New Efficient Bit-Parallel Algorithms for the δ-Matching Problem with α-Bounded Gaps in Musical Sequences

Domenico Cantone, Salvatore Cristofaro, and Simone Faro

Università degli Studi di Catania, Dipartimento di Matematica e Informatica
Viale Andrea Doria 6, I-95125, Catania, Italy
{cantone, cristofaro, faro}@dmi.unict.it

Abstract

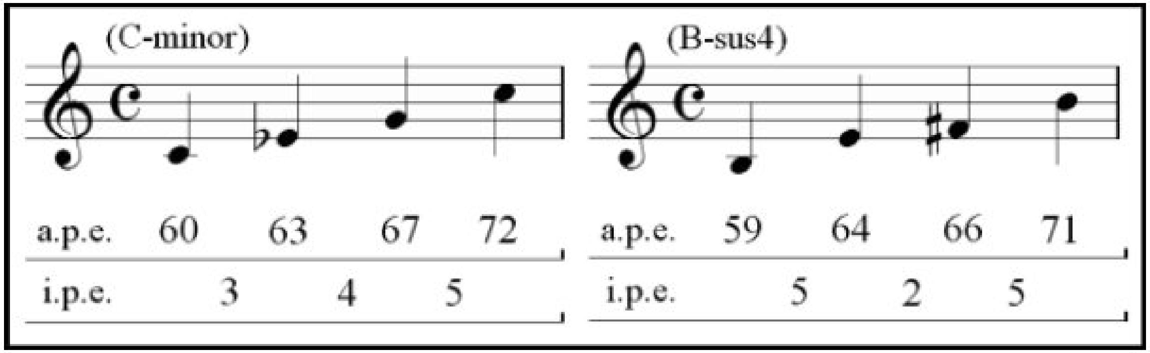
We present new efficient variants of the (δ,α)-Sequential-Sampling algorithm, recently introduced by the authors, for the δ-approximate string matching problem with α-bounded gaps. These algorithms, which have practical applications in music information retrieval and analysis, make use of the well-known technique of bit-parallelism. An extensive comparison with the most efficient algorithms present in the literature for the same search problem shows that our newly proposed solutions achieve very good results in practice, in terms of both space and time complexity, and, in most cases, they outperform existing algorithms.

δ-Matching with α-Bounded Gaps

The δ-approximate string matching problem with α-bounded gaps is a generalization of the δ-approximate string matching problem and arise in many questions in music information retrieval and music analysis. This is particularly true in the context of monophonic music, where one wants to retrieve occurrences of a given melody from a complex musical score

δ-matching

musical sequences have a δ-approximate matching if they have the same length (i.e., they contain the same number of notes) and notes at the same positions differ by at most δ semitones

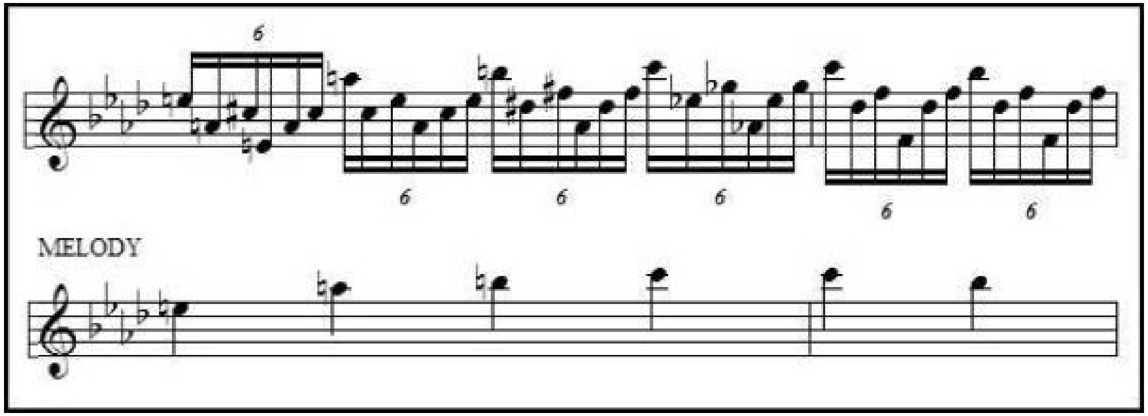


δ-Matching with α-Bounded Gaps

The δ-approximate string matching problem with α-bounded gaps is a generalization of the δ-approximate string matching problem and arise in many questions in music information retrieval and music analysis. This is particularly true in the context of monophonic music, where one wants to retrieve occurrences of a given melody from a complex musical score

δ-matching with α-bounded gaps

It is allowed in the musical sequence to skip up to a fixed number α of notes (the gap) between any two consecutive positions.



δ-Matching with α-Bounded Gaps

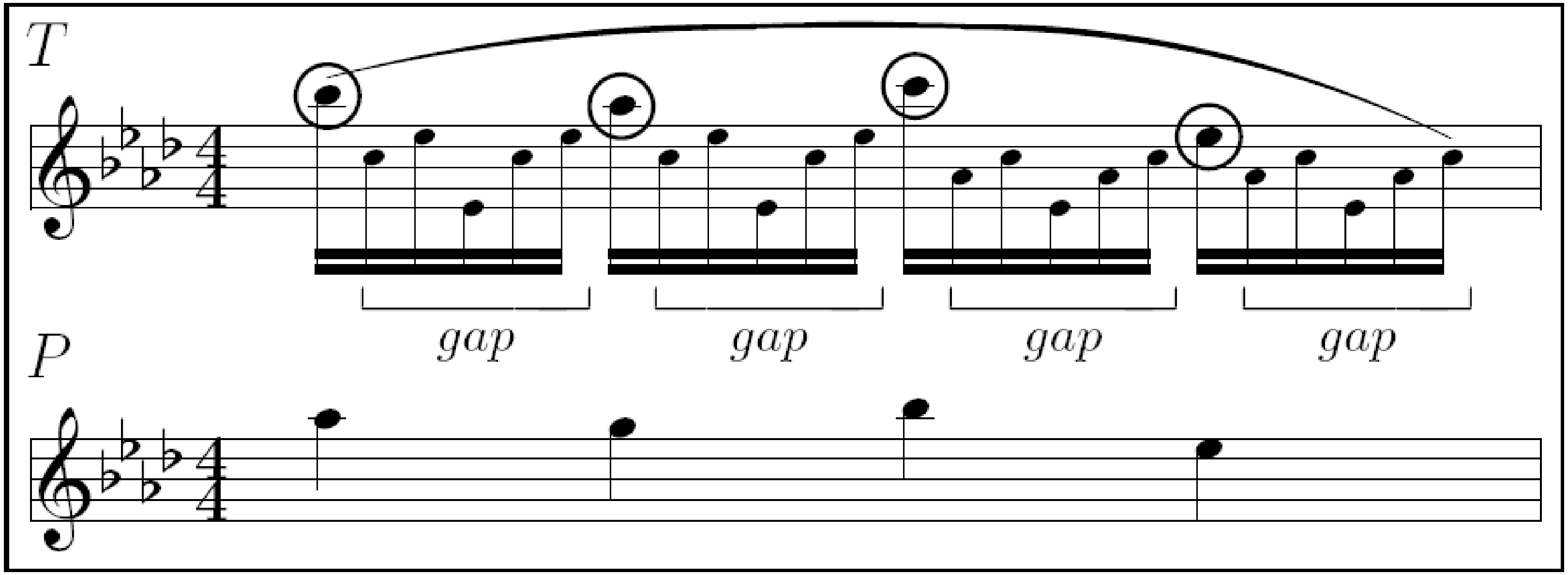


Figure 1. An excerpt from study *Op. 25 Nr. 1 for Piano Solo* by F. Chopin (first score). Melody P has a δ-approximate occurrence with α-bounded gaps in T, for $\delta \geq 2$ and $\alpha \geq 5$, indicated by the circled notes. Tiny notes represent arpeggios and form the gaps. Notice that in this case the gaps are all of the same size 5. Observe also that the first and the third note of P differ from the corresponding matchings in T (circled notes) by 2 semitones; the second note differ by 1 semitone, while the last note equals its matching. In any case, the difference between a note and its matching does not exceed 2 semitones, so that we have a (δ, α)-occurrence of P in T, for any $\delta \geq 2$ and $\alpha \geq 5$.

δ-Matching with α-Bounded Gaps

Figure 3: An excerpt of a piece of J.S. Bach (first score). The second score shows how the musical ornaments must be played. Two musical ornaments are present: a *mordent*, attached to the 4^{th} note; a *trill*, attached to the 11^{th} note.

The (δ,α)-Shift-And Algorithm

The problem can be solved in O(n)-time by simulating the behavior of an automaton which recognizes the pattern P. The (δ,α)-Shift-And algorithm **[1]**, uses bit-parallelism to simulate the automaton in its nondeterministic form. This simulation is performed by representing the automata as a list of mα bits, where mα is the number of states of the automata, and each state corresponds to a bit in the list.

(δ,α)-Shift-And	
Preprocessing Time Complexity	$O(m\alpha\vert\Sigma\vert)$
Searching Time Complexity	$O(\lceil mn/w \rceil)$
Space Complexity	$O(m\alpha\vert\Sigma\vert)$

[1] D. Cantone, S. Cristofaro, and S. Faro: On tuning the (δ,α)-sequential-sampling algorithm for δ-approximate matching with α-bounded gaps in musical sequences, in Proceedings of 6-th International Conference on Music Information Retrieval (ISMIR'05), S. D. Reiss and G. A. Wiggins, eds., 2005, pp. 454–459.

The (δ,α)-Shift-And Algorithm

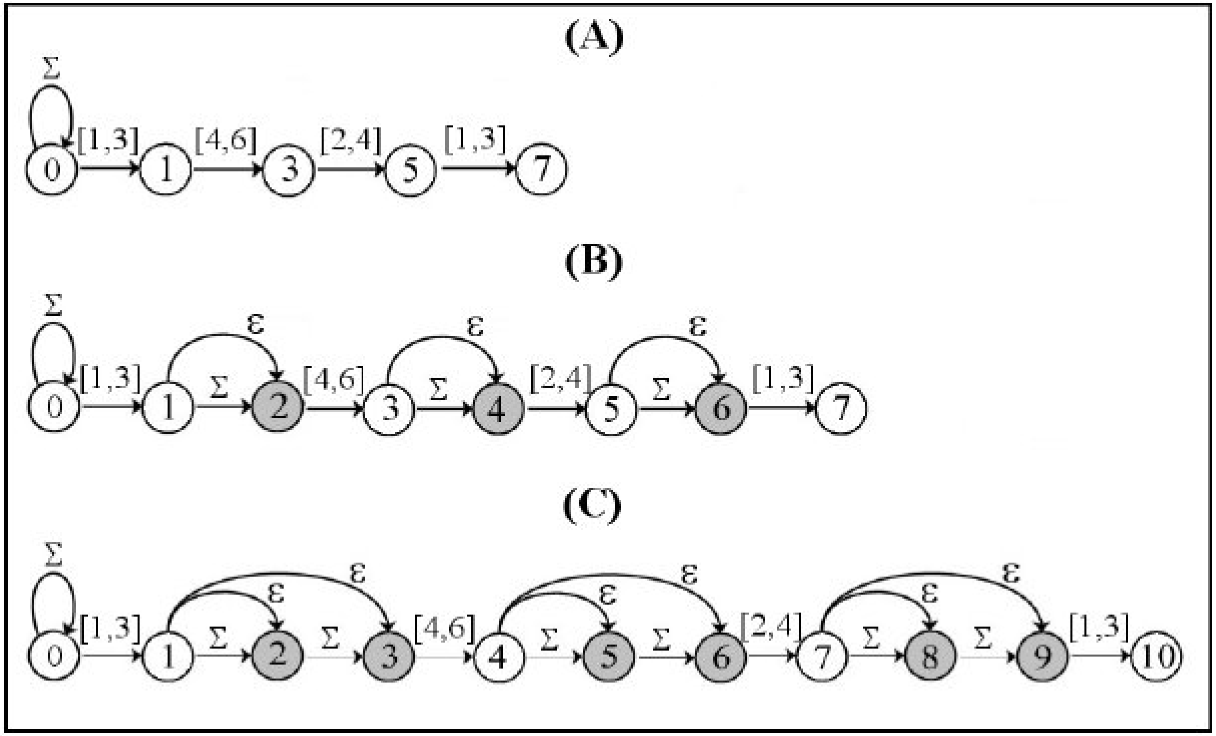


Figure 5: Three non-deterministic automata of the numeric pattern $P = 2, 5, 3, 2$ for approximate matching with $\delta = 1$. **(A)** non-deterministic automata with $\alpha = 0$ **(B)** non-deterministic automata with $\alpha = 1$ **(C)** non-deterministic automata with $\alpha = 2$

The DA-mloga-bits Algorithm

The DA-mloga-bits algorithm **[2]** is based on a compact representation, in the form of a systolic array, of the nondeterministic automaton used in the algorithm (δ,α)-Shift-And. The systolic array is composed of m building blocks, one for each symbol of the pattern, and is represented as a bit mask of length $(m-1)(\lceil\log(\alpha+1)\rceil)+1$.

DA-mloga-bits	
Worst Case Complexity	$\mathcal{O}(n\lceil(m\log_2\alpha)/w\rceil)$
Space Complexity	$\mathcal{O}(\lceil(m\log_2\alpha)/w\rceil)$

[2] K. Fredriksson and S. Grabowski: Efficient bit-parallel algorithms for (δ,α)-matching, in Proceedings of 5-th Workshop on Efficient and Experimental Algorithms, LNCS 4007, Springer–Verlag, 2006, pp. 170–181.

The δ-Bounded-Gaps Algorithm

The δ-Bonded-Gaps algorithm **[3]** is presented as an incremental procedure, based on the dynamic programming approach.

More specifically:

• it starts by first computing all the δ-occurrences with α-bounded gaps in T of the prefix of P of length 1.

• then, during the i-th iteration, it looks for all the (δ,α)-occurrences in T of the prefix of P of lerngth i.

• at the end of the last iteration, the (δ,α)-occurrences of the whole pattern P have been computed.

[3] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and W. Rytter: Finding motifs with gaps, in Proceedings of the International Symposium on Music Information Retrieval (ISMIR'00), Plymouth, USA, 2000, pp. 306–317.

The δ-Bounded-Gaps Algorithm

More specifically, the algorithm δ-Bounded-Gaps fills a matrix D of dimension m×n, where D[i, j] is computed according to the following recursive relation:

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j-\alpha \\ -1 & \text{otherwise} \end{cases}$$

DA-mloga-bits	
Worst Case Complexity	$\mathcal{O}(nm)$
Space Complexity	$\mathcal{O}(nm)$

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j-\alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	**3**	**7**	**4**	**2**	**5**	**6**	**8**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0																						
5	1																						
3	2																						
2	3																						
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j - \alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	**3**	**7**	**4**	**2**	**5**	**6**	**8**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	-1	1	1	-1	4	4	4	-1	8	9	10	11	11	13	13	15	16	17	18	18	20	20
5	1																						
3	2																						
2	3																						
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ \ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ \ j \geq 1\,,\ \text{and } D[i,j-1] \geq j-\alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	**3**	**7**	**4**	**2**	**5**	**6**	**8**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	-1	1	1	-1	4	4	4	-1	8	9	10	11	11	13	13	15	16	17	18	18	20	20
5	1	-1	-1	-1	3	3	5	6	6	6	-1	-1	-1	12	12	12	-1	-1	-1	-1	19	19	19
3	2																						
2	3																						
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j - \alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	**3**	**7**	**4**	**2**	**5**	**6**	**8**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	-1	1	1	-1	4	4	4	-1	8	9	10	11	11	13	13	15	16	17	18	18	20	20
5	1	-1	-1	-1	3	3	5	6	6	6	-1	-1	-1	12	12	12	-1	-1	-1	-1	19	19	19
3	2	-1	-1	-1	-1	4	4	4	-1	8	8	8	-1	-1	13	13	13	-1	-1	-1	-1	20	20
2	3																						
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j - \alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	3	7	4	2	5	6	8	2	1	3	3	5	3	5	3	3	2	1	6	2	5
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	-1	1	1	-1	4	4	4	-1	8	9	10	11	11	13	13	15	16	17	18	18	20	20
5	1	-1	-1	-1	3	3	5	6	6	6	-1	-1	-1	12	12	12	-1	-1	-1	-1	19	19	19
3	2	-1	-1	-1	-1	4	4	4	-1	8	8	8	-1	-1	13	13	13	-1	-1	-1	-1	20	20
2	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	9	10											
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The δ-Bounded-Gaps Algorithm

$$D[i,j] = \begin{cases} j & \text{if } T[j] =_\delta P[i]\,,\ i,j \geq 1\,,\ \text{and } D[i-1,j-1] \geq 0 \\ j & \text{if } T[j] =_\delta P[i] \text{ and } i = 0 \\ D[i,j-1] & \text{if } T[j] \neq_\delta P[i]\,,\ j \geq 1\,,\ \text{and } D[i,j-1] \geq j - \alpha \\ -1 & \text{otherwise} \end{cases}$$

		4	3	7	4	2	5	6	8	2	1	3	3	5	3	5	3	3	2	1	6	2	5
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	-1	1	1	-1	4	4	4	-1	8	9	10	11	11	13	13	15	16	17	18	18	20	20
5	1	-1	-1	-1	3	3	5	6	6	6	-1	-1	-1	12	12	12	-1	-1	-1	-1	19	19	19
3	2	-1	-1	-1	-1	4	4	4	-1	8	8	8	-1	-1	13	13	13	-1	-1	-1	-1	20	20
2	3	-1	-1	-1	-1	-1	-1	-1	-1	-1	9	10											
6	4																						
3	5																						
8	6																						
2	7																						

m

n

The SD-Simple Algorithm

The SD-Simple algorihtm **[4]** inherits the basic idea from the δ-Bounded-Gaps algorithm and uses bit-parallelism to compute an $m \times n$ bit-matrix in combination with sparse dynamic programming techniques

Basically, the algorithm partitions each row of the matrix as a sequence of $\lceil n/w \rceil$ consecutive bit masks, each of which represents a group of w bits on that row.

The SD-Simple algorithm turns out to be among the most efficient ones, in terms of running time, in many practical cases, especially for small values of α

SD-Simple	
Time Complexity	$O(\lceil mn/w \rceil)$
Space Complexity	$\mathcal{O}(n)$

[4] K. Fredriksson and S. Grabowski: Efficient algorithms for pattern matching with general gaps, character classes, and transposition invariance. Information Retrieval, March 2008.

The (δ,α)-Sequential-Sampling Algorithm

The (δ,α)-Sequential-Sampling algorithm **[5]** is based on dynamic programming and scans the text T from left to right and for each position i of T it looks for the (δ,α)-occurrences at position i of all prefixes of the pattern P. The (δ,α)-Sequential-Sampling algorithm has an O(nm) running time and requires O(mα)-space. A much more efficient variant of it is the (δ,α)-Tuned-Sequential-Sampling algorithm, which has an average case running time of O(n), in the case in which α is assumed constant.

(δ,α)-Sequential-Sampling	
Average Case Complexity	$\mathcal{O}(n)$
Worst Case Complexity	$\mathcal{O}(nm)$
Space Complexity	$\mathcal{O}(m\alpha)$

[5] D. Cantone, S. Cristofaro, and S. Faro: An efficient algorithm for δ-approximate matching with α-bounded gaps in musical sequences, in Proceedings of 4-th International Workshop on Experimental and Efficient Algorithms, S. E. Nikoletseas, ed., vol. 3503 of LNCS, Springer-Verlag, 2005, pp. 428–439.

The (δ,α)-Sequential-Sampling Algorithm

		4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0																			
5	1	0	0	1																			
3	2	0	0	0																			
2	3	0	0	0																			
6	4	0	0	0																			
3	5	0	0	0																			
8	6	0	0	0																			

m

n

The (δ,α)-Sequential-Sampling Algorithm

		4	3	5	3	3	5	3	7	2	1	3	3	5	3	5	3	3	2	1	6	2	5
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0	1																		
5	1	0	0	1	0																		
3	2	0	0	0	1																		
2	3	0	0	0	0																		
6	4	0	0	0	0																		
3	5	0	0	0	0																		
8	6	0	0	0	0																		

m

n

The (δ,α)-Sequential-Sampling Algorithm

		4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0	1	1																	
5	1	0	0	1	0	0																	
3	2	0	0	0	1	1																	
2	3	0	0	0	0	1																	
6	4	0	0	0	0	0																	
3	5	0	0	0	0	0																	
8	6	0	0	0	0	0																	

m

n

The (δ,α)-Sequential-Sampling Algorithm

		4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0	1	1	0																
5	1	0	0	1	0	0	1																
3	2	0	0	0	1	1	0																
2	3	0	0	0	0	1	0																
6	4	0	0	0	0	0	1																
3	5	0	0	0	0	0	0																
8	6	0	0	0	0	0	0																

n

m

The (δ,α)-Sequential-Sampling Algorithm

		4	3	5	3	3	5	3	7	2	1	3	3	5	3	5	3	3	2	1	6	2	5
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0	1	1	0	1															
5	1	0	0	1	0	0	1	0															
3	2	0	0	0	1	1	0	1															
2	3	0	0	0	0	1	0	1															
6	4	0	0	0	0	0	1	0															
3	5	0	0	0	0	0	0	1															
8	6	0	0	0	0	0	0	0															

m

n

The (δ,α)-Sequential-Sampling Algorithm

$\alpha+2$

		4	3	5	3	3	5	3	7	2	1	3	3	5	3	5	3	3	2	1	6	2	5
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	0	0	1	0	1	1	0	1	0	1	1	1	1										
5	1	0	0	1	0	0	1	0	0	0	0	0	0										
3	2	0	0	0	1	1	0	1	0	0	0	0	0										
2	3	0	0	0	0	1	0	1	0	1	1	0	0										
6	4	0	0	0	0	0	1	0	1	0	0	0											
3	5	0	0	0	0	0	0	1	0	1	0	0											
8	6	0	0	0	0	0	0	0	1	0	0	0											

m

n

The (δ,α)-Sequential-Sampling Algorithm

4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

								C
2	0	0	1	0	1			1
5	1	0	0	1	0			1
3	2	0	0	0	1			0
2	3	0	0	0	0			0
6	4	0	0	0	0			0
3	5	0	0	0	0			0
8	6	0	0	0	0			0

$\longleftarrow \alpha+2 \longrightarrow$

The (δ,α)-Sequential-Sampling Algorithm

4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

							C
2	0	1	0	1	1		2
5	1	0	1	0	0		1
3	2	0	0	1	1		1
2	3	0	0	0	1		0
6	4	0	0	0	0		0
3	5	0	0	0	0		0
8	6	0	0	0	0		0

← α+2 →

The (δ,α)-Sequential-Sampling Algorithm

4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

						C
2	0	0	1	1	0	2
5	1	1	0	0	1	1
3	2	0	1	1	0	2
2	3	0	0	1	0	1
6	4	0	0	0	1	0
3	5	0	0	0	0	0
8	6	0	0	0	0	0

← α+2 →

The (δ,α)-Sequential-Sampling Algorithm

4	3	5	3	3	5	3	7	2	1	3	3	5	3	5	3	3	2	1	6	2	5
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

		3	4	5	6		C
2	0	1	1	0	1		2
5	1	0	0	1	0		1
3	2	1	1	0	1		2
2	3	0	1	0	1		1
6	4	0	0	1	0		1
3	5	0	0	0	1		0
8	6	0	0	0	0		0

$\longleftarrow \alpha+2 \longrightarrow$

The (δ,α)-Sequential-Sampling Algorithm

4	**3**	**5**	**3**	**3**	**5**	**3**	**7**	**2**	**1**	**3**	**3**	**5**	**3**	**5**	**3**	**3**	**2**	**1**	**6**	**2**	**5**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

						C
2	0	1	0	1	0	2
5	1	0	1	0	0	1
3	2	1	0	1	0	2
2	3	1	0	1	0	2
6	4	0	1	0	1	1
3	5	0	0	1	0	1
8	6	0	0	0	1	0

$\leftarrow \alpha+2 \rightarrow$

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that $\alpha < w$.

$\mathcal{M}_i$

0	0	1	0
1	0	0	1
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that $\alpha < w$.

$\mathcal{M}_i$

2	0	0	1	0	$\mathcal{C}_i^{(0)}$
5	1	0	0	1	$\mathcal{C}_i^{(1)}$
3	2	0	0	0	⋮
2	3	0	0	0	⋮
6	4	0	0	0	⋮
3	5	0	0	0	⋮
8	6	0	0	0	$\mathcal{C}_i^{(m-1)}$

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that α < w.

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$ →

$\mathcal{M}_i$

			$\mathcal{C}_i^{(0)}$
			$\mathcal{C}_i^{(1)}$
			.
			.
			.
			.
			$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=2

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that α < w.

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$

$$\mathcal{C}_i^{(j)}[0\,..\,\alpha-1] = \mathcal{C}_{i-1}^{(j)}[1\,..\,\alpha] \quad 0 \leq j < m$$

$\mathcal{M}_i$

1	0		$\mathcal{C}_i^{(0)}$
0	1		$\mathcal{C}_i^{(1)}$
0	0		.
0	0		.
0	0		.
0	0		.
0	0		$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=1

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that α < w.

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$ $\longrightarrow$

If j=0 the last bit of the bit-mask is:

- equal to 1 if T[i] δ= P[j]
- equal to 0 otherwise

1	0	$\boxed{1}$	$\mathcal{C}_i^{(0)}$
0	1		$\mathcal{C}_i^{(1)}$
0	0		.
0	0		.
0	0		.
0	0		.
0	0		$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=1

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that α < w.

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$

If j>0 the last bit of the bit-mask:

- is 1, if $T[i] =_\delta P[j]$ and $\mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}$
- otherwise it is 0

$\mathcal{M}_i$

1	0	1	$\mathcal{C}_i^{(0)}$
0	1	$\boxed{0}$	$\mathcal{C}_i^{(1)}$
0	0		.
0	0		.
0	0		.
0	0		.
0	0		$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=1

The (δ,α)-Sequential-Sampling-HBP

The basic idea is to represent each row of the matrices M as a bit mask of length α+1. Consequently, the whole matrix $\mathcal{M}_i$ can be represented as an array of m bit masks, each of which corresponds to a row of $\mathcal{M}_i$, and each of which fits in a single computer word in the case that α < w.

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$

If j>0 the last bit of the bit-mask:

- is 1, if $T[i] =_\delta P[j]$ and $\mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}$
- otherwise it is 0

$\mathcal{M}_i$

1	0	1	$\mathcal{C}_i^{(0)}$
0	1	0	$\mathcal{C}_i^{(1)}$
0	0	$\boxed{1}$	.
0	0		.
0	0		.
0	0		.
0	0		$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=1

The (δ,α)-Sequential-Sampling-HBP

Therefore, if we put $I = 01^\alpha$, we obtain

$$\mathcal{C}_i^{(j)} = \begin{cases} ((\mathcal{C}_{i-1}^{(j)} \,\&\, I) \ll 1) \,|\, 0^\alpha 1, & \text{if } T[i] =_\delta P[j] \text{ AND } (j = 0 \text{ OR } \mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}) \\ (\mathcal{C}_{i-1}^{(j)} \,\&\, I) \ll 1, & \text{otherwise}, \end{cases}$$

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$ $\longrightarrow$

$\mathcal{M}_i$

1	0	1	$\mathcal{C}_i^{(0)}$
0	1	0	$\mathcal{C}_i^{(1)}$
0	0	1	.
0	0	0	.
0	0	0	.
0	0	0	.
0	0	0	$\mathcal{C}_i^{(m-1)}$

Suppose T[i] = 3 and δ=1

The (δ,α)-Sequential-Sampling-HBP

Therefore, if we put $I = 01^\alpha$, we obtain

$$\mathcal{C}_i^{(j)} = \begin{cases} ((\mathcal{C}_{i-1}^{(j)} \,\&\, I) \ll 1) \,|\, 0^\alpha 1, & \text{if } T[i] =_\delta P[j] \text{ AND } (j = 0 \text{ OR } \mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}) \\ (\mathcal{C}_{i-1}^{(j)} \,\&\, I) \ll 1, & \text{otherwise}, \end{cases}$$

$\mathcal{M}_{i-1}$

2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	0	0	0
8	6	0	0	0

Compute $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$

$\mathcal{C}_{i-1}^{(j)}$	0	0	1	&
I	0	1	1	=
	0	0	1	$\ll 1$
$\mathcal{C}_i^{(j)}$	0	1	0	

Suppose T[i] = 3 and δ=1

$\mathcal{M}_i$

1	0	1	$\mathcal{C}_i^{(0)}$
0	1	0	$\mathcal{C}_i^{(1)}$
0	0	1	.
0	0	0	.
0	0	0	.
0	0	0	.
0	0	0	$\mathcal{C}_i^{(m-1)}$

The (δ,α)-Sequential-Sampling-HBP

- This algorithm improves the space complexity to $O(m\lceil\alpha/w\rceil)$

- The running time, which is $O(nm\lceil\alpha/w\rceil)$, is worse than that of the previous algorithm. The reason is that, in general, we need $\lceil(\alpha + 1)/w\rceil$ computer words to represent a bit mask of length $\alpha + 1$. Consequently, any update of the entry $C[j]$ costs $O(\lceil\alpha/w\rceil)$-time.

- However, we notice that in almost all practical applications in music information retrieval the value of the gap bound α is at most 10 (or less), therefore smaller than the size w of a computer word (which is 32 or 64)

The (δ,α)-Sequential-Sampling-HBP

(δ, α)-**Sequential-Sampling-HBP**$(P, m, T, n, \delta, \alpha)$

```
for i := 0 to m − 1 do
    C[i] := 0^(α+1)
I := 01^α
for i := 0 to n − 1 do
    for j := m − 1 downto 1 do
        C[j] := (C[j] & I) ≪ 1
        if T[i] =_δ P[j] AND C[j − 1] ≠ 0^(α+1)
        then C[j] := C[j] | 0^α 1
    C[0] := (C[0] & I) ≪ 1
    if T[i] =_δ P[0] then
        C[0] := C[0] | 0^α 1
    if (C[m − 1] & 0^α 1) ≠ 0^(α+1) then
        print(i)
```

The (δ,α)-Sequential-Sampling-HBP

(δ, α)-**Sequential-Sampling-HBP**$(P, m, T, n, \delta, \alpha)$

1. **for** $i := 0$ **to** $m-1$ **do**
2. $\quad C[i] := 0^{\alpha+1}$
3. $I := 01^{\alpha}$
4. **for** $i := 0$ **to** $n-1$ **do**
5. $\quad$ **for** $j := m-1$ **downto** 1 **do**
6. $\quad\quad C[j] := (C[j]\,\&\,I) \ll 1$
7. $\quad\quad$ **if** $T[i] =_\delta P[j]$ AND $C[j-1] \neq 0^{\alpha+1}$
8. $\quad\quad$ **then** $C[j] := C[j] \,|\, 0^{\alpha}1$
9. $\quad C[0] := (C[0]\,\&\,I) \ll 1$
10. $\quad$ **if** $T[i] =_\delta P[0]$ **then**
11. $\quad\quad C[0] := C[0] \,|\, 0^{\alpha}1$
12. $\quad$ **if** $(C[m-1]\,\&\,0^{\alpha}1) \neq 0^{\alpha+1}$ **then**
13. $\quad\quad$ **print**(i)

Observe that during an iteration the for-loop at line 5 relative to a value of $j > 0$ has no effect if the items C[j] and C[j –1] are both null.

Therefore, it is enough to scan only positions j of the array C such that $C[j] \neq 0^{\alpha+1}$

The (δ,α)-Tuned-Sequential-Sampling-HBP

To perform such process, the positions j of the nonnull items of C (i.e., the j's such that $C[j] \neq 0^{\alpha+1}$) are maintained into an ordered, linked list L, which is scanned from the highest value of j up to the lowest one.

$\mathcal{M}_{i-1}$

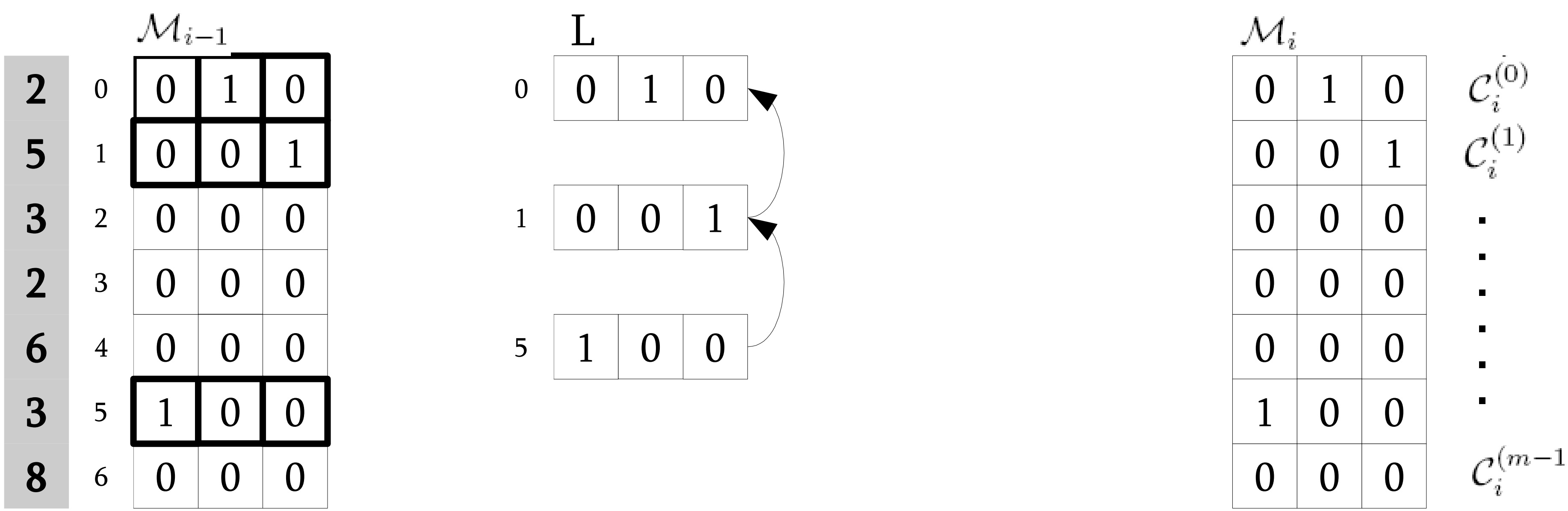
2	0	0	1	0
5	1	0	0	1
3	2	0	0	0
2	3	0	0	0
6	4	0	0	0
3	5	1	0	0
8	6	0	0	0

$\mathcal{M}_{i}$

0	1	0	$\mathcal{C}_i^{(0)}$
0	0	1	$\mathcal{C}_i^{(1)}$
0	0	0	⋮
0	0	0	
0	0	0	
1	0	0	
0	0	0	$\mathcal{C}_i^{(m-1)}$

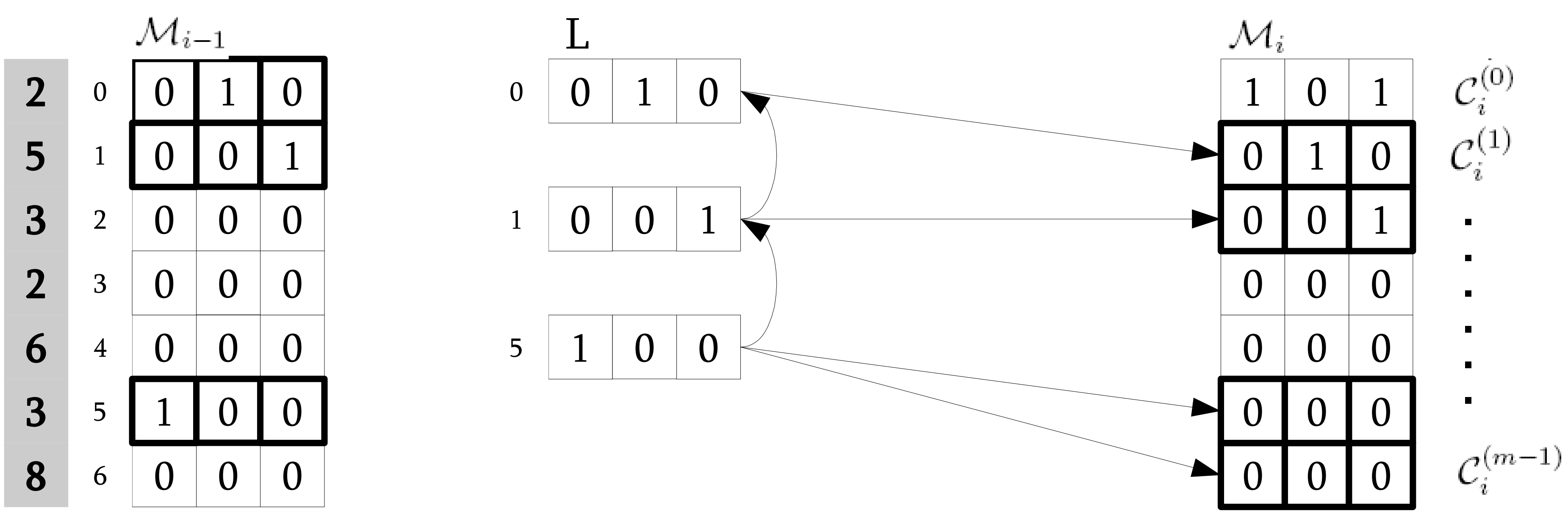
The (δ,α)-Tuned-Sequential-Sampling-HBP

To perform such process, the positions j of the nonnull items of C (i.e., the j's such that $C[j] \neq 0^{\alpha+1}$) are maintained into an ordered, linked list L, which is scanned from the highest value of j up to the lowest one.



The (δ,α)-Tuned-Sequential-Sampling-HBP

To perform such process, the positions j of the nonnull items of C (i.e., the j's such that C[j] ≠ $0^{\alpha+1}$) are maintained into an ordered, linked list L, which is scanned from the highest value of j up to the lowest one.



The (δ,α)-Tuned-Sequential-Sampling-HBP

(δ, α)-**Tuned-Sequential-Sampling-HBP**$(P, m, T, n, \delta, \alpha)$

```
for i := 0 to m − 1 do C[i] := 0^(α+1)
next[0] := next[m] := m
I := 01^α
for i := 0 to n − 1 do
   p := m
   j := next[p]
   while j < m do
      if j < m − 1 AND T[i] =_δ P[j + 1] then
         C[j + 1] := C[j + 1] | 0^α 1
         if p > j + 1 then
            next[p] := j + 1
            next[j + 1] := j
            p := j + 1
      C[j] := (C[j] & I) ≪ 1
      if C[j] = 0^(α+1) then next[p] := next[j]
      else p := j
      j := next[p]
   if T[i] =_δ P[0] then
      C[0] := C[0] | 0^α 1
      if p > 0 then next[p] := 0
   if (C[m − 1] & 0^α 1) ≠ 0^(α+1) then print(i)
```

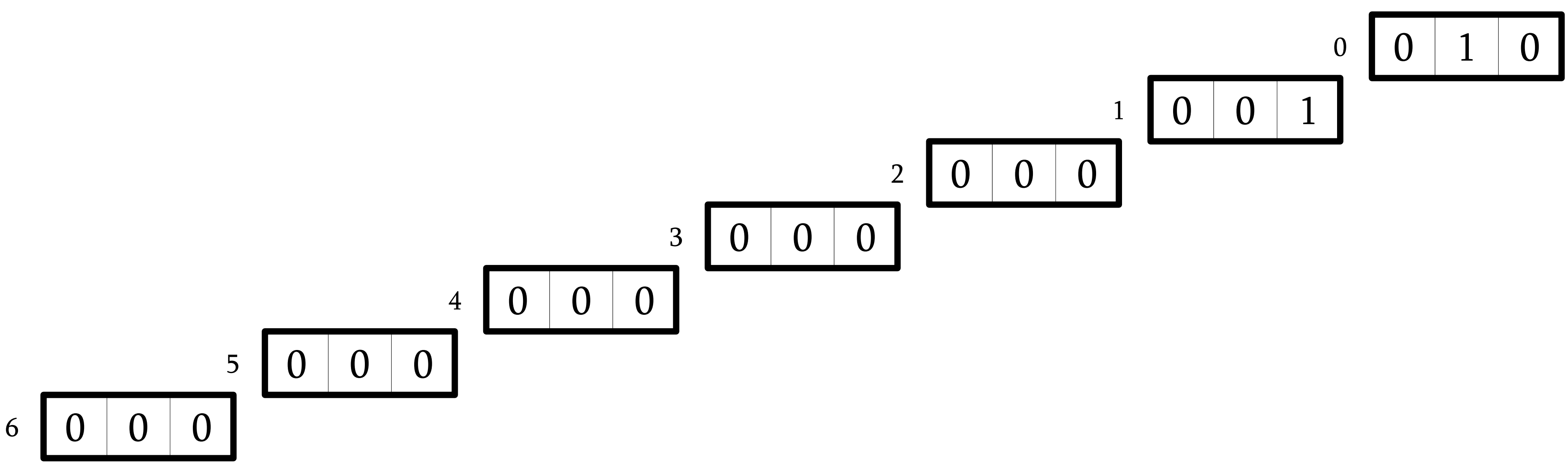
The (δ,α)-Sequential-Sampling-BP

In the last variant of the (δ, α)-Sequential-Sampling algorithm the matrix M is represented as a single bit mask of length $L = (\alpha + 1)m$, obtained by concatenating the bit masks corresponding to the rows of M.

0	0	1	0
1	0	0	1
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0

The (δ,α)-Sequential-Sampling-BP

In the last variant of the (δ, α)-Sequential-Sampling algorithm the matrix M is represented as a single bit mask of length $L = (\alpha + 1)m$, obtained by concatenating the bit masks corresponding to the rows of M.



The (δ,α)-Sequential-Sampling-BP

In the last variant of the (δ, α)-Sequential-Sampling algorithm the matrix M is represented as a single bit mask of length L = (α + 1)m, obtained by concatenating the bit masks corresponding to the rows of M.

6			5			4			3			2			1			0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

The (δ,α)-Sequential-Sampling-BP

In the last variant of the (δ, α)-Sequential-Sampling algorithm the matrix M is represented as a single bit mask of length L = (α + 1)m, obtained by concatenating the bit masks corresponding to the rows of M.

6			5			4			3			2			1			0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

$$\mathcal{B}_i = \mathcal{C}_i^{(m-1)} \mathcal{C}_i^{(m-2)} \ldots \mathcal{C}_i^{(0)}$$

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

Suppose T[i] = 3 and δ=1

		8			**3**			**6**			**2**			**3**			**5**			**2**			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$

3 3

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

Suppose T[i] = 3 and δ=1

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$

$$\mathcal{X}_i^{(j)} = \begin{cases} 0^{\alpha}1, & \text{if } T[i] =_{\delta} P[j] \text{ AND } (j = 0 \text{ OR } \mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}) \\ 0^{\alpha+1}, & \text{otherwise}, \end{cases}$$

3 3

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

Suppose T[i] = 3 and δ=1

		8			3			6			2			3			5			2			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$

$$\mathcal{X}_i^{(j)} = \begin{cases} 0^{\alpha}1, & \text{if } T[i] =_{\delta} P[j] \text{ AND } (j = 0 \text{ OR } \mathcal{C}_{i-1}^{(j-1)} \neq 0^{\alpha+1}) \\ 0^{\alpha+1}, & \text{otherwise}, \end{cases}$$

3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

$$\mathcal{X}_i = \mathcal{X}_i^{(m-1)} \mathcal{X}_i^{(m-2)} \ldots \mathcal{X}_i^{(0)}$$

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \mid \mathcal{X}_i$$

		8			3			6			2			3			5			2			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
		0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	I =
3	3																						

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$ &
		0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	I =
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 0 0	
3	3								

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8			**3**			**6**			**2**			**3**			**5**			**2**		
		6			5			4			3			2			1			0		
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	3																					

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$ &
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 0 0	
3	3								

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8	3	6	2	3	5	2	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 0 0	$\ll 1$
3	3								

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \mid \mathcal{X}_i$$

		8			3			6			2			3			5			2			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	$\ll 1$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
3	3																						

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8			3			6			2			3			5			2			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	

3	3

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 1 0	0 0 0	
3	3								

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \mid \mathcal{X}_i$$

		8			**3**			**6**			**2**			**3**			**5**			**2**			
		6			5			4			3			2			1			0			
5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	$\vert$
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	$\mathcal{X}_i =$
3	3																						

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \mid \mathcal{X}_i$$

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 1 0	0 0 0	$\mid$
		0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 0 0	0 0 1	$\mathcal{X}_i =$
3	3	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	0 0 1	$\mathcal{B}_i$

The (δ,α)-Sequential-Sampling-BP

Assuming such representation for the matrices M_i as single bit masks, the task is to find an efficient way to compute bit-parallely the bit mask B_i from the bit mask B_{i-1}.

$$\mathcal{B}_i = ((\mathcal{B}_{i-1} \,\&\, I) \ll 1) \,|\, \mathcal{X}_i$$

		8	**3**	**6**	**2**	**3**	**5**	**2**	
		6	5	4	3	2	1	0	
5	2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	$\mathcal{B}_{i-1}$
3	3	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1	0 1 0	0 0 1	$\mathcal{B}_i$

The (δ,α)-Sequential-Sampling-BP

Now we need only to be able to compute effectively the bit mask $\mathcal{X}_i$ from the bit mask $\mathcal{B}_{i-1}$, which we do as follows. For each symbol s of the alphabet Σ and each $0 \leq j < m$, let $\mathsf{b}_s^{(j)}$ be the bit value 1, if $s =_\delta P[j]$ holds, otherwise let $\mathsf{b}_s^{(j)}$ be the bit value 0.

$$\mathcal{H}(s) = \mathsf{0}^\alpha(\mathsf{b}_s^{(m-1)}\mathsf{0}^\alpha)(\mathsf{b}_s^{(m-2)}\mathsf{0}^\alpha)\ldots(\mathsf{b}_s^{(1)}\mathsf{0}^\alpha)\mathsf{b}_s^{(0)}$$

The (δ,α)-Sequential-Sampling-BP

Now we need only to be able to compute effectively the bit mask $\mathcal{X}_i$ from the bit mask $\mathcal{B}_{i-1}$, which we do as follows. For each symbol s of the alphabet Σ and each 0 ≤ j < m, let $\mathsf{b}_s^{(j)}$ be the bit value 1, if s =δ P[j] holds, otherwise let $\mathsf{b}_s^{(j)}$ be the bit value 0.

$$\mathcal{H}(s) = 0^\alpha(\mathsf{b}_s^{(m-1)}0^\alpha)(\mathsf{b}_s^{(m-2)}0^\alpha)\dots(\mathsf{b}_s^{(1)}0^\alpha)\mathsf{b}_s^{(0)}$$

P	**8**	**3**	**6**	**2**	**3**	**5**	**2**

$\mathcal{H}(1)$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
$\mathcal{H}(2)$	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1
$\mathcal{H}(3)$	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((((\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{(\mathcal{B}_{i-1} \,\&\, I)} + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

6			5			4			3			2			1			0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{(\mathcal{B}_{i-1} \& I) + I}) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \& \mathcal{H}(T[i]))$$

6			5			4			3			2			1			0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	+
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	0	1	0	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = (((((\underline{(\mathcal{B}_{i-1} \,\&\, I) + I}) \mid \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \mid 0^{L-1}1) \,\&\, \mathcal{H}(T[i]))$$

6			5			4			3			2			1			0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	+
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	0	1	0	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = (((((\underline{(\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1}} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

6			5			4			3			2			1			0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ &
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	+
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	I =
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	0	1	0	1	|
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	$\mathcal{B}_{i-1}$ =
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{(\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1}} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

6			5			4			3			2			1			0		
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{(\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i]))$$

6			5			4			3			2			1			0			
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	$\&$
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$01^{L-1} =$
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = (((\underline{((\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \;\ll 1}) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

6			5			4			3			2			1			0			
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	$\&$
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$01^{L-1} =$
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	$\ll 1 =$
1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{((\mathcal{B}_{i-1} \,\&\, I) + I) \mid \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \mid 0^{L-1}1)} \,\&\, \mathcal{H}(T[i]))$$

6			5			4			3			2			1			0			
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	$\&$
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	$01^{L-1} =$
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	$\ll 1 =$
1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	$\mid$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	$0^{L-1}1 =$
1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((\underline{(\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \,\&\, 01^{L-1}) \ll 1) \,|\, 0^{L-1}1)} \,\&\, \mathcal{H}(T[i]))$$

6			5			4			3			2			1			0		
1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = (\underline{((((\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])}$$

	6			5			4			3			2			1			0			
	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	$\&$
	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	$\mathcal{H}(3)$ =
$\mathcal{X}_i$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	

The (δ,α)-Sequential-Sampling-BP

Then $\mathcal{X}_i$ can be obtained in constant time with the following relation.

$$\mathcal{X}_i = ((((((\mathcal{B}_{i-1} \,\&\, I) + I) \,|\, \mathcal{B}_{i-1} \;\&\; 01^{L-1}) \ll 1) \,|\, 0^{L-1}1) \,\&\, \mathcal{H}(T[i])$$

	6			5			4			3			2			1			0		
$\mathcal{X}_i$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

The (δ,α)-Sequential-Sampling-BP

(δ, α)-**Sequential-Sampling-BP**$(P,\ m,\ T,\ n,\ \delta,\ \alpha)$

$L := (\alpha + 1)m$
for $s \in \Sigma$ **do** $H[s] := 0^L$
$I := 0^{L-\alpha}1^{\alpha}$
$U := 0^{L-1}1$
for $j := 0$ **to** $m - 1$ **do**
  for $s \in \Sigma \cap [P[j] - \delta \mathinner{..} P[j] + \delta]$ **do**
    $H[s] := H[s] \,|\, U$
  if $j < m - 1$ **then**
    $I := (I \ll (\alpha + 1)) \,|\, 0^{L-\alpha}1^{\alpha}$
    $U := U \ll (\alpha + 1)$
$F := 01^{L-1}$
$B := 0^L$
for $i := 0$ **to** $n - 1$ **do**
  $W := ((B \,\&\, I) + I) \,|\, B$
  $X := (((W \,\&\, F) \ll 1) \,|\, 0^{L-1}1) \,\&\, H[T[i]]$
  $B := ((B \,\&\, I) \ll 1) \,|\, X$
  if $(B \,\&\, U) \neq 0^L$ **then print**(i)

(δ,α)-Sequential-Sampling
Time Complexity
$\mathcal{O}((\sigma + n + m\delta)\lceil (m\alpha)/w \rceil)$
Space Complexity
$\mathcal{O}(\sigma\lceil (m\alpha)/w \rceil)$

Experimental Results

We have performed two main sets of experimental data:

ES1: (for general length patterns)

- (δ,α)-Tuned-Sequential-Sampling-HBP algorithm (TSS-HBP)
- SDP-Simple algorithm (SD-S)

ES2: (for short patterns)

- (δ,α)-Tuned-Sequential-Sampling-HBP algorithm (TSS-HBP)
- (δ,α)-Sequential-Sampling-BP (SS-BP)
- (δ,α)-Shift-And algorithm (DA-NFA)
- DA-mloga-bits algorithm (DA-CNFA)

Experimental Results on Real Data

Tests on the real music text buffer have been performed on a fixed text sequence T of length n = 2.982.507 obtained by combining a set of various classical pieces in MIDI format, with an overall alphabet of 76 distinct symbols, i.e., the MIDI values of the notes of the pieces. For each m we have randomly selected a set of 150 substrings of T of length m which subsequently have been searched for in T

Experimental Results on Real Data

Tests on the real music text buffer have been performed on a fixed text sequence T of length n = 2.982.507 obtained by combining a set of various classical pieces in MIDI format, with an overall alphabet of 76 distinct symbols, i.e., the MIDI values of the notes of the pieces. For each m we have randomly selected a set of 150 substrings of T of length m which subsequently have been searched for in T

Experimental results on a Real Music Problem (Es1)

ALGS	(δ, α)	$m=6$	$m=8$	$m=10$	$m=20$	$m=30$	$m=40$	$m=50$	$m=60$	$m=70$	$m=85$	$m=100$
TSS-HBP	$(1,2)$	**2.36**	**2.40**	**2.44**	**2.50**	**2.54**	**2.30**	**2.27**	**2.39**	**2.42**	**2.44**	**2.48**
SDP-S	$(1,2)$	3.50	3.38	3.79	3.75	3.87	3.42	3.76	3.70	3.66	3.81	3.73
TSS-HBP	$(1,5)$	**4.14**	**4.33**	**4.95**	**4.93**	**5.17**	**4.35**	**4.53**	**4.85**	**4.72**	**4.85**	**4.63**
SDP-S	$(1,5)$	4.93	5.15	5.95	6.03	6.16	5.39	5.55	5.58	5.75	5.83	5.71
TSS-HBP	$(1,8)$	**5.65**	**6.36**	**7.69**	**7.93**	**8.13**	**6.67**	**7.05**	**7.45**	**7.31**	**7.61**	**7.33**
SDP-S	$(1,8)$	6.15	6.79	8.06	8.76	8.65	7.58	7.83	8.17	8.05	8.52	8.07
TSS-HBP	$(3,2)$	**5.11**	**4.78**	**5.27**	**5.16**	**5.90**	**4.87**	**5.05**	**5.53**	**5.41**	**4.97**	**5.57**
SDP-S	$(3,2)$	6.60	6.27	7.00	6.81	7.38	6.40	6.77	7.01	7.08	6.77	7.06
TSS-HBP	$(3,5)$	**9.57**	**10.00**	**12.40**	**12.96**	**14.61**	**12.68**	**12.50**	**13.42**	**13.17**	**12.59**	**13.49**
SDP-S	$(3,5)$	10.59	10.73	12.92	14.52	16.32	14.35	14.11	15.27	14.75	14.12	15.23
TSS-HBP	$(3,8)$	**11.13**	**12.63**	**16.59**	**20.43**	**24.57**	**22.56**	**21.45**	**24.19**	**23.53**	**21.83**	**23.60**
SDP-S	$(3,8)$	12.73	14.20	18.11	23.41	28.91	27.10	25.09	28.86	28.97	26.04	28.49
TSS-HBP	$(5,2)$	**9.03**	**9.05**	**10.54**	**10.58**	**15.46**	**18.78**	**19.95**	**18.98**	**19.32**	**18.88**	**21.63**
SDP-S	$(5,2)$	10.39	10.49	11.68	12.44	18.66	22.22	23.59	22.60	22.92	22.83	25.07
TSS-HBP	$(5,5)$	**13.14**	**15.02**	**19.46**	**23.73**	**24.83**	**25.06**	**27.69**	**51.78**	**33.51**	**28.93**	**37.44**
SDP-S	$(5,5)$	15.85	17.91	22.64	28.41	30.73	31.15	35.34	65.30	41.79	36.76	47.81
TSS-HBP	$(5,8)$	**12.94**	**15.92**	**21.29**	**30.10**	**36.26**	**36.67**	**47.64**	**48.03**	**52.02**	**55.14**	**52.40**
SDP-S	$(5,8)$	17.59	20.36	26.91	38.40	46.99	48.06	63.97	64.66	70.58	76.90	75.33

Experimental Results on Real Data

Tests on the real music text buffer have been performed on a fixed text sequence T of length n = 2.982.507 obtained by combining a set of various classical pieces in MIDI format, with an overall alphabet of 76 distinct symbols, i.e., the MIDI values of the notes of the pieces. For each m we have randomly selected a set of 150 substrings of T of length m which subsequently have been searched for in T

Experimental results on a Real Music Problem (Es2)

ALGS	(δ,α)	$m=6$	$m=8$	$m=10$	$m=12$	$m=14$	$m=16$		ALGS	(δ,α)	$m=6$	$m=8$	$m=10$
TSS-HBP	$(1,1)$	2.00	1.88	2.04	2.06	2.00	1.98		TSS-HBP	$(1,2)$	2.36	2.27	2.42
SS-BP	$(1,1)$	**1.00**	**0.84**	**0.96**	**0.94**	**0.94**	**0.88**		SS-BP	$(1,2)$	**0.92**	**0.90**	**0.82**
DA-NFA	$(1,1)$	1.02	1.00	1.00	1.00	1.04	1.00		DA-NFA	$(1,2)$	1.02	1.00	1.05
DA-CNFA	$(1,1)$	9.15	9.14	9.03	9.12	9.13	9.17		DA-CNFA	$(1,2)$	9.22	9.19	9.09
TSS-HBP	$(3,1)$	3.26	3.28	3.29	3.41	3.39	3.48		TSS-HBP	$(3,2)$	5.14	4.58	4.99
SS-BP	$(3,1)$	**0.94**	**0.94**	**0.96**	**0.88**	**0.94**	**0.98**		SS-BP	$(3,2)$	**0.92**	**0.94**	**0.94**
DA-NFA	$(3,1)$	1.06	1.04	1.05	1.02	1.04	1.02		DA-NFA	$(3,2)$	1.16	1.14	1.06
DA-CNFA	$(3,1)$	9.34	9.35	9.23	9.49	9.24	9.20		DA-CNFA	$(3,2)$	9.40	9.25	9.30
TSS-HBP	$(5,1)$	5.13	5.35	4.93	5.69	6.02	5.28		TSS-HBP	$(5,2)$	8.70	8.87	9.91
SS-BP	$(5,1)$	1.08	**0.92**	**0.90**	**0.88**	**0.96**	**0.84**		SS-BP	$(5,2)$	**1.18**	**1.06**	**1.02**
DA-NFA	$(5,1)$	**1.07**	1.06	1.04	1.06	1.04	1.06		DA-NFA	$(5,2)$	1.20	1.08	1.06
DA-CNFA	$(5,1)$	9.38	9.25	9.26	9.25	9.33	9.29		DA-CNFA	$(5,2)$	9.54	9.44	9.28

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

Experimental results on a Rand50 problem (Es1)

ALGS	(δ, α)	$m = 6$	$m = 8$	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$	$m = 70$	$m = 85$	$m = 100$
TSS-HBP	(1,2)	**3.02**	**2.92**	**2.98**	**2.84**	**2.94**	**2.94**	**3.03**	**2.90**	**2.92**	**2.98**	**2.96**
SDP-S	(1,2)	4.60	4.78	4.58	4.69	4.75	4.73	4.80	4.77	4.63	4.65	4.77
TSS-HBP	(1,5)	**4.33**	**4.17**	**4.35**	**4.29**	**4.35**	**4.27**	**4.39**	**4.32**	**4.23**	**4.25**	**4.35**
SDP-S	(1,5)	6.19	6.11	6.25	6.21	6.17	6.19	6.15	6.01	6.19	6.09	6.11
TSS-HBP	(1,8)	**5.65**	**5.59**	**5.79**	**5.89**	**5.89**	**5.69**	**5.73**	**5.73**	**5.77**	**5.68**	**5.79**
SDP-S	(1,8)	7.43	7.65	7.79	7.61	7.71	7.67	7.81	7.59	7.63	7.67	7.67
TSS-HBP	(3,2)	**5.89**	**5.81**	**5.79**	**5.95**	**5.94**	**5.91**	**5.77**	**5.84**	**6.03**	**5.84**	**5.71**
SDP-S	(3,2)	8.85	8.73	8.85	8.83	8.83	8.62	8.87	8.79	8.88	8.85	8.79
TSS-HBP	(3,5)	**12.24**	**12.96**	**13.31**	**13.84**	**13.75**	**13.47**	**13.73**	**13.71**	**14.09**	**13.95**	**13.58**
SDP-S	(3,5)	13.10	14.63	15.56	16.27	16.09	15.80	16.13	16.28	16.57	16.28	16.11
TSS-HBP	(3,8)	17.38	20.48	22.83	**26.10**	**26.29**	**25.95**	**26.79**	**26.46**	**26.99**	**51.53**	**50.98**
SDP-S	(3,8)	**17.22**	**19.63**	**22.61**	29.15	29.75	29.28	30.56	30.32	30.64	59.02	58.44
TSS-HBP	(5,2)	**11.49**	**11.79**	**11.68**	**11.79**	**11.99**	**11.94**	**17.55**	**22.87**	**21.73**	**22.27**	**22.07**
SDP-S	(5,2)	14.11	14.82	15.28	15.12	15.28	15.51	22.95	29.34	28.47	29.10	28.77
TSS-HBP	(5,5)	**22.91**	**27.01**	**29.66**	**35.88**	**37.35**	**37.61**	**68.71**	**39.97**	**36.32**	**64.85**	**36.72**
SDP-S	(5,5)	24.04	27.65	30.35	40.83	44.26	45.15	82.58	47.06	43.83	77.20	43.95
TSS-HBP	(5,8)	**26.53**	**34.68**	**43.38**	**121.11**	**175.83**	**212.14**	**231.21**	**257.04**	**285.96**	**329.08**	**320.51**
SDP-S	(5,8)	29.82	37.62	46.58	135.99	205.92	254.88	281.20	318.26	357.93	429.03	422.84

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

Experimental results on a Rand90 problem (Es1)

ALGS	(δ, α)	$m = 6$	$m = 8$	$m = 10$	$m = 20$	$m = 30$	$m = 40$	$m = 50$	$m = 60$	$m = 70$	$m = 85$	$m = 100$
TSS-HBP	(1,2)	**2.27**	**2.27**	**2.40**	**2.42**	**2.40**	**2.38**	**2.38**	**2.26**	**2.32**	**2.30**	**2.36**
SDP-S	(1,2)	3.70	3.78	3.64	3.71	3.71	3.71	3.63	3.77	3.69	3.69	3.67
TSS-HBP	(1,5)	**2.94**	**3.03**	**3.11**	**3.00**	**3.05**	**2.93**	**2.96**	**2.96**	**2.86**	**2.94**	**2.97**
SDP-S	(1,5)	4.42	4.31	4.27	4.43	4.25	4.37	4.32	4.39	4.49	4.43	4.36
TSS-HBP	(1,8)	**3.57**	**3.21**	**3.61**	**3.36**	**3.35**	**3.43**	**3.36**	**3.41**	**3.27**	**3.57**	**3.45**
SDP-S	(1,8)	4.97	4.81	4.87	5.00	4.97	4.87	4.91	4.93	4.95	4.97	4.87
TSS-HBP	(3,2)	**3.41**	**3.51**	**3.55**	**3.39**	**3.47**	**3.49**	**3.50**	**4.93**	**6.59**	**6.53**	**6.66**
SDP-S	(3,2)	5.40	5.37	5.40	5.39	5.42	5.40	5.36	7.47	10.37	10.15	10.17
TSS-HBP	(3,5)	**5.59**	**5.55**	**5.71**	**5.57**	**5.81**	**5.71**	**5.59**	**5.63**	**5.65**	**5.61**	**5.61**
SDP-S	(3,5)	7.66	7.63	7.92	7.62	7.74	7.64	7.76	7.68	7.60	7.56	7.66
TSS-HBP	(3,8)	**8.06**	**8.25**	**8.33**	**8.32**	**8.51**	**8.45**	**8.24**	**8.28**	**8.39**	**8.29**	**8.15**
SDP-S	(3,8)	9.59	10.17	10.56	10.28	10.30	10.19	10.38	10.24	10.51	10.24	10.31
TSS-HBP	(5,2)	**7.11**	**7.01**	**9.86**	**9.66**	**9.28**	**9.68**	**9.76**	**9.63**	**9.63**	**9.72**	**9.75**
SDP-S	(5,2)	9.55	9.57	14.81	14.63	14.36	14.65	14.46	14.60	14.53	14.77	14.68
TSS-HBP	(5,5)	**10.04**	**10.54**	**10.81**	**10.79**	**10.45**	**10.85**	**10.75**	**10.77**	**10.79**	**10.93**	**10.82**
SDP-S	(5,5)	11.43	12.43	13.28	13.17	12.77	13.15	13.09	12.99	12.93	13.29	13.39
TSS-HBP	(5,8)	**14.48**	16.77	**17.86**	**19.45**	**19.39**	**19.80**	**19.46**	**19.57**	**19.59**	**19.85**	**19.86**
SDP-S	(5,8)	14.53	**16.72**	18.82	21.70	21.55	21.81	21.69	21.71	21.62	21.99	22.06

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

EXPERIMENTAL RESULTS ON A Rand130 PROBLEM (Es1)

ALGS	(δ,α)	$m=6$	$m=8$	$m=10$	$m=20$	$m=30$	$m=40$	$m=50$	$m=60$	$m=70$	$m=85$	$m=100$
TSS-HBP	$(1,2)$	**2.16**	**2.12**	**2.14**	**2.12**	**2.14**	**2.10**	**2.12**	**2.14**	**2.24**	**2.14**	**2.15**
SDP-S	$(1,2)$	3.37	3.30	3.44	3.53	3.41	3.38	3.34	3.47	3.32	3.33	3.34
TSS-HBP	$(1,5)$	**2.61**	**2.56**	**2.52**	**2.60**	**2.62**	**2.44**	**2.50**	**2.52**	**2.53**	**2.50**	**2.54**
SDP-S	$(1,5)$	3.72	3.83	3.74	3.66	3.70	3.78	3.70	3.81	3.80	3.83	3.75
TSS-HBP	$(1,8)$	**2.92**	**2.78**	**2.96**	**2.78**	**2.88**	**2.80**	**2.76**	**2.89**	**2.85**	**2.82**	**2.80**
SDP-S	$(1,8)$	4.15	4.33	4.09	4.12	4.06	4.08	4.15	4.07	4.06	4.11	4.09
TSS-HBP	$(3,2)$	**2.83**	**2.95**	**2.83**	**2.84**	**2.83**	**2.84**	**2.74**	**2.81**	**2.89**	**2.88**	**2.80**
SDP-S	$(3,2)$	4.42	4.57	4.59	4.52	4.62	4.59	4.59	4.52	4.60	4.39	4.58
TSS-HBP	$(3,5)$	**4.07**	**4.04**	**4.15**	**4.04**	**3.94**	**4.05**	**4.03**	**4.01**	**4.10**	**4.07**	**7.65**
SDP-S	$(3,5)$	5.66	5.66	5.66	5.68	5.79	5.77	5.68	5.72	5.70	5.78	10.76
TSS-HBP	$(3,8)$	**5.08**	**4.96**	**5.23**	**5.17**	**5.13**	**5.11**	**5.18**	**5.22**	**5.20**	**5.04**	**5.19**
SDP-S	$(3,8)$	6.87	6.95	7.04	7.01	6.99	6.94	6.99	6.96	6.97	6.89	6.99
TSS-HBP	$(5,2)$	**3.78**	**3.78**	**3.84**	**3.68**	**3.72**	**6.14**	**7.14**	**7.06**	**7.09**	**7.05**	**6.91**
SDP-S	$(5,2)$	5.79	5.74	5.93	5.71	5.66	9.69	10.89	10.91	10.95	10.96	10.77
TSS-HBP	$(5,5)$	**9.14**	**9.39**	**9.78**	**9.53**	**9.77**	**9.56**	**9.62**	**9.82**	**9.86**	**9.71**	**9.46**
SDP-S	$(5,5)$	10.39	11.27	11.52	11.48	11.52	11.39	11.53	11.40	11.67	11.50	11.31
TSS-HBP	$(5,8)$	**10.23**	**10.15**	**12.34**	**11.96**	**11.82**	**12.00**	**11.92**	**11.97**	**12.14**	**11.94**	**11.69**
SDP-S	$(5,8)$	12.20	12.29	16.42	15.68	15.88	15.78	15.88	15.96	15.89	15.94	15.72

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

EXPERIMENTAL RESULTS ON A Rand50 PROBLEM (Es2)

ALGS	(δ, α)	$m = 6$	$m = 8$	$m = 10$	$m = 12$	$m = 14$	$m = 16$		ALGS	(δ, α)	$m = 6$	$m = 8$	$m = 10$
TSS-HBP	$(1, 1)$	2.68	2.76	2.66	2.82	2.68	2.78		TSS-HBP	$(1, 2)$	3.05	2.98	2.92
SS-BP	$(1, 1)$	**1.68**	**1.68**	**1.58**	**1.52**	**1.64**	**1.58**		SS-BP	$(1, 2)$	**1.72**	**1.68**	1.78
DA-NFA	$(1, 1)$	1.94	1.70	1.84	1.92	1.86	1.76		DA-NFA	$(1, 2)$	1.90	1.81	**1.70**
DA-CNFA	$(1, 1)$	16.07	15.88	15.96	15.95	16.04	15.92		DA-CNFA	$(1, 2)$	16.07	16.06	15.95
TSS-HBP	$(3, 1)$	4.52	4.46	4.46	4.60	4.58	4.56		TSS-HBP	$(3, 2)$	5.95	5.81	5.73
SS-BP	$(3, 1)$	**1.74**	**1.64**	**1.74**	**1.67**	**1.55**	1.73		SS-BP	$(3, 2)$	**1.79**	**1.78**	**1.78**
DA-NFA	$(3, 1)$	1.92	1.89	1.84	1.74	1.92	**1.72**		DA-NFA	$(3, 2)$	1.84	1.80	1.92
DA-CNFA	$(3, 1)$	16.68	16.28	16.30	16.36	16.34	16.34		DA-CNFA	$(3, 2)$	16.53	16.33	16.24
TSS-HBP	$(5, 1)$	10.37	13.13	13.49	13.55	13.49	13.91		TSS-HBP	$(5, 2)$	11.35	11.57	11.57
SS-BP	$(5, 1)$	**2.46**	**3.44**	**3.30**	**3.36**	**3.43**	**3.22**		SS-BP	$(5, 2)$	**1.82**	**1.74**	**1.68**
DA-NFA	$(5, 1)$	2.70	3.56	3.51	3.46	3.54	3.50		DA-NFA	$(5, 2)$	1.94	1.86	1.84
DA-CNFA	$(5, 1)$	23.32	31.23	31.16	31.28	31.05	31.15		DA-CNFA	$(5, 2)$	16.58	16.35	16.31

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

EXPERIMENTAL RESULTS ON A Rand90 PROBLEM (Es2)

ALGS	(δ,α)	$m=6$	$m=8$	$m=10$	$m=12$	$m=14$	$m=16$		ALGS	(δ,α)	$m=6$	$m=8$	$m=10$
TSS-HBP	(1,1)	2.24	2.24	2.21	2.28	2.25	2.30		TSS-HBP	(1,2)	2.38	2.30	2.36
SS-BP	(1,1)	**1.68**	**1.71**	**1.64**	**1.68**	**1.70**	**1.68**		SS-BP	(1,2)	**1.68**	**1.70**	**1.68**
DA-NFA	(1,1)	1.84	1.78	1.86	1.76	1.82	1.80		DA-NFA	(1,2)	2.02	1.78	1.84
DA-CNFA	(1,1)	16.12	15.92	15.98	16.02	15.94	15.96		DA-CNFA	(1,2)	16.14	15.94	15.88
TSS-HBP	(3,1)	3.16	3.03	3.09	3.03	3.05	2.97		TSS-HBP	(3,2)	3.59	3.29	3.48
SS-BP	(3,1)	1.79	**1.78**	**1.70**	**1.74**	**1.74**	1.84		SS-BP	(3,2)	**1.76**	**1.83**	**1.72**
DA-NFA	(3,1)	**1.76**	1.84	1.80	1.82	1.82	**1.74**		DA-NFA	(3,2)	1.89	1.88	1.84
DA-CNFA	(3,1)	16.57	16.22	16.34	16.39	16.28	16.30		DA-CNFA	(3,2)	16.56	16.33	16.38
TSS-HBP	(5,1)	3.99	4.09	4.07	4.00	4.05	3.97		TSS-HBP	(5,2)	5.26	4.97	4.96
SS-BP	(5,1)	**1.86**	**1.68**	**1.74**	1.77	**1.60**	**1.68**		SS-BP	(5,2)	**1.72**	**1.76**	**1.76**
DA-NFA	(5,1)	**1.86**	1.88	1.78	**1.76**	1.94	1.88		DA-NFA	(5,2)	1.84	1.78	1.92
DA-CNFA	(5,1)	16.51	16.31	16.39	16.30	16.35	16.34		DA-CNFA	(5,2)	16.47	16.27	16.31

Experimental Results on Randσ problem

Each Randσ problem consisted in searching for a set of 150 random patterns of length m = 6, 8, 10, 20, 30, 40, 50, 60, 70, 85, 100 in a random text sequence of length n =5, 242, 880, over a common alphabet of size σ. For each Randσ problem, the values of the approximation bound δ and of the gap bound α have been set to 1, 3, 5 and to 2, 5, 8, respectively.

EXPERIMENTAL RESULTS ON A Rand130 PROBLEM (Es2)

ALGS	(δ,α)	$m=6$	$m=8$	$m=10$	$m=12$	$m=14$	$m=16$		ALGS	(δ,α)	$m=6$	$m=8$	$m=10$
TSS-HBP	$(1,1)$	2.30	2.07	2.00	2.20	2.10	2.02		TSS-HBP	$(1,2)$	2.26	2.16	2.14
SS-BP	$(1,1)$	**1.58**	**1.72**	**1.72**	**1.62**	**1.68**	**1.62**		SS-BP	$(1,2)$	**1.52**	**1.70**	**1.66**
DA-NFA	$(1,1)$	1.82	**1.72**	1.82	1.88	1.84	1.88		DA-NFA	$(1,2)$	1.98	**1.70**	1.82
DA-CNFA	$(1,1)$	16.12	15.98	15.95	16.00	15.92	15.96		DA-CNFA	$(1,2)$	16.12	16.02	16.00
TSS-HBP	$(3,1)$	2.73	2.85	2.62	2.61	2.60	2.62		TSS-HBP	$(3,2)$	2.97	2.89	2.85
SS-BP	$(3,1)$	**1.71**	**1.42**	1.86	**1.57**	**1.76**	**1.74**		SS-BP	$(3,2)$	**1.65**	**1.73**	**1.70**
DA-NFA	$(3,1)$	1.88	1.92	**1.80**	1.98	1.80	1.94		DA-NFA	$(3,2)$	1.98	1.84	1.84
DA-CNFA	$(3,1)$	16.44	16.32	16.16	16.32	16.32	16.49		DA-CNFA	$(3,2)$	16.45	16.30	16.34
TSS-HBP	$(5,1)$	3.15	3.20	3.11	5.51	6.20	6.14		TSS-HBP	$(5,2)$	3.76	3.72	3.80
SS-BP	$(5,1)$	**1.75**	**1.75**	**1.78**	**2.97**	**3.38**	**3.30**		SS-BP	$(5,2)$	**1.79**	**1.69**	**1.49**
DA-NFA	$(5,1)$	1.86	1.82	1.86	3.09	3.48	3.62		DA-NFA	$(5,2)$	1.84	1.78	1.82
DA-CNFA	$(5,1)$	16.48	16.29	16.37	27.50	31.51	31.15		DA-CNFA	$(5,2)$	16.52	16.30	16.34

Conclusions

We have presented some efficient practical algorithms for the δ-approximate string matching problem with α-bounded gaps, which have important applications in music information retrieval. Despite their non-optimal asymptotic behavior, our algorithms perform very well in practice and, in particular, one of them wins against the fastest existing algorithms in most practical cases.