O(mk) \cdot \mathbf{Pr}_T\left[\mathcal{F}(W_p, P, k) = \texttt{Check}\right] + O(1)\right)$$
$$= s \cdot \mathbf{Avg}_{\mathcal{F}}(w, P, k) + O(mk) \cdot \sum_{p=1}^{s} \mathbf{Pr}_T\left[\mathcal{F}(W_p, P, k) = \texttt{Check}\right] + O(s).$$

Conditioning according to whether W_p belongs to $\mathcal{W}$, we get for any $\mathcal{W}$ that

$$\mathbf{Pr}_T\left[\mathcal{F}(W_p, P, k) = \texttt{Check}\right] \leq \mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] + \max_{W \notin \mathcal{W}} \mathbf{Pr}[\mathcal{F}(W, P, k) = \texttt{Check}].$$

The stated bound on $\mathbf{Avg}_{\mathcal{G}}(n, P, k)$ follows immediately. The memory and randomized complexity bounds are obvious. □

The intuition behind the preceding theorem is that, given any class $\mathcal{W}$ of "interesting" windows, if we have a filter that discards the uninteresting windows with high probability, then the probability that the algorithm has to verify a given text window can be bounded by the sum of two probabilities: (i) that of the window being interesting, (ii) the maximum probability that the filter fails to discard a noninteresting window. As such, the theorem gives a general framework to analyze probabilistic filtration algorithms. An immediate consequence of the result is the following:

Corollary 2. *Under the same conditions of Theorem 1, if in addition*

$$\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] = \max_{W \notin \mathcal{W}} \mathbf{Pr}[\mathcal{F}(W, P, k) = \texttt{Check}] = O\left(1/m^2\right),$$

then $\mathbf{Avg}_{\mathcal{G}}(n, P, k) = O(s \cdot \mathbf{Avg}_{\mathcal{F}}(w, P, k))$. This also holds if $\mathcal{E}$ is the classical $O(m^2)$ time dynamic programming algorithm.

[2] If $pw - m - k + 1 < 1$, pad T to the left when running algorithm $\mathcal{E}$.

Algorithm 2 Probabilistic filter based on q-grams

procedure Q-PE-$\mathcal{F}_{c,\rho,q}(W, P, k)$ ▷ $W \in \Sigma^w, P \in \Sigma^m, k \in \mathbf{N}$
  $ctr \leftarrow 0$
  for $i \in \{1, \ldots, c\}$ **do**
    Choose j_i uniformly at random in $\{1, \ldots, |W| - q + 1\}$
    if $W_{j_i..j_i+q-1}$ is a substring of P **then** $ctr \leftarrow ctr + 1$
  if $ctr > \rho \cdot c$ **then return** `Check` **else return** `Discard`

The previous results suggests an obvious strategy for the design of d-ASM algorithms with errors. Indeed, it suffices to identify a small subset of windows $\mathcal{W} \subseteq \Sigma^w$ that contain all windows of length w that are at distance at most k of a pattern substring, and then design a filter $\mathcal{F}$ such that: (1) the probability that $\mathcal{F}(W, P, k) = \texttt{Check}$ is high when $W \in \mathcal{W}$ (in order not to miss pattern occurrences), and (2) the probability that $\mathcal{F}(W, P, k) = \texttt{Check}$ is low when $W \notin \mathcal{W}$ (in order to avoid running an expensive procedure over regions of the text where there are no pattern occurrences).

The next error amplification result is a simple observation whose proof we omit since it follows by standard methods (running $\mathcal{A}$ repeatedly).

Proposition 3. *Let $\mathcal{A}$ be a randomized algorithm that solves the d-ASM problem with (ϵ, ϵ')-error.*

- *Let $\alpha \leq \epsilon = \epsilon' < 1/2$ and $N = O(\log(1/\alpha)/(1-2\epsilon)^2)$. Then, there is a randomized algorithm $\mathcal{A}'$ that solves the d-ASM problem with (α, α)-error.*[3]
- *If $\mathcal{A}$ is one-sided, there is a randomized algorithm $\mathcal{A}'$ solving the d-ASM problem with $(\epsilon^N, 0)$-error.*

Moreover, it holds that $\mathbf{Avg}_{\mathcal{A}'}(n, P, k) = N \cdot \mathbf{Avg}_{\mathcal{A}}(n, P, k)$, $\mathbf{Mem}_{\mathcal{A}'}(n, P, k) = \mathbf{Mem}_{\mathcal{A}}(n, P, k) + O(\log N)$, *and* $\mathbf{Rnd}_{\mathcal{A}'}(n, P, k) = N \cdot \mathbf{Rnd}_{\mathcal{A}}(n, P, k)$.

3. Algorithms for approximate searching with errors

In this section we derive two probabilistic filters inspired on existing (errorless) filtration algorithms. Note that according to the previous section, we focus on the design of the window filters, and the rest follows from the general framework.

3.1. Algorithm based on q-gram sampling

A q-gram is a substring of length q. Thus, a pattern of length m has $(m - q + 1)$ overlapping q-grams. Each difference can alter at most q of the q-grams of the pattern, and therefore $(m - q + 1 - kq)$ pattern q-grams must appear in any approximate occurrence of the pattern in the text. Ukkonen's idea [16] is to sequentially scan the text while keeping count of the last q-grams seen. The counting is done using a suffix tree of P and keeping the relevant information attached to the $m - q + 1$ important nodes at depth q in the suffix tree. The key intuition behind the algorithm is that in random text it is difficult to find substrings of the pattern of length $q > \log_\sigma m$. The opposite is true in zones of the text where the pattern approximately occurs. Hence, by keeping count of the last q-grams seen one may quickly filter out many uninteresting areas.

We now show how to adapt the ideas mentioned so far in order to design a probabilistic filter. The filtering procedure randomly chooses several indices $i \in \{1, \ldots, |W| - q + 1\}$ and checks whether the q-gram $W_{i..i+q-1}$ is a pattern substring. Depending on the number of q-grams that are present in the pattern the filter decides whether or not to discard the window. See Algorithm 2 for a formal description of the derived probabilistic filter **Q-PE-**$\mathcal{F}_{c,\rho,q}$, where c and ρ are parameters to be tuned later. Using the filter as a subroutine for the generic algorithm with errors described in Algorithm 1 gives rise to a procedure to which we will henceforth refer to as **Q-PE**.

Remark 4. The set of the q-grams of P can be precomputed, so Algorithm 2 requires only $O(cq)$ time.

Let $\mathcal{W}$ be the collection of all windows in Σ^w for which at least β of its q-grams are substrings of the pattern. Let $w' = w - q + 1$ be the number of q-grams (counting repetitions) in a window of length w. Finally, let p denote the probability that a randomly chosen q-gram is a substring of the pattern P, that is,

$$p = \frac{1}{\sigma^q} \cdot \left|\{P_{i..i+q-1} : i = 1, \ldots, m - q + 1\}\right|.$$

The following result shows that a window chosen randomly in Σ^w is unlikely to be in $\mathcal{W}$.

Lemma 5. *Let $\beta \geq pw'$. Then,* $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] \leq \exp\left(-\frac{24(\beta - pw')^2}{25q(\beta + 2pw')}\right)$.

[3] By running the algorithm repeatedly and determining the occurrences using majority rule, see for example [5].

Proof. For $i = 1, \ldots, w'$ let Y_i be the indicator variable of the event "$W_{i..i+q-1}$ is a substring of P" when W is randomly chosen in Σ^w. Clearly, $\mathbf{Ex}[Y_i] = p$. Moreover, $W \in \mathcal{W}$ if and only if $\sum_{i=1}^{w'} Y_i \geq \beta$. Unfortunately, a standard Chernoff type bound cannot be directly applied given that the Y_i's are not independent. Nevertheless, the collection $\{Y_1, \ldots, Y_{w'}\}$ can be partitioned into q families according to i mod q, each one an independent family of random variables. The desired result follows applying a Chernoff bound for such type of independent families [9, Corollary 2.4]. □

Lemma 6. *Let $\beta \leq \rho w'$. If $W \notin \mathcal{W}$, then*

$$\mathbf{Pr}\left[\textbf{Q-PE}\text{-}\mathcal{F}_{c,\rho,q}(W, P, k) = \texttt{Check}\right] \leq \exp\left(\rho c - \frac{c\beta}{w'}\right)\left(\frac{\beta}{\rho w'}\right)^{\rho c},$$

where the probability is taken exclusively over the sequence of random bits of the probabilistic filter.

Proof. Let X_{j_i} denote the indicator of whether $W_{j_i..j_i+1-1}$ turns out to be a substring of the pattern P in line 5 of the description of **Q-PE**-$\mathcal{F}_{c,\rho,q}$. Note that the X_{j_i}'s are independent, each with expectation at most β/w' when $W \notin \mathcal{W}$. The claim follows by a standard Chernoff type bound from the fact that:

$$\mathbf{Pr}_{W \leftarrow \Sigma^w}\left[\textbf{Q-PE}\text{-}\mathcal{F}_{c,\rho,q}(W, P, k) = \texttt{Check}\right] = \mathbf{Pr}\left[\sum_{i=1}^{c} X_{j_i} > \rho \cdot c\right]. \quad \square$$

Lemma 7. *If $kq \leq w'(1-\rho)$, then **Q-PE**-$\mathcal{F}_{c,\rho,q}$ is an α-filter for*

$$\alpha \leq \exp\left((1-\rho)c - \frac{ckq}{w'}\right)\left(\frac{kq}{w'(1-\rho)}\right)^{c(1-\rho)}.$$

Proof. Let $W \in \Sigma^w$. Assume $d(P_{i..j}, W) \leq k$ for some pattern substring $P_{i..j}$. Then, at least $w' - kq$ of W's q-grams are substrings of P. Defining X_{j_i} as in Lemma 6 we still have that the X_{j_i}'s are independent but now their expectation is at least $1 - kq/w'$. The claim follows by a standard Chernoff type bound from the fact that:

$$\mathbf{Pr}\left[\textbf{Q-PE}\text{-}\mathcal{F}_{c,\rho,q}(W, P, k) = \texttt{Discard}\right] = \mathbf{Pr}\left[\sum_{i=1}^{c} X_{j_i} \leq \rho \cdot c\right],$$

where the probabilities are taken exclusively over the sequence of random bits of the probabilistic filter. □

Theorem 8. *Let $t > 0$ be a fixed constant. If $k < (m - 4\log_\sigma m)/(1 + 8\log_\sigma m)$, then **Q-PE** is a d-ASM algorithm with one-sided error $O((k\log_\sigma m/m)^t)$ and average time complexity $\mathbf{Avg}_{\textbf{Q-PE}}(n, P, k) = O(tn\log_\sigma m/m)$.*

Proof. Choose $q = 2\lceil\log_\sigma m\rceil$, so $p \leq m/\sigma^q \leq 1/m^2$. Taking $\beta = \Theta(\log^2 m)$ where the hidden constant is sufficiently large, we have by Lemma 5 that $\mathbf{Pr}_{W \leftarrow \mathcal{W}}[W \in \mathcal{W}] = O(1/m^2)$. By Lemma 6 and taking $\rho = 1/2$ and c a sufficiently large constant, we get that $\mathbf{Pr}\left[\textbf{Q-PE}\text{-}\mathcal{F}_{c,\rho,q}(W, P, k) = \texttt{Check}\right] = O(1/m^2)$ when $W \notin \mathcal{W}$.

Choose $c(1-\rho) \geq t$. Note that the bound on k in the theorem's statement implies that $kq \leq w'(1-\rho)$. Lemma 7 thus applies and yields that **Q-PE**-$\mathcal{F}_{c,\rho,q}$ has $O((k\log_\sigma m/m)^t)$-error.

Clearly $\mathbf{Avg}_{\textbf{Q-PE}\text{-}\mathcal{F}_{c,\rho,q}}(w, P, k) = O(cq) = O(t\log_\sigma m)$. The result follows from Theorem 1 and Corollary 2. □

We can obtain different trade-offs between the time and error of the algorithm **Q-PE** by adjusting the parameters of the algorithm. By choosing a non-constant t in the proof of Theorem 8, it is easy to see that **Q-PE** is a d-ASM algorithm with any desired error tolerance ϵ and average time complexity $\mathbf{Avg}_{\textbf{Q-PE}}(n, P, k) = O(tn\log_\sigma m/m)$ where $t = \log_\sigma(1/\epsilon)/\log_\sigma(m/(k\log_\sigma m))$.

3.2. Algorithm based on covering by pattern substrings

In 1994 Chang and Marr [4] proposed a variant of SET [3] with running time $O(n(k+\log_\sigma m)/m)$ for $k/m \leq 1/3 - O(\sigma^{-1/2})$. As in SET, Chang and Marr consider text windows of size $(m-k)/2$, and pinpoint occurrences of the pattern by identifying blocks that approximately match a substring of the pattern. This identification is based on splitting the text window into contiguous blocks of length $\ell = \lceil\alpha\log_\sigma m\rceil$. Those blocks are considered consecutively. For each one, the algorithm counts the minimum number differences for the block to match inside P. The scanning of blocks continues until the accumulated number of differences exceeds k or all the blocks are considered. If k differences occur before $(m-k)/2$ text characters are covered with the scanned blocks, then the rest of the window can be safely skipped.

The adaptation of Chang and Marr's approach to the design of probabilistic filters is quite natural. Instead of looking at ℓ-grams sequentially we just randomly choose sufficiently many non-overlapping ℓ-substrings in each block. We then determine the fraction of them that approximately appear in the pattern. If this fraction is small enough, then the block is discarded. See Algorithm 3 for a formal description of the derived probabilistic filter CM-PE-$\mathcal{F}_{c,\rho,\ell,g}$. Using the filter as a subroutine for the generic algorithm with errors described in Algorithm 1 gives rise to a procedure to which we will henceforth refer to as CM-PE.

Algorithm 3 Probabilistic filter based on covering by pattern substrings.

procedure CM-PE-$\mathcal{F}_{c,\rho,\ell,g}(W, P, k)$ ▷ $W \in \Sigma^w, P \in \Sigma^m, k \in \mathbf{N}$
  $ctr \leftarrow 0$
  for $i \in \{1, \ldots, c\}$ **do**
    Choose j_i uniformly at random in $\{1, \ldots, \lfloor w/\ell \rfloor\}$
    if $asm(W_{(j_i-1)\ell+1..j_i\ell}, P) \leq g$ **then** $ctr \leftarrow ctr + 1$ ▷ $asm(S, P) = \min_{a \leq b} d(S, P_{a..b})$
  if $ctr > \rho \cdot c$ **then return** `Check` **else return** `Discard`

Remark 9. Values $asm(S, P)$ of Algorithm 3 can be precomputed for all values of $S \in \Sigma^\ell$, so Algorithm 3 performs in $O(c\ell)$ time.

The analysis of Algorithm 3 establishes results such as Lemmas 5–7 and Theorem 8 but concerning CM-PE-$\mathcal{F}_{c,\rho,\ell,g}$. Below we provide the corresponding details. Throughout this section let $d = \lfloor w/\ell \rfloor$. Let $\mathcal{W}$ be the collection of all windows W in Σ^w for which at least β of the $W_{(i-1)\ell+1..i\ell}$'s with i varying in $\{1, \ldots, d\}$ are at distance at most g of some substring of P.

Lemma 10. *There are $\alpha > 0$ and $0 < \epsilon < 1$ such that for $\ell = \lceil \alpha \log_\sigma m \rceil$, $g = \epsilon\ell$, $d \leq \beta m^3$ and m sufficiently large,* $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] \leq \exp\left(\beta - dm^{-3}\right)\left(\frac{d}{\beta m^3}\right)^\beta$.

Proof. For $1 \leq i \leq d$ let Y_i be the indicator variable of the event "$asm(W_{(i-1)\ell+1..i\ell}, P) \leq g$" when W is randomly chosen in Σ^w. The Y_i's are clearly independent and have all the same expected value which we will denote $p = p(g, \ell)$. In [4] it is shown that there exists $\alpha > 0$ and $0 < \epsilon < 1$ such that $p(\epsilon\ell, \ell) = O(m^{-3})$ for $\ell = \lceil \alpha \log_\sigma m \rceil$. Since $W \in \mathcal{W}$ if and only if $\sum_{i=1}^d Y_i \geq \beta$, the desired result follows applying a standard Chernoff type bound. □

Lemma 11. *Let $\beta \leq \rho d$. If $W \notin \mathcal{W}$, then*

$$\mathbf{Pr}\left[\text{CM-PE-}\mathcal{F}_{c,\rho,\ell,g}(W, P, k) = \texttt{Check}\right] \leq \exp\left(\rho c - \frac{c\beta}{d}\right)\left(\frac{\beta}{\rho d}\right)^{\rho c},$$

where the probability is taken exclusively over the sequence of random bits of the probabilistic filter.

Proof. For $1 \leq i \leq d$, let X_{j_i} be the indicator variable of the event "$asm(W_{(j_i-1)\ell+1..j_i\ell}, P) \leq g$" when the j_i's are randomly chosen in $\{1, \ldots, d\}$. Note that the X_{j_i}'s are independent, each with expectation at most β/d when $W \notin \mathcal{W}$. The claim follows by a standard Chernoff type bound from the fact that

$$\mathbf{Pr}\left[\text{CM-PE-}\mathcal{F}_{c,\rho,\ell,g}(W, P, k) = \texttt{Check}\right] = \mathbf{Pr}\left[\sum_{i=1}^{c} X_{j_i} > \rho \cdot c\right]. \quad \square$$

Lemma 12. *If $k \leq dg(1-\rho)$, then* CM-PE-$\mathcal{F}_{c,\rho,\ell,g}$ *is an α-filter where*

$$\alpha \leq \exp\left((1-\rho)c - \frac{ck}{dg}\right)\left(\frac{k}{dg(1-\rho)}\right)^{c(1-\rho)}.$$

Proof. Let $W \in \Sigma^w$. Assume $d(P_{i..j}, W) \leq k$ for some pattern substring $P_{i..j}$. Then, the fraction of j's in $\{1, \ldots, d\}$ such that $asm(W_{(j-1)\ell+1..j\ell}, P) > g$ is at most k/dg. Defining X_{j_i} as in Lemma 11 we still have that the X_{j_i}'s are independent but now their expectation is at least $1 - k/dg$. The claim follows by a standard Chernoff type bound from the fact that:

$$\mathbf{Pr}\left[\text{CM-PE-}\mathcal{F}_{c,\rho,\ell,g}(W, P, k) = \texttt{Discard}\right] = \mathbf{Pr}\left[\sum_{i=1}^{c} X_{j_i} \leq \rho \cdot c\right]. \quad \square$$

Theorem 13. *There is a sufficiently small constant C such that if $k < Cm$ and $t > 0$, then* CM-PE *is a d-*ASM *algorithm with one-sided error $O((k/m)^t)$. Its average running time is $\mathbf{Avg}_{\text{CM-PE}}(n, P, k) = O(tn \log_\sigma m/m)$.*

Proof. Choose $\rho = 1/2, c = \max\{2t, 6\}, g = \epsilon\ell$, and $\ell = \lceil \alpha \log_\sigma m \rceil$, where ϵ and α are the constants of Lemma 10. Taking $\beta = 2$ and applying Lemma 10 we get that $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] = O(1/m^4)$. Since $w \geq 4\ell$ for m sufficiently large, it holds that $\beta \leq \rho d$. Hence, by Lemma 11 and observing that $\rho \cdot c \geq 3$, we get that $\mathbf{Pr}\left[\text{CM-PE-}\mathcal{F}_{c,\rho,\ell,g}(W, P, k) = \texttt{Check}\right] = O(\log_\sigma^3 m/m^3) = O(1/m^2)$ when $W \notin \mathcal{W}$. If $k \leq (\epsilon m)/(4+\epsilon)$, then $k \leq dg(1-\rho)$, and applying Lemma 12 we get that CM-PE-$\mathcal{F}_{c,\rho,\ell,g}$ has $O((k/m)^t)$-error.

Clearly $\mathbf{Avg}_{\text{CM-PE-}\mathcal{F}_{c,\rho,\ell,g}}(w, P, k) = O(c\ell) = O(t \log_\sigma m)$. The result follows from Theorem 1 and Corollary 2. □

We can obtain different trade-offs between the time and error of CM-PE by adjusting the parameters of the algorithm. It can be shown that CM-PE is a d-ASM algorithm with any desired error tolerance ϵ and average time complexity $\mathbf{Avg}_{\text{CM-PE}}(n, P, k) = O(tn \log_\sigma m/m)$ where $t = \log_\sigma(1/\epsilon)/\log_\sigma(m/k)$.

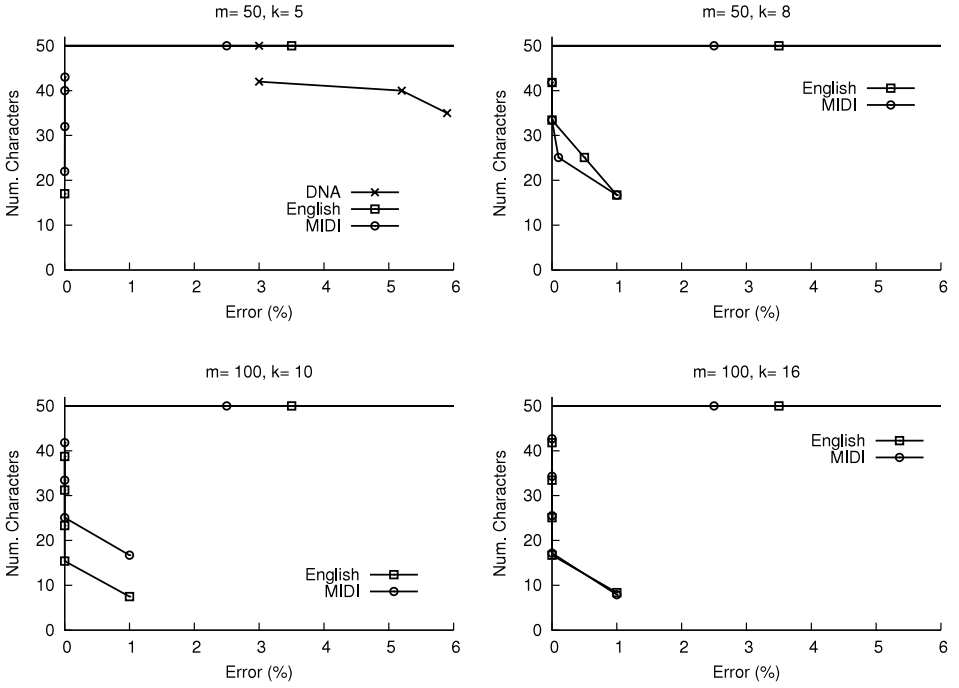

Fig. 2. Experimental results for **Q-PE**. Straight horizontal lines correspond to the errorless version. The y axis represents the number of character inspections times 1024.

4. Experimental results

We implemented the algorithms of Sections 3.1 and 3.2. We extracted three real-life texts of 50 MB from *Pizza&Chili* (http://pizzachili.dcc.uchile.cl): English text, DNA, and MIDI pitches. We used patterns of length 50 and 100, randomly extracted from the text, and some meaningful k values. Each data point is the average over 50 such search patterns, repeating each search 15 times in the case of the randomized algorithms. We measured the average number of character inspections and the average percentage of missed occurrences.

We used the following setup for the algorithms. For q-gram algorithms (Section 3.1), we used $q = 4$. Our preliminary results show that $\rho = 0.7$ is a good choice. For covering by pattern substrings (Section 3.2), we used $\epsilon = 0.2$ and $\rho = 0.3$. In our algorithms, we only moved parameter c in order to change the accuracy/time tradeoff. We compared our algorithms with the corresponding errorless filtering algorithms.

Fig. 2 shows the experimental results for the q-gram based procedure, and Fig. 3 for the covering by pattern substrings process. The errorless version of the q-grams algorithm inspects all text characters. In contrast, our q-gram based procedure achieves less than 1% error rate and looks at up to 6 times less characters on English and MIDI corpora. For our second algorithmic proposal, the result of the comparison against the errorless version is not as good. Nevertheless, we emphasize that it beats the average-optimal (errorless) algorithm by a wide margin, specifically it inspects about half the characters with 15% errors on the English corpus.

5. Extensions to multiple pattern approximate string matching

The d-Multiple Pattern Approximate String Matching (d-MASM) problem is the natural generalization of the d-ASM problem to the case that r patterns are given as part of the input (see formal definition given below).

Problem	d-Approximate Multi-pattern String Matching
Input	Text $T \in \Sigma^*$, multi-pattern $\vec{P} = (P^{(1)}, \ldots, P^{(r)}) \in (\Sigma^*)^r$ and parameter $k \in \mathbf{N}$.
Output	$S = S(T, \vec{P}, k) \subseteq \{1, \ldots, r\} \times \{1, \ldots, n\}$ such that $(a, j) \in S$ if and only if there is an i such that $d(T_{i..j}, P^{(a)}) \leq k$.

It is straightforward to extend to the multi-pattern case the notions of solving the d-ASM problem with errors, the concept of probabilistic filters, and the notion of d-ASM algorithm with errors. We henceforth refer to the latter type of algorithms as *d-approximate multi-pattern string matching algorithms with error*.

The generic principle described in Section 2 for designing d-ASM algorithms with errors is well adapted to the design of similar algorithms for the d-MASM problem. Indeed, one can again focus on deriving a probabilistic filter that again either

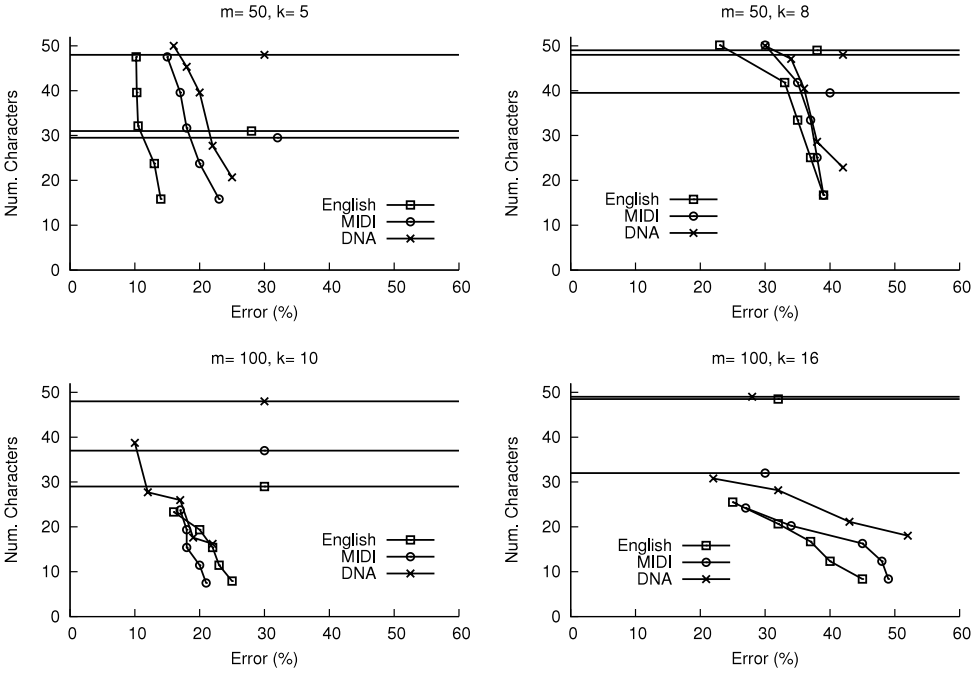

Fig. 3. Experimental results for CM-PE. Straight horizontal lines correspond to the errorless version. The y axis represents the number of character inspections times 1024.

checks/discards text windows of length $w = \lfloor (m-k)/2 \rfloor$. Each time a window is chosen for verification, an appropriate neighborhood of the window in the text is verified for pattern occurrences. The only required adaptations to the multi-pattern scenario is in the design of the probabilistic filter. We now briefly discuss how to carry out this adaptation for the filtering algorithms discussed in Sections 3.1 and 3.2.

The adaptation of the q-gram filter of Section 3.1 to the multi-pattern case consists in sampling q-grams from the text window in the same way as in the single pattern case, but we now discard a window provided at most $\rho \cdot c$ of the c sampled q-grams are not present in *any* of the r patterns $P^{(1)}, \ldots, P^{(r)}$. We denote the resulting probabilistic filter and d-approximate multi-pattern string matching algorithm with errors by QMP-PE-$\mathcal{F}_{c,\rho,q}$ and QMP-PE respectively.

For the analysis of QMP-PE, let $w' = w - q + 1$ and p denote the probability that a randomly chosen q-gram is a substring of one of the patterns $P^{(1)}, \ldots, P^{(r)}$. Also, let $\mathcal{W}$ be the collection of windows in Σ^w for which at least β of its q-grams are substrings of some pattern $P^{(1)}, \ldots, P^{(r)}$. Lemmas 5 through 7 are still valid, and we can use them to derive the following:

Theorem 14. *Let $u > 0$ and $r = O(m^u)$. Let $t > 0$ be a fixed constant. If $k < (m - 4\log_\sigma(rm))/(1 + 8\log_\sigma(rm))$, then* QMP-PE *is a d-approximate multi-pattern string matching algorithm with one sided $O((k\log_\sigma(rm)/m)^t)$-error. Moreover,* $\mathbf{Avg}_{\text{QMP-PE}}(n, m, k) = O(tn\log_\sigma(rm)/m)$.

Proof. Since there are at most rm distinct q-grams that can show up in $P^{(1)}, \ldots, P^{(r)}$, we have that $p \leq rm/\sigma^q$. Thus, taking $q = 2\lceil \log_\sigma(rm) \rceil$ it follows that $p \leq 1/(rm)$. Taking $\beta = \Theta(\log_\sigma^2(rm))$ with a sufficiently large hidden constant, by Lemma 5 we get that $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] = O(1/rm^2)$. Now, taking $\rho = 1/2$ and c a constant strictly greater than $2(2+u)$, by Lemma 6 we get that

$$\mathbf{Pr}\left[\text{QMP-PE-}\mathcal{F}_{c,\rho,q}(W, P^{(1)}, \ldots, P^{(r)}, k) = \texttt{Check}\right] = O\left(\frac{1}{rm^2}\right).$$

By hypothesis, $k < (m - 4\log_\sigma(rm))/(1 + 8\log_\sigma(rm))$, so $kq \leq w'(1-\rho)$ and Lemma 7 can be applied. Taking $c \geq 2t$, we thus conclude that QMP-PE-$\mathcal{F}_{c,\rho,q}$ is an α-filter with $\alpha = O((k\log_\sigma(rm)/m)^t)$.

Observing that QMP-PE-$\mathcal{F}_{c,\rho,q}(W, P^{(1)}, \ldots, P^{(r)}, k)$ has an average case complexity of $O(cq)$ (provided all q-grams of $P^{(1)}, \ldots, P^{(r)}$ are precomputed), recalling that all occurrences of W in $P^{(1)}, \ldots, P^{(r)}$ can be determined in time $O(rm^2)$, and following the argument used to derive Theorem 1, yields the desired conclusion. □

The adaptation of the covering by pattern substrings filter of Section 3.2 to the multi-pattern case consists in sampling ℓ-grams from the text in the same way as in the single pattern case, but now we discard a window if at most $\rho \cdot c$ of the c sampled ℓ-grams are at distance at most g of some substring of *any* of the r patterns $P^{(1)}, \ldots, P^{(r)}$. We denote the resulting probabilistic filter and d-approximate multi-pattern string matching algorithm with errors by CMMP-PE-$\mathcal{F}_{c,\rho,\ell,g}$ and CMMP-PE respectively.

For the analysis of CMMP-PE, we let $w' = w - \ell + 1$ and $d = \lfloor w/\ell \rfloor$. Also, let $\mathcal{W}$ be the collection of windows W in Σ^w such that at least β of the $W_{(i-1)\ell+1..i\ell}$'s with i varying in $\{1, \ldots, d\}$ are at distance at most g of some pattern $P^{(1)}, \ldots, P^{(r)}$. In addition, let

$$multAsm(S, P^{(1)}, \ldots, P^{(r)}) = \min_{s=1,\ldots,r} \min_{a \le b} d(S, P^{(s)}_{a..b}).$$

It is easy to verify that Lemmas 11 and 12 still hold. An argument similar to the one used to prove Lemma 10 can be applied to obtain the following:

Lemma 15. *Let $u > 0$ and $r = O(m^u)$. There are $\alpha > 0$ and $0 < \epsilon < 1$ such that for $\ell = \lceil \alpha \log_\sigma m \rceil$, $g = \epsilon\ell$, $d \le \beta m^3$ and m sufficiently large,* $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}] \le \exp\left(\beta - dm^{-3}\right)\left(\frac{d}{\beta m^3}\right)^\beta$.

Proof. For $1 \le i \le d$ let Y_i be the indicator variable of the event "$multAsm(W_{(i-1)\ell+1..i\ell}, P^{(1)}, \ldots, P^{(r)}) \le g$" when W is randomly chosen in Σ^w. The Y_i's are clearly independent and have all the same expected value which we will denote $p = p(g, \ell, r)$. Clearly, $p \le rp(g, \ell)$ where $p(g, \ell)$ is as defined in the proof of Lemma 10. In [4] it is shown that for any $u > 0$ there exists $\alpha > 0$ and $0 < \epsilon < 1$ such that $p(\epsilon\ell, \ell) = O(m^{-u-3})$ for $\ell = \lceil \alpha \log_\sigma m \rceil$. Since $W \in \mathcal{W}$ if and only if $\sum_{i=1}^d Y_i \ge \beta$, the desired result follows applying a standard Chernoff type bound. □

We can now establish our main result concerning multiple pattern approximate string matching with bounded errors.

Theorem 16. *Let $u, t > 0$ and let $r = O(m^u)$. There is a sufficiently small constant C such that if $k \le Cm$, then* CMMP-PE *is a d-ASM algorithm with one sided $O((k/m)^t)$-error. Moreover,* $\mathbf{Avg}_{\mathbf{Q\text{-}PE}}(n, m, k) = O(tn \log_\sigma m/m)$.

Proof. Let α, ϵ, ℓ and g be as in Lemma 15, set $\beta = u/2 + 2$, and apply the lemma to obtain that $\mathbf{Pr}_{W \leftarrow \Sigma^w}[W \in \mathcal{W}]$ is $O(1/rm^2)$. Let $\rho = 1/2$, choose c a constant strictly greater than $2(2+u)$, and apply Lemma 11 to obtain that when $W \notin \mathcal{W}$,

$$\mathbf{Pr}\left[\text{CMMP-PE-}\mathcal{F}_{c,\rho,\ell,g}(W, P^{(1)}, \ldots, P^{(r)}, k) = \texttt{Check}\right] = O\left(\frac{1}{rm^2}\right).$$

Assuming that C is small enough so $Cm \le w\epsilon/2$, we can apply Lemma 12 with $c \ge 2t$ and conclude that CMMP-PE-$\mathcal{F}_{c,\rho,\ell,g}$ has $O((k/m)^t)$-error.

Observing that CMMP-PE-$\mathcal{F}_{c,\rho,\ell,g}(W, P^{(1)}, \ldots, P^{(r)}, k)$ has an average case complexity of $O(c\ell)$ (provided all ℓ-grams of $P^{(1)}, \ldots, P^{(r)}$ are precomputed), recalling that all occurrences of W in $P^{(1)}, \ldots, P^{(r)}$ can be determined in time $O(rm^2)$, and following the argument used to derive Theorem 1, yields the desired conclusion. The result follows from Theorem 1 and Corollary 2. □

6. Final comments

In this paper we have advocated considering a new dimension of the approximate string matching problem, namely the probability of missing an approximate occurrence. This relaxation is particularly natural for a problem that usually arises when modeling processes where errors have to be tolerated, and it opens the door to novel approaches to approximate string matching which break the average-case lower bound of the original problem. In particular, we have shown that much faster text scanning is possible if one allows a small probability of missing occurrences. We achieved an average time complexity of $O(n \log_\sigma m/m)$ (which is the complexity of exact string matching, $k = 0$) with error probability bounded by any polynomial in k/m. Empirically, we have shown that our algorithms inspect a fraction of the text with virtually no mistakes.

We have just scratched the surface of this new area. In particular, we have not considered filtration algorithms that use sliding instead of fixed windows. Sliding-window algorithms have the potential of being more efficient (cf. Fredriksson and Navarro's variant [7] with the original Chang and Marr's average-optimal algorithms [4]). It is not hard to design those variants, yet analyzing them is more challenging. On the other hand, it is rather simple to extend our techniques to multiple ASM, as we have shown. Preliminary explorations suggest that the approach and techniques proposed in this work can also be applied to tackle pattern matching problems where preprocessing of the text is possible, that is, indexed algorithms can be used [12]. Several indexes build on sequential filtration algorithms, and thus adapting them is rather natural.

Another interesting aspect is to determine the average time complexity of this relaxed problem, considering the error probability ϵ in the formula. This would give an idea of how much can one gain by allowing errors in the outcome of the search. For example, our algorithms break the $\Omega(kn/m)$ term in the problem complexity, yet a term poly(k/m) appears in the error probability. Which are the best tradeoffs one can achieve?

Finally, and potentially even more fruitful, is to extend the approach put forth in this work to other information retrieval problems.

Acknowledgements

The first author gratefully acknowledges the support of CONICYT via FONDAP-Basal in Applied Mathematics, Anillo en Redes ACT08, and FONDECYT 1090227. The second author was funded in part by Fondecyt Grant 1-080019, Chile. The third author gratefully acknowledges the support of CONICYT via Anillo en Redes ACT08.

References

[1] M. Blum, M. Luby, R. Rubinfeld, Self-testing/correcting with applications to numerical problems, J. Comput. Syst. Sci. 47 (3) (1993) 549–595.
[2] W. Chang, E. Lawler, Approximate string matching in sublinear expected time, in: Proceedings of the ACM-SIAM 31st Annual Symposium on Foundations of Computer Science, 1990, pp. 116–124.
[3] W. Chang, E. Lawler, Sublinear approximate string matching and biological applications, Algorithmica 12 (4–5) (1994) 327–344.
[4] W. Chang, T. Marr, Approximate string matching and local similarity, in: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, in: LNCS, vol. 807, Springer-Verlag, 1994, pp. 259–273.
[5] Devdatt Dubhashi, Alessandro Panconesi, Concentration of Measure for the Analysis of Randomized Algorithms, 1st ed., Cambridge University Press, New York, NY, USA, 2009.
[6] E. Fischer, The computational complexity column: the art of uninformed decisions, Bulletin of the EATCS 75 (2001) 97–126.
[7] K. Fredriksson, G. Navarro, Average-optimal single and multiple approximate string matching, ACM Journal of Experimental Algorithmics 9 (article 1.4) (2004).
[8] O. Goldreich, Combinatorial property testing (a survey), Electronic Colloquium on Computational Complexity 4 (56) (1997).
[9] S. Janson, Large deviations for sums of partly dependent random variables, Random Structure & Algorithms 24 (3) (2004) 234–248.
[10] M. Kiwi, G. Navarro, C. Telha, On-line approximate string matching with bounded errors, in: Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching, in: LNCS, vol. 5029, Springer-Verlag, 2008, pp. 130–142.
[11] G. Navarro, A guided tour to approximate string matching, ACM Computing Surveys 33 (1) (2001) 31–88.
[12] G. Navarro, R. Baeza-Yates, E. Sutinen, J. Tarhio, Indexing methods for approximate string matching, IEEE Data Engineering Bulletin 24 (4) (2001) 19–27.
[13] D. Ron, Handbook of Randomized Computing II, in: Combinatorial Optimization, vol. 9, Springer, 2001 (Chapter Property Testing).
[14] R. Rubinfeld, R. Kumar, Algorithms column: sublinear time algorithms, SIGACT News 34 (4) (2003) 57–67.
[15] J. Tarhio, E. Ukkonen, Approximate Boyer-Moore string matching, SIAM Journal on Computing 22 (2) (1993) 243–260.
[16] E. Ukkonen, Approximate string-matching with q-grams and maximal matches, Theoretical Computer Science 92 (1992) 191–211.