Faster Optimal Algorithms For Segment Minimization With Small Maximal Value*

Therese Biedl[1], Stephane Durocher[2], Céline Engelbeen[3], Samuel Fiorini[4], and Maxwell Young[1]

[1] David R. Cheriton School of Computer Science, University of Waterloo, ON, Canada
{biedl,m22young}@uwaterloo.ca
[2] Department of Computer Science, University of Manitoba, MB, Canada
durocher@cs.umanitoba.ca
[3] Département de Mathématique, Université Libre de Bruxelles, Brussels, Belgium
{celine.engelbeen,sfiorini}@ulb.ac.be

Abstract. The segment minimization problem consists of finding the smallest set of integer matrices (*segments*) that sum to a given intensity matrix, such that each summand has only one non-zero value (the *segment-value*), and the non-zeroes in each row are consecutive. This 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. Here, we obtain an almost-quadratic speedup.

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 $m \times n$ pixels, where m is typically the number of pairs of leaves of the MLC. This determines

* 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.).

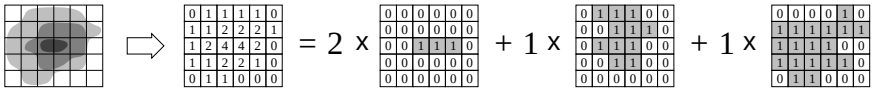

Fig. 1. An example of a segmentation of an intensity matrix where $H = 4$.

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 pixel.

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 have reached 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 a $m \times n$ 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 $v_{\mathcal{S}}(S)$ or simply $v(S)$ when $\mathcal{S}$ is understood. 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 the 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.[4] 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.

Related Work: The segmentation problem is known to be NP-complete in the strong sense, even for a single row [9, 2, 3], 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

[4] 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.

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.* [16] 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, 18, 11, 14] as well as approaches for obtaining optimal (exact) solutions [7, 1, 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.

Kalinowski [15] 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 a $O(2^H \sqrt{H}\, m\, n^{2H+2})$ 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 that $T \in O(H^2\, n)$.

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 algorithms 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 $O(m\, n^H / 2^{(1-\epsilon)(H)})$ time for an arbitrarily small constant $\epsilon > 0$. In contrast, applying the variant of Kalinowski's algorithm mentioned above yields a worst-case run-time of $\Omega(m\, n^{2H+3})$. Therefore, our result improves the run-time by more than $\Omega(n^3)$.
– 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 $O(m\, n^H / 2^{(\frac{1}{2}-\varepsilon)H})$. In comparison to the previous best result by Kalinowski [15], our algorithm improves the run-time by more than $\Omega(n^2)$.

Therefore, our algorithms represent a significant asymptotic speed-up and the techniques required to achieve these improvements is non-trivial. Due to space restrictions, we omit some proofs and details; these can be found in [5].

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 $A[1..n]$. Let $\Delta[j] := A[j] - A[j-1]$ for $j \in [n+1]$ (for the purpose of such definitions, we will assume that $A[0] = A[n+1] = 0$.) We say that there is a *marker* between index $j-1$ and j if $\Delta[j] \neq 0$, i.e., if the value in A changes.

Any segmentation of a row can be *standardized* as follows: (1) Every segment S *begins* (i.e., has its first non-zero entry) and *ends* (i.e., has its last non-zero entry) adjacent to a marker. For if it doesn't, then some other segment(s) must end where S begins (or vice versa), and by moving all these endpoints to the nearest marker, we retain a segmentation without adding new segments. (2) Whenever a segment ends at a marker, then no other segment of the same value begins at that marker. For otherwise the two segments could be combined into one. Note that standardization of a segmentation can only decrease the number of t-segments for all t; hence it can only improve the cardinality of the segmentation and its beam-on time.

For the single-row problem, we can improve segments even further. Call a segmentation of $A[1..n]$ *compact* if any two segments in it begin at different indices end end at different indices. Similarly as above one can show:

Lemma 1. *For any segmentation $\mathcal{S}$ of a single row, there exists a compact segmentation $\mathcal{S}'$ with $|\mathcal{S}'| \leq |\mathcal{S}|$.*

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 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 use operations such as $\cup, \cap$, set-difference, subset, adding/deleting elements generalized to multi-sets in the obvious way.

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.

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. The following recursive formula for f can be shown:

Lemma 2. *For $j \geq 1$, $f(j,\phi) = \min_{\psi \in \Phi_{j-1}(\phi)} \{f(j-1,\psi) + ||\phi - \psi||\}$*

Theorem 1. *The single-row segmentation problem can be solved in $O(p(H)\,H\cdot n)$ time and $O(n+p(H)H)$ space, where $p(H)$ is the number of partitions of H.*

Proof. The idea is to compute $f(j,\phi)$ with Lemma 2 recursively with a dynamic programming approach; the optimal value can then be found in $f(n+1,\emptyset)$. To achieve the time complexity, we need to store the partitions in a suitable data structure. 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. We store partitions using a trie where each node uses $O(H)$ space but allows access to the correct child in constant time; a partition can then be located in $O(\sqrt{H})$ time.

So to compute $f(n+1,\emptyset)$, go through $j=1,\ldots,n$ and through all partitions ϕ of $A[j]$. For each distinct integer $t\in\phi$, compute the partition $\psi\in\Phi_{j-1}(\phi)$ obtained by deleting t and then adding one element so that ψ is a partition of $A[j-1]$. Look up ψ (and the value of $f(j-1,\psi)$ stored with it) in the trie, add $||\phi-\psi||$ to it, and update $f(j,\phi)$ if the result is smaller than what we had before. Analyzing these loops, we see that the runnning time is $O(n\cdot p(H)\cdot\sqrt{H}\cdot\sqrt{H})$ as desired. □

Note that the algorithm is fixed-parameter tractable with respect to parameter H. It is known that $p(H)\leq \mathrm{e}^{\pi\cdot\sqrt{\frac{2\cdot H}{3}}}$ [12], so this algorithm is in fact polynomial as long as $H\in O(\log^2 n)$. In the present form, it only returns the size of the smallest segmentation, but standard dynamic programming techniques can be used to retrieve the segmentation in the same run-time with an $O(\log n)$ space overhead.

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 Segmenting a row under constraints

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.

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 yes ($[1\ 3\ 2\ 4] = [1\ 1\ 0\ 0] + [0\ 2\ 2\ 0] + [0\ 0\ 0\ 1] + [0\ 0\ 0\ 3]$), 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.

We claim that $f'(\cdot,\cdot,\cdot)$ has a simple recursive formula. The base case is again $j = 0$ and $f'(0,\emptyset,\nu) = 1$ 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 (details are in the full paper):

Lemma 3. *For all $j \geq 1$,*

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

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

We now turn to the run-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.

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. Recall that a marker is a place where the values within a row change; let ρ_i be the number of markers in row i, and $\rho = \max_i \rho_i$. One can show the following:

Lemma 4. *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 $t \in [H]$.*

Now let $\mathbb{M}$ be all those multi-sets over $[H]$ where all multiplicities are at most $\rho/2$; this contains all relevant multi-sets. We store these in an H-dimensional array with indices in $[0..\rho/2]$; this takes $O((\rho/2)^H)$ space, and allows lookup of a multi-set in $O(H)$ time. We can then compute the values $f'(j,\phi,\nu)$ with Algorithm 1.

The run-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.

So lines 9-11 require $O(H)$ time. They are executed $p(H)$ times from line 8, $p(H)$ times from line 6, $|\mathbb{M}|$ times from line 5, and $n+1$ times from line 4; the run-time is hence $O(n(\rho/2+1)^H p(H)^2 H)$.

Algorithm 1

```
Let 𝕄 be all multi-sets where all multiplicities are at most ρ/2.
for all multi-sets ν in 𝕄 do
    Initialize f'(0, ∅, ν) = 1.
    for j = 1, ..., n + 1 do
        for all multi-sets ν in 𝕄 do
            for all partitions φ of A[j] do
                Initialize f'(j, φ, ν) = 0
                for all partitions ψ of A[j − 1] do
                    Compute ν' = ν − (φ − ψ)
                    if f'(j − 1, ψ, ν') = 1 then
                        Set f'(j, φ, ν) = 1 and break
                    end if
                end for
            end for
        end for
    end for
end for
```

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 $O(p(H)\,H(\rho/2)^H)$.

Lemma 5. *Consider one row $A[1..n]$. In $O(n(\rho/2)^H p(H)^2 H)$ time and $O(p(H)\,H(\rho/2)^H)$ space we can compute an H-dimensional binary array $\mathcal{F}$ such that for any $m_1, \ldots, m_H \leq \rho/2$ we have $\mathcal{F}(m_1, \ldots, m_H) = 1$ if and only if there exists a segmentation of $A[1..n]$ that uses at most m_t segments of value t for $t \in [H]$.*

3.2 Full-matrix

To solve the full-matrix problem, compute for all rows i the table $\mathcal{F}_i$ described in Lemma 5. This takes time $O(mn(\rho/2)^H p(H)^2 H)$ total. The space is $O(p(H)\,H(\rho/2)^H)$ per row, but once done with a row i we only need to keep the $O((\rho/2)^H)$ values for the corresponding table $\mathcal{F}_i$; therefore, in total, it is $O(\max\{m, p(H)H\}(\rho/2)^H)$.

Now, in $O(m(\rho/2)^H)$ time find the numbers $m_1, \ldots, m_H$ for which $\mathcal{F}_i(m_1, \ldots, m_H)$ is 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 $(\rho/2)^H$ multi-sets across m rows, we have the following result:

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

Note that one could view our result as FPT in parameter is $H + \rho$. 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 size of the optimum solution), then $\rho = n + 1$. We therefore prefer to re-phrase our theorem to express the worst-case run-time in terms of m, n and H only. Note that $\rho \leq n + 1$ always, so the run-time becomes $O(mn^{H+1}p(H)^2 H/2^H)$. Recall that $p(H) \leq e^{\pi\sqrt{\frac{2H}{3}}} \leq e^{2.6\sqrt{H}}$ and, therefore, $Hp(H)^2 \leq He^{5.2\sqrt{H}} = 2^{\lg(H)+5.2\sqrt{H}\lg(e)} \leq 2^{8.6\sqrt{H}}$, implying that $p(H)^2 H/2^H \in O(2^{-(1-\epsilon)H})$ for arbitrarily small $\epsilon > 0$ if H is sufficiently large.

Corollary 1. *The full-matrix segmentation problem can 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.*

3.3 Further improvements of the complexity

We sketch a further improvement that removes a factor of n from the running time. Recall that the function $f'(j, \phi, \nu)$ was defined to be 1 if and only if there exists a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu$. In its place, we can instead define a function $f''(j, \phi, \nu)$, which contains the minimum number of 1-segments in a segmentation $\mathcal{S}$ of $A[1..j]$ with signature ϕ and multi-set $\mathcal{M}(\mathcal{S}) \subseteq \nu + \nu_1$. Here, ν_1 is the multi-set that has $m_1(\nu_1) = \infty$ and $m_t(\mathcal{M}_1) = 0$ for all $t \neq 1$. 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 first multiplicity of ν, and hence must be computed only for those ν with $m_1(\nu) = 0$; there are only $(\rho/2 + 1)^{H-1}$ such multi-sets ν.

It remains to argue that f'' can be computed efficiently, with a similar formula as for f'. This is quite simple. To compute $f''(j, \phi, \nu)$, try all possible partitions ψ of $A[j-1]$, compute $\nu' = \nu - (\phi - \psi)$, and let ν'' be ν' with its first multiplicity changed to 0. Look up the value $f''(j-1, \psi, \nu'')$ and add to it the number of 1s in $\phi - \psi$. This gives one possible candidate for a segmentation; we find the best one by minimizing over all ψ. We leave the formal proof of correctness to the reader.

We can hence compute $f''(n+1, \emptyset, \nu)$ for all $(\rho/2)^{H-1}$ multi-sets ν in $O(n(\rho/2)^{H-1} p(H)^2 H)$ time. Doing this for all rows, we can compute the maximum of the values $f''(n+1, \emptyset, \nu)$ over all rows. The optimum segmentation can then be found by choosing the one that minimizes this maximum plus $||\nu||$ over all ν. As before, this only adds an extra $O(m)$ factor to the run-time, which is hence $O(mn(\rho/2)^{H-1}p(H)^2 H)$, and similarly as before this can be simplified to $O(mn^H/2^{(1-\varepsilon)H})$.

Theorem 3. *The full-matrix segmentation problem can be solved in $O(mn^H/2^{(1-\epsilon)H})$ time, for $\epsilon > 0$ an arbitrarily small constant, and $O(mn^{H-1})$ 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, is was shown in [14] 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. Instead of the relevant multi-sets as used earlier, we need all multi-sets ν such that $\sum_{t=1}^{H} t \cdot m_t(\nu)$ equals the minimum beam-on time. Let $\mathbb{M}_{lex}$ be the set of these multi-sets and their subsets. While Lemma 4 no longer applies, we still obtain a useful bound on the size $\mathbb{M}_{lex}$, whose proof is in the full paper.

Lemma 6. *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 $\rho - 1$ t-segments for all $t \in [H]$. Moreover, for $t > H/2$, there are at most $\rho/2$ t-segments.*

We can hence find and store a (super-set of) $\mathbb{M}_{lex}$ by using all entries in an H-dimensional array $[0, \rho]^{\lfloor H/2 \rfloor} \times [0, \rho/2]^{\lceil H/2 \rceil}$, and there are $O(\rho^H/2^{H/2})$ such multi-sets. We will compute $f''(n+1, \emptyset, \nu)$ for all such multi-sets ν, and then pick a multi-set ν for which $||\nu|| + \sum_{t=1}^{H} m_t(\nu)$ is minimized, and for which $\sum_{t=1}^{H} t m_t(\nu)$ 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, and the improvement described in the previous Section 3.3, we have:

Theorem 4. *The lex-min problem can be solved in $O(mn^H/2^{(\frac{1}{2}-\varepsilon)})$ time and with $O(mn^{H-1})$ space.*

Recall that Kalinowski's algorithm in [15] has a time complexity of $O(2^H\sqrt{H} \cdot m \cdot n^{2H+2})$. So we obtain a significant improvement in the time complexity. Finally, we note that it is intuitively reasonable that our algorithm can be applied to the lex-min problem since the restriction on the space of feasible solutions that the beam-on time be minimized can be captured by modifying appropriately the set of interesting multi-sets $\mathbb{M}_{lex}$.

4 The special case of $H = 2$

For $H = 2$ (i.e., a 0/1/2-matrix), the algorithm of Section 3.3 has run-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 a nutshell, the function f'' of Section 3.3 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. We call $A[j'..j'']$ a *tower* if $A[j'-1]$ and $A[j''+1]$ both equal 0, a *simple 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, simple steps and double steps, respectively. Figure 2 illustrates how interpreting $A[i] = t$ as t blocks atop each other gives rise to these descriptive names.

Fig. 2. Two kinds of simple 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 7. *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 at most $g(d)$ 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, which can be done 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 the 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 $s_2 + t_2 + u_2 \le d$. So to get a lower bound on the number of 1-segments, we should minimize $c(A) - s_2 - 2t_2$, subject to $s_2 + t_2 + u_2 \le d$ and the obvious $0 \le s_2 \le s$, $0 \le t_2 \le t$ and $0 \le u_2 \le u$. The bound now easily follows by distinguishing whether $d < t$ (the minimum is at $t_2 = d$, $s_2 = u_2 = 0$), or $t \le d < t + s$ (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 $\min\{d, t\}$ 2-segments for towers, then $\min\{d - t, s\}$ 2-segments for stairs 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 $O(n \cdot \rho/2)$ 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 $d + \max_i\{f_i''(n + 1, \emptyset, \{0, d\})\} =: D$, where $f_i''(\cdot)$ is function $f''(\cdot) = g(\cdot)$ for row i. We can hence find the optimal segmentation of A as follows. Compute the complexity and the number of towers and stairs in each row; this takes $O(mn)$ time total. Each $f_i''(\cdot)$ is then the maximum of three lines defined by these numbers. Hence $d + \max_i\{f_i''(n + 1, \emptyset, \{0, d\})\}$ is the maximum of $3m$ lines. We hence can compute D (and with it d^*) by taking the intersection of the upper half-spaces defined by the $3m$ lines (this can be done in $O(m)$ expected time easily, and in $O(m)$ worst-case time with a complicated algorithm [13]), and then finding the grid point with the smallest y-coordinate in it.

Once we found 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 7) 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 run-time is $O(mn)$.

Theorem 5. *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 approximation guarantee remains unchanged, this should result in improved solutions in practice while not substantially increasing the running time.

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 1-segments given that at most d_2 2-segments and d_3-3-segments are used. It seems likely that this function would be piecewise linear (just like $g(d)$ 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 drastic running time improvements for the minimum cardinality problem. At this point, a couple interesting problems remain open. Does the full-matrix problem admit a FPT result if $m > 1$ but m is small (i.e., a small number of rows)? Is the full-matrix problem $W[1]$-hard in H?

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 APPROX-RANDOM*, pages 27–38, 2006.
5. Therese Biedl, Stephane Durocher, Celine Engelbeen, Samuel Fiorini, and Maxwell Young. Faster optimal algorithms for segments minimization with small maximal value. Technical Report CS-2011-08, University of Waterloo, 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 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)*, pages 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)*, pages 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. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry (3rd edition)*. Springer-Verlag, 2008.
14. Konrad Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Applied Mathematics*, 152(1–3):35–51, 2005.
15. 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.
16. Shuang Luan, Jared Saia, and Maxwell Young. Approximation algorithms for minimizing segments in radiation therapy. *Information Processing Letters*, 101(6):239–244, 2007.
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.