SIAM J. COMPUT.
Vol. 34, No. 2, pp. 373–387

AN OPTIMAL ALGORITHM FOR THE MAXIMUM-DENSITY SEGMENT PROBLEM*

KAI-MIN CHUNG† AND HSUEH-I LU‡

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 $w_{\min} \le w_i + w_{i+1} + \cdots + w_j \le w_{\max}$. The best previously known algorithm for the problem, due to Goldwasser, Kao, and Lu [*Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, R. Guigó and D. Gusfield, eds., Lecture Notes in Comput. Sci. 2452, Springer-Verlag, New York, 2002, pp. 157–171], 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 [*J. Comput. System Sci.*, 65 (2002), pp. 570–586]. 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.

Key words. bioinformatics, biological sequence analysis, maximum-density segment, slope selection, computational geometry, sequence algorithm, data structure

AMS subject classifications. 68P05, 68Q25, 68R05, 68U05, 68W05, 68W40, 92-08, 92D20

DOI. 10.1137/S0097539704440430

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 from position i to position j. The *width* $w(i,j)$ of $S(i,j)$ is $w_i + w_{i+1} + \cdots + w_j$. The *density* $d(i,j)$ of $S(i,j)$ is $(a_i + a_{i+1} + \cdots + a_j)/w(i,j)$. 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 $w_{\min} \le w(i,j) \le w_{\max}$. The *maximum-density segment* problem is to find a maximum-density segment over all $O(n(w_{\max} - w_{\min} + 1))$ feasible segments.

This problem arises from the investigation of the nonuniformity of nucleotide composition within genomic sequences, which was first revealed through thermal melting and gradient centrifugation experiments [21, 28]. The GC content of the DNA sequences in all organisms varies from 25% to 75%. GC-ratios have the greatest variation among bacterial DNA sequences, while the typical GC-ratios of mammalian

*Received by the editors February 3, 2004; accepted for publication (in revised form) July 7, 2004; published electronically December 1, 2004. A preliminary version of this paper appeared in *Proceedings of the* 11*th Annual European Symposium on Algorithms* (Budapest, Hungary, 2003), G. Di Battista and U. Zwick, eds., Lecture Notes in Comput. Sci. 2832, Springer-Verlag, New York, 2003, pp. 136–147.

http://www.siam.org/journals/sicomp/34-2/44043.html

†Institute of Information Science, Academia Sinica. Part of this work was done while this author was an undergraduate student in the Department of Computer Science and Information Engineering, National Taiwan University.

‡Corresponding author. Institute of Information Science, Academia Sinica, 128 Academia Road, Section 2, Taipei 115, Taiwan, Republic of China (hil@iis.sinica.edu.tw, www.iis.sinica.edu.tw/∼hil/).
This author's research was supported in part by NSC grants 91-2215-E-001-001 and 92-2218-E-001-001.

genomes stay in the range 45–50%. Despite intensive research effort in the past two decades, the underlying causes of the observed heterogeneity remain debatable [2, 3, 5, 8, 9, 10, 11, 18, 40, 42]. Researchers [32, 39] observed that the extent of the compositional heterogeneity in a genomic sequence strongly correlates with its GC content regardless of genome size. Other investigations showed that gene length [7], gene density [44], patterns of codon usage [37], distribution of different classes of repetitive elements [7, 38], number of isochores [2], lengths of isochores [32], and recombination rate within chromosomes [12] are all correlated with GC content. More research related to GC-rich segments can be found in [16, 17, 20, 23, 29, 31, 35, 41, 43] and 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 a G or C, and $a_i = 0$ otherwise. In the work of Huang [19], 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 occurs very often. In such applications, a_i could be the relative frequency with which the corresponding DNA segments appear in the given patterns. Further natural applications of this problem can be designed for sophisticated sequence analysis such as mismatch density [36], ungapped local alignments [1], annotated multiple sequence alignments [39], promoter mapping [22], and promoter recognition [33].

For the *uniform* case, i.e., $w_i = 1$ for all indices i, Nekrutenko and Li [32] and Rice, Longden, and Bleasby [34] 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 [19] 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 [27] gave an $O(n \log w_{\min})$-time algorithm for this case based on right-skew decompositions of a sequence. (See [26] for related software.) The case with general $w_{\max}$ was first investigated by Goldwasser, Kao, and Lu [13, 14], who gave an $O(n)$-time algorithm for the uniform case. Recently, Kim [25] 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, 24, 30]. Unfortunately, Kim's analysis of time complexity has a flaw which seems difficult to fix.[1] For the general (i.e., *nonuniform*) 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 time required 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 the

[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 p. 340 in [25].) 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.

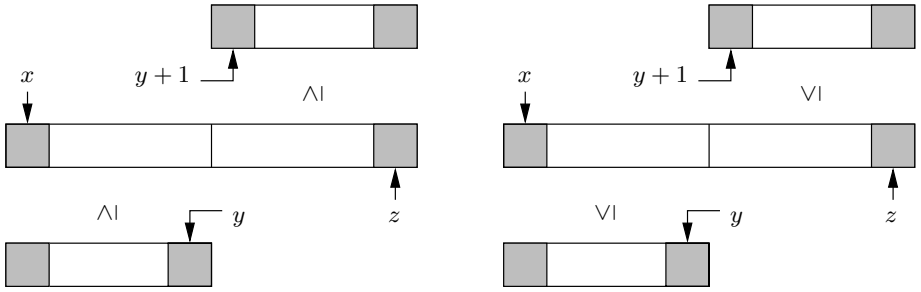

FIG. 1.1. *An illustration for* (1.1)*: There exist only two possibilities for the order among* $d(x,y), d(x,z), d(y+1,z)$.

order of any two of them:

$$\begin{aligned} &d(x,y) \le d(y+1,z) \Leftrightarrow d(x,y) \le d(x,z) \Leftrightarrow d(x,z) \le d(y+1,z), \\ &d(x,y) \ge d(y+1,z) \Leftrightarrow d(x,y) \ge d(x,z) \Leftrightarrow d(x,z) \ge d(y+1,z). \end{aligned} \tag{1.1}$$

(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 1.1.) 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, 22, 33, 36, 39], 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 $n_{j-1} < i \le n_j$ and $1 \le j \le m$. If $w'_j = 1$ holds for all $1 \le j \le m$, 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 $\{x, x+1, \ldots, y\}$. Without loss of generality, we may assume that $w_1 + w_2 + \cdots + w_n \ge w_{\min}$ and $w_i \le w_{\max}$ holds for each $i = 1, 2, \ldots, n$. 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 $w_{\max}$. Define $\phi(x,y)$ 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 $S(x,y)$. Let j_0 be the smallest index with $w(1,j_0) \ge w_{\min}$. Let $J = [j_0, n]$. For each $j \in J$, let ℓ_j be the smallest index i with $w(i,j) \le w_{\max}$, and let r_j be the largest index i with $w(i,j) \ge w_{\min}$. That is, $S(i,j)$ is feasible if and only if $i \in [\ell_j, r_j]$. (Figure 2.1 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 know only that ℓ_j and r_j are both (not necessarily strictly) 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 $d(k,j) = \max\{d(i,j) \mid i \in [\ell_j, r_j]\}$. There must be an index j^* such that $S(i^*_{j^*}, j^*)$ is a maximum-density segment of

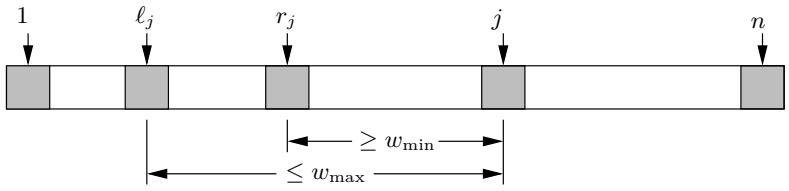

FIG. 2.1. *An illustration for the definitions of ℓ_j and r_j.*

```
    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;
```

FIG. 2.2. *Our main algorithm.*

S. Therefore, a natural but seemingly difficult possibility for solving the maximum-density segment problem would be to compute i_j^* for all indices $j \in J$ in $O(n)$ time. Instead, our approach is to compute a pair (i_j, j) of indices with $i_j \in [\ell_j, r_j]$ for each index $j \in J$ by the algorithm shown in Figure 2.2. More specifically, each iteration of our algorithm, based upon (1.1), keeps chopping off the lowest-density prefix without affecting the feasibility of the remaining segment until its density does not go any higher. For brevity of presentation, each algorithm throughout the paper outputs a linear number of index pairs (i, j). (See, e.g., step 4 of algorithm MAIN shown in Figure 2.2.) Clearly, it takes a linear-time postprocessing for the produced index pairs to obtain one that maximizes $d(i, j)$. The rest of the section ensures the correctness of our algorithm by showing $i_{j^*} = i_{j^*}^*$, and thus reduces the maximum-density segment problem to implementing our algorithm to run in $O(n)$ time.

LEMMA 2.1. *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 (1.1), 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 (1.1) 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 (1.1) and the first inequality, (c) the third inequality is by $i^* \le \phi(i, r-1)$ and the definition of $\phi(i, r-1)$, and (d) the last inequality is by (1.1) and the third inequality. It follows from $d(i^*, j) \ge d(i^*, \phi(i, r-1))$ and (1.1) that $d(\phi(i, r-1)+1, j) \ge d(i^*, j)$, contradicting the definition of i^* by $i^* < \phi(i, r-1)+1$. □

THEOREM 2.2. *Algorithm* MAIN *correctly solves the maximum-density segment problem.*

Proof. We prove the theorem by showing $i_{j^*} = i^*_{j^*}$. By $\ell_{j_0} = i_{j_0-1} = 1$ and Lemma 2.1, the equality holds if $j^* = j_0$. The rest of the proof assumes $j^* > j_0$. By Lemma 2.1 and $\ell_{j^*} \le i^*_{j^*}$, it suffices to ensure $i_{j^*-1} \le i^*_{j^*}$. Assume for contradiction that there is an index $j \in [j_0, j^*-1]$ with $i_{j-1} \le i^*_{j^*} < i_j$. By $j < j^*$, we know $\ell_j \le i^*_{j^*}$. By Lemma 2.1 and $\max(\ell_j, i_{j-1}) \le i^*_{j^*} < i_j \le r_j$, we have $d(i_j, j) \ge d(i^*_{j^*}, j)$. It follows from (1.1) and $i^*_{j^*} < i_j$ that $d(i^*_{j^*}, j) \ge d(i^*_{j^*}, i_j - 1)$. By $\ell_{j^*} \le i^*_{j^*} < i_j \le r_{j^*}$ and the definition of j^*, we know $d(i^*_{j^*}, j^*) > d(i_j, j^*)$. It follows from $i^*_{j^*} < i_j$ and (1.1) that $d(i^*_{j^*}, i_j - 1) > d(i^*_{j^*}, j^*)$. Therefore, $d(i_j, j) \ge d(i^*_{j^*}, j) \ge d(i^*_{j^*}, i_j - 1) > d(i^*_{j^*}, j^*)$, contradicting the definition of j^*. □

One can verify that the value of i increases by at least 1 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 an $O(1)$-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., $w_{\max} \ge w(1, n)$, we have $\ell_j = 1$ for all $j \in J$. 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 $p \le q$ 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 3.1 for an illustration.)

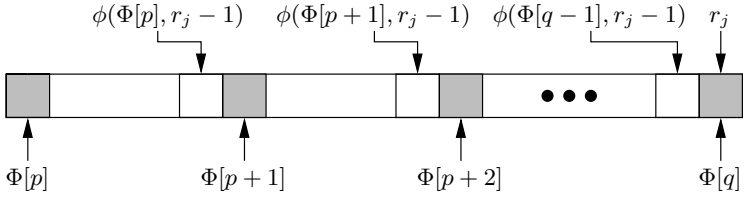

FIG. 3.1. *An illustration for Condition* C_j.

Then, the subroutine call to BEST(i_{j-1}, r_j, j) can clearly be replaced by LBEST(j), as defined in Figure 3.2. 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. With the initialization of letting $p = q = r_{j_0-1} = \Phi[1] = 1$, as described at step 1 of algorithm LMAIN, Condition C_{j_0-1} clearly holds before the subroutine call to UPDATE(j_0). The following lemma is crucial in ensuring the correctness and efficiency of our implementation shown in Figure 3.2.

LEMMA 3.1. *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.*

```
    ALGORITHM LMAIN.
1       let p = q = r_{j_0-1} = Φ[1] = 1;
2       for j = j_0 to n do {
3           call UPDATE(j);
4           let i_j = LBEST(j);
5           output (i_j, j);
6       }

    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       }
```

FIG. 3.2. *An efficient implementation for the case that $w_{\max}$ is ineffective.*

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$(\Phi[p], \Phi[q], j)$.

Proof. Statement 1. We need the following condition.

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

Clearly, Condition C_j is exactly Condition D_{r_j}. To prove the statement, we show that if Condition D_{r-1} holds, then Condition D_r holds right after executing steps 2–5 of UPDATE (i.e., an iteration of the for-loop of UPDATE), although the value of q may change.

Consider the moment when step 4 of UPDATE is about to be executed. We first show that if $p < q$, then $\Phi[t] = \phi(\Phi[t-1], r-1) + 1$ holds for each $t \in [p+1, q]$. By the definition of ϕ, we have

$$\phi(\ell, r-1) = \begin{cases} r-1 & \text{if } d(\ell, \phi(\ell, r-2)) \geq d(\ell, r-1), \\ \phi(\ell, r-2) & \text{otherwise.} \end{cases} \tag{3.1}$$

By Condition D_{r-1}, we know $\phi(\Phi[q-1], r-2) = \Phi[q] - 1$. With $\ell = \Phi[q-1]$ plugged into (3.1), we know that if $d(\Phi[q-1], \Phi[q]-1) < d(\Phi[q-1], r-1)$, then $\phi(\Phi[q-1], r-1) = \phi(\Phi[q-1], r-2)$. It follows from the condition of step 2 of UPDATE that $\phi(\Phi[q-1], r-1) = \phi(\Phi[q-1], r-2)$. 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 (1.1), we have $d(\ell, m) < d(\ell, \phi(m+1, r-1)) < d(m+1, \phi(m+1, r-1))$. As a result, we have $d(\ell, m) < d(m+1, r-1)$, which by (1.1) implies $d(\ell, m) < d(\ell, r-1)$. Thus $\phi(\ell, r-1) = \phi(\ell, r-2)$.

Therefore, $\phi(\Phi[t], r-1) = \phi(\Phi[t], r-2)$ implies $\phi(\Phi[t-1], r-1) = \phi(\Phi[t-1], r-2)$. As a result, $\phi(\Phi[t], r-1) = \phi(\Phi[t], r-2)$ holds for each $t \in [p, q-1]$. By Condition D_{r-1}, we know $\Phi[t] = \phi(\Phi[t-1], r-1) + 1$ holds for each $t \in [p+1, q]$.

Since the value of q will be incremented by 1 after executing step 4 of UPDATE and $\Phi[q+1]$ will equal r after executing step 5 of UPDATE, it remains to show $\phi(\Phi[q], r-1) = r-1$. Clearly, the equality holds if $\Phi[q] = r-1$, i.e., step 3 was not executed. If step 3 was executed at least once, then we know $d(\Phi[q], \Phi[q+1]-1) \geq d(\Phi[q], r-1)$, i.e., $d(\Phi[q], \phi(\Phi[q], r-2)) \geq d(\Phi[q], r-1)$. By plugging $\ell = \Phi[q]$ into (3.1), we know $\phi(\Phi[q], r-1) = r-1$.

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

Proof. By Lemma 3.1(1) and the 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 3.1(2), LBEST(j) is 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 2.2.

As for the efficiency of LMAIN, observe that $q - p \geq 0$ 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. Since step 4 of UPDATE, which is the only place that increases the value of $q-p$, 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 this 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.

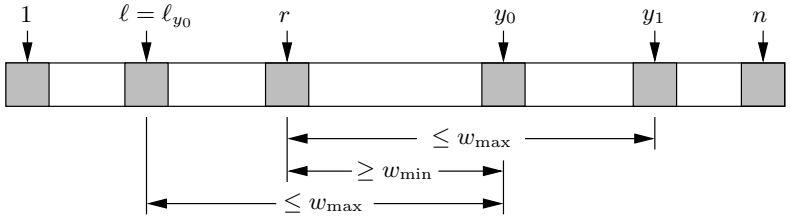

FIG. 4.1. *An illustration for the relation among* ℓ, r, y_0, y_1.

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 y_1 is the largest index in J with $w(r, y_1) \leq w_{\max}$. See Figure 4.1 for an illustration. The *variant version* of the maximum-density segment problem is to look for a maximum-density segment over

```
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   }
```

FIG. 4.2. *Our algorithm for the variant version of the maximum-density segment problem, where function* BEST *is as defined in Figure* 2.2.

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 4.2, 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 role of the index r_j in MAIN. We have the following lemma whose proof is very similar to that of Theorem 2.2.

LEMMA 4.1. *Algorithm* VMAIN *solves the variant version of the maximum-density problem correctly.*

Proof. We prove the theorem by showing $x_{y^*} = x_{y^*}^*$. By $\ell_{y_0} = x_{y_0 - 1} = \ell$ and Lemma 2.1, the equality holds if $y^* = y_0$. The rest of the proof assumes $y^* > y_0$. By Lemma 2.1 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 2.1 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 (1.1) 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 the definition of y^*, we know $d(x_{y^*}^*, y^*) > d(x_y, y^*)$. It follows from $x_{y^*}^* < x_y$ and (1.1) 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 4.3. The following lemma ensures the correctness and efficiency of our implementation VARIANT shown in Figure 4.3.

LEMMA 4.2. *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. One can easily verify that if $\Psi[i] = \phi(i, r - 1)$ holds for each index $i \in$

```
    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       }
```

FIG. 4.3. *An efficient implementation for the algorithm* VMAIN.

$[\ell, r-1]$, then VBEST is a faithful implementation of BEST. Therefore, by Lemma 4.1, the correctness of VARIANT can be ensured by showing that after calling INIT$(\ell, r-1)$ at step 3 of algorithm VARIANT , $\Psi[i] = \phi(i, r-1)$ holds for each index $i \in [\ell, r-1]$. Note that for brevity of the following proof, we slightly abuse the notation r in Figure 4.3. That is, although the subroutine call at step 3 of algorithm VARIANT is INIT$(\ell, r-1)$, the second parameter in the definition of subroutine INIT in Figure 4.3 becomes r. Let us make it clear that the rest of the proof (i) lets r denote the one in the definition of subroutine INIT(ℓ, r), and (ii) proves that $\Psi[i] = \phi(i, r)$ holds for each index $i \in [\ell, r]$ at the end of the subroutine call to subroutine INIT(ℓ, 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.

> 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 (1.1), 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 (1.1), 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 $\phi(z+1, r) = \Psi[z+1] \leq$

```
    ALGORITHM GENERAL.
1       let p = q = r_{j0-1} = Φ[1] = 1;
2       for j = j0 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      }
```

FIG. 4.4. *Our algorithm for the general case, where* UPDATE *and* LBEST *are defined in Figure* 3.2 *and* VARIANT *is defined in Figure* 4.3.

$\phi(x,r)$. By $\phi(z+1,r) \le \phi(x,r) \le r$ and the definition of $\phi(z+1,r)$, we have $d(z+1,\phi(x,r)) \ge d(z+1,\phi(z+1,r))$. By definition of $\phi(x,r)$ and (1.1), we have $d(x,z) \ge d(x,\phi(x,r)) \ge d(z+1,\phi(x,r))$. By $d(x,z) \ge d(z+1,\phi(z+1,r))$ and (1.1), we have $d(x,z) \ge d(x,\phi(z+1,r))$. 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.

As for the time complexity, we first observe that ℓ and y_1 can be found from r and y_0 in $O(r-\ell+y_1-y_0+1)$ time:

- Let $\ell = r$, and then repeatedly decrease ℓ by 1 as long as $w(\ell-1,y_0) \le w_{\max}$ holds.
- Let $y_1 = y_0$, and then repeatedly increase y_1 by 1 as long as $w(r,y_1+1) \le w_{\max}$ holds.

Secondly, one can see that the rest of the implementation also runs in $O(r-\ell+y_1-y_0+1)$ time 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 $O(r-\ell+1)$ iterations. To see statement (a), just observe that the value of index x (i) never decreases, (ii) stays in $[\ell,r]$, and (iii) 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 $|Z_s| \ge 1$ 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 4.4. Algorithm GENERAL is obtained by inserting four lines of code (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 $\ell_j \le \Phi[p]$. 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 4.5. This is when the variant version comes in: As shown in the next theorem, we can remedy the problem by calling VARIANT$(\Phi[p], j)$.

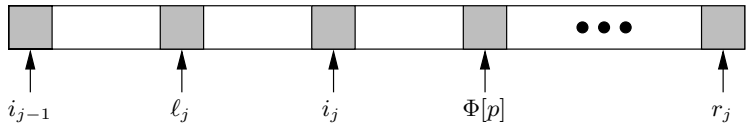

FIG. 4.5. *An illustration for the situation when Steps* 6 *and* 7 *of* GENERAL *are needed.*

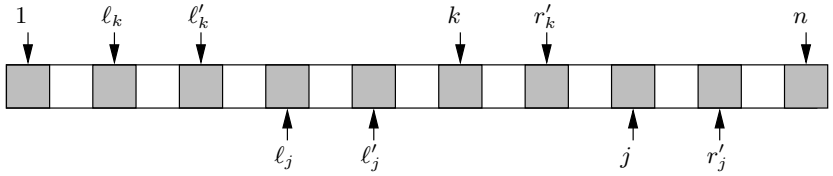

FIG. 4.6. *An illustration showing that the overall running time of all subroutine calls to* VARIANT(ℓ'_j, j) *in* GENERAL *is* $O(n)$.

THEOREM 4.3. *Algorithm* GENERAL *solves the maximum-density segment problem in an online manner in linear time.*

Proof. We prove the correctness of GENERAL by showing that $i^*_{j^*} \neq i_{j^*}$ implies $i^*_{j^*} = x_{j^*}$. By Lemma 3.1(1), Condition C_j holds after the subroutine call UPDATE(j) at step 3 of GENERAL. Observe that steps 4 and 5 of GENERAL, which may increase the value of p, do not affect the validity of Condition C_j. Also, 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 3.1(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 $i_{j^*} \neq i^*_{j^*}$, there must be an index $j \in [j_0, j^*]$ with $i_{j-1} \leq i^*_{j^*} < i_j$. It can be proved as follows that $i^*_{j^*} \leq \ell'_j - 1$.

> Assume $\ell'_j \leq i^*_{j^*}$ for contradiction. It follows from Lemma 3.1(2) and (1.1) that $d(i_j, j) \geq d(i^*_{j^*}, j) \geq d(i^*_{j^*}, i_j - 1)$. By the definition of j^*, we have $d(i^*_{j^*}, j^*) > d(i_j, j^*)$, which by (1.1) 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 $i_{j-1} \leq i^*_{j^*} \leq \ell'_j - 1$, we know $w(\ell'_j - 1, j^*) \leq w(i^*_{j^*}, j^*) \leq w_{\max}$. Thus, $S(i^*_{j^*}, j^*)$ is a feasible segment in the variant version of the maximum-density segment problem for S with respect to indices $r = \ell'_j$ and $y_0 = j$. By Lemma 4.2, the subroutine call VARIANT(ℓ'_j, j) at step 7 of algorithm GENERAL has to output 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 $q - p \geq 0$ holds throughout the execution of GENERAL. Step 4 of UPDATE, which is the only place that increases the value of $q - p$, increases the value of $q - p$ by 1 for $O(n)$ times. Note that each iteration of the while-loops of GENERAL, LBEST, and UPDATE decreases the value of $q - p$ by 1. 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 4.2, it suffices to show $\ell'_k < \ell_j$ and $r'_k < j$ as follows. (See Figure 4.6.) By the definition of GENERAL, we know that $i_{j-1} < \ell_j$, which is ensured by the situation illustrated in Figure 4.5. By $k < j$, we have $\ell'_k \leq i_{j-1}$, implying $\ell'_k < \ell_j$. Moreover, by the definitions of ℓ_j and

r'_k, one can easily verify that $\ell'_k < \ell_j$ implies $r'_k < j$.

It is not difficult to see that our algorithm shown in Figure 4.4 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 the input sequence $S = (a_1, w_1), (a_2, w_2), \ldots, (a_n, w_n)$ is run-length encoded [15], i.e., 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 that $a_i = a'_j$ holds for all indices i and j with $n_{j-1} < i \le n_j$ and $1 \le j \le m$. Our algorithm for solving the maximum-density problem for the $O(m)$-space representable sequence S is shown in Figure 5.1.

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), ..., (n'_m a'_m, n'_m);
4    let (i', j') be an optimal output of GENERAL(w_min, w_max, S');
5    output (n_{i'−1} + 1, n_{j'});
6    for k = 1 to m do {
7        if n_k ≥ w_min then
8            output (ℓ_{n_k}, n_k) and (r_{n_k}, n_k);
9        if n_{k−1} + w_min ≤ n then
10           output (n_{k−1} + 1, n_{k−1} + w_min) and (n_{k−1} + 1, min(n, n_{k−1} + w_max));
11   }
```

FIG. 5.1. *Our algorithm that handles sparse input sequence for the uniform case, where* GENERAL *is defined in Figure* 4.4.

THEOREM 5.1. *Algorithm* SPARSE *solves the maximum-density problem for the above $O(m)$-space representable sequence in $O(m)$ time.*

Proof. By Theorem 4.3, 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 as follows 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 $a_{j^*} = a_{j^*+1}$. It follows from the optimality of $S(i^*, j^*)$ that $a_{i^*} \ge a_{j^*+1}$ and $a_{i^*-1} \le a_{j^*}$, implying $a_{i^*-1} = a_{i^*} = a_{j^*} = a_{j^*+1}$. Therefore, $S(i^* + 1, j^* + 1)$ is also a maximum-density segment.

It remains to show that our algorithm works correctly for each possible case.

- 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 1–5 of SPARSE take care of this case.
- Case 2: $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \in \{n_1, n_2, \ldots, n_m\}$. Clearly, if $i^* \in \{\ell_{j^*}, r_{j^*}\}$, $S(i^*, j^*)$ can be discovered by steps 7 and 8 of SPARSE. Since $i^* - 1 \notin \{n_0, n_1, \ldots, n_{m-1}\}$, we have $a_{i^*-1} = a_{i^*}$. If $a_{i^*-1} = a_{i^*} \ne d(i^*, j^*)$, then by (1.1) we have either $d(i^* - 1, j^*) > d(i^*, j^*)$ or $d(i^* + 1, j^*) > d(i^*, j^*)$,

which implies that either $S(i^* - 1, j^*)$ or $S(i^* + 1, j^*)$ is infeasible, and thus $i^* \in \{\ell_{j^*}, r_{j^*}\}$. On the other hand, if $a_{i^*-1} = a_{i^*} = d(i^*, j^*)$ and $i^* \neq \ell_j^*$, then $S(i^*-1, j^*)$ is also a feasible segment with maximum density. We can continue the same argument until we have 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 7 and 8 of SPARSE.

- Case 3: $i^* - 1 \in \{n_0, n_1, \ldots, n_{m-1}\}$ and $j^* \notin \{n_1, n_2, \ldots, n_m\}$. The proof of this case, omitted for brevity, is very similar to that of Case 2.

The theorem is proved. □

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

REFERENCES

[1] N. N. Alexandrov and V. V. Solovyev, *Statistical significance of ungapped sequence alignments*, in Proceedings of the Pacific Symposium on Biocomputing, Maui, HI, 1998, World Scientific, River Edge, NJ, Vol. 3, pp. 461–470.

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

[3] G. Bernardi and G. Bernardi, *Compositional constraints and genome evolution*, J. Molec. Evolution, 24 (1986), pp. 1–11.

[4] H. Brönnimann and B. Chazelle, *Optimal slope selection via cuttings*, Comput. Geom., 10 (1998), pp. 23–29.

[5] B. Charlesworth, *Genetic recombination: Patterns in the genome*, Current Biol., 4 (1994), pp. 182–184.

[6] R. Cole, J. S. Salowe, W. L. Steiger, and E. Szemerédi, *An optimal-time algorithm for slope selection*, SIAM J. Comput., 18 (1989), pp. 792–810.

[7] L. Duret, D. Mouchiroud, and C. Gautier, *Statistical analysis of vertebrate sequences reveals that long genes are scarce in GC-rich isochores*, J. Mol. Evol., 40 (1995), pp. 308–371.

[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 Res., 20 (1992), pp. 1497–1501.

[9] A. Eyre-Walker, *Recombination and mammalian genome evolution*, Proc. Roy. Soc. London Ser. B, 252 (1993), pp. 237–243.

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

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

[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*, Mol. Biol. Evol., 18 (2001), pp. 1139–1142.

[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 Proceedings of the Second International Workshop on Algorithms in Bioinformatics (Rome, Italy, 2002), R. Guigó and D. Gusfield, eds., Lecture Notes in Comput. Sci. 2452, Springer-Verlag, New York, 2002, pp. 157–171.

[14] M. H. Goldwasser, M.-Y. Kao, and H.-I. Lu, *Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications*, J. Comput. System Sci., to appear.

[15] S. W. Golomb, *Run-length encodings*, IEEE Trans. Inform. Theory, 12 (1966), pp. 399–401.

[16] 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 Res., 26 (1998), pp. 1548–1549.

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

[18] G. P. Holmquist, *Chromosome bands, their chromatin flavors, and their functional features*, Am. J. Hum. Genet., 51 (1992), pp. 17–37.

[19] X. Huang, *An algorithm for identifying regions of a DNA sequence that satisfy a content requirement*, Comput. Appl. Biosci., 10 (1994), pp. 219–225.
[20] 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 Res., 24 (1996), pp. 4249–4255.
[21] R. B. Inman, *A denaturation map of the* 1 *phage DNA molecule determined by electron microscopy*, J. Mol. Biol., 18 (1966), pp. 464–476.
[22] I. P. Ioshikhes and M. Q. Zhang, *Large-scale human promoter mapping using CpG islands*, Nature Genet., 26 (2000), pp. 61–63.
[23] 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?*, EMBO J., 16 (1997), pp. 4456–4466.
[24] M. J. Katz and M. Sharir, *Optimal slope selection via expanders*, Inform. Process. Lett., 47 (1993), pp. 115–122.
[25] S. K. Kim, *Linear-time algorithm for finding a maximum-density segment of a sequence*, Inform. Process. Lett., 86 (2003), pp. 339–342.
[26] Y.-L. Lin, X. Huang, T. Jiang, and K.-M. Chao, *MAVG: Locating nonoverlapping maximum average segments in a given sequence*, Bioinformatics, 19 (2003), pp. 151–152.
[27] Y.-L. Lin, T. Jiang, and K.-M. Chao, *Algorithms for locating the length-constrained heaviest segments, with applications to biomolecular sequence analysis*, J. Comput. System Sci., 65 (2002), pp. 570–586.
[28] G. Macaya, J.-P. Thiery, and G. Bernardi, *An approach to the organization of eukaryotic genomes at a macromolecular level*, J. Mol. Biol., 108 (1976), pp. 237–254.
[29] 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*, J. Biol. Chem., 272 (1997), pp. 29842–29851.
[30] J. Matoušek, *Randomized optimal algorithm for slope selection*, Inform. Process. Lett., 39 (1991), pp. 183–187.
[31] S.-I. Murata, P. Herman, and J. R. Lakowicz, *Texture analysis of fluorescence lifetime images of AT- and GC-rich regions in nuclei*, J. Histochem. Cytochem., 49 (2001), pp. 1443–1452.
[32] A. Nekrutenko and W.-H. Li, *Assessment of compositional heterogeneity within and between eukaryotic genomes*, Genome Res., 10 (2000), pp. 1986–1995.
[33] 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 (2001), pp. S199–S206.
[34] P. Rice, I. Longden, and A. Bleasby, *EMBOSS: The European molecular biology open software suite*, Trends Genet., 16 (2000), pp. 276–277.
[35] 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*, Mol. Cell. Biol., 13 (1993), pp. 3588–3597.
[36] P. H. Sellers, *Pattern recognition in genetic sequences by mismatch density*, Bull. Math. Biol., 46 (1984), pp. 501–514.
[37] P. M. Sharp, M. Averof, A. T. Lloyd, G. Matassi, and J. F. Peden, *DNA sequence evolution: The sounds of silence*, Philos. Trans. Soc. Lond. B Biol. Sci., 349 (1995), pp. 241–247.
[38] P. Soriano, M. Meunier-Rotival, and G. Bernardi, *The distribution of interspersed repeats is nonuniform and conserved in the mouse and human genomes*, Proc. Natl. Acad. Sci. USA, 80 (1983), pp. 1816–1820.
[39] 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 Res., 27 (1999), pp. 3899–3910.
[40] N. Sueoka, *Directional mutation pressure and neutral molecular evolution*, Proc. Natl. Acad. Sci. USA, 80 (1988), pp. 1816–1820.
[41] 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*, J. Biol. Chem., 277 (2002), pp. 4446–4454.
[42] K. H. Wolfe, P. M. Sharp, and W.-H. Li, *Mutation rates differ among regions of the mammalian genome*, Nature, 337 (1989), pp. 283–285.
[43] 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 Res., 27 (1999), article e9.
[44] S. Zoubak, O. Clay, and G. Bernardi, *The gene distribution of the human genome*, Gene, 174 (1996), pp. 95–102.