An Optimal Algorithm for the Maximum-Density Segment Problem*

Kai-min Chung† Hsueh-I Lu‡

November 29, 2003

Abstract

We address a fundamental problem arising from analysis of biomolecular sequences. The input consists of two numbers $w_{\min}$ and $w_{\max}$ and a sequence S of n number pairs (a_i, w_i) with $w_i > 0$. Let *segment* $S(i, j)$ of S be the consecutive subsequence of S between indices i and j. The *density* of $S(i, j)$ is $d(i, j) = (a_i + a_{i+1} + \cdots + a_j)/(w_i + w_{i+1} + \cdots + w_j)$. The *maximum-density segment problem* is to find a maximum-density segment over all segments $S(i, j)$ with $w_{\min} \leq w_i + w_{i+1} + \cdots + w_j \leq w_{\max}$. The best previously known algorithm for the problem, due to Goldwasser, Kao, and Lu, runs in $O(n \log(w_{\max} - w_{\min} + 1))$ time. In the present paper, we solve the problem in $O(n)$ time. Our approach bypasses the complicated *right-skew decomposition*, introduced by Lin, Jiang, and Chao. As a result, our algorithm has the capability to process the input sequence in an online manner, which is an important feature for dealing with genome-scale sequences. Moreover, for a type of input sequences S representable in $O(m)$ space, we show how to exploit the sparsity of S and solve the maximum-density segment problem for S in $O(m)$ time.

1 Introduction

We address the following fundamental problem: The input consists of two numbers $w_{\min}$ and $w_{\max}$ and a sequence S of number pairs (a_i, w_i) with $w_i > 0$ for $i = 1, \ldots, n$. A *segment* $S(i, j)$ is a consecutive subsequence of S starting with index i and ending with index j. For a segment $S(i, j)$, the *width* is $w(i, j) = w_i + w_{i+1} + \cdots + w_j$, and the *density* is $d(i, j) = (a_i + a_{i+1} + \cdots + a_j)/w(i, j)$. It is not difficult to see that with an $O(n)$-time preprocessing to compute all $O(n)$ prefix sums $a_1 + a_2 + \cdots + a_j$ and $w_1 + w_2 + \cdots + w_j$, the density of any segment can be computed in $O(1)$ time. $S(i, j)$ is *feasible* if $w_{\min} \leq w(i, j) \leq w_{\max}$. The *maximum-density segment* problem is to find a maximum-density segment over all $O(n^2)$ feasible segments.

This problem arises from the investigation of non-uniformity of nucleotide composition within genomic sequences, which was first revealed through thermal melting and gradient centrifugation

*An early version of this paper was presented at 11th Annual European Symposium on Algorithms, Budapest, Hungary, September 15-20, 2003.

†Institute of Information Science, Academia Sinica Taipei 115, Taiwan, Republic of China. Part of this work was done while this author was an undergraduate student at Department of Computer Science and Information Engineering, National Taiwan University.

‡Corresponding author. Institute of Information Science, Academia Sinica, Taipei 115, Taiwan, Republic of China. Email: hil@iis.sinica.edu.tw. URL: www.iis.sinica.edu.tw/~hil/. Research supported in part by NSC grant 91-2215-E-001-001.

experiments [19, 26]. The GC content of the DNA sequences in all organisms varies from 25% to 75%. GC-ratios have the greatest variations among bacteria's DNA sequences, while the typical GC-ratios of mammalian genomes stay in 45-50%. Despite intensive research effort in the past two decades, the underlying causes of the observed heterogeneity remain debatable [2, 3, 5, 8–11, 16, 38, 40]. Researchers [30, 37] observed that the compositional heterogeneity is highly correlated to the GC content of the genomic sequences. Other investigations showed that gene length [7], gene density [42], patterns of codon usage [35], distribution of different classes of repetitive elements [7, 36], number of isochores [2], lengths of isochores [30], and recombination rate within chromosomes [12] are all correlated with GC content. More research related to GC-rich segments can be found in [14, 15, 18, 21, 27, 29, 33, 39, 41] and the references therein.

In the most basic form of the maximum-density segment problem, the sequence S corresponds to the given DNA sequence, where $a_i = 1$ if the corresponding nucleotide in the DNA sequence is G or C; and $a_i = 0$ otherwise. In the work of Huang [17], sequence entries took on values of p and $1-p$ for some real number $0 \leq p \leq 1$. More generally, we can look for regions where a given set of patterns occur very often. In such applications, a_i could be the relative frequency that the corresponding DNA character appears in the given patterns. Further natural applications of this problem can be designed for sophisticated sequence analysis such as mismatch density [34], ungapped local alignments [1], annotated multiple sequence alignments [37], promoter mapping [20], and promoter recognition [31].

For the *uniform* case, i.e., $w_i = 1$ for all indices i, Nekrutendo and Li [30], and Rice, Longden and Bleasby [32] employed algorithms for the case $w_{\min} = w_{\max}$, which is trivially solvable in $O(n)$ time. More generally, when $w_{\min} \neq w_{\max}$, the problem is also easily solvable in $O(n(w_{\max} - w_{\min} + 1))$ time, linear in the number of feasible segments. Huang [17] studied the case where $w_{\max} = n$, i.e., there is effectively no upper bound on the width of the desired maximum-density segments. He observed that an optimal segment exists with width at most $2w_{\min} - 1$. Therefore, this case is equivalent to the case with $w_{\max} = 2w_{\min} - 1$ and can be solved in $O(nw_{\min})$ time in a straightforward manner. Lin, Jiang, and Chao [25] gave an $O(n \log w_{\min})$-time algorithm for this case based on right-skew decompositions of a sequence. (See [24] for a related software.) The case with general $w_{\max}$ was first investigated by Goldwasser, Kao, and Lu [13], who gave an $O(n)$-time algorithm for the uniform case. Recently, Kim [23] showed an alternative algorithm based upon a geometric interpretation of the problem, which basically relates the maximum-density segment problem to the fundamental *slope selection problem* in computational geometry [4, 6, 22, 28]. Unfortunately, Kim's analysis of time complexity has some flaw which seems difficult to fix.[1] For the general (i.e., *non-uniform*) case, Goldwasser, Kao, and Lu [13] also gave an $O(n \log(w_{\max} - w_{\min} + 1))$-time algorithm. By bypassing the complicated preprocessing step required in [13], we successfully reduce the required time for the general case down to $O(n)$. Our result is based upon the following set of equations, stating that the order of $d(x, y)$, $d(y + 1, z)$, and $d(x, z)$ with $x \leq y < z$ can be determined by that of any two of them:

$$\begin{aligned} &d(x, y) \leq d(y + 1, z) \Leftrightarrow d(x, y) \leq d(x, z) \Leftrightarrow d(x, z) \leq d(y + 1, z); \\ &d(x, y) \geq d(y + 1, z) \Leftrightarrow d(x, y) \geq d(x, z) \Leftrightarrow d(x, z) \geq d(y + 1, z). \end{aligned} \tag{1}$$

[1] Kim claims that all the progressive updates of the lower convex hulls $L_j \cup R_j$ can be done in overall linear time. The paper only sketches how to obtain $L_{j+1} \cup R_{j+1}$ from $L_j \cup R_j$. (See the fourth-to-last paragraph of page 340 in [23].) Unfortunately, Kim seems to overlook the marginal cases when the upper bound $w_{\max}$ forces the p_z of $L_j \cup R_j$ to be deleted from $L_{j+1} \cup R_{j+1}$. As a result, obtaining $L_{j+1} \cup R_{j+1}$ from $L_j \cup R_j$ could be much more complicated than Kim's sketch. A naive implementation of Kim's algorithm still takes $\Omega(n(w_{\max} - w_{\min} + 1))$ time in the worst case. We believe that any correct implementation of Kim's algorithm requires $\Omega(n \log(w_{\max} - w_{\min} + 1))$ time in the worse case.

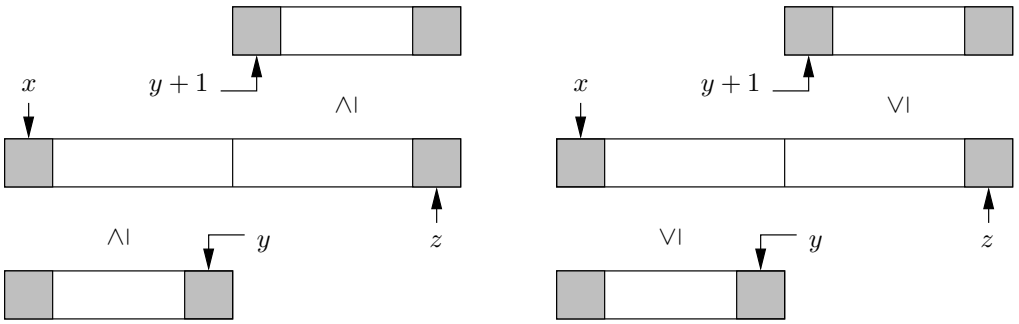

Figure 1: An illustration for Equation (1): There are only two possibilities for the order among $d(x, y), d(x, z), d(y + 1, z)$.

(Both equations can be easily verified by observing the existence of some number ρ with $0 < \rho < 1$ and $d(x, z) = \rho \cdot d(x, y) + (1-\rho) \cdot d(y+1, z)$. See Figure 1.) Our algorithm is capable of processing the input sequence in an online manner, which is an important feature for dealing with genome-scale sequences.

For bioinformatics applications, e.g., in [1, 20, 31, 34, 37], the input sequence S is usually very *sparse*. That is, S can be represented by $m = o(n)$ triples $(a'_1, w'_1, n_1), (a'_2, w'_2, n_2), \ldots, (a'_m, w'_m, n_m)$ with $0 = n_0 < n_1 < n_2 < \cdots < n_m = n$ to signify that $(a_i, w_i) = (a'_j, w'_j)$ holds for all indices i and j with $n_{j-1} < i \leq n_j$ and $1 \leq j \leq m$. If $w'_j = 1$ holds for all $1 \leq j \leq m$, we show how to exploit the sparsity of S and solve the maximum-density problem for S given in the above compact representation in $O(m)$ time.

The remainder of the paper is organized as follows. Section 2 shows the main algorithm. Section 3 explains how to cope with the simple case that the width upper bound $w_{\max}$ is ineffective. Section 4 takes care of the more complicated case that $w_{\max}$ is effective. Section 5 explains how to exploit the sparsity of the input sequence for the uniform case.

2 The main algorithm

For any integers x and y, let $[x, y]$ denote the set $\{x, x + 1, \ldots, y\}$. Throughout the paper, we need the following definitions and notation with respect to the input length-n sequence S and width bounds $w_{\min}$ and $w_{\max}$. Let j_0 be the smallest index with $w(1, j_0) \geq w_{\min}$. Let $J = [j_0, n]$. For each $j \in J$, let ℓ_j (respectively, r_j) be the smallest (respectively, largest) index i with $w_{\min} \leq w(i, j) \leq w_{\max}$. That is, $S(i, j)$ is feasible if and only if $i \in [\ell_j, r_j]$. (Figure 3 is an illustration for the definitions of ℓ_j and r_j.) Clearly, for the uniform case, we have $\ell_{i+1} = \ell_i + 1$ and $r_{i+1} = r_i + 1$. As for the general case, we only know that ℓ_j and r_j are both (not necessarily monotonically) increasing. One can easily compute all ℓ_j and r_j in $O(n)$ time. Let i^*_j be the largest index $k \in [\ell_j, r_j]$ with $d(k, j) = \max\{d(i, j) \mid i \in [\ell_j, r_j]\}$. Clearly, there must be an index j^* such that $S(i^*_{j^*}, j^*)$ is a maximum-density segment of S. Therefore, a natural but seemingly difficult possibility to solve the maximum-density segment problem would be to compute i^*_j for all indices $j \in J$ in $O(n)$ time. Instead, our strategy is to compute an index $i_j \in [\ell_j, r_j]$ for each index $j \in J$ by the algorithm shown in Figure 2, where $\phi(x, y)$ is defined to be the largest index $z \in [x, y]$ that minimizes $d(x, z)$. That is, $S(x, \phi(x, y))$ is the longest minimum-density prefix of $S(x, y)$. The rest of the section ensures the correctness of our algorithm by showing $i_{j^*} = i^*_{j^*}$, and thus reduces the

```
    algorithm MAIN
1       let i_{j_0-1} = 1;
2       for j = j_0 to n do {
3           let i_j = BEST(max(i_{j-1}, ℓ_j), r_j, j);
4           output (i_j, j);
5       }

    function BEST(ℓ, r, j)
1       let i = ℓ;
2       while i < r and d(i, φ(i, r − 1)) ≤ d(i, j) do
3           let i = φ(i, r − 1) + 1;
4       return i;
```

Figure 2: Our main algorithm.

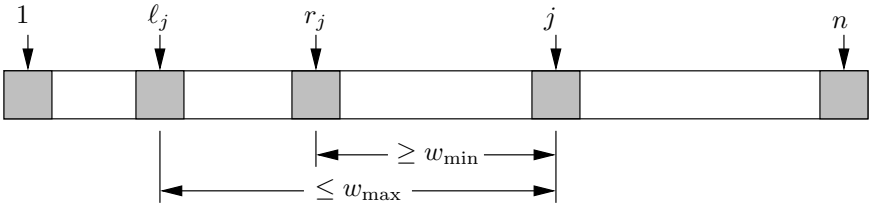

Figure 3: An illustration for the definitions of ℓ_j and r_j.

maximum-density segment problem to implementing our algorithm to run in $O(n)$ time.

Lemma 1 *The index returned by function call* BEST(ℓ, r, j) *is the largest index* $i \in [\ell, r]$ *that maximizes* $d(i, j)$.

Proof. Let i^* be the largest index in $[\ell, r]$ that maximizes $d(i, j)$, i.e., $d(i^*, j) = \max_{i \in [\ell, r]} d(i, j)$. Let i_j be the index returned by function call BEST(ℓ, r, j). We show $i_j = i^*$ as follows. If $i_j < i^*$, then $i_j < r$. By the condition of the while-loop at Step 2 of BEST, we know $d(i_j, \phi(i_j, r-1)) > d(i_j, j)$. By $d(i_j, j) \leq d(i^*, j)$ and Equation (1), we have $d(i_j, i^* - 1) \leq d(i_j, j)$. It follows that $d(i_j, i^* - 1) < d(i_j, \phi(i_j, r-1))$, contradicting the definition of $\phi(i_j, r-1)$.

On the other hand, suppose that $i_j > i^*$. By definition of BEST, there must be an index $i \in [\ell, r]$ with $i < r$, $d(i, \phi(i, r-1)) \leq d(i, j)$, and $i \leq i^* < \phi(i, r-1) + 1$. If $i = i^*$, by Equation (1) we have $d(i^*, \phi(i^*, r-1)) \leq d(i^*, j) \leq d(\phi(i^*, r-1) + 1, j)$, where the last inequality contradicts the definition of i^*. Now that $i < i^*$, we have $d(i^*, j) \geq d(i, j) \geq d(i, i^* - 1) \geq d(i, \phi(i, r-1)) \geq d(i^*, \phi(i, r-1))$, where (a) the first inequality is by definition of i^*, (b) the second inequality is by Equation (1) and the first inequality, (c) the third inequality is by $i^* \leq \phi(i, r-1)$ and definition of $\phi(i, r-1)$, and (d) the last inequality is by Equation (1) and the third inequality. It follows from $d(i^*, j) \geq d(i^*, \phi(i, r-1))$ and Equation (1) that $d(\phi(i, r-1) + 1, j) \geq d(i^*, j)$, contradicting the definition of i^* by $i^* < \phi(i, r-1) + 1$. □

Theorem 1 *Algorithm* MAIN *correctly solves the maximum-density problem.*

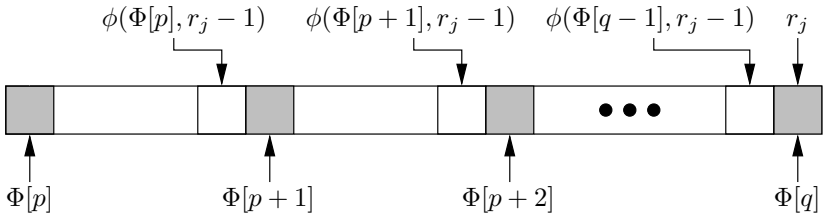

Figure 4: An illustration for Condition C_j.

Proof. We prove the theorem by showing $i_{j^*} = i^*_{j^*}$. Clearly, by $\ell_{j_0} = i_{j_0-1} = 1$ and Lemma 1, the equality holds if $j^* = j_0$. The rest of the proof assumes $j^* > j_0$. By Lemma 1 and $\ell_{j^*} \leq i^*_{j^*}$, it suffices to ensure $i_{j^*-1} \leq i^*_{j^*}$. Assume for contradiction that there is an index $j \in [j_0, j^* - 1]$ with $i_{j-1} \leq i^*_{j^*} < i_j$. By $j < j^*$, we know $\ell_j \leq i^*_{j^*}$. By Lemma 1 and $\max(\ell_j, i_{j-1}) \leq i^*_{j^*} < i_j \leq r_j$, we have $d(i_j, j) \geq d(i^*_{j^*}, j)$. It follows from Equation (1) and $i^*_{j^*} < i_j$ that $d(i^*_{j^*}, j) \geq d(i^*_{j^*}, i_j - 1)$. By $\ell_{j^*} \leq i^*_{j^*} < i_j \leq r_{j^*}$ and definition of j^*, we know $d(i^*_{j^*}, j^*) > d(i_j, j^*)$. It follows from $i^*_{j^*} < i_j$ and Equation (1) that $d(i^*_{j^*}, i_j - 1) > d(i^*_{j^*}, j^*)$. Therefore, $d(i_j, j) \geq d(i^*_{j^*}, j) \geq d(i^*_{j^*}, i_j - 1) > d(i^*_{j^*}, j^*)$, contradicting the definition of j^*. □

One can verify that the value of i increases by at least one each time Step 3 of BEST is executed. Therefore, to implement the algorithm to run in $O(n)$ time, it suffices to maintain a data structure to support $O(1)$-time query for each $\phi(i, r_j - 1)$ in Step 2 of BEST.

3 Coping with ineffective width upper bound

When $w_{\max}$ is ineffective, i.e., $w_{\max} \geq w(1, n)$, we have $\ell_j = 1$ for all $j \in F$. Therefore, the function call in Step 3 of MAIN is exactly BEST(i_{j-1}, r_j, j). Moreover, during the execution of the function call BEST(i_{j-1}, r_j, j), the value of i can only be i_{j-1}, $\phi(i_{j-1}, r_j - 1) + 1$, $\phi(\phi(i_{j-1}, r_j - 1) + 1, r_j - 1) + 1, \ldots, r_j$. Suppose that a subroutine call to UPDATE(j) yields an array Φ of indices and two indices p and q of Φ with $p \leq q$ and $\Phi[p] = i_{j-1}$ such that the following condition holds.

Condition C_j : $\Phi[q] = r_j$ and $\Phi[t] = \phi(\Phi[t-1], r_j - 1) + 1$ holds for each index $t \in [p+1, q]$.

(See Figure 4 for an illustration.) Then, the subroutine call to BEST(i_{j-1}, r_j, j) can clearly be replaced by LBEST(j), as defined in Figure 5. That is, LBEST(j) can access the value of each $\phi(i, r_j - 1)$ by looking up Φ in $O(1)$ time. It remains to show how to implement UPDATE(j) such that all $O(n)$ subroutine calls to UPDATE from Step 3 of LMAIN run in overall $O(n)$ time. The following lemma is crucial in ensuring the correctness and efficiency of our implementation shown in Figure 5, where Condition C_{j_0-1} stands for $p = 1$, $q = 0$, and $i_{j_0-1} = 1$.

Lemma 2 *For each index $j \in J$, the following statements hold.*

1. *If Condition C_{j-1} holds right before calling* UPDATE(j)*, then Condition C_j holds right after the subroutine call.*
2. *If Condition C_j holds right before calling* LBEST(j)*, then the index returned by the function call is exactly that returned by* BEST$(\Phi[p], \Phi[q], j)$.

```
    algorithm LMAIN
1       let p = 1, q = 0, and i_{j0-1} = 1;
2       for j = j0 to n do {
3           call UPDATE(j);
4           let i_j = LBEST(j);
5           output (i_j, j);
5       }

    function LBEST(j)
1       while p < q and d(Φ[p], Φ[p+1]-1) ≤ d(Φ[p], j) do
2           let p = p + 1;
3       return Φ[p];

    subroutine UPDATE(j)
1       for r = r_{j-1} + 1 to r_j do {
2           while p < q and d(Φ[q-1], Φ[q]-1) ≥ d(Φ[q-1], r-1) do
3               let q = q - 1;
4           let q = q + 1;
5           let Φ[q] = r;
6       }
```

Figure 5: An efficient implementation for the case that $w_{\max}$ is ineffective.

Proof. Statement 1. It is not difficult to verify that with the initialization $p = 1$, $q = 0$, and $i_{j_0-1} = 1$, Condition C_{j_0} holds with $p = 1$ and $q \geq 1$ after calling UPDATE(j_0). The rest of the proof assumes $j \in J - \{j_0\}$. For each $r \in [r_{j-1} + 1, r_j]$, one can see from the definition of ϕ that $\phi(\ell, r-1)$ is either $\phi(\ell, r-2)$ or $r-1$. More precisely, we have $\phi(\ell, r-1) = r-1$ if and only if $d(\ell, \phi(\ell, r-2)) \geq d(\ell, r-1)$. Furthermore, if $\phi(\ell, r-2) < r-2$, then one can prove as follows that $\phi(\phi(\ell, r-2)+1, r-1) = \phi(\phi(\ell, r-2)+1, r-2)$ implies $\phi(\ell, r-1) = \phi(\ell, r-2)$.

> Let $m = \phi(\ell, r-2)$. By $\phi(m+1, r-1) = \phi(m+1, r-2)$, we have $d(m+1, \phi(m+1, r-1)) < d(m+1, r-1)$. By definition of ϕ and Equation (1), we have $d(\ell, m) < d(m, \phi(m+1, r-1)) < d(m+1, \phi(m+1, r-1))$. As a result, we have $d(\ell, m) < d(m+1, r-1)$, which by Equation (1) implies $d(\ell, m) < d(\ell, r-1)$. Thus $\phi(\ell, r-1) = \phi(\ell, r-2)$.

Therefore, at the end of each iteration of the for-loop of UPDATE(j), we have that $\Phi[q] = r$ and $\Phi[t] = \phi(\Phi[t-1], r-1)+1$ holds for each index $t \in [p+1, q]$. (The value of q may change, though.) It follows that at the end of the for-loop, Condition C_j holds.

Statement 2. By Condition C_j, one can easily verify that LBEST(j) is a faithful implementation of best($\Phi[p], \Phi[q], j$). Therefore, the statement holds. □

Lemma 3 *The implementation* LMAIN *solves the maximum-density problem for the case with ineffective* $w_{\max}$ *in* $O(n)$ *time.*

Proof. By Lemma 2(1) and definitions of UPDATE and LBEST, both Condition C_j and $\Phi[p] = i_{j-1}$ hold right after the subroutine call to UPDATE(j). By Condition C_j and Lemma 2(2), LBEST(j) is

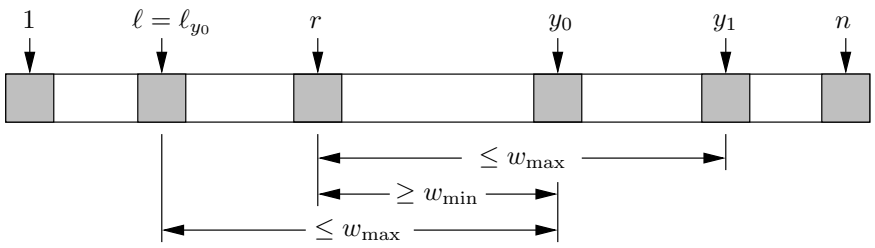

Figure 6: An illustration for the relation among ℓ, r, y_0, y_1.

a faithful implementation of BEST($\Phi[p], \Phi[q], j$). Therefore, the correctness of LMAIN follows from $\Phi[p] = i_{j-1}$, $\Phi[q] = r_j$, and Theorem 1.

As for the efficiency of LMAIN, observe that $q - p \geq -1$ holds throughout the execution of LMAIN. Note that each iteration of the while-loops of LBEST and UPDATE decreases the value of $q - p$ by one. Clearly, Step 4 of UPDATE is the only place that increases the value of $q - p$. Since it increases the value of $q - p$ by one for $O(n)$ times, the overall running time of LMAIN is $O(n)$. □

4 Coping with effective width upper bound

In contrast to the previous simple case, when $w_{\max}$ is arbitrary, ℓ_j may not always be 1. Therefore, the first argument of the function call in Step 3 of MAIN could be ℓ_j with $\ell_j > i_{j-1}$. It seems quite difficult to update the corresponding data structure Φ in overall linear time such that both $\Phi[p] = \max(i_{j-1}, \ell_j)$ and Condition C_j hold throughout the execution of our algorithm. To overcome the difficulty, our algorithm sticks with Condition C_j but allows $\Phi[p] > \max(i_{j-1}, \ell_j)$. As a result, $\max_{j \in J} d(i_j, j)$ may be less than $\max_{j \in J} d(i^*_j, j)$. Fortunately, this potential problem can be resolved if we simultaneously solve a series of variant versions of the maximum-density segment problem.

4.1 A variant version of the maximum-density segment problem

Suppose that we are given two indices r and y_0 with $w(r, y_0) \geq w_{\min}$. Let $X = [\ell, r]$ and $Y = [y_0, y_1]$ be two intervals such that $\ell = \ell_{y_0}$ and and y_1 is the largest index in J with $w(r, y_1) \leq w_{\max}$. See Figure 6 for an illustration. The *variant version* of the maximum-density segment problem is to look for a maximum-density segment over all feasible segments $S(x, y)$ with $x \in X$, $y \in Y$, and $w_{\min} \leq w(x, y) \leq w_{\max}$ such that $d(x, y)$ is maximized.

For each $y \in Y$, let x^*_y be the largest index $x \in X$ with $w_{\min} \leq w(x, y) \leq w_{\max}$ that maximizes $d(x, y)$. Let y^* be an index in Y with $d(x^*_{y^*}, y^*) = \max_{y \in Y} d(x^*_y, y)$. Although solving the variant version can naturally be reduced to computing the index x^*_y for each index $y \in Y$, the required running time is more than what we can afford. Instead, we compute an index $x_y \in X$ with $w_{\min} \leq w(x_y, y) \leq w_{\max}$ for each index $y \in Y$ such that $x_{y^*} = x^*_{y^*}$. By $w(r, y_0) \geq w_{\min}$ and $w(r, y_1) \leq w_{\max}$, one can easily see that, for each $y \in Y$, r is always the largest index $x \in X$ with $w_{\min} \leq w(x, y) \leq w_{\max}$. Our algorithm for solving the variant problem is as shown in Figure 7, presented in a way to emphasize the analogy between VMAIN and MAIN. For example, the index x_y in VMAIN is the counterpart of the index i_j in MAIN. Also, the index r in VMAIN plays the

```
    algorithm VMAIN(r, y_0)
1       let ℓ be the smallest index in [1, n] with w(ℓ, y_0) ≤ w_max;
2       let y_1 be the largest index in [1, n] with w(r, y_1) ≤ w_max;
3       let x_{y_0 − 1} = ℓ;
4       for y = y_0 to y_1 do {
5           let x_y = BEST(max(x_{y−1}, ℓ_y), r, y);
6           output (x_y, y);
7       }
```

Figure 7: Our algorithm for the variant version of the maximum-density segment problem, where function BEST is as defined in Figure 2.

role of the index r_j in MAIN. We have the following lemma whose proof is very similar to that of Theorem 1.

Lemma 4 *Algorithm* VMAIN *correctly solves the variant version of the maximum-density problem.*

Proof. We prove the theorem by showing $x_{y^*} = x^*_{y^*}$. Clearly, by $\ell_{y_0} = x_{y_0-1} = \ell$ and Lemma 1, the equality holds if $y^* = y_0$. The rest of the proof assumes $y^* > y_0$. By Lemma 1 and $\ell_{y^*} \leq x^*_{y^*}$, it suffices to ensure $x_{y^*-1} \leq x^*_{y^*}$. Assume for contradiction that there is an index $y \in [y_0, y^* - 1]$ with $x_{y-1} \leq x^*_{y^*} < x_y$. By $y < y^*$, we know $\ell_y \leq x^*_{y^*}$. By Lemma 1 and $\max(\ell_y, x_{y-1}) \leq x^*_{y^*} < x_y \leq r$, we have $d(x_y, y) \geq d(x^*_{y^*}, y)$. It follows from Equation (1) and $x^*_{y^*} < x_y$ that $d(x^*_{y^*}, y) \geq d(x^*_{y^*}, x_y - 1)$. By $\ell_{y^*} \leq x^*_{y^*} < x_y \leq r$ and definition of y^*, we know $d(x^*_{y^*}, y^*) > d(x_y, y^*)$. It follows from $x^*_{y^*} < x_y$ and Equation (1) that $d(x^*_{y^*}, x_y - 1) > d(x^*_{y^*}, y^*)$. Therefore, $d(x_y, y) \geq d(x^*_{y^*}, y) \geq d(x^*_{y^*}, x_y - 1) > d(x^*_{y^*}, y^*)$, contradicting the definition of y^*. □

Again, the challenge lies in supporting each query to $\phi(i, r-1)$ of BEST in $O(1)$ time during the execution of VMAIN. Fortunately, unlike during the execution of MAIN where both parameters of $\phi(i, r-1)$ may change, the second parameter $r-1$ is now fixed. Therefore, to support each query to $\phi(i, r-1)$ in $O(1)$ time, we can actually afford $O(r-\ell+1)$ time to compute a data structure Ψ such that $\Psi[i] = \phi(i, r-1)$ for each $i \in [\ell, r-1]$. As a result, the function BEST can be implemented as the function VBEST shown in Figure 8. The following lemma ensures the correctness and efficiency of our implementation VARIANT shown in Figure 8.

Lemma 5 *The implementation* VARIANT *correctly solves the variant version of the maximum-density segment problem in $O(r - \ell + y_1 - y_0 + 1)$ time.*

Proof. Clearly, if $\Psi[i] = \phi(i, r-1)$ holds for each index $i \in [\ell, r-1]$, then VBEST is a faithful implementation of BEST. By Lemma 4, the correctness of VARIANT can be ensured by showing that after calling INIT(ℓ, r), $\Psi[i] = \phi(i, r)$ holds for each index $i \in [\ell, r]$. By Step 1 of INIT, we have $\Psi[r] = r = \phi(r, r)$. Now suppose that $\Psi[i] = \phi(i, r)$ holds for each index $i \in [x+1, r]$ right before INIT is about to execute the iteration for index $x \in [\ell, r]$. It suffices to show $\Psi[x] = \phi(x, r)$ after the iteration. Let $Z_x = \{x, \Psi[x+1], \Psi[\Psi[x+1]+1], \Psi[\Psi[\Psi[x+1]+1]+1], \ldots, r\}$. Let $|Z_x|$ denote the cardinality of Z_x. We first show $\phi(x, r) \in Z_x$ as follows.

```
algorithm VARIANT(r, y_0)
1    let ℓ be the smallest index in [1, n] with w(ℓ, y_0) ≤ w_max;
2    let y_1 be the largest index in [1, n] with w(r, y_1) ≤ w_max;
3    call INIT(ℓ, r − 1);
4    let x_{y_0 − 1} = ℓ;
5    for y = y_0 to y_1 do {
6        let x_y = VBEST(max(x_{y−1}, ℓ_y), r, y);
7        output (x_y, y);
8    }

function VBEST(ℓ, r, y)
1    let x = ℓ;
2    while x < r and d(x, Ψ[x]) ≤ d(x, y) do
3        let x = Ψ[x] + 1;
4    return x;

subroutine INIT(ℓ, r)
1    let Ψ[r] = r;
2    for s = r − 1 downto ℓ do {
3        let t = s;
4        while t < r and d(s, t) ≥ d(s, Ψ[t + 1]) do
5            let t = Ψ[t + 1];
6        let Ψ[s] = t;
7    }
```

Figure 8: An efficient implementation for the algorithm VMAIN.

Assume for contradiction that $\phi(x,r) \notin Z_x$, i.e., there is an index $z \in Z_x$ with $z < \phi(x,r) < \Psi[z+1] = \phi(z+1,r)$. By definition of ϕ and Equation (1), we have $d(z+1,\phi(x,r)) > d(z+1,\phi(z+1,r)) > d(\phi(x,r)+1,\phi(z+1,r))$ and $d(\phi(x,r)+1,\phi(z+1,r)) \geq d(x,\phi(z+1,r)) \geq d(x,\phi(x,r))$. By $d(z+1,\phi(x,r)) > d(x,\phi(x,r))$ and Equation (1), we have $d(x,\phi(x,r)) > d(x,z)$, contradicting the definition of $\phi(x,r)$.

For any index $z \in Z_x$ with $z < \phi(x,r)$, we know $z < r$ and $\phi(z+1,r) = \Psi[z+1] \leq \phi(x,r)$. By $\phi(z+1,r) \leq \phi(x,r) \leq r$ and definition of $\phi(z+1,r)$, we have $d(z+1,\phi(x,r)) \geq d(z+1,\phi(z+1,r))$. By definition of $\phi(x,r)$ and Equation (1), we have $d(x,z) \geq d(x,\phi(x,r)) \geq d(z+1,\phi(x,r))$. By $d(x,z) \geq d(z+1,\phi(z+1,r))$ and Equation (1), we have $d(x,z) \geq d(x,\phi(z+1,r))$. Therefore, if $z < \phi(x,r)$, then Step 5 of INIT will be executed to increase the value of z. Observe that $\phi(x,r) = z < r$ and $\Psi[z+1] > z$ imply $d(x,z) < d(x,\Phi[z+1])$. It follows that as soon as $z = \phi(x,r)$ holds, whether $\phi(x,r) = r$ or not, the value of $\Psi[x]$ will immediately be set to z at Step 6 of INIT.

One can see that the running time is indeed $O(r - \ell + y_1 - y_0 + 1)$ by verifying that throughout the execution of the implementation, (a) the while-loop of VBEST runs for $O(r - \ell + y_1 - y_0 + 1)$ iterations, and (b) the while-loop of INIT runs for $O(r - \ell + 1) = O(r - \ell + 1)$ iterations. To see statement (a), just observe that the value of index x (i) never decreases, (ii) stays in $[\ell, r]$, and (iii)

```
    algorithm GENERAL
1       let p = 1, q = 0, and i_{j_0-1} = 1;
2       for j = j_0 to n do {
3           call UPDATE(j);
4           while Φ[p] < ℓ_j do
5               let p = p + 1;
6           if i_{j-1} < Φ[p] then
7               call VARIANT(Φ[p], j);
8           let i_j = LBEST(j);
9           output (i_j, j);
10      }
```

Figure 9: Our algorithm for the general case, where UPDATE and LBEST are defined in Figure 5 and VARIANT is defined in Figure 8.

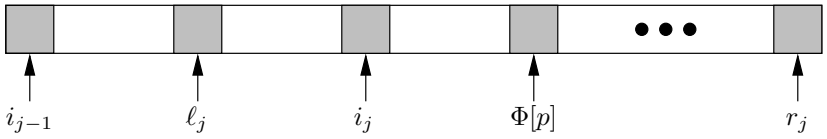

Figure 10: An illustration for the situation when Steps 6 and 7 of GENERAL are needed.

increases by at least one each time Step 3 of VBEST is executed. As for statement (b), consider the iteration with index s of the for-loop of INIT. Note that if Step 6 of INIT executes t_s times in this iteration, then $|Z_s| = |Z_{s+1}| - t_s + 1$. Since $|Z_s| \geq 1$ holds for each $s \in X$, we have $\sum_{s \in X} t_s = O(r - \ell + 1)$, and thus statement (b) holds. □

4.2 Our algorithm for the general case

With the help of VARIANT, we have a linear-time algorithm for solving the original maximum-density segment problem as shown in Figure 9. Algorithm GENERAL is obtained by inserting four lines of codes (i.e., Steps 4–7 of GENERAL) between Steps 3 and 4 of LMAIN in order to handle the case $i_{j-1} < \ell_j$. Specifically, when $i_{j-1} < \ell_j$, we cannot afford to appropriately update the data structure Φ. Therefore, instead of moving i to ℓ_j, Steps 4 and 5 move i to $\Phi[p]$, where p is the smallest index with $\ell_j \leq \Phi[p]$. Of course, these two steps may cause our algorithm to overlook the possibility of $i_j \in [i_{j-1}, \Phi[p] - 1]$, as illustrated in Figure 10. This is when the variant version comes in: As shown in the next theorem, we can remedy the problem by calling VARIANT$(\Phi[p], j)$.

Theorem 2 *The linear-time algorithm* GENERAL *solves the maximum-density segment problem in an on-line manner.*

Proof. We prove the correctness of GENERAL by showing that $i^*_{j^*} \neq i_{j^*}$ implies $i^*_{j^*} = x_{j^*}$. By Lemma 2(1), after the subroutine call UPDATE(j) at Step 3 of GENERAL, Condition C_j holds. Clearly, Steps 4 and 5 of GENERAL, which may increase the value of p, do not affect the validity of

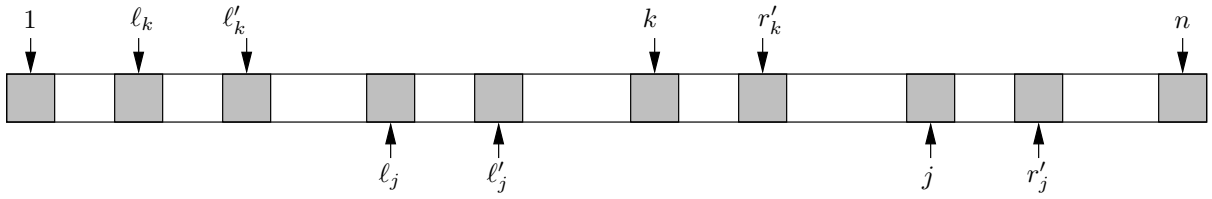

Figure 11: An illustration for showing that the overall running time of all subroutine calls to VARIANT(ℓ'_j, j) in GENERAL is $O(n)$.

Condition C_j. Clearly, Steps 6 and 7 do not modify p, q, and Φ. Let ℓ'_j be the value of $\Phi[p]$ right before executing Step 8 of GENERAL. By Lemma 2(2), the index i_j returned by LBEST(j) is the largest index in $[\ell'_j, r_j]$ that maximizes $d(i_j, j)$. Clearly, $i_{j^*} = i^*_{j^*}$ implies the correctness of GENERAL. If $i_{j^*} \neq i^*_{j^*}$, then there must be an index $j \in [j_0, j^*]$ such that $i_{j-1} \leq i^*_{j^*} < i_j$. It can be proved as follows that $i^*_{j^*} < \ell'_j$.

> Assume $\ell'_j \leq i^*_{j^*}$ for contradiction. It follows from Lemma 2(2) and Equation (1) that $d(i_j, j) \geq d(i^*_{j^*}, j) \geq d(i^*_{j^*}, i_j - 1)$. By definition of j^*, we have $d(i^*_{j^*}, j^*) > d(i_j, j)$, which by Equation (1) implies $d(i^*_{j^*}, i_j - 1) > d(i^*_{j^*}, j^*)$. Therefore, $d(i_j, j) > d(i^*_{j^*}, j^*)$, contradicting the definition of j^*.

Since $i_{j-1} \leq i^*_{j^*} < \ell'_j$, we know $w(\ell'_j - 1, j^*) \leq w(i^*_{j^*}, j^*) \leq w_{\max}$. It follows from Lemma 5 that there is an index pair (x, y) with $w_{\min} \leq w(x, y) \leq w_{\max}$ and $d(x, y) = d(i^*_{j^*}, j^*)$.

As for the running time, observe that $q - p \geq -1$ holds throughout the execution of GENERAL. Note that each iteration of the while-loops of GENERAL, LBEST and UPDATE decreases the value of $q - p$ by one. Clearly, Step 4 of UPDATE is the only place that increases the value of $q - p$. Moreover, it increases the value of $q - p$ by one for $O(n)$ times. Therefore, to show that the overall running time of GENERAL is $O(n)$, it remains to ensure that all those subroutine calls to VARIANT at Step 7 of GENERAL take overall $O(n)$ time. Suppose that j and k are two arbitrary indices with $k < j$ such that GENERAL makes subroutine calls to VARIANT(ℓ'_k, k) and VARIANT(ℓ'_j, j). Let r'_k be the largest index in $[1, n]$ with $w(\ell'_k, r'_k) \leq w_{\max}$. By Lemma 5, it suffices to show that $\ell'_k < \ell_j$ and $r'_k < j$ as follows. (See Figure 11.) By definition of GENERAL, we know that $i_{j-1} < \ell_j$, which is ensured by the situation illustrated in Figure 10. By $k < j$, we have $\ell'_k \leq i_{j-1}$, implying $\ell'_k < \ell_j$. Moreover, by definitions of ℓ_j and r'_k, one can easily verify that $\ell'_k < \ell_j$ implies $r'_k < j$.

It is clear that our algorithm shown in Figure 9 is already capable of processing the input sequence in an online manner, since the only preprocessing required is to obtain ℓ_j, r_j, and the prefix sums of $a_1, a_2, \ldots, a_j$ and $w_1, w_2, \ldots, w_j$ (for the purpose of evaluating the density of any segment in $O(1)$ time), which can easily be computed on the fly. □

5 Exploiting sparsity for the uniform case

In this section, we assume that S is represented by m pairs $(a'_1, n_1), (a'_2, n_2), \ldots, (a'_m, n_m)$ with $0 = n_0 < n_1 < n_2 < \cdots < n_m = n$ to signify that $w_1 = w_2 = \cdots = w_n = 1$ and $a_i = a'_j$ holds for all indices i and j with $n_{j-1} < i \leq n_j$ and $1 \leq j \leq m$. Our algorithm for solving the maximum-density problem for the $O(m)$-space representable sequence S is shown in Figure 12.

algorithm SPARSE
1 **for** $k = 1$ **to** m **do**
2 let $n'_k = n_k - n_{k-1}$;
3 let S' be the length-m sequence $(n'_1 a'_1, n'_1), (n'_2 a'_2, n'_2), \ldots, (n'_m a'_m, n'_m)$;
4 **output** $(n_{i'-1} + 1, n_{j'})$, where (i', j') is an optimal output of GENERAL$(w_{\min}, w_{\max}, S')$;
5 **for** $k = 1$ **to** m **do** {
6 **if** $n_k \geq w_{\min}$ **then**
7 **output** (ℓ_{n_k}, n_k) and (r_{n_k}, n_k);
8 **if** $n_{k-1} + w_{\min} \leq n$ **then**
9 **output** $(n_{k-1} + 1, n_{k-1} + w_{\min})$ and $(n_{k-1} + 1, \min(n, n_{k-1} + w_{\max}))$;
11 }

Figure 12: Our algorithm that handles sparse input sequence for the uniform case, where GENERAL is defined in Figure 9.

Theorem 3 *Algorithm* SPARSE *solves the maximum-density problem for the above $O(m)$-space representable sequence in $O(m)$ time.*

Proof. By Theorem 2, SPARSE runs in $O(m)$ time. Let $S(i^*, j^*)$ be a feasible segment with maximum density. We first show that without loss of generality $i^* - 1 \in \{n_0, n_1, \ldots, n_{m-1}\}$ or $j^* \in \{n_1, n_2, \ldots, n_m\}$ holds. More specifically, we show that if $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \notin \{n_1, n_2, \ldots, n_m\}$, then $S(i^* + 1, j^* + 1)$ is also a feasible segment with maximum density: By $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$, we know $a_{i^*-1} = a_{i^*}$. By $j^* \notin \{n_1, n_2, \ldots, n_m\}$, we know $a_{j^*} = a_{j^*+1}$. By the optimality of $S(i^*, j^*)$, we have $a_{i^*} \geq a_{j^*+1}$ and $a_{i^*-1} \leq a_{j^*}$, implying $a_{i^*-1} = a_{i^*} = a_{j^*} = a_{j^*+1}$. Therefore, $S(i^* + 1, j^* + 1)$ is also a maximum-density segment.

- Case 1: $i^* - 1 \in \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \in \{n_1, n_2, \ldots, n_m\}$. Clearly, Steps 1–4 of SPARSE take care of this case.
- Case 2: $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \in \{n_1, n_2, \ldots, n_m\}$. By Equation (1), we know that $a_{i^*-1} = a_{i^*} \neq d(i^*, j^*)$ implies $d(i^*-1, j^*) > d(i^*, j^*)$ or $d(i^*+1, j^*) > d(i^*, j^*)$. If (i^*, j^*) is not discovered by Steps 6 and 7 of SPARSE, then $\ell_{j^*} < i^* < r_{j^*}$. Since $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$ implies $a_{i^*-1} = a_{i^*}$, we know that $\ell_{j^*} < i^* < r_{j^*}$ implies $d(i^*-1, j^*) = d(i^*, j^*) = d(i^*+1, j^*)$. Thus, $S(i^*-1, j^*)$ is also a feasible segment with maximum density. Clearly, we can continue the same argument until having a maximum-density segment $S(i, j^*)$ such that either $i-1 \in \{n_0, n_1, \ldots, n_{m-1}\}$, which is handled in Case 1, or $i = \ell_{j^*}$, which is handled by Steps 6 and 7 of SPARSE.
- Case 3: $i^* - 1 \in \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \notin \{n_1, n_2, \ldots, n_m\}$. By Equation (1), we know that $a_{j^*} = a_{j^*+1} \neq d(i^*, j^*)$ implies $d(i^*, j^* - 1) > d(i^*, j^*)$ or $d(i^*, j^* + 1) > d(i^*, j^*)$. If (i^*, j^*) is not discovered by Steps 8 and 9 of SPARSE, then $\ell_{j^*} < i^* < r_{j^*}$. Since $j^* \notin \{n_1, n_2, \ldots, n_m\}$ implies $a_{j^*} = a_{j^*+1}$, we know that $\ell_{j^*} < i^* < r_{j^*}$ implies $d(i^*, j^*-1) = d(i^*, j^*) = d(i^*, j^*+1)$. Thus, $S(i^*, j^*+1)$ is also a feasible segment with maximum density. Clearly, we can continue the same argument until having a maximum-density segment $S(i^*, j)$ such that either $j \in$

$\{n_1, n_2, \ldots, n_m\}$, which is handled in Case 1, or $i^* = \ell_j$, which is handled by Steps 8 and 9 of SPARSE.

The theorem is proved. □

Acknowledgments

We thank Yi-Hsuan Hsin and Hsu-Cheng Tsai for discussions in the preliminary stage of this research. We thank Tien-Ching Lin for useful comments.

References

[1] N. N. Alexandrov and V. V. Solovyev. Statistical significance of ungapped sequence alignments. In *Proceedings of Pacific Symposium on Biocomputing*, volume 3, pages 461–470, 1998.

[2] G. Barhardi. Isochores and the evolutionary genomics of vertebrates. *Gene*, 241:3–17, 2000.

[3] G. Bernardi and G. Bernardi. Compositional constraints and genome evolution. *Journal of Molecular Evolution*, 24:1–11, 1986.

[4] H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Computational Geometry — Theory and Applications*, 10(1):23–29, 1998.

[5] B. Charlesworth. Genetic recombination: patterns in the genome. *Current Biology*, 4:182–184, 1994.

[6] R. Cole, J. S. Salowe, W. L. Steiger, and E. Szemeredi. An optimal-time algorithm for slope selection. *SIAM Journal on Computing*, 18(4):792–810, 1989.

[7] L. Duret, D. Mouchiroud, and C. Gautier. Statistical analysis of vertebrate sequences reveals that long genes are scarce in GC-rich isochores. *Journal of Molecular Evolution*, 40:308–371, 1995.

[8] A. Eyre-Walker. Evidence that both G+C rich and G+C poor isochores are replicated early and late in the cell cycle. *Nucleic Acids Research*, 20:1497–1501, 1992.

[9] A. Eyre-Walker. Recombination and mammalian genome evolution. *Proceedings of the Royal Society of London Series B, Biological Science*, 252:237–243, 1993.

[10] J. Filipski. Correlation between molecular clock ticking, codon usage fidelity of DNA repair, chromosome banding and chromatin compactness in germline cells. *FEBS Letters*, 217:184–186, 1987.

[11] M. P. Francino and H. Ochman. Isochores result from mutation not selection. *Nature*, 400:30–31, 1999.

[12] S. M. Fullerton, A. B. Carvalho, and A. G. Clark. Local rates of recombination are positively correlated with GC content in the human genome. *Molecular Biology and Evolution*, 18(6):1139–1142, 2001.

[13] M. H. Goldwasser, M.-Y. Kao, and H.-I. Lu. Fast algorithms for finding maximum-density segments of a sequence with applications to bioinformatics. In R. Guigó and D. Gusfield, editors, *Proceedings of the Second International Workshop of Algorithms in Bioinformatics*, Lecture Notes in Computer Science 2452, pages 157–171, Rome, Italy, 2002. Springer-Verlag.

[14] P. Guldberg, K. Gronbak, A. Aggerholm, A. Platz, P. thor Straten, V. Ahrenkiel, P. Hokland, and J. Zeuthen. Detection of mutations in GC-rich DNA by bisulphite denaturing gradient gel electrophoresis. *Nucleic Acids Research*, 26(6):1548–1549, 1998.

[15] W. Henke, K. Herdel, K. Jung, D. Schnorr, and S. A. Loening. Betaine improves the PCR amplification of GC-rich DNA sequences. *Nucleic Acids Research*, 25(19):3957–3958, 1997.

[16] G. P. Holmquist. Chromosome bands, their chromatin flavors, and their functional features. *American Journal of Human Genetics*, 51:17–37, 1992.

[17] X. Huang. An algorithm for identifying regions of a DNA sequence that satisfy a content requirement. *Computer Applications in the Biosciences*, 10(3):219–225, 1994.

[18] K. Ikehara, F. Amada, S. Yoshida, Y. Mikata, and A. Tanaka. A possible origin of newly-born bacterial genes: significance of GC-rich nonstop frame on antisense strand. *Nucleic Acids Research*, 24(21):4249–4255, 1996.

[19] R. B. Inman. A denaturation map of the 1 phage DNA molecule determined by electron microscopy. *Journal of Molecular Biology*, 18:464–476, 1966.

[20] I. P. Ioshikhes and M. Q. Zhang. Large-scale human promoter mapping using CpG islands. *Nature Genetics*, 26:61–63, 2000.

[21] R. Jin, M.-E. Fernandez-Beros, and R. P. Novick. Why is the initiation nick site of an AT-rich rolling circle plasmid at the tip of a GC-rich cruciform? *The EMBO Journal*, 16(14):4456–4466, 1997.

[22] M. J. Katz and M. Sharir. Optimal slope selection via expanders. *Information Processing Letters*, 47(3):115–122, 1993.

[23] S. K. Kim. Linear-time algorithm for finding a maximum-density segment of a sequence. *Information Processing Letters*, 86(6):339–342, 2003.

[24] Y.-L. Lin, X. Huang, T. Jiang, and K.-M. Chao. MAVG: locating non-overlapping maximum average segments in a given sequence. *Bioinformatics*, 19(1):151–152, 2003.

[25] Y.-L. Lin, T. Jiang, and K.-M. Chao. Algorithms for locating the length-constrained heaviest segments, with applications to biomolecular sequence analysis. *Journal of Computer and System Sciences*, 65(3):570–586, 2002.

[26] G. Macaya, J.-P. Thiery, and G. Bernardi. An approach to the organization of eukaryotic genomes at a macromolecular level. *Journal of Molecular Biology*, 108:237–254, 1976.

[27] C. S. Madsen, C. P. Regan, and G. K. Owens. Interaction of CArG elements and a GC-rich repressor element in transcriptional regulation of the smooth muscle myosin heavy chain gene in vascular smooth muscle cells. *Journal of Biological Chemistry*, 272(47):29842–29851, 1997.

[28] J. Matoušek. Randomized optimal algorithm for slope selection. *Information Processing Letters*, 39(4):183–187, 1991.

[29] S.-i. Murata, P. Herman, and J. R. Lakowicz. Texture analysis of fluorescence lifetime images of AT- and GC-rich regions in nuclei. *Journal of Hystochemistry and Cytochemistry*, 49:1443–1452, 2001.

[30] A. Nekrutenko and W.-H. Li. Assessment of compositional heterogeneity within and between eukaryotic genomes. *Genome Research*, 10:1986–1995, 2000.

[31] U. Ohler, H. Niemann, G. Liao, and G. M. Rubin. Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. *Bioinformatics*, 17(S1):S199–S206, 2001.

[32] P. Rice, I. Longden, and A. Bleasby. EMBOSS: The European molecular biology open software suite. *Trends in Genetics*, 16(6):276–277, June 2000.

[33] L. Scotto and R. K. Assoian. A GC-rich domain with bifunctional effects on mRNA and protein levels: implications for control of transforming growth factor beta 1 expression. *Molecular and Cellular Biology*, 13(6):3588–3597, 1993.

[34] P. H. Sellers. Pattern recognition in genetic sequences by mismatch density. *Bulletin of Mathematical Biology*, 46(4):501–514, 1984.

[35] P. M. Sharp, M. Averof, A. T. Lloyd, G. Matassi, and J. F. Peden. DNA sequence evolution: the sounds of silence. *Philosophical Transactions of the Royal Society of London Series B, Biological Sciences*, 349:241–247, 1995.

[36] P. Soriano, M. Meunier-Rotival, and G. Bernardi. The distribution of interspersed repeats is nonuniform and conserved in the mouse and human genomes. *Proceedings of the National Academy of Sciences of the United States of America*, 80:1816–1820, 1983.

[37] N. Stojanovic, L. Florea, C. Riemer, D. Gumucio, J. Slightom, M. Goodman, W. Miller, and R. Hardison. Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions. *Nucleic Acids Research*, 27:3899–3910, 1999.

[38] N. Sueoka. Directional mutation pressure and neutral molecular evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 80:1816–1820, 1988.

[39] Z. Wang, E. Lazarov, M. O'Donnel, and M. F. Goodman. Resolving a fidelity paradox: Why Escherichia coli DNA polymerase II makes more base substitution errors in at- compared to GC-rich DNA. *Journal of Biological Chemistry*, 277:4446–4454, 2002.

[40] K. H. Wolfe, P. M. Sharp, and W.-H. Li. Mutation rates differ among regions of the mammalian genome. *Nature*, 337:283–285, 1989.

[41] Y. Wu, R. P. Stulp, P. Elfferich, J. Osinga, C. H. Buys, and R. M. Hofstra. Improved mutation detection in GC-rich DNA fragments by combined DGGE and CDGE. *Nucleic Acids Research*, 27(15):e9, 1999.

[42] S. Zoubak, O. Clay, and G. Bernardi. The gene distribution of the human genome. *Gene*, 174:95–102, 1996.