Faster Optimal Algorithms for Segment Minimization with Small Maximal Value

Therese Biedl

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Stephane Durocher

Department of Computer Science, University of Manitoba, Winnipeg, Canada

Céline Engelbeen

Département de Mathématique, Université Libre de Bruxelles, Brussels, Belgium

Samuel Fiorini

Département de Mathématique, Université Libre de Bruxelles, Brussels, Belgium

Maxwell Young

School of Computing, National University of Singapore, Singapore

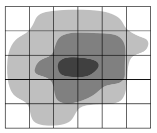
Abstract

The segment minimization problem consists of finding a smallest set of binary matrices (*segments*), where non-zero values in each row of each matrix are consecutive, each matrix is assigned a positive integer weight (a *segment-value*), and the weighted sum of the matrices corresponds to the given input intensity matrix. This problem has direct applications in intensity-modulated radiation therapy, an effective form of cancer treatment.

We study here the special case when the largest value H in the intensity matrix is small. We show that for an intensity matrix with one row, this problem is fixed-parameter tractable (FPT) in H; our algorithm obtains a significant asymptotic speedup over the previous best FPT algorithm. We also show how to solve the full-matrix problem faster than all previously known algorithms. Finally, we address a closely related problem that deals with minimizing the number of segments subject to a minimum *beam-on time*, defined as the sum of the segment-values, and again improve the running time of previous algorithms. Our algorithms have running time $O(mn)$ in the case that the matrix has only entries in $\{0, 1, 2\}$.

Keywords: Intensity-modulated radiation therapy (IMRT), multileaf collimator (MLC), segmentation problem, algorithm

Email addresses: `biedl@uwaterloo.ca` (Therese Biedl), `durocher@cs.umanitoba.ca` (Stephane Durocher), `celine.engelbeen@ulb.ac.be` (Céline Engelbeen), `sfiorini@ulb.ac.be` (Samuel Fiorini), `m22young@uwaterloo.ca` (Maxwell Young)



$$\Rightarrow \begin{array}{|c|c|c|c|c|c|}\hline 0&1&1&1&1&0\\\hline 1&1&2&2&2&1\\\hline 1&2&4&4&2&0\\\hline 1&1&2&2&1&0\\\hline 0&1&1&0&0&0\\\hline\end{array} = 2 \times \begin{array}{|c|c|c|c|c|c|}\hline 0&0&0&0&0&0\\\hline 0&0&0&0&0&0\\\hline 0&0&1&1&1&0\\\hline 0&0&0&0&0&0\\\hline 0&0&0&0&0&0\\\hline\end{array} + 1 \times \begin{array}{|c|c|c|c|c|c|}\hline 0&1&1&1&0&0\\\hline 0&0&1&1&1&0\\\hline 0&1&1&1&0&0\\\hline 0&0&1&1&0&0\\\hline 0&0&0&0&0&0\\\hline\end{array} + 1 \times \begin{array}{|c|c|c|c|c|c|}\hline 0&0&0&0&1&0\\\hline 1&1&1&1&1&1\\\hline 1&1&1&1&0&0\\\hline 1&1&1&1&1&0\\\hline 0&1&1&0&0&0\\\hline\end{array}$$

Figure 1: An example of a segmentation of an intensity matrix where $H = 4$.

1. Introduction

Intensity-modulated radiation therapy (IMRT) is an effective form of cancer treatment, where radiation produced by a linear accelerator is delivered to the patient through a multileaf collimator (MLC). The MLC is mounted on an arm that can revolve freely around the patient so that he or she can be irradiated from several angles. We focus on the so-called *step-and-shoot* mode, where the radiation is delivered in a series of steps. In each step, two banks of independent metal leaves in the MLC are positioned to obstruct certain portions of the radiation field, while leaving others exposed. Neither the head of the MLC, nor its leaves move during irradiation. A treatment plan specifies the amount of radiation to be delivered along each angle.

For any given angle, the radiation field is discretized and decomposed into $m \times n$ bixels [4], where m is typically the number of pairs of leaves of the MLC. This determines a decomposition of the radiation beam into $m \times n$ *beamlets*. The amount of radiation is represented as an $m \times n$ *intensity matrix* A of non-negative integer values, whose entries represent the amount of radiation to be delivered through the corresponding bixel.

The leaves of the MLC can be seen as partially covering rows of A; for each row i of A there are two leaves, one of which may slide inwards from the left to cover the elements in columns 1 to $\ell - 1$ of that row, while the other may slide inwards from the right to cover the elements in columns $r + 1$ to n. Thus the entries of A that are not covered form an interval $[\ell, r] := \{\ell, \ell + 1, \ldots, r\}$ of consecutive columns. After each step, the amount of radiation applied in that step (this can differ per step) is subtracted from each entry of A that has not been covered. The irradiation is completed when all entries of A are equal to 0.

Setting leaf positions in each step requires time. Minimizing the number of steps reduces treatment time, which increases patient comfort, and can result in increased patient throughput, reduced machine wear, and overall reduced cost of the procedure. Minimizing the number of steps for a given treatment plan is the primary objective of this paper.

Formally, a *segment* is an $m \times n$ binary matrix S such that ones in each row of S are consecutive. Each segment S has an associated non-negative integer weight which we call the *segment-value*, denoted by $v(S)$. We call a segment a t-segment if its value is t. A *segmentation* of A is a set of segments whose weighted sum equals A. So, $\mathcal{S}$ is a segmentation of A if and only if we have $A = \sum_{S \in \mathcal{S}} v(S)S$. Figure 1 illustrates a segmentation of an intensity matrix.

The *(minimum-cardinality) segmentation problem* is, given an intensity matrix A, to find a minimum-cardinality segmentation of A. We also consider the special case of a matrix A with one row, which we call the *single-row segmentation problem*, in contrast with the more general *full-matrix segmentation problem* with m rows.

We also briefly examine a different, but closely related, *lex-min* problem: find a minimum-cardinality segmentation among those with minimum *beam-on time*, defined as the total value

$\sum_{S \in \mathcal{S}} v(S)$ of the segmentation.[1] As the segmentation problem focuses on the time incurred for establishing leaf positions, optimizing the beam-on time also has implications for making procedures more efficient by reducing the time spent administering the treatment corresponding to the segments themselves.

1.1. Related Work

The segmentation problem is known to be NP-complete in the strong sense, even for a single row [2, 3, 9], as well as APX-complete [4]. Bansal *et al.* [4] provide a 24/13-approximation algorithm for the single-row problem and give better approximations for more constrained versions. Work by Collins *et al.* [10] shows that the single-*column* version of the problem is NP-complete and provides some non-trivial lower bounds given certain constraints. Work by Luan *et al.* [15] gives two approximation algorithms for the full $m \times n$ segmentation problem, and Biedl *et al.* [6] extend this work to achieve better approximation factors.

A number of heuristics are known [3, 11, 13, 18] as well as approaches for obtaining optimal (exact) solutions [1, 7, 17]. Particularly relevant to our work is that of Cambazard *et al.* [8] who show that the segmentation of a single row is fixed-parameter tractable (FPT); specifically, they give an algorithm which achieves an optimal segmentation in $O(p(H)^2\, n)$ time, where H is the largest value in A and $p(H)$ is the number of partitions of H. Recall that a partition of H is an expression of H as an unordered sum of positive integers.

Kalinowski [14] studies the lex-min problem and gives polynomial time algorithms for the case when H is a constant. In the single-row case, he gives an $O(p(H)^2\, n)$ time algorithm. The solution output by this first algorithm is also optimal for the minimum-cardinality segmentation problem (this follows from known results, e.g. [4]). For general $m \times n$ intensity matrices, he provides an $O(2^H \sqrt{H}\, m\, n^{2H+2})$ time algorithm. From this second algorithm, one can derive an algorithm for the full $m \times n$ minimum segmentation problem with time complexity $O(2^H H^{5/2}\, m\, n^{2H+3})$ by guessing the beam-on time T of a minimum-cardinality segmentation and appending a row to the intensity matrix to increase its minimum beam-on time to T; it can be shown (e.g. see our Lemma 3.3) that $T \in O(H^2\, n)$.

1.2. Our Contributions

We summarize our contributions below:

- For the single-row segmentation problem, we provide a faster exact algorithm. In particular, our algorithm runs in $O(p(H)\, H\, n)$ time, which is polynomial in n so long as $H \in O(\log^2 n)$. In comparison to the result of Cambazard *et al.* [8], our algorithm is faster by a factor of $\Omega(p(H)/H)$.

Significant challenges remain in solving the full-matrix problem and here we achieve two important results:

- For general H, we give an algorithm that yields an optimal solution to the full-matrix segmentation problem in $O(m\, n^H / 2^{(1-\epsilon)(H-1)})$ time for an arbitrarily small constant $\epsilon > 0$.

[1]The lex-min problem is also known as the *min DT-min DC* problem, where DT stands for *decomposition time* (i.e., the beam-on time) and DC stands for decomposition cardinality (i.e., the number of segments); however, we refer to this as the lex-min problem throughout.

In contrast, applying the variant of Kalinowski's algorithm mentioned above yields a worst-case running time of $\Omega(m\,n^{2H+3})$. Therefore, our result yields a near-quadratic improvement in the running time.

- For $H = 2$, the full matrix problem can be solved optimally in $O(m\,n)$ time in contrast to the $O(m\,n^2)$ time implied by the previous result for general H. This result also has implications for the approximation algorithms in [6] where it can be employed as a subroutine to improve results in practice.

Finally, we address the lex-min problem:

- For general H, we give an algorithm that yields an optimal solution to the full-matrix lex-min problem in time $O(m\,n^H/2^{(1/2-\epsilon)(H-1)})$. In comparison to the previous best result by Kalinowski [14], our algorithm yields a near-quadratic improvement in the running time.

Therefore, our algorithms represent a significant asymptotic speed-up and the techniques required to achieve these improvements are non-trivial.

2. Single-row segmentation

In this section, we give an algorithm for the single-row segmentation problem that is FPT in H, the largest value in the intensity matrix A. Since A has only one row, we represent it as a vector $A[1..n]$.

We say that a (row) segment S *begins* at index ℓ if it has its first non-zero entry at index ℓ, and *ends* at index r if it has its last non-zero entry at index r. We call a segmentation of A *compact* if any two segments in it begin at different indices and end at different indices.

Lemma 2.1. *For any segmentation $\mathcal{S}$ of a single row, there exists a compact segmentation $\mathcal{S}'$ with $|\mathcal{S}'| \leq |\mathcal{S}|$.*

Proof: Consider an optimal segmentation $\mathcal{S}$ that has (among all optimal segmentations) the *maximum* beam-on time; recall that this means that it maximizes $\sum_{S\in\mathcal{S}} v(S)$. We claim that $\mathcal{S}$ is compact. Otherwise, $\mathcal{S}$ has two segments with coinciding begin- or end-indices. Without loss of generality, assume that two segments S, S' of $\mathcal{S}$ begin at index ℓ. Say S and S' have non-zero values v and v' and end at indices r and r', respectively. If $r = r'$, then the two segments could be combined into one to give a smaller segmentation, a contradiction. So $r \neq r'$, say $r < r'$.

Now, define two new segments S'' and S''' as follows. Segment S'' begins at ℓ, ends at r, and has value $v + v'$. Segment S''' begins at $r + 1$, ends at r', and has value v'. Clearly $vS + v'S' = (v + v')S'' + v'S'''$, so $\mathcal{S}' = \mathcal{S} - \{S, S'\} \cup \{S'', S'''\}$ is also an optimal segmentation. But $\mathcal{S}'$ has larger beam-on time, a contradiction. $\square$

Our algorithm uses a dynamic programming approach that computes an optimal segmentation of any prefix $A[1..j]$ of A. We say that a segmentation of $A[1..j]$ is *almost-compact* if any two segments in it begin at different indices, and any two segments in it either end at different indices or both end at index j. We will only compute almost-compact segmentations; this is sufficient by Lemma 2.1. We compute the segmentation conditional on the values of the last segments in it.

Let $\mathcal{S}$ be a segmentation of vector $A[1..j]$; each $S \in \mathcal{S}$ is hence a vector $S[1..j]$. Define the *signature* of $\mathcal{S}$ to be the multi-set obtained by taking the value $v(S)$ of each segment ending in j. Note that the signature of a segmentation of $A[1..j]$ is a *partition* of $A[j]$, i.e., a multi-set of positive integers that sum to $A[j] \leq H$.

We briefly remind the reader of notation for multi-sets. Recall that a multi-set is a list of entries where entries may appear repeatedly. In all our applications, entries will be from the universe $[H] := \{1, \ldots, H\}$. The multi-set $\mathcal{M}$ can be described via the H-tuple $(m_1(\mathcal{M}), \ldots, m_H(\mathcal{M}))$, where $m_t(\mathcal{M})$ denotes the multiplicity of element t in $\mathcal{M}$. We use $\|\mathcal{M}\| := \sum_{t=1}^{H} m_t(\mathcal{M})$ to denote the cardinality of $\mathcal{M}$.

For two such multi-sets $\mathcal{M}_1$ and $\mathcal{M}_2$, let $\mathcal{M}_1 \cup \mathcal{M}_2$ be the multi-set defined by $m_t(\mathcal{M}_1 \cup \mathcal{M}_2) := m_t(\mathcal{M}_1) + m_t(\mathcal{M}_2)$ for $t \in [H]$, let $\mathcal{M}_1 \cap \mathcal{M}_2$ be the multi-set defined by $m_t(\mathcal{M}_1 \cap \mathcal{M}_2) := \min\{m_t(\mathcal{M}_1),\ m_t(\mathcal{M}_2)\}$ for $t \in [H]$, and let $\mathcal{M}_1 - \mathcal{M}_2$ be the multi-set defined by $m_t(\mathcal{M}_1 - \mathcal{M}_2) := \max\{0, m_t(\mathcal{M}_1) - m_t(\mathcal{M}_2)\}$ for $t \in [H]$. *Adding (resp. deleting)* element $t \in [H]$ from multi-set $\mathcal{M}$ means increasing (resp. decreasing) $m_t(\mathcal{M})$ by one (while keeping $m_t(\mathcal{M}) \geq 0$). Finally, we say that $\mathcal{M}_1$ *is contained* in $\mathcal{M}_2$ and write $\mathcal{M}_1 \subseteq \mathcal{M}_2$ whenever $m_t(\mathcal{M}_1) \leq m_t(\mathcal{M}_2)$ for $t \in [H]$.

The key idea of our algorithm is to compute the best almost-compact segmentation of $A[1..j]$ subject to a given signature. Thus define a function f as follows:

Given an integer j and a partition ϕ of $A[j]$, let $f(j, \phi)$ be the minimum number of segments in an almost-compact segmentation $\mathcal{S}$ of $A[1..j]$ that has signature ϕ.

We will show that $f(j, \phi)$ can be computed recursively. To simplify computation we will use $f(0, \cdot)$ as a base case; we assume that $A[0] = A[n+1] = 0$. The only possible partition of 0 is the empty partition, and so $f(0, \emptyset) = 0$ is our base case. In order to help the reader, we also aggregate some frequently used notation in Table 1.

Given a partition ϕ of $A[j]$, let $\Phi_{j-1}(\phi)$ be the set of those partitions of $A[j-1]$ that can be obtained from ϕ by deleting at most one element, and then adding at most one element. We have the following recursive formula for f.

Lemma 2.2. *For $j \geq 1$, $f(j, \phi) = \min_{\psi \in \Phi_{j-1}(\phi)} \{f(j-1, \psi) + \|\phi - \psi\|\}$*

Proof: We break the proof into two parts, proved separately below.

Claim 2.3. *For $j \geq 1$, $f(j, \phi) \geq \min_{\psi \in \Phi_{j-1}(\phi)} \{f(j-1, \psi) + \|\phi - \psi\|\}$*

Proof: Consider an almost-compact segmentation $\mathcal{S}_j$ of $A[1..j]$ that achieves the left-hand side, i.e., its signature is ϕ and $|\mathcal{S}_j| = f(j, \phi)$. We have four kinds of segments in $\mathcal{S}_j$: (1) Those that end at index $j-2$ or earlier, (2) those that end at $j-1$ (there can be at most one, since $\mathcal{S}_j$ is almost-compact), (3) those that end at j and begin at $j-1$ or earlier, and (4) those that end at j and begin at j (there can be at most one).

Let $\mathcal{S}_{j-1}$ be the segmentation of $A[1..j-1]$ obtained from $\mathcal{S}_j$ by taking all segments of type (1)–(3), and deleting the last entry (at index j). The value of each segment in $\mathcal{S}_{j-1}$ is defined as the value of the corresponding segment in $\mathcal{S}_j$. Note that $\mathcal{S}_{j-1}$ is also almost-compact. The signature ψ of $\mathcal{S}_{j-1}$ is the same as ϕ, except that the value of the (unique) segment of type (4)

Notation	Definition
A	A $m \times n$ intensity matrix. (For $m = 1$, $A[1..j]$ is a vector.)
H	Largest value in A.
$[H]$	The set $\{1, \ldots, H\}$.
$\mathcal{S}$	Set of segments that are a segmentation for A.
S	Segment. (For $m = 1$, $S[1..j]$ is a vector.)
$v(S)$	Value of segment S.
$\mathcal{M}$	Multi-set with entries from the universe $[H]$; it can be described via the H-tuple $(m_1(\mathcal{M}), \ldots, m_H(\mathcal{M}))$.
$m_t(\mathcal{M})$	Multiplicity of element t in a multi-set $\mathcal{M}$.
Signature of $\mathcal{S}$	Multi-set of all values of segments in $\mathcal{S}$ ending in index j.
$\mathcal{M}(\mathcal{S})$	Multi-set defined by values of segments in segmentation $\mathcal{S}$.
ϕ	Partition of a value in A; in our context, the partition of $A[j]$.
$\Phi_{j-1}(\phi)$	Set of partitions of $A[j-1]$ that can be obtained from ϕ by deleting at most one element and adding at most one element.
$f(j, \phi)$	Minimum number of segments in an almost-compact segmentation $\mathcal{S}$ of $A[1..j]$ that has signature ϕ.
$p(H)$	Number of partitions of the integer H.
ρ	The smallest number such that every row of A has at most ρ markers.
$c(A[i])$	Complexity of row i of A.
$f'(j, \phi, \nu)$	For partition ϕ of $A[j]$ and a multiset ν over $[H]$, this function equals 1 if there $f'(j, \phi, \nu)$ exists a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu$; 0 otherwise.
$f''(j, \phi, \nu)$	For partition ϕ of $A[j]$ and a multiset ν over $[H]$, this function equals the minimum possible number of 1-segments in a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu \cup (\infty, 0, \ldots, 0)$.
$\mathbb{M}$, $\mathbb{M}_{lex}$	Set of interesting multi-sets for the full-matrix and lex-min problem.

Table 1: Summary of frequently used notation.

(if any) has been removed, and the value of the (unique) segment of type (2) (if any) has been added. So $\psi \in \Phi_{j-1}(\phi)$.

If both a segment of type (4) and a segment of type (2) exist in $\mathcal{S}_j$, then they necessarily have different non-zero values (otherwise they could be combined, contradicting the minimality of $\mathcal{S}_j$). Hence $\|\phi - \psi\|$ is exactly the number of segments of type (4). So $|\mathcal{S}_{j-1}| = |\mathcal{S}_j| - \|\phi - \psi\|$, which proves the claim. □

Claim 2.4. *For $j \geq 1$, $f(j, \phi) \leq \min_{\psi \in \Phi_{j-1}(\phi)} \{f(j-1, \psi) + \|\phi - \psi\|\}$.*

Proof: Let $\psi \in \Phi_{j-1}(\phi)$ be a partition of $A[j-1]$ that achieves the minimum on the right-hand side. Let $\mathcal{S}_{j-1}$ be an almost-compact segmentation that achieves $f(j-1, \psi)$, i.e., it is a segmentation of $A[1..j-1]$ with signature ψ and cardinality $f(j-1, \psi)$.

Define a segmentation $\mathcal{S}_j$ of $A[1..j]$ as follows. Each segment of $\mathcal{S}_{j-1}$ that ends before index $j-1$ is extended by setting its jth entry to be 0 and added to $\mathcal{S}_j$. For each value t, if t exists in $\psi - \phi$, then there must be a t-segment in $\mathcal{S}_{j-1}$ that ends at index $j-1$; add this segment to $\mathcal{S}_j$ and let it end at $j-1$ (i.e., set its jth entry to 0). For each occurrence of value t in $\psi \cap \phi$ (there could be many), there must be a t-segment in $\mathcal{S}_{j-1}$ that ends at index $j-1$; add this segment to $\mathcal{S}_j$ and extend it to j (i.e., set its jth entry to 1). In all the preceding cases, the value of each segment in $\mathcal{S}_j$ is defined as the value of the corresponding segment in $\mathcal{S}_{j-1}$.

Finally, for each value t, if t exists in $\phi - \psi$, then define a new segment in $\mathcal{S}_j$ that begins at j and has value t. One easily verifies that $\mathcal{S}_j$ has signature ϕ, and therefore it is a segmentation of $A[1..j]$, since ϕ is a partition of $A[j]$. Since $\psi \in \Phi_{j-1}(\phi)$, at most one segment of $\mathcal{S}_j$ ends at $j-1$ or begins at j, so $\mathcal{S}_j$ is almost-compact. Also, $|\mathcal{S}_j| = |\mathcal{S}_{j-1}| + \|\phi - \psi\|$, which proves the result. □

The lemma follows from Claims 2.3 and 2.4. □

Theorem 2.5. *The single-row segmentation problem can be solved in $O(p(H)\,H\,n)$ time and $O(p(H)H)$ additional space, where $p(H)$ is the number of partitions of H.*

Proof: The idea is to compute $f(j, \phi)$ as in Lemma 2.2 recursively with a dynamic programming approach; the optimal value can then be found in $f(n+1, \emptyset)$. See Algorithm 1 for details.

To achieve the time complexity, we must look up partitions (and their associated stored values) quickly. The key property here is that any partition ϕ of $A[j] \leq H$ has $O(\sqrt{H})$ distinct integers in the set $[H] := \{1, \ldots, H\}$. Thus, we can describe a partition in $O(\sqrt{H})$ space, and store it (using a trie) so that it can be located in $O(\sqrt{H})$ time. We give the details of this data structure in Appendix A.

Therefore, line 9 of Algorithm 1 can be executed in $O(\sqrt{H})$ time. We execute this line $O(\sqrt{H})$ times from line 5 (since ϕ has $O(\sqrt{H})$ distinct integers), which proves that the time complexity is $O(n \cdot p(H) \cdot \sqrt{H} \cdot \sqrt{H})$ as desired. As for the space, there are $p(H)$ partitions that each can be described in $O(\sqrt{H})$ space. We need to keep with each partition ϕ only the values $f(j, \phi)$ and $f(j-1, \phi)$ (for the current j), so the space for the partitions is $O(p(H)\sqrt{H})$. However, the trie to store these partitions will require $O(H)$ space at each internal node to allow for quick access; since the number of internal nodes of the trie is $O(p(H))$, therefore we use $O(p(H)H)$ space for the trie. □

Algorithm 1

```
1: Initialize $f(0, \emptyset) = 0$
2: **for** $j = 1, \ldots, n+1$ **do**
3:    **for all** partitions $\phi$ of $A[j]$ **do**
4:       Initialize $f(j, \phi) = \infty$
5:       **for all** distinct integers $t$ in $\phi$ and also for $t = 0$ **do**
6:          let $t' = A[j-1] - (A[j] - t)$
7:          **if** $t' \geq 0$ **then**
8:             let $\psi = \phi - \{t\} \cup \{t'\}$  (or $\phi - \{t\}$ if $t' = 0$)
9:             look up $f(j-1, \psi)$
10:            set $f(j, \phi) = \min\{f(j, \phi), f(j-1, \psi) + \|\phi - \psi\|\}$
11:         **end if**
12:      **end for**
13:   **end for**
14: **end for**
15: Return $f(n+1, \emptyset)$
```

Note that the algorithm is fixed-parameter tractable with respect to parameter H. It is known [12] that

$$p(H) \leq \mathrm{e}^{\pi \cdot \sqrt{\frac{2 \cdot H}{3}}} \ ,$$

so this algorithm is in fact polynomial as long as $H \in O(\log^2 n)$. In the present form, it only returns the cardinality of the smallest segmentation, but standard dynamic programming techniques can be used to retrieve the segmentation in the same running time with $O(\log n)$ space overhead. We also show in the appendix that the space requirement can be improved by a factor of $\sqrt{H}$ at the expense of an additional $\log H$ factor in the running time.

3. Full-matrix segmentation

In this section, we give an algorithm that computes the optimal segmentation for a full matrix, and which is polynomial as long as H is a constant.

3.1. Bounds on the number of segments with the same value

It is necessary for our approach to the full-matrix problem to find an upper bound on the number of t-segments (i.e., segments of value t) in an optimal solution that can be assumed without loss of generality. We bound this relative to the number of 'markers' defined as follows.

Consider the ith row $A[i]$ of the intensity matrix A. Let $\Delta[i][j] = A[i][j] - A[i][j-1]$ for $i \in [m]$ and $j \in [n+1]$; assuming that $A[i][0] = A[i][n+1] = 0$. We say that there is a *marker* in row i between indices $j-1$ and j, if $\Delta[j] \neq 0$, i.e., the value in $A[i]$ changes there. We say that a row-segment *begins* at a marker if it has its first non-zero entry to the right of that marker, and it *ends* at a marker if it has its last non-zero entry to the left of that marker.

We call a segmentation of A *standardized* if the following two conditions hold for all $i \in [m]$: (1) The ith row of every segment begins and ends at a marker; (2) Whenever the ith row of some t-segment ends at a marker, then the ith row of no other t-segment begins at that marker.

Lemma 3.1. *Any segmentation $\mathcal{S}$ of A can be standardized without increasing its number of t-segments, for all $t \in [H]$.*

Proof: Assume that Condition (1) is violated in row i, such that k is an index with $A[i][k] = A[i][k+1] \neq 0$ but the ith row of some segments begins at $k+1$ or ends at k. Because $\mathcal{S}$ is a segmentation of A, the total value of its segments whose ith row ends at k equals the total value of its segments whose ith row begins at $k+1$. We redefine, respectively, the end-index and the begin-index of the ith row of these segments so that they are adjacent to the closest marker, say, to the left of k. In doing so, we retain a segmentation of A. Repeating the above argument for different values of i and k, we prove that (1) can be satisfied.

The process is similar and even easier if (2) is not satisfied. For as long as (2) is violated, say in row i at marker k for two t-segments, we combine the ith row of the two t-segments into one. Thus we eventually obtain the desired standardized segmentation. □

In particular, if $\mathcal{S}$ has minimum cardinality, then during standardization no segments can be deleted, and none are added, so its beam-on time remains unchanged. Vice versa, if $\mathcal{S}$ has minimum beam-on time, then during standardization no segments can be deleted (otherwise the beam-on time would decrease), and so its number of t-segments remains unchanged for all t.

Lemma 3.2. *If all rows of A have at most ρ markers, then there exists a minimum-cardinality segmentation that has at most $\rho/2$ segments of value t for all $t \in [H]$.*

Proof: As in the proof of Lemma 2.1, consider an optimal segmentation $\mathcal{S}$ with *maximum* beam-on time; as explained above we may assume that $\mathcal{S}$ is standardized since this does not change the beam-on time for an optimal segmentation. Assume for contradiction that there is some $t \in [H]$ for which $\mathcal{S}$ contains $r > \rho/2$ segments of value t. We will modify $\mathcal{S}$ into another segmentation $\mathcal{S}'$ that uses one fewer segment of value t, and one more segment of value $2t$. Hence $\mathcal{S}'$ has the same cardinality as $\mathcal{S}$, but larger beam-on time, contradicting the choice of $\mathcal{S}$.

To see why such an $\mathcal{S}'$ exists, consider any row of A and its segmentation by $\mathcal{S}$. If the segmentation uses at most $r-1$ segments of value t, then we use the same segmentation in $\mathcal{S}'$. If it has $r > \rho/2$ segments of value t, then there must be in this segmentation two t-segments that both begin at the same marker, or both end at the same marker, say the former. We can replace these two segments by a $2t$-segment, followed by one (possibly empty) t-segment once the first of the two segments ends. Hence again we obtain a segmentation of the row that can be used for $\mathcal{S}'$. Therefore, in every row we can change the segmentation by $\mathcal{S}$ into one that can be used for $\mathcal{S}'$, proving that $\mathcal{S}'$ exists, a contradiction. $\square$

Recall that the lex-min problem wants to find the segmentation with minimum beam-on time that has (among all such segmentations) the smallest cardinality. Lemma 3.2 does not apply for the lex-min problem, but we can obtain a similar bound on the number of t-segments in an optimal solution to the lex-min problem.

Lemma 3.3. *If all rows of A have at most ρ markers, then there exists a minimum-cardinality segmentation among all those that have minimum beam-on time that has at most ρ segments of value t for all $t \in [H]$. Moreover, for $t > H/2$, there are at most $\rho/2$ segments of value t.*

Proof: The proof is similar to the one of Lemma 3.2, except that here the beam-on time has to be kept minimum. In particular, we need to choose $\mathcal{S}'$ more carefully to ensure that it, too, has minimum beam-on time. So let $\mathcal{S}$ be a minimum-cardinality segmentation among all those that have minimum beam-on time. As usual, we assume that $\mathcal{S}$ has been standardized.

It is then near-trivial that $\mathcal{S}$ has at most $\rho/2$ segments of value t for $t > H/2$. Pick $i \in [m]$ and consider the segmentation of the ith row of A induced by $\mathcal{S}$. In this segmentation, no two segments of value $> H/2$ can overlap, since their combined value would then exceed H. It follows that no two of them can share a marker, since $\mathcal{S}$ is standardized. Because each touches two markers, there can be at most $\rho/2$ of them.

The other claim is more complicated. Assume for contradiction that there is some $t \in [H]$ for which $\mathcal{S}$ contains $r \geq \rho$ segments of value t. We will obtain a new segmentation $\mathcal{S}'$ that uses at most $r-2$ segments of value t, and one additional segment of value $2t$. For all other values, $\mathcal{S}$ and $\mathcal{S}'$ use equally many segments. Hence $\mathcal{S}'$ has the same beam-on time and a smaller cardinality as $\mathcal{S}$, contradicting the choice of $\mathcal{S}$.

To see why such an $\mathcal{S}'$ exists, consider any row i of A and the segmentation $\mathcal{S}_i$ induced by $\mathcal{S}$ in this row. If $\mathcal{S}_i$ uses at most $r-2$ segments of value t, then set $\mathcal{S}'_i = \mathcal{S}_i$, i.e., we use the same segmentation for this row in $\mathcal{S}'$.

If $\mathcal{S}_i$ uses $r-1$ segments of value t, where $r \geq \rho > 1+\rho/2$, then as in the proof of Lemma 3.2, two of them must begin or end at the same marker, and we remove them and replace them by a $(2t)$-segment and a t-segment. The resulting segmentation $\mathcal{S}_i'$ has at most $r-2$ segments of value t and one additional segment of value $2t$. While $\mathcal{S}_i'$ has a bigger beam-on time than $\mathcal{S}_i$, its beam-on time is no more than the beam-on time of $\mathcal{S}$ since $\mathcal{S}_i$ had fewer t-segments than $\mathcal{S}$.

Now finally assume that some $\mathcal{S}_i$ contains $r \geq \rho$ segments of value t. Define an auxiliary graph G as follows: G has a vertex for every marker, and an edge between two markers if and only if there exists a t-segment that begins at one of them and ends at the other. This graph G has ρ vertices and $r \geq \rho$ edges, and hence contains a cycle C. Let e_1 be a shortest edge of C, as measured by the distance between the markers at the ends, and let e_0 and e_2 be its two neighboring edges on C.

Presume $e_1 = j_1 j_2$, i.e., e_1 begins at marker j_1 and ends at j_2. Rename e_0 and e_2, if needed, so that e_0 and e_2 share marker j_1 and j_2, respectively, with e_1. Since $\mathcal{S}$ is standardized, we have $e_0 = j_1 j_3$ with $j_3 > j_1$ and $e_2 = j_0 j_2$ with $j_0 < j_2$. Since e_1 was a shortest edge of C, we have $j_0 \leq j_1 < j_2 \leq j_3$. See Figure 2.

If $j_0 = j_1$, i.e., e_1 and e_2 are two t-segments that begin and end at the same marker, then we replace them by one $(2t)$-segment with the same range. Similarly we proceed if $j_2 = j_3$. So we may assume $j_0 < j_1 < j_2 < j_3$, which in particular implies that $e_0 \neq e_1 \neq e_2 \neq e_0$.

We replace the three segments corresponding to edges e_0, e_1 and e_2 by a segment beginning at j_1 and ending at j_2 of value $2t$, and by another segment beginning at j_0 and ending at j_3 of value t. One easily verifies that this defines a segmentation $\mathcal{S}_i'$ of row i and uses at most $r-2$ t-segments and one additional $2t$-segment.

Combining these segmentations $\mathcal{S}_i'$ for the rows proves that $\mathcal{S}'$ exists, a contradiction. □

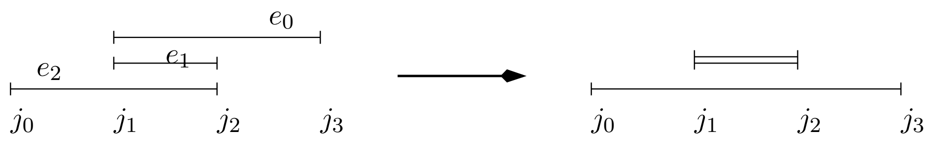


Figure 2: Combining segments to achieve the bound for the lex-min problem.

3.2. Segmenting a row under constraints

With these bounds in place, we now aim to develop an algorithm for the full-matrix problem.

The difficulty of full-matrix segmentation lies in that rows cannot be solved independently of each other, since an optimal segmentation of a full matrix does not mean that the induced segmentations of the rows are optimal. Consider for example

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

which is optimal, but the induced segmentation for the third row is not optimal. Hence we must solve a problem for each row that is restricted further. A second problem occurs because in the full-matrix problem we cannot assume that each row uses a compact segmentation. Consider for example

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

which is optimal, but the induced segmentation for the second row is not compact.

If $\mathcal{S}$ is a segmentation, then let $m_t(\mathcal{S})$ be the number of t-segments in $\mathcal{S}$; note that this defines a multi-set over $[H]$ which we refer to as the *multi-set $\mathcal{M}(\mathcal{S})$ defined by segmentation $\mathcal{S}$*. We now want to compute whether a row $A[1..n]$ has a segmentation $\mathcal{S}$ such that $\mathcal{M}(\mathcal{S}) \subseteq \nu$ for some given multi-set ν. We do this again with dynamic programming, by further restricting the segmentation to the first j elements and by restricting its signature. Thus define:

> *Given an integer j, a partition ϕ of $A[j]$, and a multiset ν over $[H]$, define $f'(j, \phi, \nu)$ to be 1 if there exists a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu$. Define $f'(j, \phi, \nu)$ to be 0 otherwise.*

For example, consider $A = [1\ 3\ 2\ 4]$, $\phi = \{1, 3\}$ and $\nu = \{1, 1, 1, 2, 3\}$. Then $f'(4, \phi, \nu)$ asks whether we can segment A such that at index 4 we use one 1-segment and one 3-segment, and overall we use at most three 1-segments, at most one 2-segment, and at most one 3-segment. The answer in this case is affirmative ($[1\ 3\ 2\ 4] = [1\ 1\ 0\ 0] + [0\ 0\ 0\ 1] + 2\,[0\ 1\ 1\ 0] + 3\,[0\ 0\ 0\ 1]$), so $f'(4, \phi, \nu) = 1$. Note that we were allowed one more 1-segment than was actually used; this is acceptable since the multi-set of the segmentation is allowed to be a subset of ν.

It can be shown that f' obeys a simple recursive formula, and therefore can be computed easily with dynamic programming. Using this, the full-matrix segmentation problem can then be solved in $O(mn^{H+1}/2^{(1-\epsilon)H})$ time, where $\epsilon > 0$ is an arbitrarily small constant, and $O(mn^H)$ space. We omit the details (they can be found in [5]), because we will improve the running time by defining a function f'' closely related to f', which has one fewer parameter and hence can be computed more efficiently.

To define this function f'', we need the notation of multiset ν_1 that has $m_1(\nu_1) = \infty$ and $m_t(\nu_1) = 0$ for all $t \neq 1$.

> *Given an integer j, a partition ϕ of $A[j]$, and a multiset ν over $[H]$, define $f''(j, \phi, \nu)$ to be the smallest number of 1-segments in a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu \cup \nu_1$. Define $f''(j, \phi, \nu)$ to be ∞ if no such segmentation exists.*

In other words, the segmentation that defines f'' is restricted in the number of t-segments only for $t > 1$, and the restriction on 1-segments is expressed in the return-value of f''. In particular, the value of $f''(j, \phi, \nu)$ is independent of the multiplicity of 1 in ν, and hence must be computed only for those ν with $m_1(\nu) = 0$.

Continuing the example from above, consider again $A = [1\ 3\ 2\ 4]$, $\phi = \{1, 3\}$ and $\nu = \{2, 3\}$. Then $f''(4, \phi, \nu)$ asks for the smallest number of 1-segments if we segment A such that at index 4 we use one 1-segment and one 3-segment, and overall we use at most one 2-segment, and at most one 3-segment. The segmentation given previously used two 1-segments, so $f''(4, \phi, \nu) \leq 2$ (and one can show this to be tight).

We now give a recursive formula for $f''(\cdot, \cdot, \cdot)$. The base case is again $j = 0$. Since $A[0] = 0$ (as before we assume $A[0] = A[n+1] = 0$), the only possible signature is $\emptyset$, and $f''(0, \emptyset, \nu) = 0$ for all possible multi-sets ν.

For $j \geq 1$, we can compute $f''(j, \phi, \nu)$ from $f''(j-1, \cdot, \cdot)$ as follows:

Lemma 3.4. *For all $j \geq 1$,*

$$f''(j, \phi, \nu) = \min_{\psi \text{ is a partition of } A[j-1]} \{f''(j-1, \psi, \nu - (\phi - \psi)) + m_1(\phi - \psi)\}.$$

Proof: The proof is similar to the proof of Lemma 2.2, except that we also need to keep track of how the multi-sets of segmentations change. Also, we use an arbitrary partition ψ (rather than restricting it to $\Phi_{j-1}(\phi)$) because we cannot restrict the attention to almost-compact segmentations.

Assume first that $f''(j, \phi, \nu) = k$, so we have a segmentation $\mathcal{S}_j$ of $A[1..j]$ with signature ϕ and $\mathcal{M}(\mathcal{S}_j) \subseteq \nu + \nu_1$, and $m_1(\mathcal{M}(\mathcal{S}_j)) = k$, i.e., $\mathcal{S}_j$ has k 1-segments. Let $\mathcal{S}_{j-1}$ be its implied segmentation of $A[1..j-1]$, and let it have signature ψ. Then $\phi - \psi$ contains exactly those values of segments of $\mathcal{S}_j$ that begin (and end) at j, and hence are not in $\mathcal{S}_{j-1}$. Therefore we can split $k = k_1 + k_2$, where $k_2 = m_1(\phi - \psi)$ while $k_1 = m_1(\mathcal{M}(\mathcal{S}_{j-1}))$.

For $t > 1$, segmentation $\mathcal{S}_j$ contains at most $m_t(\nu)$ segments of value t, therefore $\mathcal{S}_{j-1}$ contains at most $m_t(\nu) - m_t(\phi - \psi)$ segments of value t. Thus segmentation $\mathcal{S}_{j-1}$ has signature ψ and satisfies $\mathcal{M}(\mathcal{S}_{j-1}) \subseteq \nu - (\psi - \phi) \cup \nu_1$. So $f''(j-1, \psi, \nu - (\phi - \psi)) \leq m_1(\mathcal{M}(S_{j-1})) = k_1 = |\mathcal{S}_j| - k_2 = f''(j, \psi, \nu) - k_2$. Adding $k_2 = m_1(\phi - \psi)$ on both sides proves the '$\geq$'.

Now assume that the right-hand side is k, such that for partition ψ there exists a segmentation $\mathcal{S}_{j-1}$ that has signature ψ and satisfies $\mathcal{M}(\mathcal{S}_{j-1}) \subseteq \nu - (\phi - \psi) + \nu_1$ and $k = m_1(\mathcal{M}(s_{j-1})) + m_1(\phi - \psi)$. For each value t, we add $m_t(\phi - \psi)$ many new t-segments beginning and ending at j. For each value t, we extend $m_t(\phi \cap \psi)$ many t-segments to also cover index j. Any newly added 1-segment is counted in $m_1(\phi - \psi)$, so the resulting segmentation has k 1-segments, and $f''(j, \phi, \nu)$ will be no more than that as desired. $\square$

We illustrate this lemma with the above example of $A = [1\ 3\ 2\ 4]$, $\phi = \{1, 3\}$ and $\nu = \{2, 3\}$. Let $\psi = \{2\}$ and $\nu' = \{2\}$. Then $f''(3, \psi, \nu') = 1$ since $[1\ 3\ 2] = [1\ 1\ 0] + 2\,[0\ 1\ 1]$. Furthermore, we have $\phi - \psi = \{1, 3\}$ and $\nu - (\phi - \psi) = \{2\} = \nu'$. Therefore, the formula says that $f''(4, \phi, \nu)$ should be $1 + 1 = 2$, which indeed it is.

We now turn to the running time of actually computing f''. In the above definition, we have not imposed any bounds on ν, other than that it is a multi-set over $[H]$. But clearly we can restrict the multi-sets considered. Assume for a moment that we know an optimal segmentation $\mathcal{S}^*$ of the full matrix. We call a multi-set ν *relevant* if $\nu \subseteq \mathcal{M}(\mathcal{S}^*)$. Clearly it suffices to compute f'' for all relevant multi-sets. Moreover, as explained above, we only need to consider those multi-sets ν with $m_1(\nu) = 0$.

To find (a superset of) relevant multi-sets without knowing $\mathcal{S}^*$, we exploit that $\mathcal{M}(\mathcal{S}^*)$ cannot contain too many segments of the same value, as stated in Lemma 3.2. In particular, any multiplicity of a relevant multi-set is at most $\rho/2$, where ρ is the number of markers.

Now let $\mathbb{M}$ be the set of all those multi-sets over $[H]$ where the multiplicity of 1 is zero, and all other multiplicities are at most $\rho/2$. The multi-sets in $\mathbb{M}$ are called *interesting*, and it suffices to compute $f''(.,.,.)$ for them. We store the interesting multi-sets in an $(H-1)$-dimensional array with entries in $[0..\rho/2]$; this takes $O((\rho/2+1)^{H-1}) = O((\rho/2)^{H-1})$ space, and allows lookup of a multi-set in $O(H)$ time. We can then compute the values $f''(.,.,.)$ with Algorithm 2.

The running time of this algorithm is analyzed as follows. Computing ν' (given ν, ϕ and ψ) can certainly be done in $O(H)$ time. To look up $f''(j-1, \psi, \nu')$, we first look up ν' in the array in $O(H)$ time. With each multi-set $\nu \in \mathbb{M}$, we store all partitions of $A[j-1]$ and of $A[j]$ (for the current value of j), and with each of them, the values of $f''(j-1, \psi, \nu)$ and $f''(j, \psi, \nu)$, respectively. Looking up or changing these values (given ν and ψ) can then be done in $O(\sqrt{H})$ time by storing partitions in tries as explained in Appendix A. So lines 10–13 require $O(H)$ time. They are executed $p(H)$ times from line 9, $p(H)$ times from line 7, $|\mathbb{M}|$

Algorithm 2

```
Let $\mathbb{M}$ be all interesting multi-sets.
for all multi-sets $\nu$ in $\mathbb{M}$ do
    Initialize $f''(0, \emptyset, \nu) = 0$.
end for
for $j = 1, \ldots, n+1$ do
    for all multi-sets $\nu$ in $\mathbb{M}$ do
        for all partitions $\phi$ of $A[j]$ do
            Initialize $f''(j, \phi, \nu) = \infty$
            for all partitions $\psi$ of $A[j-1]$ do
                Compute $\nu' = \nu - (\phi - \psi)$
                Change the multiplicity of 1 in $\nu'$ to be zero.
                Look up $f''(j-1, \psi, \nu')$.
                Set $f''(j, \phi, \nu) = \min\{f''(j, \phi, \nu), f''(j-1, \psi, \nu') + m_1(\phi - \psi)\}$
            end for
        end for
    end for
end for
```

times from line 6, and $n+1$ times from line 5. Since $|\mathbb{M}| \leq (\rho/2)^{H-1}$, the running time is hence $O((\rho/2)^{H-1} p(H)^2 H\, n)$.

As for the space requirements, we need to store all relevant multi-sets, and with each, all partitions of $A[j-1]$ and $A[j]$, which takes $O(H)$ space per partition. So the total space is $O((\rho/2)^{H-1}) p(H) H)$.

Lemma 3.5. *Consider one row $A[1..n]$. We can compute, in $O((\rho/2)^{H-1} p(H)^2 H\, n)$ time and $O((\rho/2)^{H-1} p(H) H)$ space, an $(H-1)$-dimensional binary array $\mathcal{F}$ such that for any $m_1, m_2, \ldots, m_H \leq \rho/2$ we have $\mathcal{F}(m_2, \ldots, m_H) = m_1$ if and only if there exists a segmentation of $A[1..n]$ that uses at most m_t segments of value t for $t \in [H]$, and no such segmentation has fewer than m_1 1-segments.*

3.3. Full-matrix

To solve the full-matrix problem, compute for all rows i the table $\mathcal{F}_i$ described in Lemma 3.5. This takes time $O((\rho/2)^{H-1} p(H)^2 H\, mn)$ total. The space is $O((\rho/2)^{H-1} p(H) H)$ per row, but once done with a row i we only need to keep the $O((\rho/2)^{H-1})$ values for the corresponding table $\mathcal{F}_i$; therefore, in total, it is $O((\rho/2)^{H-1} \max\{m, p(H)H\})$.

Now, in $O((\rho/2)^{H-1} m)$ time find the numbers $m_1, \ldots, m_H$ for which $\mathcal{F}_i(m_2, \ldots, m_H) \leq m_1$ for *all* rows i and for which $m_1 + \cdots + m_H$ is minimized. Then by definition we can find a segmentation $\mathcal{S}_i$ for each row i that has at most m_t segments of value t for $t \in [H]$. We can combine these segmentations in the natural way (see also [6]) to obtain a segmentation $\mathcal{S}$ of A with at most m_t segments of value t for $t \in [H]$. This shows that an optimal segmentation has at most $m_1 + \cdots + m_H$ segments, and since we used the minimum possible such sum, no segmentation can be better than this bound.

Since the computation for this can be accomplished by scanning all $(\rho/2)^{H-1}$ multi-sets across m rows, we have the following result:

Theorem 3.6. *The full-matrix segmentation problem can be solved in $O((\rho/2)^{H-1}p(H)^2 H\, mn)$ time and $O((\rho/2)^{H-1}\max\{m, p(H)H\})$ space if each row has at most ρ markers.*

Note that one could view our result as FPT with respect to the parameter $H+\rho$. However, normally ρ will be large. In particular, if a natural pre-processing step is applied that removes from each row of A any consecutive identical numbers (this does not affect the cardinality of the optimum solution), then $\rho = n+1$. We therefore prefer to re-phrase our theorem to express the worst-case running time in terms of m, n and H only. Note that $\rho \leq n+1$ always, so the running time becomes $O(p(H)^2H/2^{H-1} \cdot mn^H)$. Recall that $p(H) \leq e^{\pi\sqrt{\frac{2H}{3}}} \leq e^{2.6\sqrt{H}}$ and, therefore, $Hp(H)^2 \leq He^{5.2\sqrt{H}} = 2^{\lg(H)+5.2\sqrt{H}\lg(e)} = O(2^{7.6\sqrt{H}})$, implying that $p(H)^2H/2^{H-1} \in O(2^{-(1-\epsilon)(H-1)})$ for arbitrarily small $\epsilon > 0$.

Corollary 3.7. *The full-matrix segmentation problem can be solved in $O(mn^H/2^{(1-\epsilon)(H-1)})$ time, where $\epsilon > 0$ is an arbitrarily small constant, and $O(mn^H)$ space.*

3.4. Solving the lex-min problem

Recall that the lex-min problem is that of finding a minimum-cardinality segmentation among those with minimum beam-on time, defined as the total value $\sum_{S\in\mathcal{S}} v(S)$ of the segmentation. Here, we show how to apply our techniques to achieve a speed up in solving this problem. To this end, we need the notion of the *complexity of row* $A[i]$ which is defined as:

$$c(A[i]) := \frac{1}{2}\sum_{j=1}^{n+1} |\Delta[i][j]| = \sum_{j=1}^{n+1} \max\{0, \Delta[i][j]\} = \sum_{j=1}^{n+1} -\min\{0, \Delta[i][j]\},$$

where as before $\Delta[i][j] := A[i][j] - A[i][j-1]$ for $j \in [n+1]$.

Importantly, it was shown in [13] that the minimum beam-on time can be computed efficiently; it is $c(A) := \max_i\{c(A[i])\}$. To solve the lex-min problem, we simply have to change our focus regarding the set $\mathbb{M}$ of interesting multi-sets. Now, each relevant multi-set $\nu \subseteq \mathcal{M}(\mathcal{S}^*)$ must satisfy $\sum_{t=1}^{H} t \cdot m_t(\nu) \leq c(A)$.[2] By Lemma 3.3, it suffices to include in the set $\mathbb{M}_{lex}$ of interesting multi-sets for the lex-min problem all multi-sets ν such that $m_t(\nu) \leq \rho - 1$ for $t \leq H/2$, $m_t(\nu) \leq \rho/2$ for $t > H/2$, and $\sum_{t=1}^{H} t \cdot m_t(\nu) \leq c(A)$. Furthermore, we may assume $m_1(\nu) = 0$, since the number of 1-segments is expressed in the return-value of f''. Hence, we only need to consider $O(\rho^{H-1}/2^{H/2})$ such multi-sets.

We will compute $f''(n+1, \emptyset, \nu)$ for all such multi-sets ν and all rows, and then pick a multi-set ν for which the maximum of $f''(n+1, \emptyset, \nu)$ over all rows plus $\sum_{t=2}^{H} m_t(\nu)$ is minimized, and for which this maximum plus $\sum_{t=2}^{H} t \cdot m_t(\nu)$ equals $c(A)$. This is then the multi-set used for a minimum segmentation among those with minimum beam-on time; we can find the actual segmentation by re-tracing the computation of $f''(n+1, \emptyset, \nu)$.

By the same analysis used for the minimum-cardinality segmentation problem, we have:

Theorem 3.8. *The lex-min problem can be solved in $O(mn^H/2^{(1/2-\epsilon)H})$ time and $O(mn^{H-1})$ space.*

Recall that Kalinowski's algorithm in [14] has a time complexity of $O(2^H\sqrt{H} \cdot m \cdot n^{2H+2})$. So we obtain a near-quadratic improvement in the time complexity.

[2]Any other variant where the restriction on the space of feasible solutions can be captured by appropriately modifying the set of interesting multi-sets could be solved similarly.

4. The special case of $H = 2$

For $H = 2$ (i.e., a 0/1/2-matrix), the algorithm of Section 3.3 has running time $O(mn^2)$. As we show in this section, however, yet another factor of n can be shaved off by analyzing the structure of the rows more carefully. In brief, the function f'' of Section 3.2 can be computed from the structure of the row alone, without needing to go through all possible signatures; we explain this now. Throughout Section 4, we assume that all entries in the intensity matrix are 0, 1, or 2.

4.1. Single row for $H = 2$

As before, let $A[1..n]$ be a single row of the matrix. Consider a maximal interval $[j', j'']$ such that $A[j'..j'']$ has all its entries equal to 2. Thus, $A[j] = 2$ for $j \in [j', j'']$ and $A[j'-1], A[j''+1] \in \{0, 1\}$. We call $A[j'..j'']$ a *tower* if $A[j'-1]$ and $A[j''+1]$ both equal 0, a *step* if one of $A[j'-1]$ and $A[j''+1]$ equals 1 and the other 0, and a *double-step* otherwise. (As usual we assume that $A[0] = A[n+1] = 0$.) We use t, s and u to denote the number of towers, steps and double-steps, respectively. Figure 3 illustrates how interpreting $A[i] = t$ as t blocks atop each other gives rise to these descriptive names.

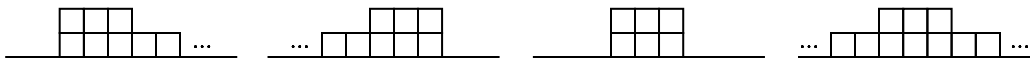


Figure 3: Two kinds of steps, a tower, and a double-step.

Recall that $c(A[i]) = \sum_{j=1}^{n+1} \max\{\Delta[i][j], 0\}$ is the complexity of a row i of a full matrix A; we use $c(A)$ for the complexity of the single row A under consideration.

Lemma 4.1. *Define $g(d)$ as follows:*

$$g(d) := \begin{cases} c(A) - 2d & \text{if } d < t, \\ c(A) - t - d & \text{if } t \le d \le s + t, \\ c(A) - 2t - s & \text{if } t + s < d. \end{cases}$$

Then for any $d \ge 0$, $f''(n+1, \emptyset, (0, d)) = g(d)$. In other words, any segmentation $\mathcal{S}$ of A with at most d segments of value 2 has at least $g(d)$ segments of value 1. Moreover, there exists a segmentation that has at most d segments of value 2 and exactly $g(d)$ segments of value 1.

Proof: Let $\mathcal{S}$ be a segmentation of A that uses at most d segments of value 2. As before, we assume that $\mathcal{S}$ has been standardized without increasing the number of 2-segments. Therefore, any tower, step or double-step of A is either entirely covered by a 2-segment, or it does not intersect any 2-segment.

Let s_2, t_2 and u_2 be the number of steps, towers, and double-steps that are entirely covered by a 2-segment. We claim that the number of 1-segments of $\mathcal{S}$ is $c(A) - s_2 - 2t_2$, and can prove this by induction on $s_2 + t_2 + u_2$. If $s_2 + t_2 + u_2 = 0$, then $\mathcal{S}$ has only 1-segments, and since $\mathcal{S}$ is standardized, the number of 1-segments equals $c(A)$. If, say, $t_2 > 0$, then let A' be the vector obtained from A by removing a tower that is covered by a 2-segment (i.e., by replacing the 2s of that tower by 0s), and let $\mathcal{S}'$ be the segmentation of A' obtained from $\mathcal{S}$ by removing the 2-segment that covers that tower. Then A' has $t_2' = t_2 - 1$ towers covered by 2-segments, and furthermore $c(A') = c(A) - 2$. Since $\mathcal{S}$ and $\mathcal{S}'$ have the same number of 1-segments, the claim easily follows by induction. Similarly one proves the claim by induction if $s_2 > 0$ or $u_2 > 0$.

Therefore the number of 1-segments in $\mathcal{S}$ is $c(A) - s_2 - 2t_2$. We also know that $s_2 + t_2 + u_2 \leq d$. So to get a lower bound on the number of 1-segments, we should minimize $c(A) - s_2 - 2t_2$, subject to $s_2 + t_2 + u_2 \leq d$ and the obvious $0 \leq s_2 \leq s$, $0 \leq t_2 \leq t$ and $0 \leq u_2 \leq u$. The bound now easily follows by distinguishing whether $d < t$ (the minimum is at $t_2 = d$, $s_2 = u_2 = 0$), or $t \leq d < t + s$ (minimum at $t_2 = t$, $s_2 = d - t$, $u_2 = 0$) or $t + s < d$ (minimum at $t_2 = t$, $s_2 = s$, $u_2 = 0$).

For the second claim, we obtain such a segmentation by using $\min\{d, t\}$ segments of value 2 for towers, then $\min\{d - t, s\}$ segments of value 2 for steps if $d \geq t$, and cover everything else by 1-segments. □

The crucial idea for $H = 2$ is that since $g(\cdot)$ can be described explicitly with only three linear equations that can easily be computed, we can save space and time by not storing $f''(n+1, \emptyset, (0, d))$ explicitly as an array of length $\rho/2 + 1$, and not spending $O(n\,\rho/2)$ time to fill it.

4.2. Full matrix segmentation for $H = 2$

As in Section 3.3, to solve the full-matrix problem we need to find the value d^* that minimizes $D(d) := d + \max_i\{g_i(d)\}$, where $g_i(d)$ is function $g(d) = f''(n+1, \emptyset, (0, d))$ for row i.

To do this, compute the complexity and the number of towers and steps in each row; this takes $O(mn)$ time in total. Each $g_i(d)$ is then the maximum of three lines defined by these values. Hence $D(d) = d + \max_i\{g_i(d))\}$ is the maximum of $3m$ lines. After this preprocessing, we can hence compute $D(d)$ for any value of d in $O(m)$ time. The optimum d is in the range $0 \leq d \leq \rho \leq n + 1$, so by computing all these values $D(d)$ we can find d^* in $O(mn)$ time.

After finding d^* we can easily compute a segmentation of each row that has at most $D - d^*$ segments of value 1 and at most d^* segments of value 2 (see the proof of Lemma 4.1) and combine them into a segmentation of the full matrix with the greedy-algorithm; this can all be done in $O(mn)$ time. Thus the overall running time is $O(mn)$.

Theorem 4.2. *A minimum-cardinality segmentation of an intensity matrix with values in $\{0, 1, 2\}$ can be found in $O(mn)$ time.*

An immediate application of this result is that it can be combined with the $O(\log H)$-approximation algorithm in [6]. While the approximation guarantee remains unchanged, this should result in improved solutions in practice without substantially increasing the running time.

The lex-min problem can also be solved easily for $H = 2$. If $\mathcal{S}$ is a segmentation of a row with d 2-segments, and it is minimal among all those, then its beam-on time is $2d + g(d)$. Let $c(A)$ be the complexity of the full-matrix. Then in searching for the optimal value d^*, only consider such values d for which $2d + g_i(d) \leq c(A)$. The segmentation obtained for this d^* then has beam-on time at most $c(A)$, which is the minimum beam-on time, and so it solves the lex-min problem.

Theorem 4.3. *The lex-min problem can be solved in $O(mn)$ time for an intensity matrix with values in $\{0, 1, 2\}$.*

One naturally asks whether this approach could be extended to higher values of H. This would be feasible if we could find (say for $H = 3$) a simpler expression for the function $f''(n + 1, \emptyset, (0, d_2, d_3))$, i.e., the minimum number of segments of value 1 given that at most d_2 segments of value 2 and at most d_3 segments of value 3 are used. It seems likely that this function would be piecewise linear (just as $g(d)$ was), but it is not clear how many pieces there are, and whether we can compute them easily from the structure of the row. Thus a faster algorithm for $H = 3$ (or higher) remains to be found.

5. Conclusion

In this work, we developed several algorithms that provide significant running time improvements for the minimum cardinality problem and the lex-min problem. At this point, a few interesting problems remain open. In particular, does the full-matrix problem admit an FPT algorithm in H, or is this problem $W[1]$-hard? If the latter, what can be said about the problem if there are only a few (but more than one) rows?

Acknowledgments

This work was supported by the "Actions de Recherche Concertées" (ARC) fund of the "Communauté française de Belgique", and the National Sciences and Engineering Research Council of Canada (NSERC). C.E. acknowledges support from the "Fonds pour la Recherche dans l'Industrie et l'Agriculture" (F.R.I.A.). Research done while the S.D. and M.Y. were at the University of Waterloo. A preliminary version of this paper appeared at WADS'11 [5]. The authors would like to thank the unknown referees for helpful suggestions.

References

[1] Davaatseren Baatar, Natashia Boland, Sebastian Brand, and Peter J. Stuckey. Minimum cardinality matrix decomposition into consecutive-ones matrices: CP and IP approaches. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pages 1–15, 2007.

[2] Davaatseren Baatar and Horst W. Hamacher. New LP model for multileaf collimators in radiation therapy. Contribution to the Conference ORP3, Universität Kaiserslautern, 2003.

[3] Davaatseren Baatar, Horst W. Hamacher, Matthias Ehrgott, and Gerhard J. Woeginger. Decomposition of integer matrices and multileaf collimator sequencing. *Discrete Applied Mathematics*, 152(1–3):6–34, 2005.

[4] Nikhil Bansal, Don Coppersmith, and Baruch Schieber. Minimizing setup and beam-on times in radiation therapy. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, Springer Lecture Notes in Computer Science, 4110:27–38, 2006.

[5] Therese Biedl, Stephane Durocher, Celine Engelbeen, Samuel Fiorini, and Maxwell Young. Faster optimal algorithms for segment minimization with small maximal value. In *Proceedings of the 13th Algorithm and Data Structures Symposium (WADS)*, Springer Lecture Notes in Computer Science, 6844:86–97, 2011.

[6] Therese Biedl, Stephane Durocher, Holger H. Hoos, Shuang Luan, Jared Saia, and Maxwell Young. A note on improving the performance of approximation algorithms for radiation therapy. *Information Processing Letters*, 111(7):326–333, 2011.

[7] Sebastian Brand. The sum-of-increments constraint in the consecutive-ones matrix decomposition problem. In *Proceedings of the 24th ACM Symposium on Applied Computing (SAC)*, pages 1417–1418, 2009.

[8] Hadrien Cambazard, Eoin O'Mahony, and Barry O'Sullivan. A shortest path-based approach to the multileaf collimator sequencing problem. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, Springer Lecture Notes in Computer Science, 5547:41–55, 2009.

[9] Danny Z. Chen, Xiaobo Sharon Hu, Shuang Luan, Shahid A. Naqvi, Chao Wang, and Cedric X. Yu. Generalized geometric approaches for leaf sequencing problems in radiation therapy. In *Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC)*, Springer Lecture Notes in Computer Science 3341:271–281, 2004.

[10] Michael J. Collins, David Kempe, Jared Saia, and Maxwell Young. Non-negative integral subset representations of integer sets. *Information Processing Letters*, 101(3):129–133, 2007.

[11] Cristian Cotrutz and Lei Xing. Segment-based dose optimization using a genetic algorithm. *Physics in Medicine and Biology*, 48(18):2987–2998, 2003.

[12] Wladimir de Azevedo Pribitkin. Simple upper bounds for partition functions. *The Ramanujan Journal*, 18(1):113–119, 2009.

[13] Konrad Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Applied Mathematics*, 152(1–3):35–51, 2005.

[14] Thomas Kalinowski. The complexity of minimizing the number of shape matrices subject to minimal beam-on time in multileaf collimator field decomposition with bounded fluence. *Discrete Applied Mathematics*, 157(9):2089–2104, 2009.

[15] Shuang Luan, Jared Saia, and Maxwell Young. Approximation algorithms for minimizing segments in radiation therapy. *Information Processing Letters*, 101(6):239–244, 2007.

[16] Robert Sedgewick. *Algorithms in Java, Parts 1–4 (Fundamental Algorithms, Data Structures, Sorting, Searching)*. Addison-Wesley, 2002.

[17] Giulia M. G. H. Wake, Natashia Boland, and Les S. Jennings. Mixed integer programming approaches to exact minimization of total treatment time in cancer radiotherapy using multileaf collimators. *Computers and Operations Research*, 36(3):795–810, 2009.

[18] Ping Xia and Lynn J. Verhey. Multileaf collimator leaf sequencing algorithm for intensity modulated beams with multiple static segments. *Medical Physics*, 25(8):1424–1434, 1998.

Appendix A. Tries to store partitions

Recall that a partition ϕ of a value $\leq H$ is a multi-set over the universe $[H] = \{1, \dots, H\}$. Let $t_1 > \cdots > t_\ell$ be those values that occur at least once in ϕ. We can then describe ϕ as a string

$$\sigma(\phi) = (t_1, m_{t_1}(\phi), \dots, t_\ell, m_{t_\ell}(\phi)),$$

where $m_{t_k}(\phi) > 0$ is the multiplicity of value t_k in ϕ, for $k = 1, \dots, \ell$. For example, we have

$$\phi = \{4, 2, 1, 1, 1\} \Longleftrightarrow \sigma(\phi) = (4, 1, 2, 1, 1, 3).$$

A key observation is that $\sigma(\phi)$ has length $O(\sqrt{H})$. For recall that ϕ is a partition of a value $\leq H$, and hence $\sum_{k=1}^{\ell} m_{t_k}(\phi) t_k \leq H$. If we had $\ell > \sqrt{2H}$ then

$$H \geq \sum_{k=1}^{\ell} m_{t_k}(\phi) t_k \geq \sum_{k=1}^{\ell} t_k \geq \sum_{k=1}^{\ell} (\ell + 1 - k) = \sum_{k=1}^{\ell} k = \frac{\ell(\ell+1)}{2} \geq \frac{\sqrt{2H}(\sqrt{2H}+1)}{2} > H,$$

a contradiction. So $|\sigma(\phi)| = 2\ell \leq 2\sqrt{2H} = O(\sqrt{H})$.

Thus, to store and access information about ϕ, we will store and access information about string $\sigma(\phi)$, which is a string with $O(\sqrt{H})$ entries in the alphabet $\Sigma = \{1, \dots, H\}$. We store such strings using a *trie*, i.e., a tree where arcs to the children of a node are labeled with distinct symbols from Σ. See for example [16] for more details about tries.

The node on level k of the trie refers to entry k of the strings $\sigma(\phi)$, i.e., it either distinguishes by the next value t_k for which $m_{t_k}(\phi)$ is non-zero, or (one level farther down) by what $m_{t_k}(\phi)$ is. To find the appropriate child, each node stores an array $C[1 \dots H]$ where $C[t]$ refers to the child where the value is t.

To find the entry for a partition ϕ (which has been stored as list $\sigma(\phi)$), we trace from the root downwards in the trie, using the kth entry in $\sigma(\phi)$ to find the appropriate child of the node on the kth level. The time required to do so is $O(\|\sigma(\phi)\|) = O(\sqrt{H})$.

The space requirement for this trie is $O(H)$ per node. If we use a compressed trie (i.e., we only split at a node if it actually has descendants in multiple children), then the number of nodes in the trie is proportional to its number of leaves, which is $p(H)$. Hence the trie needs $O(p(H)\,H)$ space.

Appendix A.1. Decreasing space by increasing time

Instead of using an array to store the children of a node, we can use a binary search table or a hash-table with constant load factor. Then the space at each node is proportional to its number of children, and hence the total space used at internal nodes is $O(p(H))$. But we still need $O(p(H)\sqrt{H})$ space to store the description $\sigma(\phi)$ for all partitions ϕ, so the total space is $O(p(H)\sqrt{H})$. This saving in space comes at an increased running time: With binary search trees, the lookup time is now $O(\log H)$ at each node, and with hash-tables, it is $O(1)$ expected time. For all but really large values of H, this rather small decrease in space does not seem to warrant the more complicated data structure and potential time-increase.

Appendix A.2. Creating partitions

We can also use this kind of trie to create all partitions for all values $\leq H$ efficiently. Let ϕ be a partition of $L \leq H$. Let t_1 be the largest value of ϕ, and let ψ be the partition obtained from ϕ by deleting one copy of t_1. Then ψ is a partition of $L - t_1$. Thus, every partition ϕ of L can be obtained by taking a partition ψ of a value $L' < L$ such that $L - L'$ is no smaller than the largest value in ψ. It is easy to see that this is a one-to-one correspondence.

To compute the set $\Phi(L)$ of all partitions of L, we assume that we have computed $\Phi(1)$, $\dots$, $\Phi(L-1)$ already and stored them in their appropriate tries. For this step, it is vital that a partition ϕ is stored using the largest integer in it first (i.e., that $t_1 > \cdots > t_\ell$ in the above definition of $\sigma(\phi)$).

Create a new trie with root node r. For each $L' < L$, we obtain the partitions of L with largest value at most $L - L'$ by scanning the trie that stores $\Phi(L')$. More precisely, ignore all partitions in $\Phi(L')$ that are located at children $C[L - L' + 1, \dots, L']$ of the root; these have largest value bigger than $L - L'$. Then scan through each remaining partition of L', add one value $L - L'$ to it to obtain a partition of L, and add it into the trie that stores $\Phi(L)$. This takes $O(\sqrt{H})$ time per partition that is inserted, and hence $O(p(L)\sqrt{H})$ time overall. Doing this for $L = 1, \dots, H$ finds all partitions of H in time $O(\sqrt{H}(p(1) + \cdots + p(H))) = O(\sqrt{H}p(H))$.