REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 10, número 1
Págs. 110- 119

Pattern-Matching with Bounded Gaps in Genomic Sequences

Yoan Pinzon†, Shu Wang*
Fecha de Recibido: 07/11/2008 Fecha de Aprobación: 20/04/2009

Abstract

Recently, some pattern matching algorithms allowing gaps were introduced in Crochemore *et al.* [Approximate string matching with gaps. Nordic Journal of Computing, 9(2002):54–65, 2002], where upper-bounded, strict-bounded and unbounded gaps were considered. In this paper we further extend these restrictions on the gaps to permit lower-bounded and (lower-upper)-bounded gaps that we simply refer to as (α,β)-bounded gaps. We give formal definitions for these problems as well as their respective algorithmic solutions.

Keywords: *string pattern matching, gaps, genomic sequences.*

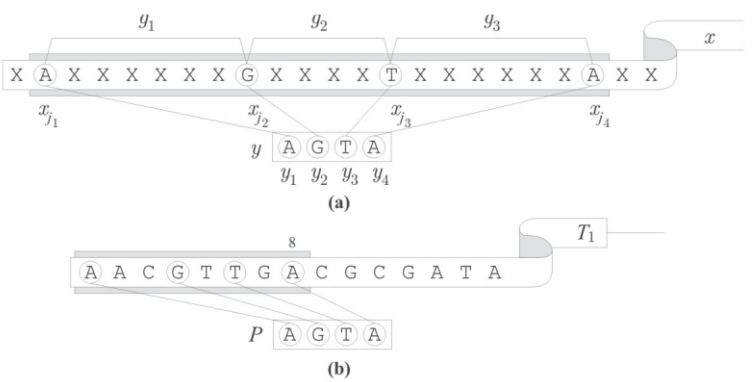
† Grupo de Investigación en Algoritmos y Combinatoria (ALGOS-UN), Universidad Nacional de Colombia, Bogotá, Colombia, ypinzon@unal.edu.co
* Department of Computing & Software, McMaster University, Canada, shuw@mcmaster.ca

1 Here we denote by α a lower bound and by β an upper bound.

1 Introduction

String matching is an important and extensively studied problem in computer science, mainly due to its direct applications to such diverse areas as text, image and signal processing, speech analysis and recognition, musical analysis, information retrieval, computational biology, etc. Formally the string matching problem consists in finding all occurrences of a given pattern in a text, over some alphabet Σ. This paper focuses in one special type of pattern matching that arise mainly in computational biology, namely, the pattern matching problem with gaps. The problem of pattern matching with gaps is defined as follows: Given a text x and a pattern y, find all occurrences of y in x such that $y_i = x_{j_i}$, $\forall$ i $\in$ 1..m where m is the length of y. Note that y occurs at position j_1 of x with a gap sequence $G = (g_1, g_2, ..., g_{m-1})$ where $g_i = j_{i+1} - j_i - 1$ and $j_1 < j_2 < ... < j_m$ (see Fig. 1a for a pictorial illustration).

In Crochemore *et al*. [1], algorithms for several versions of pattern matching with gaps were presented. They confined the gaps by certain criterion such as, β-bounded[1] (upper bounded) gaps (see Fig 1b), strict bounded (rigid length) gaps, and unbounded gaps. In this paper we introduce further restrictions to the gaps. First by bringing in a lower bound restriction on the gaps, i.e., $g_i \geq \alpha$,$\forall$ i (see Fig 1c). Then by combining the lower and upper bound, i.e.$\alpha \leq g_i \leq \beta$, $\forall$ i (see Fig 1d). And finally, by allowing different range restrictions for each individual gap, i.e., $\alpha_i \leq g_i \leq \beta_i$, $\forall$ i (see Fig 1e). In this way, the flexibility of the input and the precision of the output are both greatly increased. In the following sections we will look at each of these new versions.



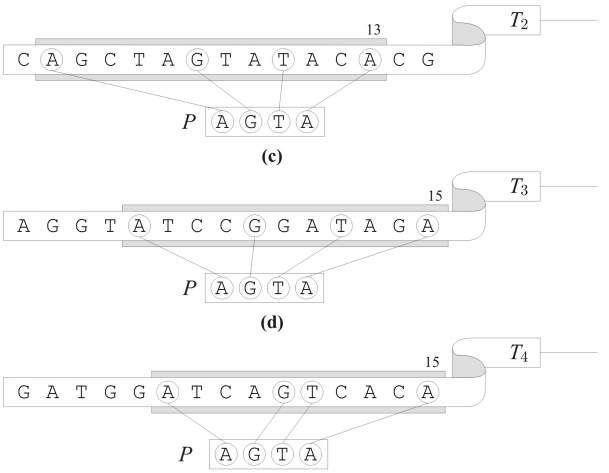


Fig.1 . Different types of gaps: (a) an occurrence with gaps in general, (b) a β-bounded gap for β = 2, (c) an α-bounded gap for α = 2, (d) an (α ,β)-bounded gap for (α ,β) = (2, 3) and (e) an (α, β)-bounded gap for (α_i, β_i) = {(2, 3), (0, 0), (3, 3)}.

2 Basic Definitions

We will uniformly adopt the four letter alphabet Σ_{DNA} = {A, C, G, T}, each letter standing for the first letter of the chemical name of the nucleotide in the polymer's chain. Let X be a string drawn from Σ_{DNA}. We represent X as an array X[1..*n*] of n ≥0 symbols, where *n* = length(X) denotes the length of the string X. By X[*i*] we denote the ith symbol in X, for 1 ≤ *i* ≤ *n*. Likewise, by X[*i..j*] we denote the substring of X contained between the ith and the jth symbol of X. For any integer j ∈ 1..*n*, we call X[1..*j*] a prefix of X.

Given a text T of length n and a pattern P of length m, an occurrence with β-bounded gaps of P in T is an increasing sequence of indices (i_1, i_2, . . . , i_m) such that (*i*) 1 ≤i_1 and i_m = *i* ≤*n* and (*ii*) i_h+1 - i_h ≤β+1, for h = 1, 2, . . . , m- 1. We write P $\ll^{i}_{\beta}$ T to mean that P has an occurrence with β-bounded gaps that terminates at position *i* in T. In the same way, an occurrence with α-bounded gaps of P in T is an increasing sequence of indices (i_1, i_2, . . . , i_m) such that (*i*) 1 ≤i_1 and i_m = *i* ≤*n* and (*ii*) i_{h+1} - i_h ≥α + 1, for h = 1, 2, . . . , *m* - 1. We write P $\ll^{i}_{\alpha}$ T to mean that P has an occurrence with α-bounded gaps that terminates at position i in T. Also, an occurrence with (α ,β)-bounded gaps of P in T is an increasing sequence of indices (i_1, i_2, . . . , i_m) such that (*i*) 1 ≤i_1 and i_m = *i*

$\leq n$ and (ii) $\alpha+1 \leq i_{h+1} - i_h \leq \beta+1$, for $h = 1,2,\ldots m-1$. We write $P \ll^{i}_{\alpha,\beta} T$ to mean that P has an occurrence with (α, β)-bounded gaps that terminates at position i in T. Finally, an occurrence with (α, β)-bounded gaps of P in T is an increasing sequence of indices $(i_1, i_2, \ldots, i_m)$ such that (i) $1 \leq i_1$ and $i_m = i \leq n$ and (ii) $\alpha_i + 1 \leq i_{h+1} - i_h \leq \beta_i + 1$, for $h, i = 1, 2, \ldots, m-1$. We write $P \ll^{i}_{\alpha,\beta} T$ to mean that P has an occurrence with (α, β)-bounded gaps that terminates at position i in T.

			LastPos$_j(\pi_i)$														
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i	π_i	L(π_i)	A	A	C	G	T	T	G	A	C	G	C	G	A	T	A
1	A	1,2,8,13,15	1	2	2	2	0	0	0	8	8	8	0	0	13	13	15
2	AG	4,10	0	0	0	4	4	4	0	0	0	10	10	10	0	0	0
3	AGT	5,6	0	0	0	0	5	6	6	6	0	0	0	0	0	0	0
4	AGTA	8	0	0	0	0	0	0	0	8	8	8	0	0	0	0	0

Table 1. *LastPos-table for T = AACGTTGACGCGATA and P = AGTA and* $\beta = 2$.

3 Pattern Matching With β-bounded Gaps

The pattern matching problem with β-bounded gaps is formally defined as follows:

Problem 1 (upper bounded gaps). Given a text T of length n, a pattern P of length m and a positive integer β, the string pattern matching problem with β-bounded gaps is to find all positions j in T such that $P \ll^{j}_{\beta} T$, for $1 \leq j \leq n$.

Let Pref(P) be the set of nonempty prefices of P $\{\pi_1, \pi_2 \ldots \pi_m\}$ where π_i = P[1..i]. For $\pi \in$ Pref(P), denote by L(π) the set of positions k in T such that $p \ll^{k}_{\beta}$ T, i.e., there is an occurrence of π with β-bounded gaps that terminates at position k in T. We compute the following table for each $\pi \in$ Pref(P): LastPos$_j(\pi_i) = \max\{0 \leq k \leq j : (k \in L(\pi_i)$ and $j - k \leq \alpha)$ or $k = 0\}$. LastPos$_j(\pi_i) = 0$ means that $\pi_i \not\ll^{j}_{\beta} T$, thus there are not occurrences of π_i at or before position j in T. The computation of a new column LastPos$_j$ is implemented by extending each of the previously occurring prefices by a single letter, or by leaving the last position of the last match unchanged otherwise. If LastPos$_j$(P) $\neq 0$ for some j then $P \ll^{j}_{\beta} T$. As an example, Table 1 shows the LastPos table for text T = AACGTTGA CGCGATA, pattern P = AGTA and $\beta = 2$.

In Crochemore *et al*. [1], it was shown that Table 1 can be obtained by computing a matrix D[0..*m*, 0..*n*] with boundaries D[, *j*] = 0; D[*i*,] = - 1; D[0, 0] = 0 and

$$D[i,j] = \begin{cases} & \text{, if } T[j] = P[i] \text{ and } D[i-1, j-1] > -1 \\ D[i, j-1] & \text{, if } j - D[i, j-1] < \pi + 1 \text{ and } T[j] \,\pi\, P[i] \text{ and } D[i-1, j-1] \,\pi\, -1 \\ -1 & \text{, otherwise} \end{cases}$$

This recursive formula is algorithmically described in Algorithm 1. The complexity of Algorithm 1 is easily seen to be O(*nm*)-time and -space. We can use the technique introduced in [2] to reduce the space complexity to O(*n*). Table 2a gives an example of Algorithm 1 applied to text T_1 = AACGTTGACGCGATA, pattern P = AGTA and β = 2. Note that $P \ll^{8}_{\beta} T_1$ (see Fig. 1b for a better illustration of the occurrence).

Algorithm 1: π-bounded gaps (*lower bounded gaps*)

Input: T, P, β, *n* = length(T), *m* = length(P)

```
01.  for j = 1 to n do D[0, j] = 0
02.  for i = 1 to m do D[i, 0] = - 1; D[0, 0] = 0
03.  for j = 1 to n do
04.        for i = 1 to m do
05.              if T[j] = P[i] and D[i - 1, j - 1] > - 1 then D[i, j] = j
06.              elseif j - D[i, j - 1] < β + 1 then D[i, j] = D[i, j - 1]
07.              else D[i, j] = - 1
08.  for j = 1 to n do if D[m, j] = j then output j
```

Fig. 2. Algorithm 1. β-bounded gaps.

Algorithm 2: π-bounded gaps (*upper bounded gaps*)

Input: T, P, π, *n* = length(T), *m* = length(P)

```
01.  for j = 1 to n do D[0, j] = 0
02.  for i = 1 to m do D[i, 0] = - 1; D[0, 0] = 0
03.  for j = 1 to n do
04.        if j > π then j' = j - π - 1 else j' = 0
05.        for i = 1 to m do
06.              if T[j] = P[i] and D[i - 1, j'] > - 1 then D[i, j] = j
07.              else D[i, j] = D[i, j - 1]
08.  for j = 1 to n do if D[m, j] = j then output j
```

Fig. 3. Algorithm 2. α-bounded gaps.

4 Pattern Matching With α-bounded Gaps

The pattern matching problem with α-bounded gaps is formally defined as follows:

Problem 2 (lower bounded gaps). Given a text T of length n, a pattern P of length m and a positive integer α, the string pattern matching problem with α-bounded gaps is to find all positions j in T such that $P \ll^{j}_{\alpha} T$, for $1 \leq j \leq n$.

The main difference between α-bounded gaps and β-bounded gaps is that when a match is found at position (i, j), instead of checking $\text{LastPos}_{j-1}(\pi_i - 1)$ we check $\text{LastPos}_{j-\alpha-1}(\pi_i - 1)$. If $\text{LastPos}_{j-\alpha-1}(\pi_i - 1) > 0$ and $j > \alpha$ then an occurrence is found at position (i, j). So, Problem 2 can be solved using Algorithm 1 with the following variations:

$$D[i, j] = \begin{cases} j & \text{, if } T[j] = P[i] \text{ and } D[i, j'_i] > -1 \\ D[i, j-1] & \text{, otherwise} \end{cases}$$

where $j' = j - \alpha - 1$ if $j > \alpha$, 0 otherwise. A more detailed description of these modifications is shown in Algorithm 2. The complexity of Algorithm 2 is the same as that of Algorithm 1. i.e., $O(nm)$. If text T_2 = AGGTATCCGGATAGA, pattern P = AGTA and $\alpha = 2$, for example, then $P \ll^{13}_{\alpha} T_2$ (cf. Table 2b and Fig. 1c).

5 Pattern Matching With (α, β)-bounded Gaps

The pattern matching problem with (α, β)-bounded gaps is formally defined as follows:

Problem 3 (lower&upper bounded gaps). Given a text T of length n and a pattern P of length m and positive integers (α ,β), the string pattern matching problem with (α ,β)-bounded gaps is to find all positions j in T such that $P \ll^{j}_{\alpha,\beta} T$, for $1 \leq j \leq n$.

We see a solution to this problem in a combination of both Algorithm 1 and Algorithm 2 as follows: Let D[,j] = j; D[i,] = - 1; D[0, 0] = 0 and

$$D[i,j] = \begin{cases} & \text{, if } T[j] = P[i] \text{ and } D[i-1, j'_i] > -1 \text{ and } j - D[i-1, j'] \leq \pi + 1 \\ D[i, j-1] & \text{, if } j - D[i, j-1] < \pi + 1 \\ -1 & \text{, otherwise} \end{cases}$$

where $j' = j - \alpha - 1$ if $j > \alpha$, 0 otherwise. Algorithm 3 describes these modifications in more detail. Results from applying Algorithm 3 to text T_3 = GATGGATCAGTCACA, pattern P = AGTA and $(\alpha, \beta) = (2, 3)$ are shown in Table 3a. Refer back to Fig. 1d to see an illustration of an occurrence ending at position 15 in T_3.

Algorithm 3: ($\pi\pi\pi\pi$-bounded gaps (*lower&upper bounded gaps*)

Input: T, P, π, $\pi\pi n$ = length(T), m = length(P)

```
01.  for j = 1 to n do D[0, j] = 0
02.  for i = 1 to m do D[i, 0] = - 1; D[0, 0] = 0
03.  for j = 1 to n do
04.          if j > π then j' = j - π - 1 else j' = 0
05.          for i = 1 to m do
06.                  T[j] = P[i] and D[i - 1, j'] > - 1 and j - D[i - 1, j'] π π + 1 then
07.                          D[i, j] = j
08.                  else D[i, j] = - 1
09.  for j = 1 to n do if D[m, j] = j then output j
```

Fig. 4. Algorithm 3. (α, β) -bounded gaps.

6 Pattern matching With (α, β) bounded Gaps

The pattern matching problem with (α, β) bounded gaps is formally defined as follows:

Problem 4 (flexible lower&upper bounded gaps). Given a text T of length n and a pattern P of length m and positive integers $\{(\alpha_1, \beta_1)..(\alpha_m, \beta_m)\}$, the string pattern matching problem with (α, β) bounded gaps is to find all positions j in T such that $P \lll^{j}_{\alpha,\beta} T$, for $1 \leq j \leq n$.

This problem differs from Problem 3 only in that individual symbols in the the pattern p_i has different gap bounds (α_i, β_i). We propose to solve this problem by the following simple recursive formula: Let D[, j] = j; D[i,] = - 1; D[0, 0] = 0 and

$$D[i,j] = \begin{cases} & \text{, if } T[j] = P[i] \text{ and } D[i-1, j'_i] > -1 \text{ and } j - D[i-1, j'_i] \leq \pi_i + 1 \\ D[i, j-1] & \text{, if } j - D[i, j-1] < \pi_i + 1 \\ -1 & \text{, otherwise} \end{cases}$$

where $j' = j - \alpha_i - 1$ if $j > \alpha_i$, 0 otherwise. The time complexity of this algorithm is the same as the previous algorithms. Table 3b gives an example for text T_4 = GATGGATCAGTCACA, pattern P = AGTA and $(\alpha_i, \beta_i) = \{(2,3), (0,0), (3,3)\}$. Notice that there is an occurrence ending at position 15 in T_4. This occurrence is also graphically illustrated in Fig. 1e.

Algorithm 4: (π_1, π_2)-bounded gaps (*flexible lower&upper bounded gaps*)

Input: T, P, $\pi_1 \pi_2 \ldots \pi_m$, $\pi_1 \pi_2 \ldots \pi_m \pi n$ = length(T), m = length(P)

```
01.  for j = 1 to n do D[0, j] = j
02.  for i = 1 to m do D[i, 0] = -1; D[0, 0] = 0
03.  for j = 1 to n do
04.        if j > π_i then j' = j - π_i - 1 else j' = 0
05.        for i = 1 to m do
06.              T[j] = P[i] and D[i - 1, j'] > -1 and j - D[i - 1, j'] π π_i + 1 then
07.                    D[i, j] = j
08.              else D[i, j] = -1
09.  for j = 1 to n do if D[m, j] = j then output j
```

Fig. 5. Algorithm 4. (α,β) bounded gaps.

7 Conclusions

New versions of pattern matching with gaps were proposed and their efficient algorithmic solutions were presented. The solutions were based on existing algorithms described in [1] but many necessary alterations were made. It was shown that all these algorithms have an O(*nm*)-time and -space complexity. After introducing these new versions more flexible and precise string matching algorithms with gaps can be achieved.

References

[1] M. Crochemore, C.S. Iliopoulos, C. Makris,W. Rytter, A. Tsakalidis, and K. Tsichlas. Approximate string matching with gaps. Nordic Journal of Computing, 9(2002):54–65, 2002.

[2] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. Communication of ACM, 18(6):341–343, 1975.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		T_1	A	A	C	G	T	T	G	A	C	G	C	G	A	T	A
0	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	A	-1	**1**	**2**	2	2	-1	-1	|-1	**8**	8	8	-1	-1	**13**	13	**15**
2	G	-1	-1	-1	-1	**4**	4	4	-1	-1	-1	**10**	10	10	-1	-1	-1
3	T	-1	-1	-1	-1	-1	**5**	**6**	6	6	-1	-1	-1	-1	-1	-1	-1
4	A	-1	-1	-1	-1	-1	-1	-1	-1	**8**	8	8	-1	-1	-1	-1	-1

(a)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		T_2	C	A	G	C	T	A	G	T	A	T	A	C	A	C	G
0	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	A	-1	-1	**2**	2	2	2	**6**	6	6	**9**	9	**11**	11	**13**	13	13
2	G	-1	-1	-1	-1	-1	-1	-1	**7**	7	7	7	7	7	7	7	**15**
3	T	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**10**	10	10	10	10	10
4	A	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**13**	13	13

(b)

Table 2. *Example of the computation of (a) a β-bounded table for pattern P = AGTAC, text T_1 = AACGTTGACGCGATA and β = 2 using Algorithm 1, (b) an α-bounded table for pattern P = AGTAC, text T_2 = CAGCTAGTATACACG and α = 2 using Algorithm 2.*

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		T_3	A	G	G	T	A	T	C	C	G	G	A	T	A	G	A
0	P	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	A	-1	**1**	1	1	1	**5**	5	5	5	-1	-1	**11**	11	**13**	13	**15**
2	G	-1	-1	-1	-1	-1	-1	-1	-1	-1	**9**	9	9	9	-1	**14**	14
3	T	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**12**	12	12	12
4	A	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**15**

(a)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		T_4	G	A	T	G	G	A	T	C	A	G	T	C	A	C	A
0	P	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	A	-1	-1	**2**	2	2	2	**6**	6	6	**9**	9	9	9	**13**	13	**15**
2	G	-1	-1	-1	-1	-1	**5**	5	5	5	5	**10**	10	10	10	10	10
3	T	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**11**	11	11	11	11
4	A	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	**15**

(b)

Table 3. *Example of the computation of (a) an (α, β)-bounded table for pattern P = AGTAC, text T_3 = AGGTATCCGGATAGA and (α, β) = (2, 3) using*

Algorithm 3, and (b) an (α, β) bounded table for pattern P = AGTAC, text T_4 = GATGGATCAGTCACA and $(\alpha_i, \beta_i) = \{(2, 3), (0, 0), (3, 3)\}$ *using Algorithm 4. We conclude that* $P \ll^{8}_{\beta} T_1$, $P \ll^{13}_{\alpha} T_2$, $P \ll^{15}_{\alpha,\beta} T_3$ *and* $P \ll^{15}_{\alpha,\beta} T_4$.