Towards an Efficient Text Sampling Approach for Exact and Approximate Matching

Simone Faro, Francesco Pio Marino, Arianna Pavone and Antonio Scardace

Department of Mathematics and Computer Science, University of Catania (Italy)
Depertment of Cognitive Science, University of Messina (Italy)

Prague Stringology Conference
August 31, 2021

String Matching

Definition

String Matching consists in finding all occurrences of a given pattern x, of length m, in a large text y, of length n, where both sequences are composed by characters drawn from an alphabet Σ of size σ.

Variants

- Exact
- Approximate

Sampled String Matching

Definition

The task of the sampled string matching problem is to find all occurrences of a given pattern x in a given text y, assuming that a fast and succint preprocessing of the text is allowed in order to build a data-structure, called partial-index, which is used to speed-up the searching phase.

Partial-Index Propriety

- Succint
- Fast to build
- Allow Fast Search
- Allow Fast Update

Sampled String Matching

Previous Solutions

- 1991 The problem was theoretically introduced by Vishkin;
- 2012 Claude *et al.* introduce a more practical solution based on alphabet reduction applied in Online and Offline string matching;
- 2020 Faro *et al.* introduce a new approach for the creation of the partial-index the Characters Distance Sampling applied in Online string matching;
- 2020 Faro and Marino presented an offline version of the Characters Distance Sampling method.
- 2021 Faro *et al.* presented an efficient version of the Characters Distance Sampling for Small Alphabets.

Alphabet Reduction Sampling (Claude *et al.*)

Definition

Let y be the input text, of length n, and let x be the input pattern, of length m, both over the same alphabet Σ of size σ. The main idea of this approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet) and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$ of length $\hat{n}$ containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$.

Position Mapping

The algorithm has to verify the corresponding occurrence of x in the text. For this reason a table ρ is maintained to map, at regular intervals, positions of the sampled text to their corresponding positions in the original text. The position mapping is maintained by an interval factor q. Suggested values of q are {8,16,32}

Characters Distance Sampling

Definition

Let $c \in \Sigma$ be the pivot character, let $n_c \leq n$ be the number of occurrences of the pivot character in the text y and let δ be the position function of y. We define the *characters distance function* $\Delta(i) = \delta(i+1) - \delta(i)$, for $1 \leq i \leq n_c - 1$, as the distance between two consecutive occurrences of the pivot character in y. The character-distance sampled version of the text is a numeric sequence, indicated by $\bar{y}$, of length n_c.

SAMPLING(y, n_c)
1. $j \leftarrow 0$
2. for $i \leftarrow 1$ to n do
3. if $y[i] \in \bar{\Sigma}$
4. $\bar{y}[j] = \delta(j) - \delta(j-1)$
5. $j = j + 1$

Characters Distance Sampling for Small Alphabets

Definition

Let y be an input sequence, of length n, over an alphabet Σ of length *sigma*. Given a constant parameter q, with $1 \leq q \leq n$ we define the condensed alphabet $\Sigma_y^{(q)}$, related to y, as:

$$\Sigma_y^{(q)} = \{c \in \Sigma^q | c = y[i...i+q-1]\} \text{ for some } 1 \leq i \leq n-q+1$$

Roughly speaking $\Sigma_y^{(q)}$ is the set of all different subsequences of length q (or q-grams) appearing in y.
Let $\Sigma^{(q)}$ be the condensed alphabet over Σ with $q > 1$, let $c \in \Sigma^{(q)}$ the pivot character we create a partial-index called $\bar{y}^{(q)}$.

Monotonic Run Length Scaling

The present work

All methods previously exposed has been designed to work for solving the exact string matching problem, being inflexible in case they have to be applied to approximate string matching problems.

The present work

- Work efficiently for exact and approximate String Matching
- Work efficiently also for small alphabets

Monotonic Run

Definition

Let y be a text of length n over a finite and totally ordered alphabet Σ of size σ.

- A Monotonic Increasing Run of y is a not extendable sub-sequence w of y whose elements are arranged in increasing order.
- Symmetrically a Monotonic Non-Increasing Run of y is a not extendable sub-sequence w of y whose elements are arranged in non-increasing order.

We will indicate with the general term Monotonic Run any sub-sequence of y that can be both monotonic increasing and monotonic non-increasing.

Monotonic Run

Definition

Two adjacent monotonic sub-sequences of a string have a single overlapping character, i.e. the rightmost character of the first sub-sequence is also the leftmost character of the second.

Example

Let $y = \langle 4, 5, 11, 7, 6, 6, 12, 12, 2, 9, 8, 6, 7, 10, 13 \rangle$ be a numeric sequence of length 15. We can identify the following monotonic runs in y: $y[0..2] = \langle 4, 5, 11 \rangle$ is a monotonic increasing run; $y[2..5] = \langle 11, 7, 6, 6 \rangle$ is a monotonic non-increasing run; $y[5..6] = \langle 6, 12 \rangle$ is a monotonic increasing run; $y[6..8] = \langle 12, 12, 2 \rangle$ is a monotonic non-increasing run; $y[8..9] = \langle 2, 9 \rangle$ is a monotonic increasing run; $y[9..11] = \langle 9, 8, 6 \rangle$ is a monotonic non-increasing run; finally, $y[11..14] = \langle 6, 7, 10, 13 \rangle$ is a monotonic increasing run.

Monotonic Run Length Scaling

Definition

Let y be a text of length n over a finite and totally ordered alphabet Σ of size σ. Let $\langle \rho_0, \rho_1, \ldots, \rho_{k-1} \rangle$ the sequence of adjacent monotonic runs of y such that ρ_0 starts at position 0 of y and ρ_{k-1} ends at position $n-1$ of y. The monotonic run length scaled version of y, indicated by $\widetilde{y}$, is a numeric sequence, defined as $\widetilde{y} = \langle |\rho_0|, |\rho_1|, .., |\rho_{k-1}| \rangle$.

Monotonic Run Length Scaling

Definition

It is straightforward to prove that there exists only a unique monotonic run length scaled version of a given string y. In addition we observe that

$$\left\lceil \sum_{i=0}^{k-1} |\rho_i| \right\rceil - k + 1 = n$$

Example

Let $y = \langle 4, 5, 11, 7, 6, 6, 12, 12, 2, 9, 8, 6, 7, 10, 13 \rangle$ be a numeric sequence of length 15. Then the scaled version of y is the sequence $\widetilde{y} = \langle 3, 4, 2, 3, 2, 3, 4 \rangle$ of length 7. We have therefore that $3 + 4 + 2 + 3 + 2 + 3 + 4 - 7 + 1 = 15$.

Monotonic Run Length Scaling

Length of Monotonic Run

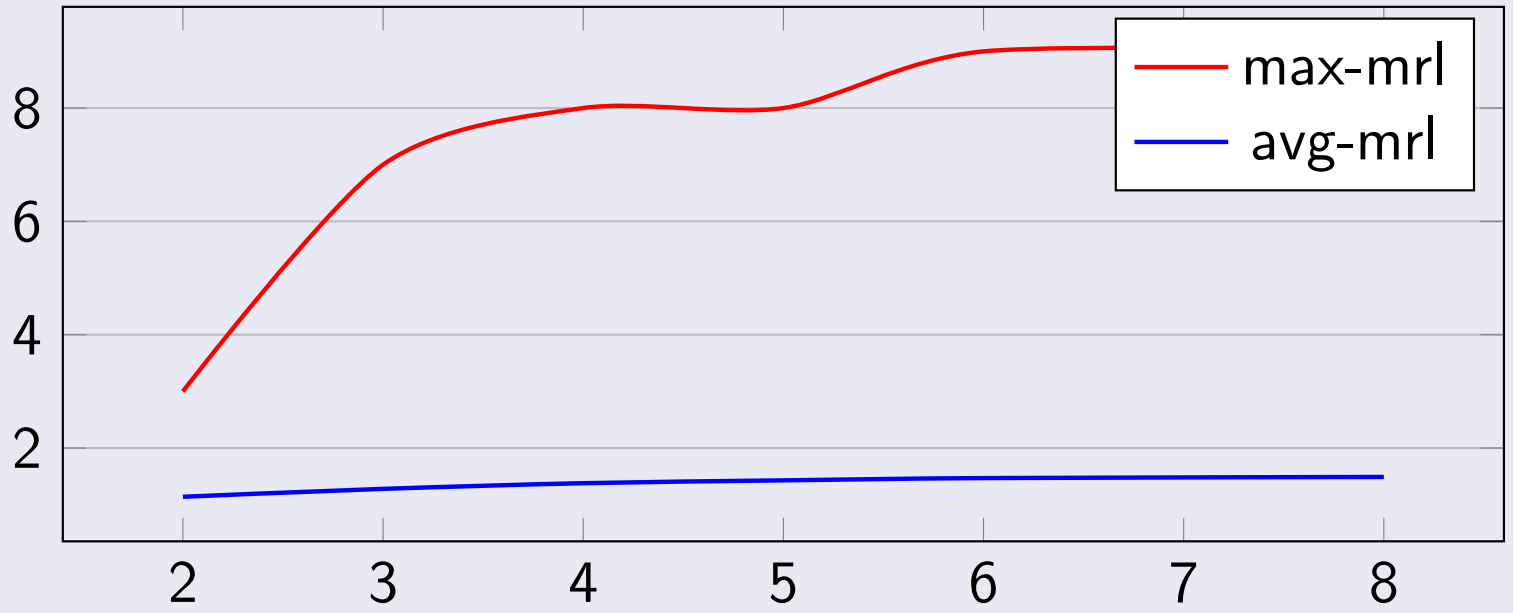

Figure: Average and maximal length of a monotonic run on a random text over an alphabet of size 2^{δ}. The ordinates show the length values while the abscissas show the values of δ.

Monotonic Run Length Scaling

Definition

It is easy to observe that the length of each monotonic run never exceeds ten characters. Furthermore, we observe that the average length of the monotonic sub-sequences is much lower being always between 1.10 (for small alphabets) and 1.50 (for large alphabets). This implies that the average length of the monotonic run length scaled version of the text is on average between 66% and 90% of the length of the original sequence, a result not particularly exciting when compared with those obtained by previous sampling techniques such as CDS and OTS.

Monotonic Run Length Scaling

Definition

However, the fact that the length of each monotonic sub-sequence is up to 10 allows us to represent the monotonic run length scaled version of the text using only 4 bits for each element of the sequence, instead of the 32 bits needed in the OTS and CDS representation. Thus the average memory consumption required by the monotonic run length scaled version of the text is on average between 33% and 45% of the memory needed to store the original sequence.

Monotonic Run Length Scaling

Definition

We discuss the use of monotonic run length scaling as a sampling approach with application to exact and approximate string matching. For the case of approximate string matching, we take the Order Preserving Pattern Matching problem.

Monotonic Run Length Scaling

Definition

We present a naïve searching procedure designed to use the monotonic scaled version of the text as a partial index in order to speed up the search for the occurrences of a given pattern within the original text. Like any other solution based on text-sampling, the solution proposed in this section requires that the partial index is used in a preliminary filter phase and that each candidate occurrence identified in this first phase is then verified within the original text using a simple verification procedure.

Monotonic Run Length Scaling

Definition

The preprocessing phase of the algorithm consists in computing the monotonic run length scaled version $\widetilde{x}$ of the input pattern x.
Let k be the length of the sequence $\widetilde{y}$ and let h be the length of the sequence $\widetilde{x}$.In addition let d be the length of the first monotonic run of x, *i.e.* $d = \widetilde{x}[0]$

Definition

During the searching phase the algorithm naively searches for all occurrences of the sub-sequence $\widetilde{x}[1..h-2]$ along the scaled version of the text. We discard the first and the last element of $\widetilde{x}$ since they are allowed to match any element in the text which is greater than or equal.

Monotonic Run Length Scaling

Example

Let $y = \langle 4, 5, 11, 7, 6, 6, 12, 12, 2, 9, 8, 6, 7, 10, 13 \rangle$ be a numeric sequence of length 15 (the text) and let $x = \langle 12, 2, 9, 8, 6, 7, 10 \rangle$ be a pattern of length 7. Then we have $\widetilde{y} = \langle 3, 4, 2, 3, 2, 3, 4 \rangle$ and $\widetilde{x} = \langle 2, 2, 3, 3 \rangle$, with $d = 2$.

The algorithm naively searches the text $\widetilde{y}$ for any occurrence of the sub-sequence $\widetilde{x}[1..2] = \langle 2, 3 \rangle$, finding two candidate occurrences at position 2 and 4, respectively.

We obtain that:

Monotonic Run Length Scaling

Example

- at the beginning of the third iteration of the **for** main loop, with $s = 2$, we have $r = 3 + 4 - 1 = 6$. Thus the verification procedure is run to compare x against the sub-sequence at position $r - d = 4$ in the original text, i.e. $\widetilde{y}[4..10] = \langle 6, 6, 12, 12, 2, 9, 8 \rangle$. Unfortunately at position 6 the verification procedure would find neither an exact match nor an order preserving match.
- at the beginning of the fifth iteration of the **for** main loop, with $s = 4$, we have $r = 3 + 4 + 2 + 3 - 3 = 8$. Thus the verification procedure is run to compare x against the sub-sequence at position $r - d = 6$ in the original text, i.e. $\widetilde{y}[6..12] = \langle 12, 12, 2, 9, 8, 6, 7, 10 \rangle$. At position 6 the verification would find both an exact match and an OPPM.

Monotonic Run Length Sampling

Definition

As observed above, the space consumption for representing the partial index obtained by monotonic run length scaling is not particularly satisfactory. This influences also the performance of the search algorithms in practical cases, as we will see in experimental evaluation.

Definition

We propose the application of an approach similar to that used by CDS sampling in order to obtain a partial index requiring a reduced amount of space, on the one hand, and is able to improve the performance of the search procedure, on the other hand.

Monotonic Run Length Sampling

Definition

Given a monotonic run w of y, and assuming $w = y[i..i+k-1]$, we use the symbol $\mu(w)$ to indicate its starting position i in the text y.

Monotonic Run Length Sampling

We introduce the *Monotonic Run Length Sampling* (MRLS) process which, given an input string y, constructs a partial index $\widetilde{y}^q$, which is the numeric sequence of all (and only) starting positions of any monotonic runs of y with length equal to q, for a given parameter $q > 1$.

Monotonic Run Length Sampling

Definition

Let y be a text of length n and let $\widetilde{y} = \langle |\rho_0|, |\rho_1|, .., |\rho_{k-1}| \rangle$ be the monotonic run length scaled version of y, with $|\widetilde{y}| = k$. In addition let ℓ be the maximal length of a monotonic run in y, i.e. $\ell = \max(|\rho_i| \ : \ 0 \leq i < k)$. If q is an integer value, with $2 \leq q \leq \ell$, we define the Monotonic Run Length Sampled version of y, with pivot length q, as the numeric sequence $\widetilde{y}^q$, defined as $\widetilde{y}^q = \langle |\rho_{i_0}|, |\rho_{i_1}|, .., |\rho_{i_{h-1}}| \rangle$, where $h \leq k$, $i_{j-1} < i_j$ for each $0 < j < h$, and $|\rho_{i_j}| = q$ for each $0 \leq j < h$.

Monotonic Run Length Sampling

Example

Let $y = \langle 4, 5, 11, 7, 6, 6, 12, 12, 2, 9, 8, 6, 7, 10, 13 \rangle$ be a numeric sequence of length 15. As already observed $\widetilde{y} = \langle 3, 4, 2, 3, 2, 3, 4 \rangle$. Thus we have:

- $\widetilde{y}^2 = \langle 5, 8 \rangle$, since $\mu(\langle 6, 12 \rangle) = 5$ and $\mu(\langle 2, 9 \rangle) = 8$;
- $\widetilde{y}^3 = \langle 0, 6, 9 \rangle$, since $\mu(\langle 4, 5, 11 \rangle) = 0$, $\mu(\langle 12, 12, 2 \rangle) = 6$ and $\mu(\langle 9, 8, 6 \rangle) = 9$;
- $\widetilde{y}^4 = \langle 2, 11 \rangle$, since $\mu(\langle 11, 7, 6, 6 \rangle) = 2$ and $\mu(\langle 6, 7, 10, 13 \rangle) = 11$.

Monotonic Run Length Sampling

Sampling Procedure

The Sampling procedure gets as input the monotonic run length scaled version $\widetilde{x}$ of the string, its length h and the pivot length q. Then it constructs the sequence $\widetilde{x}^q$ incrementally by scanning the input sequence $\widetilde{x}$ element by element, proceeding from left to right.

Monotonic Run Length Sampling

Definition

We use a simple naïve procedure to search for all occurrences (in their exact or approximate version) of a pattern x of length m inside a text y of length n.

Searching Procedure

The algorithm takes as input both the text y and its MRLS version $\widetilde{y}^q$ of length k.The searching phase of the algorithm consists in a main **for** loop which iterates over the sequence $\widetilde{y}^q$, proceeding form left to right. For each element $\widetilde{y}^q[s]$, for $0 \leq s < h$, the algorithm calls the verification procedure to check an occurrence beginning at position $\widetilde{y}^q[s] - d$ in the original text. Roughly speaking, the algorithm aligns the first monotonic of length q in the pattern with all monotonic runs of length q in the text.

Monotonic Run Length Sampling

Example

Let $y = \langle 4, 5, 11, 7, 6, 6, 12, 12, 2, 9, 8, 6, 7, 10, 13 \rangle$ be a numeric sequence of length 15 (the text) and let $x = \langle 12, 2, 9, 8, 6, 7, 10 \rangle$ be a pattern of length 7. Then we have $\widetilde{y} = \langle 3, 4, 2, 3, 2, 3, 4 \rangle$ and $\widetilde{x} = \langle 2, 2, 3, 3 \rangle$. Assuming $q = 3$ we have also $\widetilde{y}^q = \langle 0, 6, 9 \rangle$, $\widetilde{x}^q = \langle 2, 4 \rangle$ and $d = 2$.

The algorithm considers any position $r \in \widetilde{y}^q$ as a candidate occurrence of the pattern and naively checks the whole pattern against the sub-sequence $y[r - d \ldots r - d + m - 1]$ of the original text. Thus we have:

Monotonic Run Length Sampling

Example

- at the first iteration of the main **for** loop we have $s = 0$ and the algorithm would run a verification for the window starting at $\widetilde{y}^q[0] - d = 0 - 2 = -2$. However, since $-2 < 0$, such window is skipped.
- at the second iteration of the main **for** loop we have $s = 1$ and the algorithm would run a verification for the window starting at $\widetilde{y}^q[1] - d = 6 - 2 = 4$. Thus the pattern x is compared with the sub-sequence $y[4...10] = \langle 6, 6, 12, 12, 2, 9, 8 \rangle$. However in this case neither an exact occurrence nor an order preserving occurrence would be found.

Monotonic Run Length Sampling

Example

- finally, at the third iteration of the main **for** loop we have $s = 2$ and the algorithm would run a verification for the window starting at $\widetilde{y}^q[9] - d = 9 - 2 = 7$. Thus the pattern x is compared with the sub-sequence $y[7...13] = \langle 12, 2, 9, 8, 6, 7, 10 \rangle$. In this case the verification procedure would find both an exact occurrence and an order preserving match.

Space Requirements

Space Requirements

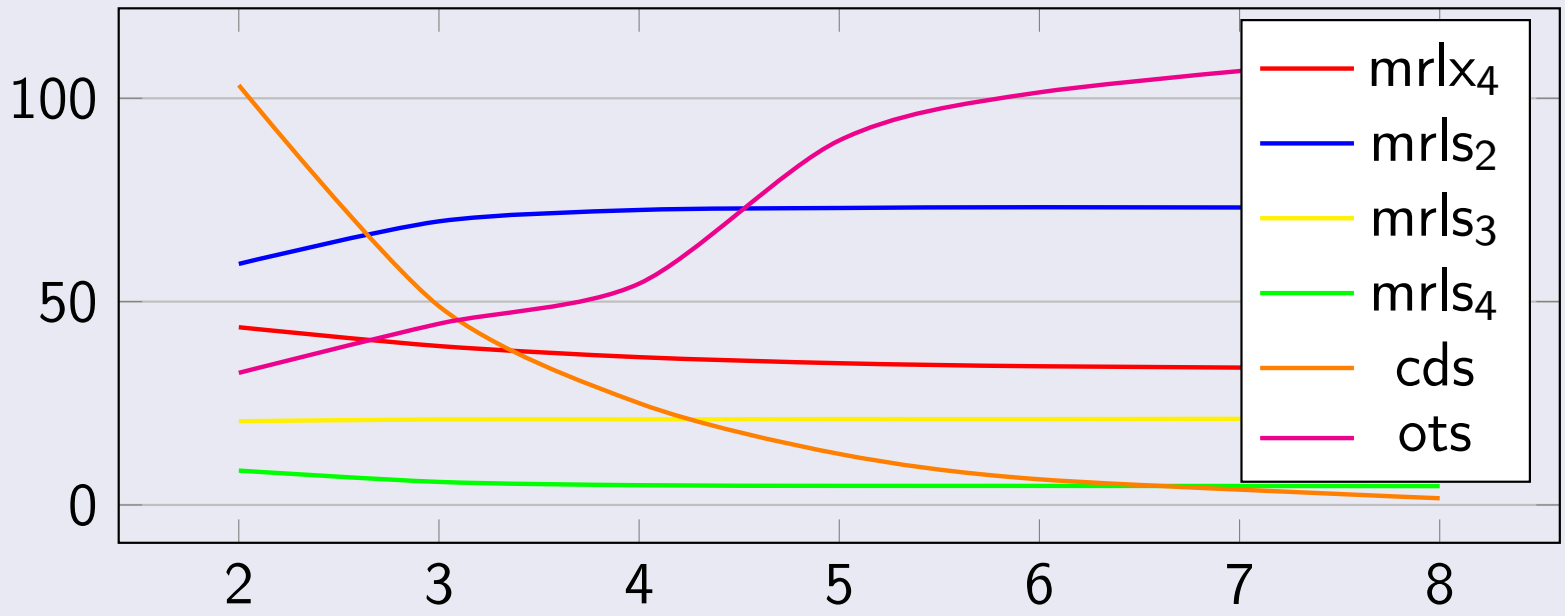

Figure: Space Consumption of sampling approaches to text searching. Data are reported as percentage values with respect to the size of the original text on which the index is built. Values are computed on six random texts with a uniform distribution and built on an alphabet of size 2^{δ} (abscissa) with $2 \leq \delta \leq 8$.

Experimental Results

Table

δ	m	NRq	MRLX$_4$	MRLX$_8$	MRLS$_2$	MRLS$_3$	MRLS$_4$
4	8	5.40	0.59	**1.19**	0.91	0.72	0.70
	16	5.21	0.74	1.34	**1.50**	0.76	0.73
	32	5.25	0.78	1.40	**3.65**	1.52	0.71
	64	5.07	0.71	1.25	3.57	**3.67**	1.16
	128	5.21	0.81	1.37	3.59	**9.65**	2.52
	256	5.61	0.77	1.40	3.98	10.58	**14.38**
	512	5.71	0.81	1.46	4.05	10.77	**22.84**
	1024	5.20	0.78	1.42	3.59	9.45	**18.57**
256	8	4.99	0.99	**1.81**	1.23	0.89	0.85
	16	4.83	1.05	1.92	**2.60**	0.93	0.77
	32	5.01	1.12	2.09	**3.48**	1.69	0.83
	64	4.76	1.08	1.91	3.33	**6.52**	0.86
	128	4.96	1.09	1.95	3.59	**8.70**	1.58
	256	4.72	1.07	1.93	3.75	**11.51**	3.87
	512	4.79	1.06	1.94	3.89	**11.97**	10.41
	1024	4.89	1.12	2.08	4.37	12.54	**37.62**

Experimental Results

Table

δ	m	HOR	CDS	OTS	$MRLS_2$	$MRLS_3$	$MRLS_4$
4	8	2.43	1.21	1.31	**1.48**	1.23	1.19
	16	2.10	1.19	1.31	**1.98**	1.27	1.25
	32	1.99	1.21	1.33	**2.65**	2.40	1.24
	64	2.16	1.23	1.33	2.96	**4.50**	2.00
	128	2.10	1.20	1.29	2.84	**6.77**	3.96
	256	1.99	1.23	1.33	2.62	6.42	**10.47**
	512	2.01	1.19	1.32	2.75	6.48	**12.56**
	1024	1.97	1.22	1.31	2.70	5.97	**12.31**
256	8	0.77	**1.54**	1.54	1.71	1.57	1.57
	16	0.58	**2.23**	2.07	1.81	2.07	1.93
	32	0.46	**3.29**	2.56	1.39	2.88	2.19
	64	0.45	**5.00**	3.45	1.36	3.46	3.21
	128	0.44	**11.00**	3.37	1.33	3.38	4.40
	256	0.45	**15.00**	4.90	1.32	3.21	5.00
	512	0.47	**23.50**	5.85	1.42	3.62	5.87
	1024	0.47	**23.50**	5.86	1.38	3.36	5.87

Conclusions and Future Works

Conclusions

The first experimental results obtained showed how the approaches are particularly versatile, obtaining considerable speed-ups on execution times, reaching gain factors of 40, in particularly favorable conditions. The new techniques presented therefore serve as good starting points for improvements in future investigations.

Future Works

One of the most interesting aspects to be analyzed in a future work is the applicability of the new sampling approach to other approximate string matching problems. We can now say that the MRLX technique is well suited to solve other problems such as Cartesian-tree pattern matching or shape preserving pattern matching, which have a strong relationship with the OPPM.