Multileaf collimator shape matrix decomposition

Thomas Kalinowski

March 7, 2007

1 Introduction

An important method in cancer treatment is the use of high energetic radiation. In order to kill tumor cells the patient is exposed to radiation that is delivered by a linear accelerator whose beam head can be rotated about the treatment couch. Inevitably the healthy tissue surrounding the tumor is also exposed to some radiation. So the problem arises to arrange the treatment in a way such that the tumor receives a sufficiently high uniform dose while the damage to the normal tissue is as small as possible. The standard approach to this problem is as follows. First the patient body is discretized into so called *voxels*. The set of voxels is then partitioned into three sets: the clinical target volume, the critical structures and the remaining tissue. There are certain dose constraints for each of these parts. Basically the dose in the target volume has to be sufficient to kill the cancerous cells and the dose in the critical structures must not destroy the functionality of the corresponding organs. The determination of a combination of radiation fields is usually done by inverse methods based on certain physical models of how the radiation passes through a body. In the early 1990's the method of intensity modulated radiation therapy (IMRT) was developed in order to obtain additional flexibility. Using a multileaf collimator (MLC) it is possible to form homogeneous fields of different shapes. By superimposing of some homogeneous fields an intensity modulated field is delivered. An MLC consists of two banks of metal leaves which block the radiation and can be shifted to form irregularly shaped beams (Fig. 1).
The most common approach in treatment planning is to divide the optimization into two phases. At first, a set of beam angles and corresponding fluence matrices are determined. In a second step a sequence of leaf positions for the MLC for each of the angles is determined that yields the desired fluence distribution. Very recently there have been attempts to combine both steps into one optimization routine [22, 9].

Figure 1: The leaf pairs of a multileaf collimator (MLC)

In this chapter we concentrate on the second step, the shape matrix decomposition problem. Suppose we have fixed the beam angles from which the radiation is released, and for each of the beam angles we are given a fluence distribution that we want the patient to be exposed to. After discretizing the beam into *bixels* we can assume that the fluence distribution is given as a nonnegative integer matrix A. Each row of the matrix corresponds to a pair of leaves of the MLC.
There are two methods in IMRT using MLCs which differ essentially in their technical realization, but the mathematical methods used to determine optimal treatment plans are quite similar. In the step–and–shoot mode the radiation is switched off whenever the leaves are moving, so the intensity modulation is the result of superimposing a finite number of homogeneous fields. In the dynamic mode the radiation is switched on during the whole treatment and the modulation is achieved by moving the leaves with varying speed. Clearly, in this setup the fluence at a particular point is proportional to the amount of time in which the point is exposed to radiation, i.e. not blocked by one of the leaves. Here we consider only the step–and–shoot mode. Essentially, the most common approach to the dynamic mode can be seen as an imitation of this case (see [15] and the references therein).

2 The mathematical model

The principle of the MLC in step–and–shoot mode is illustrated in Fig. 2. Our aim is to determine a sequence of leaf positions and corresponding irradiation times such that the given fluence distribution is realized. Suppose the given matrix has size $m \times n$, i.e. we consider m leaf pairs, and for each leaf there are $n+1$ possible positions. Then the leaf positions can be described by certain $0-1-$matrices of size $m \times n$ called shape matrices, where a $0-$entry means the radiation is blocked and a $1-$entry means that the radiation goes through.

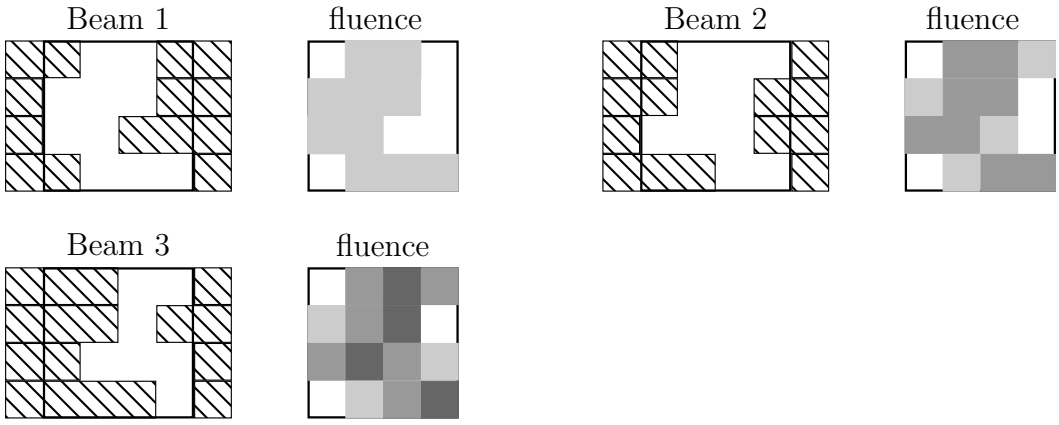

Figure 2: Intensity modulation by superimposing 3 beams of different shapes. In each step the left figure shows a leaf position and in the right figure the grey scale indicates the total fluence.

For example the first leaf position in Fig. 2 corresponds to the shape matrix

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

Clearly, the superposition of differently shaped beams corresponds to positive linear combinations of shape matrices, where the coefficient of a shape matrix measures how long the corresponding field is applied. So any representation of the given fluence matrix A as a positive integer linear combination of shape matrices is a feasible solution to our decomposition problem. For instance:

$$A = \begin{pmatrix} 1 & 3 & 3 & 0 \\ 0 & 2 & 4 & 1 \\ 1 & 1 & 4 & 4 \\ 3 & 3 & 1 & 0 \end{pmatrix} = 2 \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \quad (1)$$

We denote the set of shape matrices by $\mathcal{S}$, and consider decompositions of the form $A = \sum_{S \in \mathcal{S}} u_S S$ with $u_S \in \mathbb{N}$ for all $S \in \mathcal{S}$. There are two quantities influencing the quality of a decomposition: the total irradiation time (proportional to the sum of the coefficients) and the number of necessary beams (the number of nonzero coefficients). Let $\mathcal{S}_0$ denote the set of matrices with nonzero coefficient. We can now formulate two different optimization problems, the decomposition time (DT) problem and the decomposition cardinality (DC) problem

$$\textbf{(DT)} \qquad \min \left\{ \sum_{S \in \mathcal{S}} u_S \;\mid\; A = \sum_{S \in \mathcal{S}} u_S S,\ u_S \in \mathbb{N} \right\}, \quad (2)$$

$$\textbf{(DC)} \qquad \min \left\{ |\mathcal{S}_0| \;\mid\; \mathcal{S}_0 \subseteq \mathcal{S},\ A = \sum_{S \in \mathcal{S}_0} u_S S,\ u_S \in \mathbb{N} \right\}. \quad (3)$$

Of course, one could also minimize some weighted sum of decomposition time and decomposition cardinality, i.e. an objective function

$$\sum_{S \in \mathcal{S}_0} u_S + \alpha |\mathcal{S}_0|,$$

where α is some positive constant. This objective function can be considered as total treatment time, where the parameter α depends on the used MLC and measures the average setup time, i.e. the time needed to move the leaves and check the setting. In a still more sophisticated model one can include the possibility that the setup time between two different leaf positions depends on the amount of required leaf motion. Consequently, the order in which the beams are delivered becomes relevant and the corresponding objective function is

$$\sum_{S \in \mathcal{S}_0} u_S + \sum_{i=1}^{|\mathcal{S}_0|-1} \mu(S^{(i)}, S^{(i+1)}),$$

where $S^{(1)}, S^{(2)}, \ldots, S^{(|\mathcal{S}_0|)}$ is an ordering of the set of used shape matrices $\mathcal{S}_0$, and for two shape matrices S and S', $\mu(S, S')$ is proportional to the time necessary to change the setup of the MLC from the beam corresponding to S to the beam corresponding to S'.

The optimal value of (2) can be computed very efficiently while the problem (3) is computationally very hard (see Section 4). So the most common approach is to first compute the minimal DT, and then heuristically search for a decomposition which realizes this DT and also has a small DC.
Our model is still quite flexible: certain properties of the used MLC can be included in the definition of the shape matrices. From the design of the MLC it is clear that any shape matrix must have the *consecutive ones property*: in every row is a (possibly empty) interval of consecutive 1−entries and the remaining entries are 0. In addition, in some of the commercially available MLCs leaf overtravel is forbidden. That means the left leaf of row i and the right leaf of row $i \pm 1$ must not overlap. In this case a shape matrix cannot contain two consecutive rows as follows:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Also some MLCs have a minimum leaf distance. That means if a row is not completely covered by either the right or the left leaf, a minimum distance δ between the two leaves in this row is present. In other words, the number of ones in a row is either 0 or at least δ. Another feature of most of the MLCs is the tongue–and–groove design. To prevent radiation from going through the gap between two adjacent leaves a design similar to the one indicated in Fig. 3 is used. The small overlap between the regions that are covered

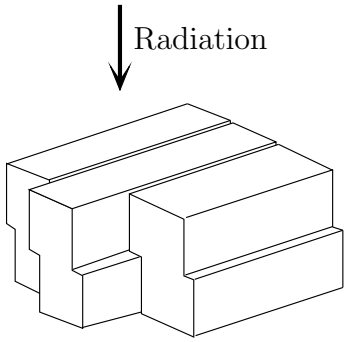

Figure 3: The tongue-and-groove design of the leaves of an MLC.

by adjacent leaves cause underdosage effects as illustrated in Fig. 4. In order to prevent such underdosage effects one has to require that $a_{i,j} \leq a_{i\pm1,j}$ implies that in each of the used beams bixel $(i \pm 1, j)$ is open whenever bixel (i, j) is open, or in terms of the shape matrices:

$$\begin{aligned} a_{i,j} \leq a_{i-1,j} \wedge s_{i,j} = 1 \quad &\Longrightarrow \quad s_{i-1,j} = 1, \\ a_{i,j} \geq a_{i-1,j} \wedge s_{i-1,j} = 1 \quad &\Longrightarrow \quad s_{i,j} = 1 \end{aligned}$$

for $i = 2, \ldots, m$ and $j = 1, \ldots, n$.

3 The decomposition time problem

Starting with [6] and [10] a number of different algorithms for the shape matrix decomposition problem have been proposed [8, 14, 16, 23, 24], some of them providing the optimal *DT* while others use heuristic methods for both objectives *DT* and *DC*. In this section

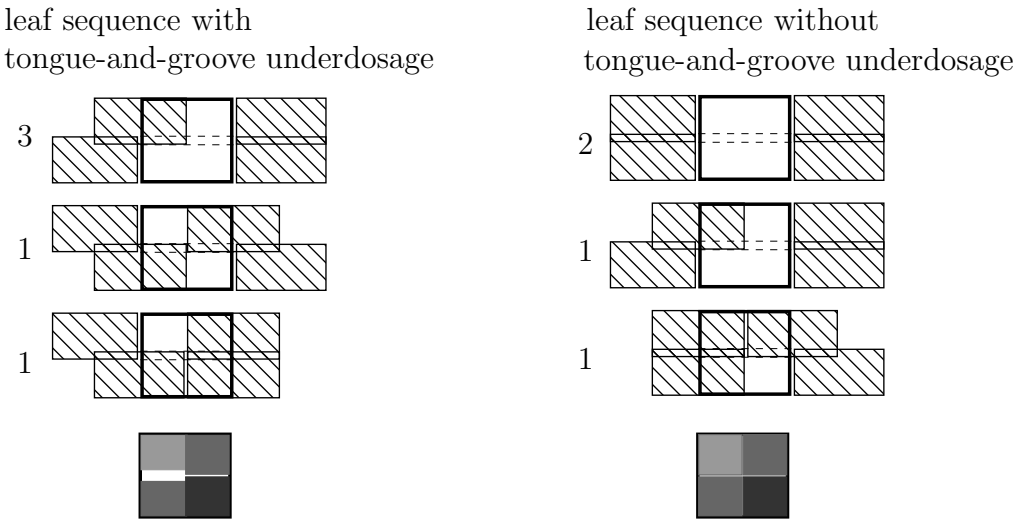

Figure 4: Two different realizations of the same fluence matrix. The numbers next to the leaf positions indicate the irradiation times for the corresponding beams. In the left version the overlap between bixels $(1,1)$ and $(2,1)$ receives no radiation at all.

we concentrate on the DT-problem, and thus without loss of generality we put all the nonzero coefficients u_s to 1, but allow that the same shape matrix S occurs several times in the decomposition $A = \sum_{t=1}^{k} S^{(k)}$. First we consider the version without additional constraints, i.e. the leaves in different rows move independently, and we neglect the tongue–and–groove underdosage. Then we can solve the decomposition problem for each row independently, and the optimal DT for the whole matrix is just the maximum of the optimal DTs of the single rows. All the algorithms that yield the exact optimum are essentially based on (disguised versions of) the following characterization of the minimal DT. For simplicity of notation we add a 0-th and a $(n+1)-$th column to the matrix A by setting

$$a_{i,0} = a_{i,n+1} = 0 \qquad \text{for } i = 1, 2, \ldots, m.$$

We define the $i-$th row complexity of A to be

$$c_i(A) = \sum_{j=1}^{n} \max\{0, a_{i,j} - a_{i,j-1}\},$$

and the complexity of A, $c(A) = \max\limits_{1 \le i \le m} c_i(A)$.

Theorem 1 ([8])**.** *The minimal DT for a matrix A equals $c(A)$.*

Proof. Let $\boldsymbol{b} = \begin{pmatrix} b_1 & b_2 & \ldots & b_n \end{pmatrix}$ denote the $i-$th row of the matrix A, i.e. $b_j = a_{i,j}$. First, we show that any representation of $\boldsymbol{b}$ as a sum of vectors with the consecutive ones property contains at least $c_i(A)$ terms. Suppose the vectors $\boldsymbol{s^{(t)}} \in \{0,1\}^n$ $(t = 1, 2, \ldots, k)$ define such a representation

$$\boldsymbol{b} = \boldsymbol{s^{(1)}} + \cdots + \boldsymbol{s^{(k)}},$$

where each vector $\boldsymbol{s^{(t)}}$ has the consecutive ones property. For $t = 1, 2, \ldots, k$ let $\boldsymbol{b^{(t)}} = \boldsymbol{s^{(1)}} + \cdots + \boldsymbol{s^{(t)}}$ be the sum of the first t terms and put

$$c^{(t)} = \sum_{j=1}^{n} \max\{0, b_j^{(t)} - b_{j-1}^{(t)}\}.$$

Now let l_t and r_t denote the positions of the leaves corresponding to $\boldsymbol{s^{(t)}}$, i.e.

$$s_j^{(t)} = \begin{cases} 1 & \text{if } l_t < j < r_t, \\ 0 & \text{otherwise.} \end{cases}$$

For $t > 1$ we obtain

$$\max\{0, b_j^{(t-1)} - b_{j-1}^{(t-1)}\} = \max\{0, b_j^{(t)} - b_{j-1}^{(t)}\} \qquad \text{for } j \notin \{l_t + 1, r_t\} \tag{4}$$

$$\max\{0, b_{l_t+1}^{(t-1)} - b_{l_t}^{(t-1)}\} = \begin{cases} \max\{0, b_{l_t+1}^{(t)} - b_{l_t}^{(t)}\} - 1 & \text{if } b_{l_t+1}^{(t)} > b_{l_t}^{(t)} \\ \max\{0, b_{l_t+1}^{(t)} - b_{l_t}^{(t)}\} & \text{otherwise} \end{cases} \tag{5}$$

$$\max\{0, b_{r_t}^{(t-1)} - b_{r_t-1}^{(t-1)}\} = \begin{cases} \max\{0, b_{r_t}^{(t)} - b_{r_t-1}^{(t)}\} + 1 & \text{if } b_{r_t}^{(t)} \geq b_{r_t-1}^{(t)} \\ \max\{0, b_{r_t}^{(t)} - b_{r_t-r}^{(t)}\} & \text{otherwise} \end{cases} \tag{6}$$

Consequently, $c^{(t-1)} \geq c^{(t)} - 1$ with equality if and only if $b_{l_t+1}^{(t)} > b_{l_t}^{(t)}$ and $b_{r_t}^{(t)} < b_{r_t-1}^{(t)}$. Summing up these inequalities for $2 \leq t \leq k$ and using $c^{(1)} = 1$ and $c^{(k)} = c_i(A)$, we obtain

$$1 + c^{(2)} + \cdots + c^{(k-1)} \geq c^{(2)} + c^{(3)} + \cdots + c^{(k-1)} + c_i(A) - (k-1),$$

or $k \geq c_i(A)$. To show that there is a decomposition of the $i-$th row with $c_i(A)$ terms we use induction on $k := c_i(A)$. If $k = 0$, then $\boldsymbol{b} = \boldsymbol{0}$ and we are done. If $k > 0$, we put $b_0 = b_{n+1} = 0$ and define

$$\begin{aligned} l &= \min\{j \mid 0 \leq j \leq n-1,\ b_j < b_{j+1}\}, \\ r &= \min\{j > l \mid l < r \leq n+1,\ b_j < b_{j-1}\}, \\ s_j^{(k)} &= \begin{cases} 1 & \text{if } l < j < r \\ 0 & \text{otherwise.} \end{cases} \qquad (j = 1, \ldots, n) \end{aligned}$$

Then by (4)–(6), for $\boldsymbol{b'} := \boldsymbol{b} - \boldsymbol{s^{(k)}}$, we have

$$\sum_{j=1}^{n} \max\{0, b'_j - b'_{j-1}\} = \sum_{j=1}^{n} \max\{0, b_j - b_{j-1}\} - 1 = k - 1.$$

By induction, there is a decomposition $\boldsymbol{b'} = \boldsymbol{s^{(1)}} + \cdots + \boldsymbol{s^{(k-1)}}$. Together with $\boldsymbol{s^{(k)}}$ this yields the required decomposition of $\boldsymbol{b}$, and this concludes the proof. □

From the proof we can immediately derive an algorithm for the construction of a DT–optimal decomposition (see Algorithm 1). Of course the choice of the l_i and r_i is not unique. In [6] this particular one is called sweep technique, because the leaves always move from left to right. As an example consider the following decomposition of a matrix from [24].

$$\begin{pmatrix} 4&5&0&1&4&5 \\ 2&4&1&3&1&4 \\ 2&3&2&1&2&4 \\ 5&3&3&2&5&3 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 1&0&0&0&0&0 \end{pmatrix} + \begin{pmatrix} 1&1&0&0&0&0 \\ 1&1&0&0&0&0 \\ 1&1&1&0&0&0 \\ 1&0&0&0&0&0 \end{pmatrix} + \begin{pmatrix} 1&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&1&1&1&1 \\ 1&1&1&0&0&0 \end{pmatrix} + \begin{pmatrix} 1&1&0&0&0&0 \\ 0&1&1&1&0&0 \\ 0&0&0&0&1&1 \\ 1&1&1&1&1&0 \end{pmatrix} + \begin{pmatrix} 0&1&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&1 \\ 1&1&1&1&1&0 \end{pmatrix}$$

$$+ \begin{pmatrix} 0&0&0&1&1&1 \\ 0&0&0&1&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$$

Algorithm 1 (DT-optimal leaf sequence in the unconstrained case)**.**
$k := 0$
while $A \neq 0$ **do**
  $k := k + 1$
  for $i = 1, 2, \ldots, m$ **do**
    if $a_{i,j} = 0$ for all $j = 1, 2, \ldots, n$ **then** $l_i := n$, $r_i := n + 1$
    else
      $l_i := \min\{j \mid 0 \leq j \leq n,\ a_{i,j} < a_{i,j+1}\}$
      $r_i := \min\{j \mid l_i < j \leq n + 1,\ a_{i,j-1} > a_{i,j}\}$
    for $j = 1, \ldots, n$ **do**
      if $l_i < j < r_i$ **then** $s_{i,j}^{(k)} := 1$ **else** $s_{i,j}^{(k)} := 0$
  $A := A - S^{(k)}$
return $S^{(1)}, \ldots, S^{(k)}$

3.1 The interleaf collision constraint

The interleaf collision constraint (ICC) is present in many of the commercially available MLCs and forbids an overlap of the left leaf in row i and the right leaf in row $i \pm 1$. If l_i and r_i denote the leaf positions in row i $(i = 1, \ldots, m)$ this amounts to

$$\textbf{(ICC)} \qquad l_i < r_{i-1} \quad \text{and} \quad r_i > l_{i-1} \qquad (i = 2, \ldots, m)$$

3.1.1 A Linear Programming approach

An important conclusion from the following algorithm is, that we can always construct a DT-optimal decomposition with unidirectional leaf movement. That means the leaves move only from left to right, or in other words, if $l_i^{(t)}$ and $r_i^{(t)}$ denote the leaf positions in row i corresponding to the $t-$th shape matrix, then $l_i^{(t)} \leq l_i^{(t+1)}$ and $r_i^{(t)} \leq r_i^{(t+1)}$ for all i and t. Such a decomposition $A = \sum_{t=1}^{k} S^{(t)}$ is completely determined once we know for each i and j, how often the leaves in row i are at position j, i.e. we have to know the numbers

$$\gamma_{i,j}^L = \left|\left\{t \mid l_i^{(t)} = j - 1\right\}\right|, \qquad \gamma_{i,j}^R = \left|\left\{t \mid r_i^{(t)} = j\right\}\right|. \tag{7}$$

The numbers $\gamma_{i,j}^L$ and $\gamma_{i,j}^R$ can be translated back into the shape matrices via

$$s_{i,j}^{(t)} = 1 \iff \sum_{j'=1}^{j} \gamma_{i,j'}^R < t \leq \sum_{j'=1}^{j} \gamma_{i,j'}^L \qquad (t = 1, \ldots, k). \tag{8}$$

This definition makes sense for any nonnegative $\gamma_{i,j}^L$ and $\gamma_{i,j}^R$. Now we formulate additional requirements for these values.

Lemma 1. *The matrices defined by (8) sum up to A if and only if*

$$\gamma_{i,j}^L - \gamma_{i,j}^R = a_{i,j} - a_{i,j-1} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n + 1). \tag{9}$$

Proof. "⇒": By hypothesis and (8) we have

$$a_{i,j} = \sum_{j'=1}^{j} \gamma_{i,j}^L - \sum_{j'=1}^{j} \gamma_{i,j}^R.$$

For $j = 1$ we obtain

$$a_{i,1} - a_{i,0} = a_{i,1} = \gamma^L_{i,1} - \gamma^R_{i,1}.$$

And for $j > 1$,

$$a_{i,j} - a_{i,j-1} = \left(\sum_{j'=1}^{j} \gamma^L_{i,j} - \sum_{j'=1}^{j} \gamma^R_{i,j}\right) - \left(\sum_{j'=1}^{j-1} \gamma^L_{i,j} - \sum_{j'=1}^{j-1} \gamma^R_{i,j}\right) = \gamma^L_{i,j} - \gamma^R_{i,j}.$$

"$\Leftarrow$": Assume that (9) is true and let $B = (b_{i,j})$ be the sum of the matrices defined by (8). Then

$$b_{i,j} = \sum_{j'=1}^{j} \gamma^L_{i,j} - \sum_{j'=1}^{j} \gamma^R_{i,j} = \sum_{j'=1}^{j} (a_{i,j'} - a_{i,j'-1}) = a_{i,j} - a_{i,0} = a_{i,j}.$$

□

The next lemma formulates the ICC in terms of the $\gamma^L_{i,j}$ and $\gamma^R_{i,j}$.

Lemma 2. *If $\gamma^L_{i,j}$ and $\gamma^R_{i,j}$ encode a decomposition (not necessarily unidirectional) $A = \sum_{t=1}^{k} S^{(t)}$ with ICC as in (7) then*

$$\sum_{j'=1}^{j} \gamma^L_{i-1,j'} \geq \sum_{j'=1}^{j} \gamma^R_{i,j'} \qquad (i = 2, \ldots, m;\ j = 1, \ldots, n+1), \tag{10}$$

$$\sum_{j'=1}^{j} \gamma^L_{i,j'} \geq \sum_{j'=1}^{j} \gamma^R_{i-1,j'} \qquad (i = 2, \ldots, m;\ j = 1, \ldots, n+1). \tag{11}$$

Proof. We have

$$\sum_{j'=1}^{j} \gamma^L_{i-1,j'} = \left|\left\{t \ :\ l^{(t)}_{i-1} < j\right\}\right|, \quad \sum_{j'=1}^{j} \gamma^R_{i,j'} = \left|\left\{t \ :\ r^{(t)}_{i} \leq j\right\}\right|.$$

The ICC implies $\left\{t \ :\ r^{(t)}_i \leq j\right\} \subseteq \left\{t \ :\ l^{(t)}_{i-1} < j\right\}$, and this gives (10). The statement in (11) is proved similarly. □

Of course, the decomposition time equals the sum of all $\gamma^L_{i,j}$ (or equivalently all $\gamma^R_{i,j}$) along any row:

$$k = \sum_{j=1}^{n+1} \gamma^L_{i,j} = \sum_{j=1}^{n+1} \gamma^R_{i,j} \qquad (i = 1, \ldots, m) \tag{12}$$

We can formulate the DT-problem with ICC as a linear program.

Theorem 2. *The DT-problem with ICC is equivalent to*

$$\min\left\{k \ \mid\ (9), (10), (11), (12),\ \gamma^L_{i,j}, \gamma^R_{i,j} \in \mathbb{N}\right\}. \tag{13}$$

Proof. The above argument shows that every decomposition with unidirectional leaf movement gives rise to a feasible solution of (13). Conversely, from every feasible solution of (13) we obtain a (unidirectional) decomposition with k shape matrices (defined according to (8)). We show that the unidirectional leaf movement is no restriction: every decomposition $A = \sum_{t=1}^{k} S^{(t)}$ with ICC yields a feasible solution of (13) with objective value k. Define $\gamma_{i,j}^L$ and $\gamma_{i,j}^R$ by (7). It is clear that (12) holds. By Lemma 2 we have (10) and (11). From

$$a_{i,j} = \left|\left\{t \;\middle|\; l_i^{(t)} < j < r_i^{(t)}\right\}\right| \quad \text{and} \quad a_{i,j-1} = \left|\left\{t \;\middle|\; l_i^{(t)} < j-1 < r_i^{(t)}\right\}\right|,$$

it follows that

$$\begin{aligned} a_{i,j} - a_{i,j-1} &= \left|\left\{t \;\middle|\; l_i^{(t)} = j-1,\ r_i^{(t)} > j\right\}\right| - \left|\left\{t \;\middle|\; l_i^{(t)} < j-1,\ r_i^{(t)} = j\right\}\right| \\ &= \left|\left\{t \;\middle|\; l_i^{(t)} = j-1,\ r_i^{(t)} \geq j\right\}\right| - \left|\left\{t \;\middle|\; l_i^{(t)} \leq j-1,\ r_i^{(t)} = j\right\}\right| \\ &= \gamma_{i,j}^L - \gamma_{i,j}^R. \end{aligned}$$

So (9) holds and this concludes the proof. □

Note that this also shows how an arbitrary leaf sequence can be transformed to an unidirectional one with the same DT: define the $\gamma_{i,j}^L$ and $\gamma_{i,j}^R$ according to (7) and the new shape matrices with (8). Obviously, the values

$$\tilde{\gamma}_{i,j}^L = \max\{0, a_{i,j} - a_{i,j-1}\}, \qquad \tilde{\gamma}_{i,j}^R = \max\{0, a_{i,j-1} - a_{i,j}\}$$

satisfy the conditions (9) and these conditions imply $\gamma_{i,j}^L \geq \tilde{\gamma}_{i,j}^L$ and $\gamma_{i,j}^R \geq \tilde{\gamma}_{i,j}^R$ for all pairs (i,j). The $\tilde{\gamma}_{i,j}^L$ and $\tilde{\gamma}_{i,j}^R$ correspond to the sweep solution for the unconstrained case coming from Algorithm 1. Since

$$\gamma_{i,j}^L - \gamma_{i,j}^R = \tilde{\gamma}_{i,j}^L - \tilde{\gamma}_{i,j}^R = a_{i,j} - a_{i,j-1},$$

we can represent $\gamma_{i,j}^L$ and $\gamma_{i,j}^R$ with a single nonnegative variable $w_{i,j}$ via

$$\gamma_{i,j}^L = \tilde{\gamma}_{i,j}^L + w_{i,j}, \qquad \gamma_{i,j}^R = \tilde{\gamma}_{i,j}^R + w_{i,j}.$$

With

$$T_i = \sum_{j=1}^{n+1} \tilde{\gamma}_{i,j}^L = \sum_{j=1}^{n+1} \tilde{\gamma}_{i,j}^R \qquad (i = 1, \ldots, m)$$

the constraints (10), (11), (12) become

$$\sum_{j'=1}^{j} \tilde{\gamma}_{i-1,j'}^L + \sum_{j'=1}^{j} w_{i-1,j'} \geq \sum_{j'=1}^{j} \tilde{\gamma}_{i,j'}^R + \sum_{j'=1}^{j} w_{i,j'} \qquad (i = 2, \ldots, m;\ j = 1, \ldots, n+1), \quad (14)$$

$$\sum_{j'=1}^{j} \tilde{\gamma}_{i,j'}^L + \sum_{j'=1}^{j} w_{i,j'} \geq \sum_{j'=1}^{j} \tilde{\gamma}_{i-1,j'}^R + \sum_{j'=1}^{j} w_{i-1,j'} \quad (i = 2, \ldots, m;\ j = 1, \ldots, n+1), \quad (15)$$

$$k = T_i + \sum_{j=1}^{n+1} w_{i,j} \qquad (i = 1, \ldots, m), \quad (16)$$

$$w_{i,j} \geq 0,\ w_{i,j} \in \mathbb{Z} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n+1). \quad (17)$$

Observe that (14) and (15) with $j = n+1$ yield

$$T_{i-1} + \sum_{j=1}^{n+1} w_{i-1,j} \overset{(14)}{\geq} T_i + \sum_{j=1}^{n+1} w_{i,j} \overset{(15)}{\geq} T_{i-1} + \sum_{j=1}^{n+1} w_{i-1,j}.$$

Consequently, we have equality and thus (16) follows from (14) and (15). This simplifies the problem, because now we just have to minimize $\sum_{j=1}^{n+1} w_{i,j}$ for any row i, and the problem becomes e.g.

$$\min \left\{ \sum_{j=1}^{n+1} w_{1,j} \;\mid\; (14), (15), (17) \right\}. \tag{18}$$

For a feasible solution $W = (w_{i,j})$ we denote the maximal index of a shape matrix having the left (right) leaf in row i to the left of column j by $I_L^{(i)}(j)$ ($I_R^{(i)}(j)$). In other words,

$$I_L^{(i)}(j) := \max\{t \;\mid\; l_i^{(t)} < j\} = \sum_{j'=1}^{j} \tilde{\gamma}_{i,j}^L + \sum_{j'=1}^{j} w_{i,j},$$

$$I_R^{(i)}(j) := \max\{t \;\mid\; r_i^{(t)} \leq j\} = \sum_{j'=1}^{j} \tilde{\gamma}_{i,j}^R + \sum_{j'=1}^{j} w_{i,j}$$

for $i = 1, \ldots, m$, $j = 1, \ldots, n+1$. In addition we put $I_L^{(i)}(0) = I_R^{(i)}(0) = 0$ for $i = 1, \ldots, m$. For the shape matrices $S^{(1)}, \ldots, S^{(k)}$ in the corresponding decomposition we have

$$s_{i,j}^{(t)} = 1 \iff I_R^{(i)}(j) < t \leq I_L^{(i)}(j),$$

hence for a feasible solution,

$$I_L^{(i)}(j) - I_R^{(i)}(j) = a_{i,j}.$$

Observe that the $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$ are exactly the terms which occur in the constraints (14) and (15). So these constraints can be rewritten as

$$I_L^{(i-1)}(j) \geq I_R^{(i)}(j), \quad I_L^{(i)}(j) \geq I_R^{(i-1)}(j) \tag{19}$$

for $i = 2, \ldots, m$, $j = 1, \ldots, n+1$. For convenience we formulate the algorithm for the solution of (18) in terms of the $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$. Clearly, knowing these values is equivalent to knowing the $w_{i,j}$, and minimizing $\sum_{j=1}^{n+1} w_{i,j}$ is the same as minimizing $I_L^{(i)}(n+1)$. The idea is to determine the values in column j depending on the values in column $j-1$. We start with the lower bounds

$$I_L^{(i)}(j) := I_L^{(i)}(j-1) + \tilde{\gamma}_{i,j}^L, \qquad I_R^{(i)}(j) := I_R^{(i)}(j-1) + \tilde{\gamma}_{i,j}^R,$$

and then we run through the rows and eliminate violations by increasing the relevant values as little as possible. Note that by increasing the values in row $i-1$ we might create a new violation between row $i-1$ and row $i-2$. The recursive call of the function `Update` takes care of this.

Theorem 3. *Algorithm 2 solves the DT-problem with ICC in time $O(m^2 n)$.*

Algorithm 2 (DT-optimal leaf sequence with ICC)**.**
for $i = 1, \dots, m$ **do** $I_L^{(i)}(0) := 0$; $I_R^{(i)}(0) := 0$
for $j = 1, \dots, n+1$ **do**
  for $i = 1, \dots, m$ **do**
    $I_L^{(i)}(j) := I_L^{(i)}(j-1) + \tilde{\gamma}_{i,j}^L$
    $I_R^{(i)}(j) := I_R^{(i)}(j-1) + \tilde{\gamma}_{i,j}^R$
  for $i = 2, \dots, m$ **do**
    if $I_L^{(i)}(j) < I_R^{(i-1)}(j)$ **then**
      $I_L^{(i)}(j) := I_R^{(i-1)}(j)$; $I_R^{(i)}(j) := I_L^{(i)}(j) - a_{i,j}$
    if $I_R^{(i)}(j) > I_L^{(i-1)}(j)$ **then** `Update(`$i-1$`)`

Function `Update(`k`)`
$I_L^{(k)}(j) := I_R^{(k+1)}(j)$; $I_R^{(k)}(j) := I_L^{(k)}(j) - a_{i,j}$
if $\left(k \geq 2 \text{ and } I_R^{(k)}(j) > I_L^{(k-1)}(j)\right)$ **then** `Update(k-1)`

Proof. It is easy to see that after termination of the algorithm $I_L^{(i)}(j) - I_R^{(i)}(j) = a_{i,j}$ for $i = 1, \dots, m$ and $j = 1, \dots, n$. So we indeed obtain a decomposition of matrix S. Also (19) and hence (14) and (15) hold, and the result corresponds to a feasible solution $W = (w_{i,j})$ of (18). Let $\hat{W} = (\hat{w}_{i,j})$ be an optimal solution corresponding to $\hat{I}_L^{(i)}(j)$ and $\hat{I}_R^{(i)}(j)$.
Claim. At any time $I_L^{(i)}(j) \leq \hat{I}_L^{(i)}(j)$ and $I_R^{(i)}(j) \leq \hat{I}_R^{(i)}(j)$.
We prove this claim by induction on j. For $j = 1$, the initialization in the first inner loop gives $I_L^{(i)}(1) = a_{i,1}$ and $I_R^{(i)}(1) = 0$. The conditions for changing these values in the second inner loop are never satisfied, so our claim follows from

$$\hat{I}_L^{(i)}(1) = \tilde{\gamma}_{i,j}^L + \hat{w}_{i,1} \geq \tilde{\gamma}_{i,j}^L, \qquad \hat{I}_R^{(i)}(1) = \tilde{\gamma}_{i,j}^R + \hat{w}_{i,1} \geq \tilde{\gamma}_{i,j}^R = 0.$$

For $j > 1$, with induction the initialization in the first inner loop yields

$$I_L^{(i)}(j) = I_L^{(i)}(j-1) + \tilde{\gamma}_{i,j}^L \leq \hat{I}_L^{(i)}(j-1) + \tilde{\gamma}_{i,j}^L \leq \hat{I}_L^{(i)}(j-1) + \tilde{\gamma}_{i,j}^L + \hat{w}_{i,j} = \hat{I}_L^{(i)}(j),$$
$$I_R^{(i)}(j) = I_R^{(i)}(j-1) + \tilde{\gamma}_{i,j}^R \leq \hat{I}_R^{(i)}(j-1) + \tilde{\gamma}_{i,j}^R \leq \hat{I}_R^{(i)}(j-1) + \tilde{\gamma}_{i,j}^R + \hat{w}_{i,j} = \hat{I}_R^{(i)}(j).$$

Now suppose our claim is false and consider the step of the algorithm where we get $I_L^{(i)}(j) > \hat{I}_L^{(i)}(j)$ or $I_R^{(i)}(j) > \hat{I}_R^{(i)}(j)$ for the first time.

Case 1. The first condition for changing $I_L^{(i)}(j)$ is satisfied. Then for the values after the update we obtain

$$I_L^{(i)}(j) := I_R^{(i-1)}(j) \leq \hat{I}_R^{(i-1)}(j) \overset{\text{ICC}}{\leq} \hat{I}_L^{(i)}(j),$$
$$I_R^{(i)}(j) := I_L^{(i)}(j) - a_{i,j} \leq \hat{I}_L^{(i)}(j) - a_{i,j} = \hat{I}_R^{(i)}(j),$$

contradicting the assumption.

Case 2. We are in the function `Update(`k`)`. Then

$$I_L^{(i)}(j) := I_R^{(i+1)}(j) \leq \hat{I}_R^{(i+1)}(j) \overset{\text{ICC}}{\leq} \hat{I}_L^{(i)}(j),$$
$$I_R^{(i)}(j) := I_L^{(i)}(j) - a_{i,j} \leq \hat{I}_L^{(i)}(j) - a_{i,j} = \hat{I}_R^{(i)}(j),$$

contradicting the assumption.

This proves the claim, and from $I_L^{(i)}(n+1) \le \hat{I}_L^{(i)}(n+1)$ and the optimality of $\hat{W}$ the optimality of W follows. Now let's consider the complexity. There are $m-1$ passes through the second inner **for**-loop, and in the worst case each of these calls the function `Update` which calls itself at most m times. So the complexity of the second inner loop is $O(m^2)$, and since we have to run $n+1$ times through the outer loop, the total complexity of $O(m^2 n)$ follows. □

Variants of this algorithm were presented in [2] and [16]. The proof given here is a mixture of these two references. The algorithm can also be adapted very easily to ensure minimum distances δ_0 and δ_1 between opposite leaves in the same row and in adjacent rows, respectively, if this is possible at all [16].

3.1.2 A network flow approach

A first network flow algorithm for DT–problem without ICC was proposed in [1]. Here we present a network flow formulation from [4] which also includes the ICC. The set of shape matrices is identified with the set of paths from D to D' in the layered directed graph (digraph) $G = (V, E)$, constructed as follows. The vertices in the $i-$th layer correspond to the possible pairs (l_i, r_i) $(1 \le i \le m)$, and two additional vertices D and D' are added:

$$V = \{(i,l,r) : i = 1,\dots,m;\ l = 1,\dots,n;\ r = l+1,\dots,n+1\} \cup \{D, D'\}.$$

Between two vertices (i,l,r) and $(i+1,l',r')$ is an arc if the corresponding leaf positions are consistent with the ICC, i.e. if $l' \le r-1$ and $r' \ge l+1$. In addition E contains all arcs from D to the first layer, all arcs from the last layer m to D' and the arc (D', D), so

$$E = E_+(D) \cup E_-(D') \cup \bigcup_{i=1}^{m-1} E(i) \cup \{(D', D)\},\ \text{where}$$

$$\begin{aligned} E_+(D) &= \{(D, (1,l,r))\ :\ (1,l,r) \in V\},\\ E_-(D') &= \{((m,l,r), D')\ :\ (m,l,r) \in V\},\\ E(i) &= \{((i,l,r),(i+1,l',r')) : l' \le r-1,\ r' \ge l+1\}. \end{aligned}$$

There is a bijection between the possible leaf positions and the cycles in G. This is illustrated in Fig. 5 which shows two cycles in G for $m = 4$, $n = 2$, corresponding to the shape matrices

$$\begin{pmatrix} 1 & 0\\ 0 & 1\\ 1 & 1\\ 1 & 0 \end{pmatrix}\ \text{(straight lines)} \qquad \text{and} \qquad \begin{pmatrix} 0 & 1\\ 1 & 1\\ 1 & 0\\ 0 & 1 \end{pmatrix}\ \text{(dashed lines)}.$$

With a segment S, given by (l_1, r_1), $(l_2, r_2), \dots, (l_m, r_m)$, we associate a unit flow on the cycle

$$D, (1, l_1, r_1), (2, l_2, r_2), \dots, (m, l_m, r_m), D', D.$$

Then any positive combination of shape matrices defines a circulation $\phi : E \to \mathbb{R}_+$ on G. For instance,

$$3\begin{pmatrix} 1 & 0\\ 0 & 1\\ 1 & 1\\ 1 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 1\\ 1 & 1\\ 1 & 0\\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 2\\ 2 & 5\\ 5 & 3\\ 3 & 2 \end{pmatrix}$$

corresponds to 3 units of flow on $(D, (1,0,2), (2,1,3), (3,0,3), (4,0,2), D')$, 2 units of flow on $(D, (1,1,3), (2,0,3), (3,0,2), (4,1,3), D')$ and 5 units of flow on (D', D). The amount

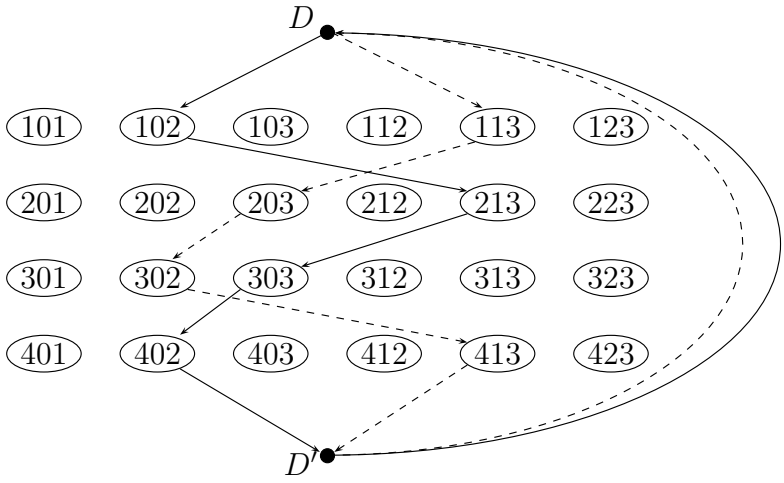

Figure 5: The vertices of G for $m = 4$, $n = 2$ and two cycles.

of radiation that is released at bixel (i, j) equals the sum of the flows going through the vertices (i, l, r) with $l < j < r$, hence the conditions that must be satisfied by the circulation in order to correspond to a segmentation of A are

$$\sum_{l=1}^{j-1} \sum_{r=j+1}^{n+1} \sum_{l'=1}^{r-1} \sum_{r'=\max\{l,l'\}-1}^{n} \phi((i, l, r), (i + 1, l', r')) = a_{i,j}, \tag{20}$$

for $1 \le i \le m - 1,\ 1 \le j \le n$, and

$$\sum_{l=1}^{j-1} \sum_{r=j+1}^{n+1} \phi((m, l, r), D') = a_{m,j}, \tag{21}$$

for $1 \le j \le n$. Since all of the flow must go through the arc (D', D), the DT of the segmentation corresponding to ϕ equals $\phi(D', D)$. Thus the DT–problem can be solved by finding a circulation satisfying conditions (20) and (21) and having minimal cost with respect to the cost function $\alpha : E \to \mathbb{R}_+$,

$$\alpha(e) = \begin{cases} 1 & \text{if } e = (D, D'), \\ 0 & \text{otherwise.} \end{cases}$$

The graph G can be expanded to a graph $\hat{G} = (\hat{V}, \hat{E})$ so that, instead of the constraints (20) and (21), the structure of $\hat{G}$ together with a capacity function on $\hat{E}$ forces the circulation to represent a decomposition of A.

$$\begin{aligned} \hat{V} = \{&(i, l, r)_1, (i, l, r)_2 \ \mid\ 1 \le i \le m,\ 0 \le l < r \le n + 1\} \\ &\cup \{(i, j) \ \mid\ 1 \le i \le m,\ 0 \le j \le n\} \cup \{D, D'\}. \end{aligned}$$

The arc set of $\hat{G}$ is $\hat{E} = \hat{E}^{\text{old}} \cup \hat{E}^1 \cup \hat{E}^2$, where

$$\begin{aligned} \hat{E}^{\text{old}} &= \{((i, l, r)_2, (i + 1, l', r')_1) \ :\ ((i, l, r), (i + 1, l', r')) \in E\} \\ &\quad \cup \{(D, (1, l, r)_1) \ :\ (1, l, r)_1 \in \hat{V}\} \cup \{((m, l, r)_2, D') \ :\ (m, l, r)_2 \in \hat{V}\} \cup \{(D', D)\}, \\ \hat{E}^1 &= \{((i, l, r)_1, (i, l)) \ :\ (i, l, r)_1 \in \hat{V}\} \cup \{((i, r - 1), (i, l, r)_2) \ :\ (i, l, r)_2 \in \hat{V}\}, \\ \hat{E}^2 &= \{((i, j - 1), (i, j)) \ :\ i = 1, \ldots, m;\ , j = 1, \ldots n\}. \end{aligned}$$

Now a shape matrix with parameters l_i, r_i $(i = 1, \ldots, m)$ corresponds to the cycle

$$\begin{aligned} D,&(1, l_1, r_1)_1, (1, l_1), (1, l_1 + 1), \ldots, (1, r_1 - 1), (1, l_1, r_1)_2, \\ &(2, l_2, r_2)_1, (2, l_2), (2, l_2 + 1), \ldots, (2, r_2 - 1), (2, l_2, r_2)_2, \\ &\ldots \\ &(m, l_m, r_m), (m, l_m), (m, l_m + 1), \ldots, (m, r_m - 1), (m, l_m, r_m)_2, D', D \end{aligned}$$

Figure 6 shows the cycles in $\hat{G}$ corresponding to the cycles in Figure 5. Now the flow

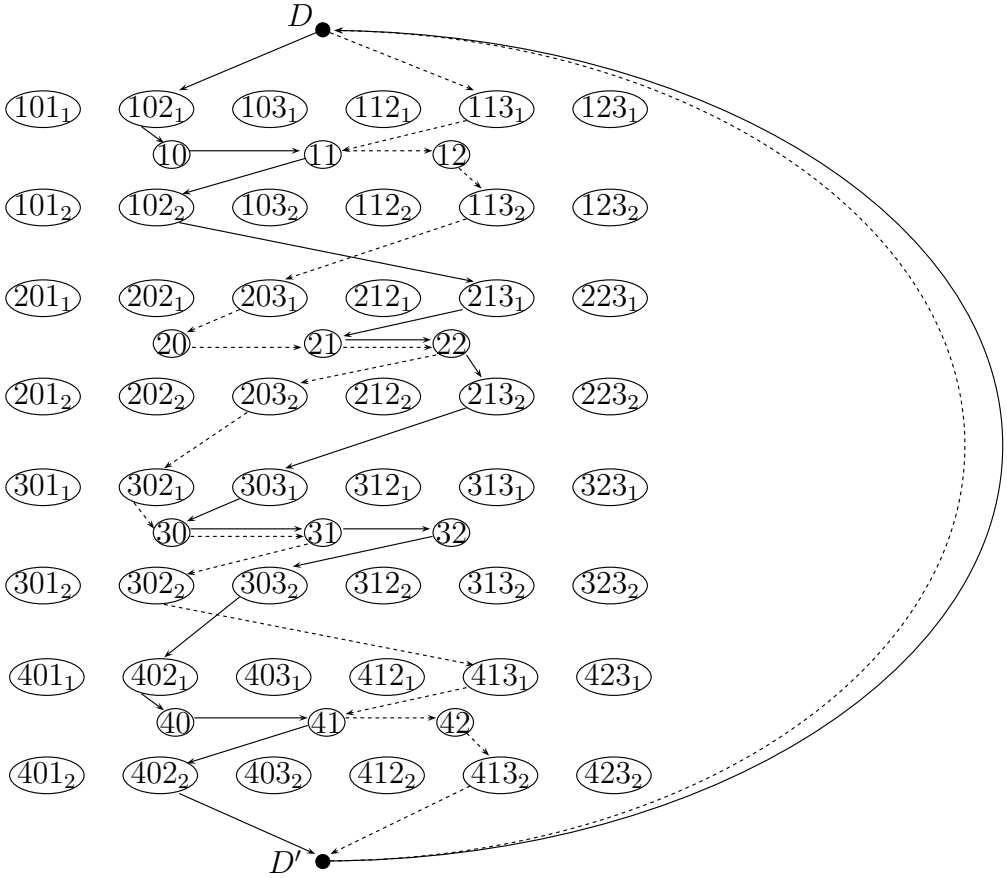

Figure 6: The vertices of $\hat{G}$ for $m = 4$, $n = 2$ and two cycles.

on the arc $((i, j-1), (i, j))$ equals the amount of radiation released at bixel (i, j) in the corresponding decomposition. Introducing lower and upper capacities $\underline{u}$ and $\overline{u}$ on the arcs of $\hat{G}$ by

$$\underline{u}(e) = \begin{cases} 0 & \text{if } e \in \hat{E}^{\text{old}} \cup \hat{E}^1 \\ a_{i,j} & \text{if } e = ((i, j-1), (i, j)) \in \hat{E}^2 \end{cases} \tag{22}$$

$$\overline{u}(e) = \begin{cases} \infty & \text{if } e \in \hat{E}^{\text{old}} \cup \hat{E}^1 \\ a_{i,j} & \text{if } e = ((i, j-1), (i, j)) \in \hat{E}^2 \end{cases} \tag{23}$$

we make sure that the fluence matrix A is realized. Now in order to obtain another reformulation of the DT–problem we just have to require that the flow on the arc $((i, l, r)_1, (i, l))$ equals the flow on the edge $((i, r-1), (i, l, r)_2)$, since both of these correspond to the total amount of radiation that is released while $l_i = l$ and $r_i = r$.

Theorem 4 ([4]). *The DT–minimization problem is equivalent to the network flow problem*

$$\text{minimize } \phi(D', D)$$

subject to ϕ a circulation in $\hat{G} = (\hat{V}, \hat{E})$ with lower and upper capacities $\underline{u}$ and $\overline{u}$, defined by (22) and (23), and satisfying, for all $(i,l,r)^{1,2} \in \hat{V}$,

$$\phi((i,l,r)_1, (i,l)) = \phi((i, r-1), (i,l,r)_2). \tag{24}$$

This formulation is quite close to a pure Min–Cost–Network–Flow problem. But the standard algorithms for this type of problem have to be adjusted in order to include the side constraint (24). Doing this one obtains a polynomial time algorithm for the DT-problem with ICC (see [4] and [19]).

3.1.3 A duality approach

Here we present another approach from [14] to the ICC-constraint, because it yields a nice characterization of the minimal DT, which can be modified to deal with the tongue-and–groove effect and also allows to derive a heuristic for the DC-problem in the next section. We only consider the problem without a minimum separation constraint, i.e. with $\delta_0 = \delta_1 = 0$ (introduced in the end of Section 3.1.1). Let the *DT-ICC-graph* $G = (V, E)$ be a digraph with vertex set V and arc set E defined as follows.

$$\begin{aligned}
V &= \{D, D'\} \cup \{(i,j) \mid 1 \le i \le m,\ 0 \le j \le n+1\} \\
E &= \{(D,(i,0)) \mid 1 \le i \le m\} \cup \{((i,n+1), D') \mid 1 \le i \le m\} \\
&\quad \cup \{((i,j),(i,j+1)) \mid 1 \le i \le m,\ 0 \le j \le n\} \\
&\quad \cup \{((i,j),(i+1,j)) \mid 1 \le i \le m-1,\ 1 \le j \le n-1\} \\
&\quad \cup \{((i,j),(i-1,j)) \mid 2 \le i \le m,\ 1 \le j \le n-1\}.
\end{aligned}$$

We define a weight function $w : E \to \mathbb{Z}$ by (recall that $a_{i,0} = a_{i,n+1} = 0$ for all i)

$$\begin{aligned}
w(D,(i,0)) = w((i,n+1),D') &= 0 && (i = 1,\dots,m) \\
w((i,j-1),(i,j)) &= \max\{0, a_{i,j} - a_{i,j-1}\} && (i = 1,\dots,m;\ j = 1,\dots,n+1) \\
w((i,j),(i+1,j)) &= -a_{i,j} && (i = 1,\dots,m-1;\ j = 1,\dots,n-1) \\
w((i,j),(i-1,j)) &= -a_{i,j} && (i = 2,\dots,m;\ j = 1,\dots,n-1).
\end{aligned}$$

Fig. 7 shows the digraph G for the matrix

$$A = \begin{pmatrix} 4 & 5 & 0 & 1 & 4 & 5 \\ 2 & 4 & 1 & 3 & 1 & 4 \\ 2 & 3 & 2 & 1 & 2 & 4 \\ 5 & 3 & 3 & 2 & 5 & 3 \end{pmatrix}.$$

As usual, the weight $w(P)$ of a path P is just the sum of the weights of the arcs contained in P. Now we can formulate the theoretical result underlying the next decomposition algorithm.

Theorem 5. *The minimal DT of a decomposition of A satisfying the ICC equals the maximal weight of a path from D to D' in G.*

In analogy with the unconstrained case we denote this maximal weight by $c(A)$.

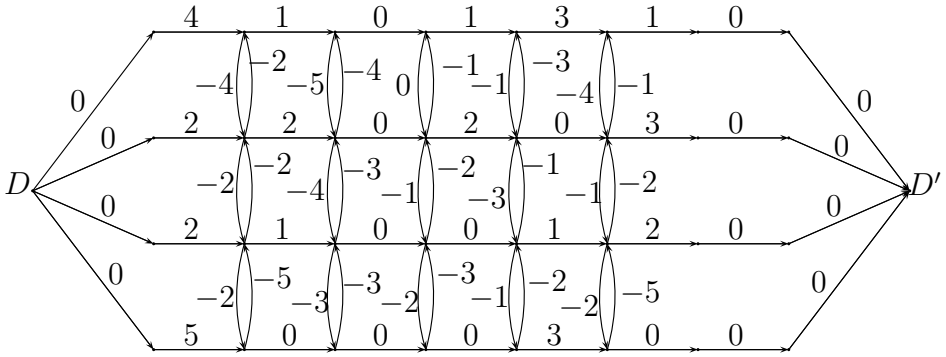

Figure 7: The DT-ICC-Graph with arc weights corresponding to matrix A.

Sketch of Proof. For the proof of this theorem we need to dualize the DT-problem (2). The LP–dual is

$$\max\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j}y_{i,j} \;\middle|\; \sum_{i=1}^{m}\sum_{j=1}^{n} s_{i,j}y_{i,j} \le 1 \text{ for all } S \in \mathcal{S}\right\}. \tag{25}$$

The basic idea of the proof is to associate with every $(D, D')-$path P a dual feasible solution $\boldsymbol{y^{(P)}}$ with objective value equal to the weight of P. By duality this gives the lower bound for DT. In a second step we will determine a shape matrix S that the maximal weight of a path with respect to $A' := A - S$ is strictly less than the maximal weight with respect to A, i.e. $c(A') < c(A)$. The value of $y_{i,j}^{(P)}$ depends on how the path P passes through the vertex (i, j).

$$y_{i,j}^{(P)} = \begin{cases} 1 & \text{if } (i,j-1),(i,j),(i,j+1) \in P \text{ and } a_{i,j-1} \le a_{i,j} > a_{i,j+1}, \\ -1 & \text{if } (i,j-1),(i,j),(i,j+1) \in P \text{ and } a_{i,j-1} > a_{i,j} \le a_{i,j+1}, \\ -1 & \text{if } (i,j-1),(i,j),(i\pm 1,j) \in P \text{ and } a_{i,j} < a_{i,j-1}, \\ -1 & \text{if } (i\pm 1,j),(i,j),(i,j+1) \in P \text{ and } a_{i,j+1} \ge a_{i,j}, \\ -1 & \text{if } (i\mp 1,j),(i,j),(i\pm 1,j) \in P, \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 8 illustrates this definition by showing the nonzero values of $y_{i,j}^{(P)}$ depending on the two neighbours of (i, j) in P. The following two lemmas establish the lower bound part

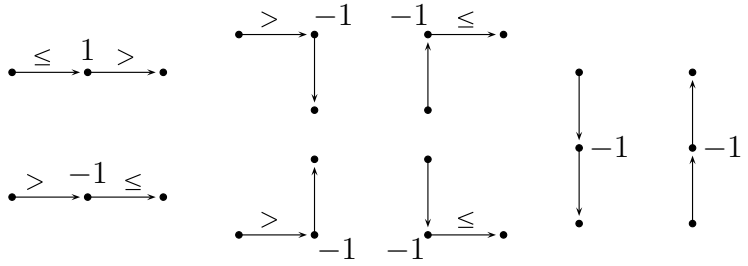

Figure 8: Illustration of the dual solution $\boldsymbol{y^{(P)}}$. The labels of the arcs indicate the relation of $a_{i,j}$ to its neighbours on P.

of the theorem.

Lemma 3. *For every $(D, D')-$path P, $\boldsymbol{y^{(P)}}$ is a feasible solution for the problem (25).*

Lemma 4. *For every $(D, D')-$path P, we have*

$$\sum_{i=1}^{m}\sum_{j=1}^{n} y_{i,j}^{(P)} a_{i,j} = w(P).$$

In order to construct a shape matrix reducing the maximal path weight we consider the following quantities.

$$\begin{aligned}
\alpha_1(i,j) &:= \max\{w(P) \mid P \text{ is a path from } D \text{ to } (i,j).\},\\
\alpha_2(i,j) &:= \max\{w(P) \mid P \text{ is a path from } (i,j) \text{ to } D.\},\\
\alpha(i,j) &:= \alpha_1(i,j) + \alpha_2(i,j).
\end{aligned}$$

In Fig. 9 we show the necessary information to determine the shape matrix. Observe

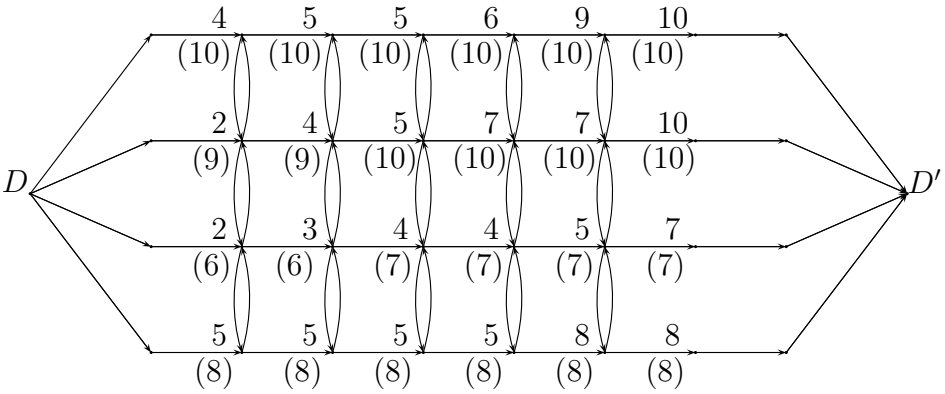

Figure 9: The values for α_1 and α (in parentheses) corresponding to the weights in Fig. 7.

that $c(A) = \max_{(i,j)} \alpha(i,j)$. We define a $0-1-$matrix by

$$s_{i,j} = 1 \iff \alpha(i,j) = c(A),\ \alpha_1(i,j) = a_{i,j} \text{ and } a_{i,j} > 0. \tag{26}$$

Lemma 5. *The matrix defined by (26) is a shape matrix satisfying the ICC.*

Lemma 6. *For the shape matrix defined by (26), the matrix $A' = A-S$ is still nonnegative and we have $c(A') = c(A) - 1$.*

Iterating this construction we obtain a decomposition of A into $c(A)$ shape matrices and this concludes the proof. □

As a consequence of the proof we obtain Algorithm 3. Note that this algorithm yields

Algorithm 3 (DT-optimal decomposition based on the DT-ICC-graph)**.**
Determine the values of α_1 and α
while $A \neq 0$ **do**
  Determine S according to (26)
  $A := A - S$
  Update α_1 and α

unidirectional decompositions, i.e. the leaves move only from left to right.

3.2 The tongue-and-groove constraint

Recall that in order to prevent underdosage effects due to the tongue–and–groove design of the leaves we have to require

$$a_{i,j} \le a_{i-1,j} \wedge s_{i,j} = 1 \quad \Longrightarrow \quad s_{i-1,j} = 1, \tag{27}$$

$$a_{i,j} \ge a_{i-1,j} \wedge s_{i-1,j} = 1 \quad \Longrightarrow \quad s_{i,j} = 1 \tag{28}$$

for $i = 2, \ldots, m$ and $j = 1, \ldots, n$. We call these the tongue-and-groove constraints (TGC). Here we construct a decomposition with unidirectional leaf movement satisfying the TGC. Recall from section 3.1.1 that such a decomposition is uniquely determined by the numbers $I_L^{(i)}(j)$, $I_R^{(i)}(j)$ $(i = 1, \ldots, m,\ j = 1, \ldots, n+1)$. The following lemmas characterize decompositions satisfying TGC (respectively ICC and TGC) in terms of the $I_R^{(i)}(j)$ and $I_L^{(i)}(j)$.

Lemma 7. *The TGC are satisfied if and only if for $i = 2, \ldots, m$, $j = 1, \ldots, n$,*

(a) $a_{i,j} = 0$ *or* $a_{i-1,j} = 0$, *or*

(b) $I_R^{(i-1)}(j) \le I_R^{(i)}(j) \le I_L^{(i)}(j) \le I_L^{(i-1)}(j)$, *or*

(c) $I_R^{(i)}(j) \le I_R^{(i-1)}(j) \le I_L^{(i-1)}(j) \le I_L^{(i)}(j)$.

Proof. Assume the TGC are satisfied and $\min\{a_{i,j}, a_{i-1,j}\} > 0$. For the $t-$th shape matrix $S^{(t)} = (s_{i,j}^{(t)})$ we have

$$s_{i,j}^{(t)} = 1 \iff I_R^{(i)}(j) < t \le I_L^{(i)}(j) \quad (i = 1, \ldots, m;\ j = 1, \ldots, n).$$

From this we derive that $a_{i,j} \le a_{i-1,j}$ and (27) imply (b), while $a_{i,j} \ge a_{i-1,j}$ and (28) imply (c). Conversely, assume $a_{i,j} \le a_{i-1,j}$ and $s_{i,j}^{(t)} = 1$. It follows, that (b) is true, and consequently $s_{i-1,j}^{(t)} = 1$. Similarly, from $a_{i,j} \ge a_{i-1,j}$ and $s_{i-1,j}^{(t)} = 1$ it follows that $s_{i,j}^{(t)} = 1$. □

Lemma 8. *The ICC and TGC are satisfied if and only if for $i = 2, \ldots, m$, $j = 1, \ldots, n$,*

(a) $I_R^{(i-1)}(j) \le I_R^{(i)}(j) \le I_L^{(i)}(j) \le I_L^{(i-1)}(j)$, *or*

(b) $I_R^{(i)}(j) \le I_R^{(i-1)}(j) \le I_L^{(i-1)}(j) \le I_L^{(i)}(j)$.

Proof. If $\min\{a_{i,j}, a_{i-1,j}\} > 0$ the proof is the same as for Lemma 7. If $a_{i,j} = 0$ we have $I_L^{(i)}(j) = I_R^{(i)}(j)$ and the ICC is equivalent to $I_R^{(i)}(j) \le I_L^{(i-1)}(j)$ and $I_L^{(i)}(j) \ge I_R^{(i-1)}(j)$, so (a) follows. Similarly, (b) follows from $a_{i-1,j} = 0$. □

Algorithm 4 can be used to obtain leaf sequences satisfying TGC and ICC. The basic idea is similar to the one in Algorithm 2. We construct the $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$ columnwise. In column j we start with the lower bounds

$$I_L^{(i)}(j) := I_L^{(i)}(j-1) + \max\{0, a_{i,j} - a_{i,j-1}\}, \quad I_R^{(i)}(j) := I_R^{(i)}(j-1) + \max\{0, a_{i,j-1} - a_{i,j}\},$$

and eliminate the violations of Lemma 8. If $a_{i,j} \le a_{i-1,j}$, condition (a) in Lemma 8 must be satisfied. This can be violated if $I_R^{(i)}(j) < I_R^{(i-1)}(j-1)$ or $I_L^{(i)}(j) > I_L^{(i-1)}(j)$. In the first case we increase $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$ by the minimum amount such that the condition

Algorithm 4 (DT-optimal leaf sequence with TGC and ICC)**.**
for $i = 1, \ldots, m$ **do** $I_L^{(i)}(0) := 0$; $I_R^{(i)}(0) := 0$
for $j = 1, \ldots, n+1$ **do**
  for $i = 1, \ldots, m$ **do**
    $I_L^{(i)}(j) := I_L^{(i)}(j-1) + \max\{0, a_{i,j} - a_{i,j-1}\}$
    $I_R^{(i)}(j) := I_R^{(i)}(j-1) + \max\{0, a_{i,j-1} - a_{i,j}\}$
  for $i = 2, \ldots, m$ **do**
    if $a_{i,j} \leq a_{i-1,j}$ **then**
      if $I_R^{(i)}(j) < I_R^{(i-1)}(j-1)$ **then**
        $\Delta := I_R^{(i-1)}(j) - I_R^{(i)}(j)$
        $I_R^{(i)}(j) := I_R^{(i)}(j) + \Delta$; $I_L^{(i)}(j) := I_L^{(i)}(j) + \Delta$
      if $I_L^{(i)}(j) > I_L^{(i-1)}(j)$ **then** `Update`$(i-1)$
    else //the case $a_{i,j} > a_{i-1,j}$
      if $I_L^{(i)}(j) < I_L^{(i-1)}(j)$ **do**
        $\Delta := I_L^{(i-1)}(j) - I_L^{(i)}(j)$
        $I_R^{(i)}(j) := I_R^{(i)}(j) + \Delta$; $I_L^{(i)}(j) := I_L^{(i)}(j) + \Delta$
      if $I_R^{(i)}(j) > I_R^{(i-1)}(j)$ **then** `Update`$(i-1)$

Function `Update`(k)
if $a_{k,j} \leq a_{k+1,j}$ **then**
  $\Delta := I_R^{(k+1)}(j) - I_R^{(k)}(j)$
  $I_R^{(k)}(j) := I_R^{(k)}(j) + \Delta$; $I_L^{(k)}(j) := I_L^{(k)}(j) + \Delta$
else //the case $a_{k,j} > a_{k+1,j}$
  $\Delta := I_L^{(k+1)}(j) - I_L^{(k)}(j)$
  $I_R^{(k)}(j) := I_R^{(k)}(j) + \Delta$; $I_L^{(k)}(j) := I_L^{(k)}(j) + \Delta$
if $k \geq 2$, $a_{k,j} \leq a_{k-1,j}$ and $I_L^{(k)}(j) > I_L^{(k-1)}(j)$ **then** `Update`$(k-1)$
if $k \geq 2$, $a_{k,j} > a_{k-1,j}$ and $I_R^{(k)}(j) > I_R^{(k-1)}(j)$ **then** `Update`$(k-1)$

holds, and in the second case we increase $I_L^{(i-1)}(j)$ and $I_R^{(i-1)}(j)$. In this second case there might be a new violation between row $i-1$ and row $i-2$. The recursive call of the function `Update`(k) takes care of this. If there is no ICC, we just have to add the condition $\min\{a_{i,j}, a_{i-1,j}\} > 0$ to the conditions for changing the values $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$. Let *Algorithm 4'* denote the result of this modification.

Theorem 6 ([17])**.** *Algorithm 4 yields DT-optimal decompositions with ICC and TGC under the additional condition of unidirectional leaf movement in time $O(m^2 n)$. Algorithm 4', yields DT-optimal decompositions with TGC and without ICC under the additional condition of unidirectional leaf movement in time $O(m^2 n)$.*

The proof of this theorem is essentially the same as the proof of Theorem 3. For the problem with ICC and TGC the condition on the unidirectional leaf movement can be dropped: using the duality based Algorithm 3 with a modified weight function yields a decomposition with unidirectional leaf movement that is optimal among all decompositions with ICC and TGC [13].

4 The decomposition cardinality problem

In this section we consider the DC-problem (3). In the first subsection we show that this problem is very hard, and in the second section we give heuristic approaches.

4.1 The computational complexity of the DC–problem

The fact that the DC-problem is NP-hard already for a single row matrix was proved first by Burkart [7] who gave a reduction from 2-partition. A similar idea was used in [2] to reduce 3-partition showing the strong NP-hardness, i.e. the nonexistence of a pseudopolynomial algorithm unless $P = NP$.

Theorem 7 ([2])**.** *The problem (3) is strongly NP-hard, even for matrices with a single row.*

Proof. The decision version of the single–row DC-problem is as follows:

Instance: A vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ with $a_i \in \mathbb{N}$, $K \in \mathbb{N}$

Question: Does a decomposition of $\boldsymbol{a}$ into at most K shape matrices exist?

Note that a shape matrix in this case is nothing else than a row vector with the consecutive ones property. We use reduction from the problem 3-partition, which is well-known to be strongly NP-hard [11].

Instance: $B, Q \in \mathbb{N}$, $b_1, b_2, \ldots, b_{3Q} \in \mathbb{N}$ with $\sum_{j=1}^{3Q} b_j = QB$ and $\frac{B}{4} < b_j < \frac{B}{2}$ for all j

Question: Does a partitioning of $\{b_1, \ldots, b_{3Q}\}$ into triples $T_1, \ldots, T_Q$ such that $\sum_{b \in T_q} b = B$ for all $q = 1, \ldots, Q$ exist?

We define an instance of the DC-problem as follows.

$$
\begin{aligned}
& n = 4Q, \\
& a_j = \begin{cases} \sum_{k=1}^{j} b_k & \text{for } j = 1, \ldots, 3Q \\ (4Q - j + 1)B & \text{for } j = 3Q+1, \ldots, 4Q, \end{cases} \\
& K = 3Q.
\end{aligned}
$$

Now it is not difficult to see, that the instance of 3-partition has answer YES if and only if the instance of the DC-problem has answer YES. □

We want to mention that this reduction proves even more, namely that it is already hard to find an approximate solution of the DC-problem. To be precise, the DC-problem is APX-hard even for single-row matrices with entries polynomially bounded in n. That means there is some $\varepsilon > 0$ such that, unless $P = NP$, there is no polynomial algorithm which decides whether the necessary number of shape matrices is K or at least $(1+\varepsilon)K$. This was shown in [3] using a result on the APX-hardness of 3-partition from [21].
The strong NP-hardness of 3-partition means that the problem remains NP-hard even if the input numbers are bounded by some constant. But in the reduction to the DC-problem we produce a vector with very large entries, because we have to sum up all the numbers from the 3-partition instance. So we can still hope for an efficient algorithm if we bound the entries of the matrix by some constant L, i.e. we require $a_{i,j} \leq L$ for all

(i,j). Observe that for the case $L = 1$ any optimal solution to the DT-problem is also optimal for the DC-problem. And indeed, there is a result in this direction: for constant L, the DC-problem without ICC and TGC can be solved in time $O(mn^{2L+2})$ [12]. In [20] the algorithm was extended in order to find the exact minimum of the DC without the restriction that the DT has to be minimal. But these pseudopolynomial algorithms are of very limited practical value, not only because of the L in the exponent but also because the constant in the $O-$notation grows very fast with L. So it is natural to require heuristic approaches to the DC-problem.

4.2 Heuristics for the DC–problem

Most of the algorithms in the literature look for a decomposition with minimum DC among all decompositions with minimum DT. So the problem (which we also call DC-problem in the following) is

$$\textbf{(DC')} \qquad \min\left\{ |\mathcal{S}_0| \;\mid\; \mathcal{S}_0 \subseteq \mathcal{S},\ A = \sum_{S\in\mathcal{S}_0} u_S S,\ u_S \in \mathbb{N},\ \sum_{S\in\mathcal{S}_0} u_S \text{ is minimal.} \right\}.$$

Note that in general it is not possible to minimize both quantities simultaneously, as can be seen by the following example (from [15]):

$$\begin{pmatrix} 2 & 6 & 3 \\ 4 & 5 & 6 \end{pmatrix} = 3\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

This is a decomposition with $DT = 6$ which cannot be achieved with 3 shape matrices. But allowing $DT = 7$, 3 shape matrices are sufficient:

$$\begin{pmatrix} 2 & 6 & 3 \\ 4 & 5 & 6 \end{pmatrix} = 4\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} + 2\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

So the problem (DC') is really different from (3). As before, let $c(A)$ denote the minimal DT, for matrix A. A very natural greedy strategy is to look for a shape matrix S that can be extracted with a large coefficient u, such that $c(A - uS) = c(A) - u$, i.e. uS can be extended to a DT-optimal decomposition.

4.2.1 The unconstrained case

The following greedy heuristic for the unconstrained case was proposed in [8]. We are looking for a pair (u, S) of a positive integer u and a shape matrix S, such that $A - uS$ is still nonnegative, $c(A - uS) = c(A) - u$ and u is maximal under these conditions. Let u_{max} be this maximal possible value. Using the notation introduced before Theorem 1, $c(A - uS) = c(A) - u$ is equivalent to

$$c_i(A - uS) \le c(A) - u \qquad (i = 1, \ldots, m).$$

Define the *complexity gap* of row i to be $g_i(A) := c(A) - c_i(A)$. As before we describe the shape matrix by the parameters l_i and r_i $(i = 1, \ldots, m)$.

Lemma 9. *There is a pair (u, S) with $u = u_{max}$ and, for all i, either $l_i = r_i - 1$ or $(a_{i,l_i} < a_{i,l_i+1}$ and $a_{i,r_i-1} > a_{i,r_i})$.*

We put $d_{i,j} = a_{i,j} - a_{i,j-1}$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$, and define

$$v_i(l,r) = \begin{cases} g_i(A) & \text{if } l = r-1, \\ g_i(A) + \min\{d_{i,l+1}, -d_{i,r}\} & \text{if } l < r-1 \text{ and } g_i(A) \le |d_{i,l+1} + d_{i,r}|, \\ (d_{i,l+1} - d_{i,r} + g_i(A))/2 & \text{if } l \le r \text{ and } g_i(A) > |d_{i,l+1} + d_{i,r}|. \end{cases}$$

Lemma 10. *$c_i(A - uS) \le c(A) - u$ if and only if $u \le v_i(l_i, r_i)$.*

For convenience we denote the set of pairs (l,r) to which we restrict our search in row i by $\mathcal{I}_i$, that is we put

$$\mathcal{I}_i := \{(l,r) \ : \ 0 \le l \le r-1 \le n \text{ and either } l = r-1 \text{ or } (d_{i,l+1} > 0 \text{ and } d_{i,r} < 0)\}.$$

Clearly the nonnegativity of $A - uS$ is equivalent to $u \le w_i(l_i, r_i)$ for all i, where

$$w_i(l,r) = \begin{cases} \infty & \text{if } l = r-1, \\ \min\limits_{l<j<r} a_{i,j} & \text{if } l < r-1. \end{cases}$$

Now we put, for $1 \le i \le m$ and $(l,r) \in \mathcal{I}_i$, $\hat{u}_i(l,r) = \min\{v_i(l,r), w_i(l,r)\}$, and for $i = 1, \ldots, m$,

$$\tilde{u}_i = \max_{(l,r)\in\mathcal{I}_i} \hat{u}_i(l,r).$$

Then

$$u_{max} = \min_{1\le i\le m} \tilde{u}_i.$$

In order to construct a shape matrix S such that, for $u = u_{max}$, $A - uS$ is nonnegative and $c(A - uS) = c(A) - u$, we just have to find, for every $i \in [m]$, a pair $(l_i, r_i) \in \mathcal{I}_i$ with $\hat{u}_i(l_i, r_i) \ge u_{max}$. A trivial way of doing this is to take a pair (l_i, r_i) where the maximum in the definition of $\tilde{u}_i$ is attained, i.e. with $\hat{u}_i(l_i, r_i) = \tilde{u}_i$. These (l_i, r_i) can be computed simultaneously with the calculation of u_{max} and this method yields $mn + n - 1$ as an upper bound for the DC of the decomposition. But there are better constructions for S after the determination of u_{max}. We put

$$q(A) = |\{(i,j) \in [m] \times [n] \ : \ d_{i,j} \ne 0\}|,$$

and choose a shape matrix S so that $q(A - uS)$ is minimized. To make this precise, for $1 \le i \le m$ and $(l,r) \in \mathcal{I}_i$, we put

$$p_i(l,r) = \begin{cases} 2 & \text{if } d_{i,l+1} = -d_{i,r} = u_{max}, \\ 1 & \text{if } d_{i,l+1} = u_{max} \ne -d_{i,r} \text{ or } d_{i,l+1} \ne u_{max} = -d_{i,r}, \\ 0 & \text{if } l = r+1 \text{ or } (d_{i,l+1} \ne u_{max} \text{ and } -d_{i,r} \ne u_{max}). \end{cases}$$

Now for (l_i, r_i) we choose among the pairs $(l,r) \in \mathcal{I}_i$ with $\hat{u}_i(l,r) \ge u_{max}$ one with maximal value of $p_i(l,r)$, and if there are several of these we choose one with maximal value of $r - l$. As an example we obtain the following decomposition.

$$\begin{pmatrix} 4&5&0&1&4&5 \\ 2&4&1&3&1&4 \\ 2&3&2&1&2&4 \\ 5&3&3&2&5&3 \end{pmatrix} = 4\begin{pmatrix} 1&1&0&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&1 \\ 1&0&0&0&0&0 \end{pmatrix} + 2\begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&1&0&0 \\ 1&1&1&0&0&0 \\ 0&1&1&1&1&0 \end{pmatrix} + \begin{pmatrix} 0&0&0&1&1&1 \\ 1&1&1&1&0&0 \\ 0&0&0&1&1&0 \\ 1&1&1&0&0&0 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&1&1 \\ 1&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&1&1 \end{pmatrix}$$
$$+ \begin{pmatrix} 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&0&1 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&1&1 \end{pmatrix}.$$

4.2.2 The interleaf collision constraint

Using the Min-Max-characterization of the minimal DT from Theorem 5, we can use the same strategy as for the unconstrained case: denoting by $c(A)$ the maximal weight with respect to A of a $(D, D')-$path in the DT-ICC-graph, we are looking for a pair (u, S) with maximal u such that $A - uS$ is nonnegative and $c(A - uS) = c(A) - u$. If this is the case we call the pair (u, S) *admissible*. An additional difficulty comes from the fact that the influence of the extraction of uS on the weight of a path through a vertex (i, j) does not only depend on the $i-$th row of S, so the determination of the values

$$\hat{u}_i(l, r) := \max\{u \mid \exists \text{ shape matrix } S \text{ with } l_i = l,\ r_i = r \text{ such that } A - uS \text{ is nonnegative and } c(A - uS) = c(A) - u\}$$

becomes much harder. But suppose these values, or at least some upper bounds $u_0(i, l, r)$ for them, are given. Then for given u, every shape matrix S, such that (u, S) is admissible, corresponds to a path

$$D, (1, l_1, r_1), (2, l_2, r_2), \ldots, (m, l_m, r_m), D'$$

in the digraph defined in the beginning of Section 3.1.2 and illustrated in Fig. 5, such that $u_0(i, l_i, r_i) \geq u$ for $i = 1, \ldots, m$. We put

$$\hat{u} = \max\{u \ :\ \text{There is a path } D, (1, l_1, r_1), \ldots, (m, l_m, r_m), D' \text{ with } u_0(i, l_i, r_i) \geq u \text{ for } i = 1, \ldots, m\}.$$

Clearly, $\hat{u}$ is an upper bound for the coefficient u in an admissible pair (u, S). The backtracking described in Algorithm 5 constructs an admissible pair (u, S) with maximal u. Starting with $u = \hat{u}$, the algorithm searches for a shape matrix S such that (u, S) is admissible, and if this is not possible, the value is decreased by one. The shape matrix is build up row by row, and the stopping criterion in row i is that after extracting the current candidates for the first i rows with coefficient u leads to a path P with all its vertices in the first i rows and $w(P) > c(A) - u$. The maximal weight of such a path is denoted by `MaxWeight`(i). Iterating Algorithm 5 with $A' = A - uS$ we obtain a decomposition of A.

Algorithm 5 (Greedy step in the heuristic for the DC-problem with ICC)**.**
Function `Construct Shape Matrix`
$u := \hat{u}$
finished:=false
$l_0 := 0$; $r_0 := n + 1$
while not finished **do**
  `Complete Shape Matrix`(1)
  if not finished **then** $u := u - 1$

Function `Complete Shape Matrix`(i)
for (l_i, r_i) with $0 \leq l_i \leq r_{i-1} - 1$, $\max\{l_i, l_{i-1}\} + 1 \leq r_i \leq n + 1$ and $u_0(i, l_i, r_i) \geq u$ **do**
  if `MaxWeight`$(i) \leq c(A) - u$ **then**
    if $i < m$ **then** `Complete Shape Matrix`$(i + 1)$ **else** finished:=true

The performance of this backtracking depends very much on the quality of the bounds $u_0(i, l, r)$. In [13] some bounds that work quite well in practice are described. A drawback of this method is that we have almost no control of the running time. Experiments with randomly generated matrices show that the algorithm is fast for the vast majority of matrices but there are some examples where it is extremely slow.

References

[1] R.K. Ahuja and H.W. Hamacher. A network flow algorithm to minimize beam-on time for unconstrained multileaf collimator problems in cancer radiation therapy. *Networks*, 45(1):36–41, 2005.

[2] D. Baatar, H.W. Hamacher, M. Ehrgott, and G.J. Woeginger. Decomposition of integer matrices and multileaf collimator sequencing. *Discrete Appl. Math.*, 152(1-3):6–34, 2005.

[3] N. Bansal, D. Coppersmith, and B. Schieber. Minimizing setup and beam-on times in radiation therapy. In J. Diaz et al., editor, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4110 of *LNCS*, pages 27–38. Springer-Verlag, 2006.

[4] N. Boland, H. W. Hamacher, and F. Lenzen. Minimizing beam-on time in cancer radiation treatment using multileaf collimators. *Networks*, 43(4):226–240, 2004.

[5] T. Bortfeld. IMRT: a review and preview. *Phys. Med. Biol.*, 51:R363–R379, 2006.

[6] T.R. Bortfeld, D.L. Kahler, T.J. Waldron, and A.L. Boyer. X–ray field compensation with multileaf collimators. *Int. J. Radiat. Oncol. Biol. Phys.*, 28:723–730, 1994.

[7] R. Burkart. Open Problem Session, Oberwolfach Conference on Combinatorial Optimization, November 24-29, 2002.

[8] K. Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Appl. Math.*, 152(1-3):35–51, 2005.

[9] K. Engel and E. Tabbert. Fast simultaneous angle, wedge, and beam intensity optimization in inverse radiotherapy planning. *Optimization and Engineering*, 6(4):393–419, 2005.

[10] J.M. Galvin, X.G. Chen, and R.M. Smith. Combining multileaf fields to modulate fluence distributions. *Int. J. Radiat. Oncol. Biol. Phys.*, 27:697–705, 1993.

[11] M.R. Garey and D.S. Johnson. *Computers and intractability, a guide to the theory of NP–completeness*. W.H. Freeman, 1979.

[12] T. Kalinowski. The algorithmic complexity of the minimization of the number of segments in multileaf collimator field segmentation. Preprint 04/1, Institut für Mathematik, Uni Rostock, 2004.

[13] T. Kalinowski. Reducing the tongue-and-groove underdosage in MLC segmentation. Preprint 04/3, Institut für Mathematik, Uni Rostock, 2004.

[14] T. Kalinowski. A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Appl. Math.*, 152(1-3):52–88, 2005.

[15] T. Kalinowski. Realization of intensity modulated radiation fields using multileaf collimators. In R. Ahlswede et al., editor, *Information Transfer and Combinatorics*, volume 4123 of *LNCS*, pages 1010–1055. Springer-Verlag, 2006.

[16] S. Kamath, S. Sahni, J. Li, J. Palta, and S. Ranka. Leaf sequencing algorithms for segmented multileaf collimation. *Phys. Med. Biol.*, 48(3):307–324, 2003.

[17] S. Kamath, S. Sartaj, J. Palta, S. Ranka, and J. Li. Optimal leaf sequencing with elimination of tongue–and–groove underdosage. *Phys. Med. Biol.*, 49:N7–N19, 2004.

[18] S. Kamath, S. Sartaj, S. Ranka, J. Li, and J. Palta. A comparison of step–and–shoot leaf sequencing algorithms that eliminate tongue–and–groove effects. *Phys. Med. Biol.*, 49:3137–3143, 2004.

[19] F. Lenzen. An integer programming approach to the multileaf collimator problem. Master's thesis, University of Kaiserslautern, Dept. of Mathematics, 2000.

[20] M. Nußbaum. Min cardinality C1 decomposition of integer matrices. Master's thesis, Faculty for Mathematics, TU Kaiserslautern, 2006.

[21] E. Petrank. The hardness of approximation: gap location. *Computational complexity*, 4:133–157, 1994.

[22] H.E. Romeijn, R.K. Ahuja, J.F. Dempsey, and A. Kumar. A column generation approach to radiation therapy treatment planning using aperture modulation. *SIAM J. on Optimization*, 15(3):838–862, 2005.

[23] R.A.C. Siochi. Minimizing static intensity modulation delivery time using an intensity solid paradigm. *Int. J. Radiat. Oncol. Biol. Phys.*, 43:671–680, 1999.

[24] P. Xia and L. Verhey. Multileaf collimator leaf–sequencing algorithm for intensity modulated beams with multiple static segments. *Med. Phys.*, 25:1424–1434, 1998.