# Realization of Intensity Modulated Radiation Fields Using Multileaf Collimators

T. Kalinowski

**Abstract.** In the treatment of cancer using high energetic radiation the problem arises how to irradiate the tumor without damaging the healthy tissue in the immediate vicinity. In order to do this as efficiently as possible intensity modulated radiation therapy (IMRT) is used. A modern way to modulate the homogeneous radiation field delivered by an external accelerator is to use a multileaf collimator in the static or in the dynamic mode. In this paper several aspects of the construction of optimal treatment plans are discussed and some algorithms for this task are described.

## 1 Introduction

In cancer treatment high energetic radiation is used to destroy the tumor. To achieve this goal the irradiation process must be planned in such a way that the tumor (target volume) receives a sufficiently high dose while the organs close to it (organs at risk) are not damaged. In clinical practice the radiation is delivered by a linear accelerator which is part of a gantry that can be rotated about the treatment couch (see Figure 1).

**Fig. 1.** A linear accelerator with a treatment couch

The first step in the treatment planning after the target volume and the organs at risk have been localized is to discretize the radiation beam head into bixels and the irradiated volume into voxels. Then a set of gantry angles from which

R. Ahlswede et al. (Eds.): Information Transfer and Combinatorics, LNCS 4123, pp. 1010–1055, 2006.

**Fig. 2.** The leaf pairs of a multileaf collimator

radiation is released has to be determined. In order to increase the efficiency of the treatment it is often desirable to modulate the intensity profile of the radiation beam. So for each gantry angle an intensity function is prescribed, i.e. an amount of radiation released at each bixel. Finally, we have to find a way to realize this modulation. Here we consider only the last step of this planning process. That is as our starting point we take an intensity function for a fixed irradiation angle. We assume that the radiation head is a rectangle and choose a partition into equidistant cells as discretization. Then the intensity function can be described as a nonnegative matrix whose entries are the desired doses at the corresponding bixels. A modern approach to the modulation of homogeneous fields is the usage of a multileaf collimator (MLC). A multileaf collimator consists of one pair of metal leaves for each row of the intensity matrix (see Figure 2). These leaves can be inserted between the beam head and the patient in order to protect parts of the irradiated area. So differently shaped homogeneous fields are generated and by superimposing a number of these the given modulated intensity can be realized. There are two essentially different ways to generate intensity modulated fields with multileaf collimators: in the static mode (stop–and–shoot) the beam is switched off when the leaves are moving while in the dynamic mode the beam is switched on during the whole treatment and the modulation is achieved by varying the speed of the leaf motion. Two important criteria for the quality of a treatment plan are the total irradiation time and the total treatment time. The total irradiation time should be small since there is always a small amount of radiation transmitted through the leaves, and if the used model ignores this leaf transmission the error increases with the total irradiation time. A small total treatment time is desirable for efficiency reasons. In the dynamic mode the two criteria coincide. So here the problem is to determine a velocity function for each leaf such that the given intensity is realized in the shortest possible time. In the static mode the whole treatment consists of the irradiation

and the intervals in between when the leaves are moved. Thus we have two parameters which influence the total treatment time: the irradiation time and the number of homogeneous fields that are needed. How these parameters have to be weighted depends on the used technology: the longer the time intervals between the different fields are, the more important becomes the reduction of the number of fields. The lengths of these time intervals is influenced by the leaf velocity and by the so called verification and record overhead, which is the time necessary to check the correct positions of the leaves. In a more realistic model one should also take the shapes of the fields into account, because clearly the necessary leaf travel time between two fields depends on the shapes of these fields (see [18,2]). The dynamic mode has the advantage of a smaller total treatment time, but the static mode involves no leaf movement with radiation on and so the verification of the correct realization of the treatment plan is easier which makes the method less sensitive to malfunctions of the technology.

There are additional machine–dependent restrictions which have to be considered when determining the leaf positions:

**Interleaf collision constraint (ICC):** In some widely used MLC's it is forbidden that opposite leaves of adjacent rows overlap, because otherwise these leaves collide. So leaf positions as illustrated in Figure 3 are not allowed.

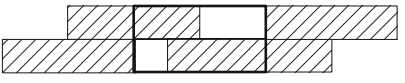

**Fig. 3.** Leaf position that is excluded by the ICC. The shading indicates the area that is covered by the leaves

**Tongue and groove constraint:** In order to reduce leakage radiation between adjacent leaves the commercially available MLC's use a tongue–and–groove (or similar) design (see Figure 4) with the effect that there is a small overlap of the regions that are covered by adjacent leaves.

Consider two bixels $x$ and $y$ that are adjacent along a column and two homogeneous fields, where in the first field $x$ is irradiated and $y$ is covered and in the second field $y$ is irradiated and $x$ is covered. Then in the composition of these fields along the border of $x$ and $y$ there is a narrow strip (the overlap of the regions that are covered by the leaves in the rows of $x$ and $y$, respectively) that receives no radiation at all. Figure 5 illustrates this for the intensity map $\left(\begin{smallmatrix} 2 & 3 \\ 3 & 4 \end{smallmatrix}\right)$.

To avoid this effect one may require that two bixels that are adjacent along a column are irradiated simultaneously for the time the lower of the two doses is delivered. Then the border region receives this lower dose. If this is the case for all the relevant pairs of adjacent bixels the treatment plan is said to satisfy the tongue and groove constraint.

In this paper we collect some of the known algorithms for the intensity modulation of radiation beams with multileaf collimators in a unified notation.

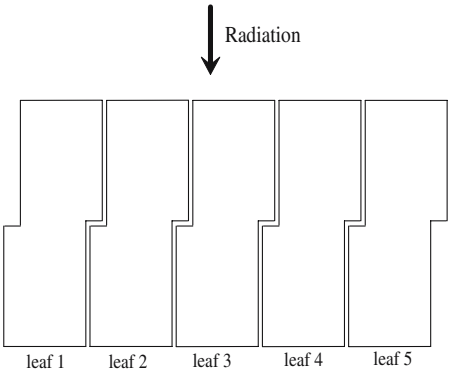


**Fig. 4.** The principle of the tongue–and–groove design. The picture shows a cut through the leaf bank perpendicular to the direction of leaf motion.

## 2 Static Methods

This chapter is organized as follows. In the first section some notation is introduced and we describe a linear programming formulation of the total irradiation time minimization (taken from [7]). In the second section we prove that the minimization of the number of homogeneous fields that are needed is an NP–complete problem. The remaining sections are devoted to the discussion of some concrete algorithms.

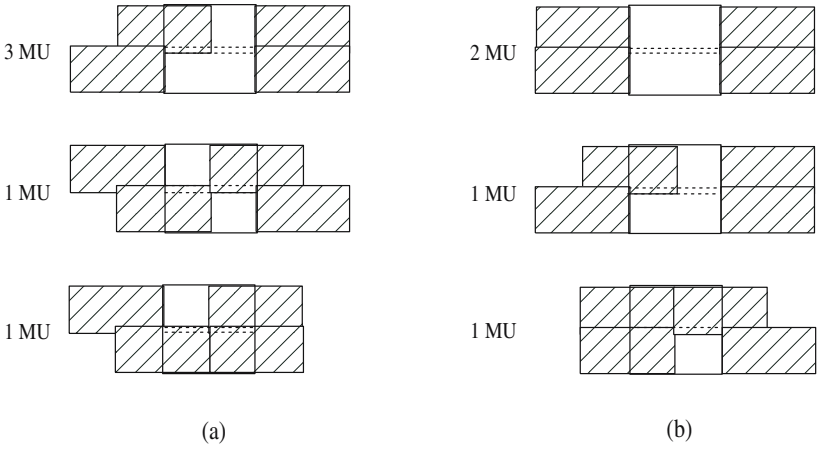


**Fig. 5.** Two realizations of the same intensity map. (a) The overlap of bixels $(1, 1)$ und $(2, 1)$ receives no radiation because of the tongue and groove effect. (b) The overlaps of bixels that are adjacent along a column receive the smaller one of the doses delivered to the overlapping bixels.

### 2.1 Notation and LP–Formulation

Throughout we use the notation $[n] := \{1, 2, \ldots, n\}$ for positive integers $n$. The given intensity function can be considered as a nonnegative integer matrix $A = (a_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$. A segment is a matrix that corresponds to a leaf position of an MLC. This is made precise in the following definition.

**Definition 1.** *A segment is an $m \times n$–matrix $S = (s_{i,j})$, such that there exist integers $l_i,\ r_i \quad (i \in [m])$ with the following properties:*

$$1 \le l_i \le r_i + 1 \le n + 1 \qquad (i \in [m]), \tag{1}$$

$$s_{i,j} = \begin{cases} 1 \text{ if } l_i \le j \le r_i \\ 0 \text{ otherwise} \end{cases} \qquad (i \in [m],\ j \in [n]). \tag{2}$$

So $l_i - 1$ and $r_i + 1$ have to be interpreted as the positions of the $i$–th left and right leaf, respectively. A *segmentation* of $A$ is a representation of $A$ as a sum of segments, i.e.

$$A = \sum_{i=1}^{k} u_i S_i$$

with segments $S_i$ and positive numbers $u_i$ $(i = 1, 2, \ldots, k)$. Every segmentation corresponds in the obvious way to a treatment plan realizing the given intensity matrix $A$. Our goal is to minimize the total number of monitor units (TNMU) and the number of segments (NS), which in the segmentation correspond to $\sum_{i=1}^{k} u_i$ and $k$, respectively. First of all, observe that in general it is not possible to minimize both of these parameters simultaneously. For the segmentation problem with ICC this was shown by an example in [9]. Here we give an example that is independent of the ICC, that means the simultaneous minimization is not possible, no matter if the ICC is taken into account or not. The matrix $\left(\begin{smallmatrix} 2 & 6 & 3 \\ 4 & 5 & 6 \end{smallmatrix}\right)$ has a segmentation with 6 monitor units

$$\left(\begin{smallmatrix} 2 & 6 & 3 \\ 4 & 5 & 6 \end{smallmatrix}\right) = 3\left(\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{smallmatrix}\right) + 1\left(\begin{smallmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{smallmatrix}\right) + 1\left(\begin{smallmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{smallmatrix}\right) + 1\left(\begin{smallmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right),$$

and this cannot be done with 3 segments. However, if we allow to use 7 monitor units, 3 segments are sufficient:

$$\left(\begin{smallmatrix} 2 & 6 & 3 \\ 4 & 5 & 6 \end{smallmatrix}\right) = 4\left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{smallmatrix}\right) + 2\left(\begin{smallmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{smallmatrix}\right) + 1\left(\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{smallmatrix}\right).$$

But it will be an easy consequence of Lemma 1 below, that for a single row $A$, i.e. in the case $m = 1$, both parameters can be minimized simultaneously. By $\mathcal{F}$ we denote the subsets of $V := [m] \times [n]$ that correspond to segments, that is

$$\mathcal{F} = \{T \subseteq V : \text{ There exists a segment } S \text{ with } ((i,j) \in T \iff s_{i,j} = 1)\}.$$

Now an LP–relaxation of the TNMU–minimization problem is given by:

$$(P)\begin{cases} \sum\limits_{T\in\mathcal{F}} f(T) \;\to\; \min & \text{subject to} \\ f(T)\geq 0 & \forall T\in\mathcal{F}, \\ \sum\limits_{T\in\mathcal{F}:(i,j)\in T} f(T)=a_{i,j} & \forall (i,j)\in V. \end{cases}$$

In order to show that a certain algorithm is optimal with respect to the TNMU one can use the dual of this program:

$$(D)\begin{cases} \sum\limits_{(i,j)\in V} a_{i,j}g(i,j) \;\to\; \max & \text{subject to} \\ \sum\limits_{(i,j)\in T} g(i,j)\leq 1 & \forall T\in\mathcal{F}. \end{cases}$$

Following [7] one can define the functions $g_s$ $(1\leq s\leq m)$ by

$$g_s(i,j)=\begin{cases} 1 \text{ if } i=s,\ a_{i,j}\geq a_{i,j-1} \text{ and } a_{i,j+1}<a_{i,j} \\ -1 \text{ if } i=s,\ a_{i,j}<a_{i,j-1} \text{ and } a_{i,j+1}\geq a_{i,j} \\ 0 \text{ otherwise,} \end{cases}$$

where we put $a_{i,0}=a_{i,n+1}=0$ for all $i$. It is easy to see that the $g_s$ are feasible for $(D)$ and that

$$\sum_{(i,j)\in V} a_{i,j}g_s(i,j)=\sum_{j=1}^{n}\max\{0,a_{s,j}-a_{s,j-1}\}.$$

Thus

$$\max_{1\leq i\leq m}\sum_{j=1}^{n}\max\{0,a_{i,j}-a_{i,j-1}\}$$

is a lower bound for the TNMU of a segmentation, and in order to show the optimality of a given algorithm it is sufficient to show that it realizes this bound.

In order to include the ICC into our model we have to add the following conditions to the definition of a segment:

$$\text{(ICC)}\qquad l_i\leq r_{i+1}+1,\qquad r_i\geq l_{i+1}-1\qquad (i\in[m-1]). \tag{3}$$

### 2.2 NS–Minimization is NP–Complete

According to [1] R.E. Burkard showed that the NS–minimization is NP–complete for $m\geq 2$. Here we describe a formulation of the $(m=1)$–case which was found independently by the author and the authors of [1], and yields the NP–completeness in this case. The intensity map is an $n$–dimensional row vector

$\mathbf{a} = (a_1, a_2, \ldots, a_n)$ with nonnegative integer entries, and a segment is an $n$–dimensional $(0,1)-$vector $\mathbf{s} = \mathbf{s}(l,r)$ with

$$s_i(l,r) = \begin{cases} 1 \text{ if } l \le i \le r \\ 0 \text{ otherwise,} \end{cases}$$

for some integers $l$ and $r$. Now the decision version of the NS–minimization problem is the following: given a row vector $\mathbf{a} = (a_1, \ldots, a_n)$ and an integer $N$, is there a segmentation of $\mathbf{a}$ with at most $N$ segments? In order to prove the NP–completeness of this problem we give a network flow formulation of the segmentation problem. We define the digraph $\Gamma = (V, E)$, where

$$V = [n+1],$$
$$E = \{(i,j) \ : \ 1 \le i < j \le n+1\}$$

Now a flow $y_{l,r+1} > 0$ on an arc $(l, r+1) \in E$ can be associated with the vector $y_{l,r+1}\mathbf{s}(l,r)$. Then a segmentation of $\mathbf{a}$ corresponds to a flow on $\Gamma$, such that the net flow at vertex $i$ is $-d_i$, where $d_i := a_i - a_{i-1}$ for $i = 1, 2, \ldots, n$ and $d_{n+1} := -a_n$, i.e.

$$\sum_{j=1}^{i-1} y_{j,i} - \sum_{j=i+1}^{n+1} y_{i,j} = -d_i \qquad (i \in [n+1]).$$

In order to count the segments in the considered segmentation we introduce the $(0,1)-$variables $x_{i,j}$ for $1 \le i < j \le n+1$, where $x_{l,r+1} = 1$ iff the segment $\mathbf{s}(l,r)$ has nonzero coefficient. So we can write the problem of finding a segmentation with minimal number of segments as the following fixed charge network flow problem:

$$\sum_{1 \le i < j \le n+1} x_{i,j} \to \min \qquad \text{subject to} \tag{4}$$

$$y_{i,j} \le L x_{i,j} \qquad (1 \le i < j \le n+1) \tag{5}$$

$$\sum_{j=1}^{i-1} y_{j,i} - \sum_{j=i+1}^{n+1} y_{i,j} = -d_i \qquad (i \in [n+1]) \tag{6}$$

$$x_{i,j} \in \{0,1\},\ y_{i,j} \in \mathcal{R}_+ \qquad (1 \le i < j \le n+1), \tag{7}$$

where $L$ is an upper bound for the coefficients in the segmentation, e.g. the maximum entry of $A$.

**Lemma 1.** *There is an optimal solution to* (4)–(7) *with* $y_{i,j} = x_{i,j} = 0$ *for all* $(i,j)$ *with* $d_i \le 0$ *or* $d_j \ge 0$.

*Proof.* Let $(\mathbf{x}, \mathbf{y})$ be an optimal solution and assume there is positive flow $y_{i,j}$ on some arc $(i,j)$ with $d_i \le 0$ or $d_j \ge 0$. Let

$$\phi(\mathbf{x}, \mathbf{y}) = |\{(i,j) \in E \ : \ y_{i,j} > 0 \text{ and } d_i \le 0 \text{ or } d_j \ge 0\}|.$$

We construct another optimal solution $(\mathbf{x}', \mathbf{y}')$ with $\phi(\mathbf{x}', \mathbf{y}') < \phi(\mathbf{x}, \mathbf{y})$. Repeating this step if necessary, we finally obtain a solution $(\mathbf{x}'', \mathbf{y}'')$ with $\phi(\mathbf{x}'', \mathbf{y}'') = 0$, and thus $(\mathbf{x}'', \mathbf{y}'')$ is the required solution. Let $(i_1, i_2, \ldots, i_t)$ be a path with the following properties:

1. $y_{i_k,i_{k+1}} > 0$ and ($d_{i_k} \leq 0$ or $d_{i_{k+1}} \geq 0$) for $1 \leq k \leq t-1$.
2. For $i < i_1$, $y_{i,i_1} > 0$ implies ($d_i > 0$ and $d_{i_1} < 0$).
3. For $i > i_t$, $y_{i_t,i} > 0$ implies ($d_{i_t} > 0$ and $d_i < 0$).

Such a path with $t \geq 2$ exists by assumption.

**Case 1:** $d_{i_1} > 0$ and $d_{i_t} < 0$.

Let $\alpha = \min\{y_{i_k,i_{k+1}} \ : \ 1 \leq k \leq t-1\}$, and put

$$\begin{aligned} y'_{i_k,i_{k+1}} &= y_{i_k,i_{k+1}} - \alpha \qquad (1 \leq k \leq t-1), \\ x'_{i_k,i_{k+1}} &= \begin{cases} 1 \text{ if } y'_{i_k,i_{k+1}} > 0, \\ 0 \text{ if } y'_{i_k,i_{k+1}} = 0, \end{cases} \\ x'_{i_1,i_t} &= 1, \\ y'_{i_1,i_t} &= y_{i_1,i_t} + \alpha \end{aligned}$$

and $x'_{i,j} = x_{i,j}$, $y'_{i,j} = y_{i,j}$ for all the remaining $(i, j)$. Obviously, the transition from $(\mathbf{x}, \mathbf{y})$ to $(\mathbf{x}', \mathbf{y}')$ preserves the net flows at the vertices, hence $(\mathbf{x}', \mathbf{y}')$ is a feasible solution. Now for at least one $k \in [t-1]$, $x_{i_k,i_{k+1}} = 1$ and $x'_{i_k,i_{k+1}} = 0$ and since the only $x$–component which might change from 0 to 1 is $x_{i_1,i_t}$, we obtain

$$\sum_{1 \leq i < j \leq n+1} x'_{i,j} \leq \sum_{1 \leq i < j \leq n+1} x_{i,j},$$

hence $(\mathbf{x}', \mathbf{y}')$ is also optimal. Finally, for a $k \in [t-1]$ with $y_{i_k,i_{k+1}} = \alpha$, $y'_{i_k,i_{k+1}} = 0$. And since $(i_1, i_t)$ is the only arc with increasing flow and does not contribute to $\phi$,

$$\phi(\mathbf{x}', \mathbf{y}') < \phi(\mathbf{x}, \mathbf{y}).$$

**Case 2:** $d_{i_1} > 0$ and $d_{i_t} \geq 0$.

Since the net flow in $i_t$ is nonpositive there is some $i_{t+1} > i_t$ with $y_{i_t,i_{t+1}} > 0$ and by condition 3, $d_{i_{t+1}} < 0$. Now we make the same construction as in Case 1 with the path $(i_1, \ldots, i_t, i_{t+1})$.

**Case 3:** $d_{i_1} \leq 0$ and $d_{i_t} < 0$.

Since the net flow on $i_1$ is nonnegative there is some $i_0 < i_1$ with $y_{i_0,i_1} > 0$ and by condition 2, $d_{i_0} > 0$. Now we make the same construction as in Case 1 with the path $(i_0, i_1, \ldots, i_t)$.

**Case 4:** $d_{i_1} \leq 0$ and $d_{i_t} \geq 0$.

As in the Cases 2 and 3, there are $i_0 < i_1$ and $i_{t+1} > i_t$ with $y_{i_0,i_1} > 0$, $y_{i_t,i_{t+1}} > 0$, $d_{i_0} > 0$ and $d_{i_{t+1}} < 0$, and we can make the same construction with the path $(i_0, i_1, \ldots, i_{t+1})$.

So we can restrict our search to the arc set

$$E_0 = \{(i,j) \ : \ 1 \le i < j \le n+1, \ d_i > 0, \ d_j < 0\}$$

and thus we have reduced the problem to a fixed charge transportation problem with sources $S = \{i \ : \ d_i > 0\}$ and sinks $T = \{j \ : \ d_j < 0\}$.

**Example 1.** *The segmentation*

$$\begin{aligned}(\,2\,4\,1\,3\,1\,4\,) = 2(\,1\,1\,0\,0\,0\,0\,) + (\,0\,1\,0\,0\,0\,0\,) + (\,0\,1\,1\,1\,1\,1\,) + 2(\,0\,0\,0\,1\,0\,0\,) \\ + 3(\,0\,0\,0\,0\,0\,1\,)\end{aligned} \tag{8}$$

*corresponds to the flow in Fig. 6.*

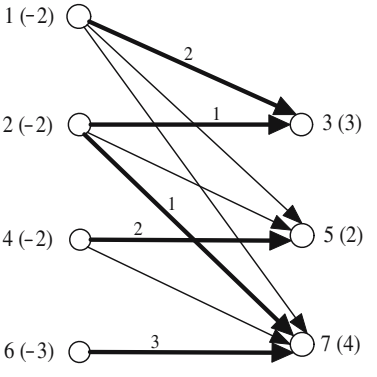


**Fig. 6.** The flow corresponding to the segmentation (8). The numbers in parentheses are the net flows at the vertices.

**Remark 1.** *Observe that in an optimal flow satisfying the conditions of Lemma 1 the sum of the flows over all arcs, i.e. the TNMU of the corresponding segmentation, equals the sum of the net flows at the sinks. Clearly, this is a lower bound for the TNMU, hence the corresponding segmentation is also optimal with respect to the TNMU. So for $m = 1$, in contrast to the general case, the TNMU and the NS can be minimized simultaneously.*

Using this transportation formulation we can now prove the NP–completeness.

**Theorem 1.** *The NS–minimization problem is NP–complete.*

*Proof.* The problem is obviously in NP, and to show the NP–hardness we reduce the $0-1-$knapsack problem: given positive integers $c_1, \ldots, c_{n-1}, K$, is there a subset $I \subseteq \{1, \ldots, n-1\}$ with $\sum_{i \in I} c_i = K$ ? We put

$$a_i = \sum_{j=1}^{i} c_j \quad (i = 1, 2, \ldots, n-1) \text{ and } a_n = K,$$

and claim that the answer to the $0-1-$knapsack problem $(c_1, c_2, \ldots, c_{n-1}, K)$ is yes iff the answer to the NS-minimization problem $(a_1, a_2, \ldots, a_n, n-1)$ is yes. We distinguish 3 cases.

**Case 1:** $K > \sum_{i=1}^{n-1} c_i = a_{n-1}$.

The answer to the knapsack problem $(c_1, c_2, \ldots, c_{n-1}, K)$ is no, and in the transportation problem corresponding to the segmentation we have $n$ sources and 1 sink, so we need $n$ edges with nonzero flow and hence the answer to the NS–minimization problem $(a_1, \ldots, a_n, n-1)$ is also no.

**Case 2:** $K = \sum_{i=1}^{n-1} c_i = a_{n-1}$.

The answer to the knapsack problem $(c_1, c_2, \ldots, c_{n-1}, K)$ is yes, and in the transportation problem corresponding to the segmentation we have $n-1$ sources and 1 sink, so $n-1$ edges with nonzero flow are sufficient and the answer to the NS–minimization problem $(a_1, \ldots, a_n, n-1)$ is also yes.

**Case 3:** $K < \sum_{i=1}^{n-1} c_i = a_{n-1}$.

In the transportation problem we have $n-1$ sources with supplies $c_1, c_2, \ldots, c_{n-1}$ and 2 sinks with demands $K$ and $a_{n-1} - K$. At every source there must be at least one outgoing arc with nonzero flow. So altogether there are at least $n-1$ edges with nonzero flow and $n-1$ arcs are sufficient iff at every source there is exactly one outgoing arc with nonzero flow. But this is equivalent to the existence of a subset $I \subset \{1, 2, \ldots, n-1\}$ with $\sum_{i \in I} c_i = K$.

Due to this result it is reasonable to look for a good approximative algorithm for the NS–minimization.

### 2.3 The Algorithm of Galvin, Chen and Smith

In [8] the authors propose a heuristic algorithm which aims at finding a segmentation with a small NS. As several of the algorithms below it works according to the following general strategy: depending on the given matrix $A$ a coefficient $u > 0$ and a number of segments $S_1, S_2, \ldots, S_t$ are determined such that $A' = A - u(S_1 + S_2 + \cdots + S_t)$ is still nonnegative, and then the algorithm is iterated with $A'$ instead of $A$. It is clear that this always yields a segmentation of $A$. Since in a number of algorithms the maximal entry of $A$ is a parameter, it is convenient to give it a name. So let $L$ denote the maximal entry of the considered matrix $A$ for the rest of this thesis. The algorithm from [8] works as follows

1. In a preliminary step eliminate the background intensity, that is put $A := A - uJ$, where $u$ is the smallest entry of $A$ and $J$ is the $m \times n$ all–one matrix.
2. Let $u$ be the smallest integer such that $\frac{1}{2}u(u+1) \geq L$.
3. Mark all the entries of $A$ which are greater or equal to $u$, i.e. which can be irradiated with $u$ MU.

4. Determine a sequence of segments whose sum is a $(0,1)-$matrix $S$ which has a 1 at position $(i,j)$ iff the entry $(i,j)$ is marked.
5. Put $A := A - uS$, $u := u - 1$ and continue with step 3.

If we do not consider interleaf collision constraints the rows can be treated independently and step 4 can be realized as follows. In each row $i$, we find the maximal intervals of entries which are greater than or equal to $u$. These intervals can be described by their left and right boundaries, that is by numbers $l_{i,1}, \ldots, l_{i,t(i)}$ and $r_{i,1}, \ldots, r_{i,t(i)}$, such that

$$\begin{aligned} & 1 \le l_{i,1}, \quad r_{i,t(i)} \le n, \\ & l_{i,k} \le r_{i,k} && (1 \le k \le t(i)), \\ & r_{i,k} < l_{i,k+1} - 1 && (1 \le k \le t(i) - 1), \\ & a_{i,j} \begin{cases} \ge u \text{ if } l_{i,k} \le j \le r_{i,k} \text{ for some } k, \\ < u \text{ otherwise.} \end{cases} \end{aligned}$$

With the additional convention that $t(i) = 0$ for rows without entries greater or equal to $u$ and $l_{i,k} = n + 1$, $r_{i,k} = n$ for $k > t(i)$, the whole procedure is summarized in Algorithm 1.

**Algorithm 1.** Galvin, Chen and Smith

$u := \min\{a_{i,j} \ : \ 1 \le i \le m, 1 \le j \le n\}$
$A := A - uJ$
$u := \min\left\{k \ : \ \frac{1}{2}k(k+1) \ge L\right\}$
**while** $A \ne 0$ **do**
  **for** $i = 1$ to $m$ **do**
    determine $l_{i,1}, l_{i,2}, \ldots, l_{i,t(i)}, r_{i,1}, r_{i,2}, \ldots, r_{i,t(i)}$
    $t := \max\limits_{1 \le i \le m} t(i)$
    for $1 \le k \le t$ let $S_k$ be the segment determined by the $l_{i,k}, r_{i,k}$
    $S := \sum\limits_{k=1}^{t} S_k$
  $A := A - uS$; $u := u - 1$

**Example 2.** *We will illustrate some of the described algorithms by construction of a segmentation for the benchmark matrix (from [4,14])*

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

*For* $u = 3$ *we obtain* $S = \begin{pmatrix} 1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1 \\ 0\,1\,0\,0\,0\,1 \\ 1\,1\,1\,0\,1\,1 \end{pmatrix}$ *with residual matrix* $\begin{pmatrix} 1\,2\,0\,1\,1\,2 \\ 2\,1\,1\,0\,1\,1 \\ 2\,0\,2\,1\,2\,1 \\ 2\,0\,0\,2\,2\,0 \end{pmatrix}$.

*For* $u = 2$ *we obtain* $S = \begin{pmatrix} 0\,1\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0\,0 \\ 1\,0\,1\,0\,1\,0 \\ 1\,0\,0\,1\,1\,0 \end{pmatrix}$ *with residual matrix* $\begin{pmatrix} 1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0 \end{pmatrix}$.

*So the total segmentation is*

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

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

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

### 2.4 The Algorithm of Bortfeld *et al.*

The first segmentation algorithm which is optimal with respect to the TNMU was introduced in [3]. Again we neglect additional constraints like ICC, and so the rows can be treated independently. Let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ be a row of $A$. In addition we put $a_0 = a_{n+1} = 0$ and $L = \max\limits_{1\le i\le n} a_i$. Now, for $1 \le k \le L$, we determine the index sets

$$P_k = \{i \in [n] \ : \ a_{i-1} < k \le a_i\}, \quad Q_k = \{i \in [n] \ : \ a_i \ge k > a_{i+1}\},$$

and put $P = \bigcup_k P_k$, $Q = \bigcup_k Q_k$ where the unions have to be understood in the multiset sense. Observe that, for each $k$, $|P_k| = |Q_k|$, and that

$$c := \sum_{k=1}^{L} |P_k| = \sum_{i=1}^{n} \max\{0, a_i - a_{i-1}\}.$$

If $P = (p_1, p_2, \ldots, p_c)$ and $Q = (q_1, q_2, \ldots, q_c)$ are ordered such that $q_i \ge p_i$ for all $i$, then we can write $\mathbf{a}$ as a sum of $c$ segments $\mathbf{b^{(1)}}, \ldots, \mathbf{b^{(c)}}$ defined by

$$b_j^{(i)} = \begin{cases} 1 \text{ if } p_i \le j \le q_i, \\ 0 \text{ otherwise.} \end{cases}$$

In [3] two variants of the segmentation algorithm are deduced from this. For the sweep technique $P$ and $Q$ are ordered independently by magnitude, i.e.

$$p_1 \le p_2 \le \ldots \le p_c, \quad q_1 \le q_2 \le \ldots \le q_c.$$

For the close–in technique the $P_k$ and the $Q_k$ are ordered by magnitude, each element of a $P_k$ is paired with the corresponding element of $Q_k$ and the resulting pairs $(p, q)$ are ordered by the magnitude of the first component.

Combining the segmentations of the single rows one can produce segmentations for general intensity matrices.

**Example 3.** *For the second row of* $A = \begin{pmatrix} 4&5&0&1&4&5\\ 2&4&1&3&1&4\\ 2&3&2&1&2&4\\ 5&3&3&2&5&3 \end{pmatrix}$ *we obtain*

$$P_1 = \{1\},\ P_2 = \{1,4,6\},\ P_3 = \{2,4,6\},\ P_4 = \{2,6\},$$
$$Q_1 = \{6\},\ Q_2 = Q_3 = \{2,4,6\},\ Q_4 = \{2,6\}$$

*and the sequence of pairs* $(p,q)$ *using the sweep technique is*

$$(1,2),(1,2),(2,2),(2,4),(4,4),(4,6),(6,6),(6,6),(6,6),$$

*while the close–in technique yields*

$$(1,6),(1,2),(2,2),(2,2),(4,4),(4,4),(6,6),(6,6),(6,6).$$

*The corresponding segmentations of the whole matrix are*

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

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

$$+ 1\begin{pmatrix} 0&0&0&0&1&1\\ 0&0&0&0&0&1\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{pmatrix} + 1\begin{pmatrix} 0&0&0&0&0&1\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{pmatrix} \qquad \textit{and}$$

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

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

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

### 2.5 The Algorithm of Engel

Engel proposes an algorithm which is optimal with respect to the TNMU and almost optimal with respect to the NS. The theoretical result underlying that algorithm is

**Theorem 2 ([7]).** *The minimal TNMU of a segmentation of* $A$ *equals*

$$c(A) := \max_{1\le i\le m} c_i(A),\ \textit{where} \tag{9}$$

$$c_i(A) := \sum_{j=1}^{n} \max\{0, a_{i,j} - a_{i,j-1}\}. \tag{10}$$

*(Recall that* $a_{i,0} = a_{i,n+1} = 0$ *for all* $i$*.)*

Using this terminology the reason for the optimality of the algorithm of Bortfeld *et al.* can be summarized as follows: if $A'$ is the residual matrix after the first step, then by construction

$$c_i(A') = c_i(A) - 1, \tag{11}$$

for all $i$ with $c_i(A) > 0$, in particular $c(A') = c(A) - 1$, and thus after $c(A)$ steps $A$ is reduced to the zero matrix. The drawback of this method is that *a priori* all the segments have coefficient 1, and thus the NS is rather large. Obviously, if the algorithm yields the same segment $S$ in $u$ different steps, these can be combined to obtain one segment with coefficient $u$. In view of (11) this amounts to the search for a pair $(u, S)$ of a positive integer $u$ and a segment $S$ such that $A - uS$ is still nonnegative and

$$c_i(A - uS) = c_i(A) - u,$$

for all $i$ with $c_i(A) > 0$. But this condition is unnecessary strong: we only need

$$c(A - uS) = c(A) - u, \tag{12}$$

i.e.

$$c_i(A - uS) \le c(A) - u \tag{13}$$

for all $i$. For the choice of the coefficient $u$ it is a suggestive strategy to take the maximal $u$ for which there exists a segment $S$ such that $A - uS$ is nonnegative and (12) is true. Let $u_{max}$ be this maximal possible value for $u$. According to [7], $u_{max}$ can be determined as follows. We put

$$d_{i,j} = a_{i,j} - a_{i,j-1} \qquad (1 \le i \le m,\ 1 \le j \le n+1)$$

and consider some segment $S$, given by $l_1, \ldots, l_m$ and $r_1, \ldots, r_m$. One can prove (see [7]) that it is no restriction to assume that, for all $i$, either $l_i = r_i + 1$ or $(d_{i,l_i} > 0$ and $d_{i,r_i+1} < 0)$, and that under these assumptions $c_i(A - uS) \le c(A) - u$ is equivalent to $u \le v_i(l_i, r_i)$, where

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

with $g_i(A) := c(A) - c_i(A)$. 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

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

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

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

Now we put, for $1 \leq i \leq m$ and $(l,r) \in \mathcal{I}_i$,

$$\hat{u}_i(l,r) = \min\{v_i(l,r), w_i(l,r)\},$$

and for $1 \leq i \leq m$,

$$\tilde{u}_i = \max_{(l,r)\in\mathcal{I}_i} \hat{u}_i(l,r). \tag{14}$$

Then

$$u_{max} = \min_{1\leq i\leq m} \tilde{u}_i. \tag{15}$$

In order to construct a segment $S$ such that, for $u = u_{max}$, $A - uS$ is nonnegative and (12) is true, we just have to find, for every $i \in [m]$, a pair $(l_i, r_i) \in \mathcal{I}_i$ with

$$\hat{u}_i(l_i, r_i) \geq u_{max}.$$

A trivial way of doing this is to take a pair $(l_i, r_i)$ where the maximum in (14) 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 NS of the segmentation (see [7]). But there are better constructions for $S$ after the determination of $u_{max}$. We describe a construction of $S$ which, on randomly generated test matrices, yields slightly better results than the one given in [7]. We put

$$q(A) = |\{(i,j) \in [m] \times [n] \ : \ d_{i,j} \neq 0\}|, \tag{16}$$

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

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

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

**Example 4.** *For the benchmark matrix the algorithm yields*

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

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

### 2.6 The Algorithm of Kalinowski

In [11] the approach of [7] is generalized to include the ICC. For this purpose we reformulate Theorem 2 as follows: let $\overrightarrow{G_0} = (V, E_0)$ be a digraph with

$$\begin{aligned} V &= [m] \times [n+1] \cup \{s, t\}, \\ E_0 &= E_1 \cup E_2 \quad \text{where} \\ E_1 &= \{(s, (i, 1)) \ : \ i \in [m]\} \cup \{((i, n+1), t) \ : \ i \in [m]\}, \\ E_2 &= \{((i, j), (i, j+1)) \ : \ i \in [m], j \in [n-1]\}, \end{aligned}$$

and define a weight function $\delta$ on $\overrightarrow{G_0}$ (depending on $A$) by

$$\begin{aligned} \delta(s, (i, 1)) &= a_{i,1} && i \in [m], \\ \delta((i, n+1), t) &= 0 && i \in [m], \\ \delta((i, j), (i, j+1)) &= \max\{0, d_{i,j+1}\} && i \in [m], j \in [n]. \end{aligned}$$

An equivalent formulation of Theorem 2 is

**Theorem 2′ 1.** *The minimal TNMU of a segmentation of $A$ (without ICC) equals the maximal weight of an $(s,t)$–path in $\overrightarrow{G_0}$ with respect to $A$.*

The vertices $(i, n+1)$ $(i \in [m])$ are not necessary here, since the arcs $((i, n), (i, n+1))$ have weight 0 anyway. But we have added them to avoid case distinctions below. In order to model the ICC in the graph we have to add some additional arcs. We define the digraph $\overrightarrow{G} = (V, E)$ with $E = E_0 \cup E_3 \cup E_4$, where

$$\begin{aligned} E_3 &= \{((i, j), (i+1, j)) \ : \ 1 \le i \le m-1, \ 1 \le j \le n-1\}, \\ E_4 &= \{((i, j), (i-1, j)) \ : \ 2 \le i \le m, \ 1 \le j \le n-1\}, \end{aligned}$$

and we extend $\delta$ to $E$ by

$$\begin{aligned} \delta((i, j), (i+1, j)) &= -a_{i,j} && 1 \le i \le m-1, \ 1 \le j \le n-1, \\ \delta((i, j), (i-1, j)) &= -a_{i,j} && 2 \le i \le m, \ 1 \le j \le n-1. \end{aligned}$$

In Figure 7 the construction is illustrated for the matrix

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

The main result of [11] is

**Theorem 3.** *The minimal TNMU of a segmentation of $A$ with ICC equals the maximal weight of an $(s,t)$–path in $\overrightarrow{G}$ with respect to $A$.*

We denote this maximal weight by $c(A)$:

$$c(A) = \max\{\delta(P) \ : \ P \text{ is an } (s,t) - \text{path in } \overrightarrow{G}\}.$$

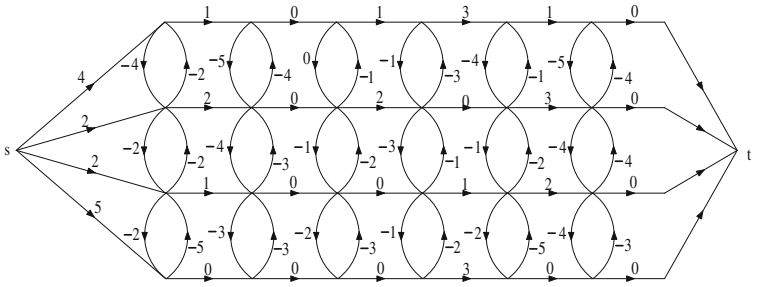


**Fig. 7.** The weighted digraph corresponding to the benchmark matrix

The proof of the theorem consists of two parts. First with an $(s,t)$–path $P$ in $\overrightarrow{G}$ we associate a function $g_P : [m] \times [n] \to \{0, 1, -1\}$ such that $g$ is dually feasible for the TNMU–minimization, and for some $(s,t)$–path $P$ with $\delta(P) = c(A)$ we have

$$\sum_{(i,j)\in[m]\times[n]} a_{i,j}g(i,j) = \delta(P).$$

From this by duality we conclude that the TNMU of a segmentation is greater or equal to $c(A)$. The function $g_P$ that does the job is

$$g_P(i,j) = \begin{cases} 1 \text{ if } \{(i,j),(i,j+1),(i,j+2)\} \subset P,\ d_{i,j} \geq 0,\ d_{i,j+1} < 0, \\ 1 \text{ if } \{(i,j),(i,j+1)\} \subset P,\ (i,j+2) \notin P,\ d_{i,j} \geq 0, \\ -1 \text{ if } \{(i,j),(i,j+1),(i,j+2)\} \subset P,\ d_{i,j} < 0,\ d_{i,j+1} \geq 0, \\ -1 \text{ if } (i,j) \in P,\ (i,j+1) \notin P, \\ -1 \text{ if } \{(i-1,j),(i,j),(i+1,j)\} \subset P, \\ -1 \text{ if } (i,j) \notin P,\ (i,j+1) \in P,\ d_{i,j+1} \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

The second part of the proof is the construction of a segmentation of $A$ with TNMU $c(A)$. For this we put $A_0 = A$, and in the $i$–th step we construct a segment $S = S_i$ such that $c(A_{i-1} - S) = c(A_{i-1}) - 1$, and put $A_i = A_{i-1} - S$. So for $k = c(A)$ after $k$ steps we obtain $c(A_k) = 0$ and this implies $A_k = 0$. By construction

$$A = \sum_{i=1}^{k} S_i$$

is the required segmentation. In order to describe the construction of $S$ for a fixed $A$ we put

$$\begin{aligned} \alpha_1(i,j) &= \max\{\delta(P) \ :\ P \text{ is an } (s,(i,j)) - \text{path in } \overrightarrow{G}\}, \\ \alpha_2(i,j) &= \max\{\delta(P) \ :\ P \text{ is an } ((i,j),t) - \text{path in } \overrightarrow{G}\}, \\ \alpha(i,j) &= \alpha_1(i,j) + \alpha_2(i,j), \end{aligned}$$

and define two subsets of $[m] \times [n]$,

$$V_1 = \{(i,j) \in [m] \times [n] : \ d_{i,j} \geq 0, \ d_{i,j+1} < 0\},$$
$$V_2 = \{(i,j) \in V_1 \ : \ \alpha(i,j) = c(A), \ \alpha_1(i,j) = a_{i,j}\}.$$

Now the segment $S$ (described by the $l_i$, $r_i$ $(i \in [m])$) can be constructed according to Algorithm 2. The proofs that the $g_P$ have the claimed properties, and that the algorithm yields the required results are quite technical and we omit them here (see [11] for the details).

In order to reduce the NS one can proceed analogous to the algorithm of Engel. In [12] is described a backtracking algorithm, that determines a pair $(u, S)$ of an integer $u$ and a segment $S$ such that $A - uS$ is nonnegative, $c(A - uS) = c(A) - u$ and $u$ is maximal under the condition that a segment with these properties exists.

**Example 5.** *For the benchmark matrix the algorithm from [12] yields*

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

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

**Algorithm 2.** Segment $S(A, V_2)$

```
    for (i,j) ∈ V_2 do
      l_i := max{j' ≤ j : a_{i,j'} = 0}+1
      r_i := j
    for i = 1 to i_1 − 1 do
 5:   l_i := l_{i_1}; r_i := l_i − 1
    for i = i_t + 1 to m do
      l_i := l_{i_t}; r_i := l_i − 1
    for k = 1 to t − 1 do
      if j_k > j_{k+1} then
10:     i := i_k
        while i < i_{k+1} and l_i > r_{i_{k+1}} + 1 do
          i := i + 1
          r_i := l_{i−1} − 1
          l_i := max{j ≤ r_i : a_ij = 0} + 1
15:     for i' = i + 1 to i_{k+1} − 1 do
          r_{i'} := r_{i_{k+1}}; l_{i'} := r_{i'} + 1
      else
        i := i_{k+1}
        while i > i_k and l_i > r_{i_k} + 1 do
20:       i := i − 1
          r_i := l_{i+1} − 1
          l_i := max{j ≤ r_i : a_ij = 0} + 1
        for i' = i_k + 1 to i − 1 do
          r_{i'} := r_{i_k}; l_{i'} := r_{i'} + 1
```

### 2.7 The Algorithm of Xia and Verhey

In [24] another heuristic method for the construction of a segmentation with small NS is proposed. Here again the general principle is to determine a coefficient $u$ and a segment $S$ and to continue with $A - uS$. The coefficient is chosen to be a power of 2 which is close to half of the maximal entry of $A$, precisely

$$u = 2^{\lceil \log L \rceil - 1},$$

where the base of the logarithm is 2. The next step towards the algorithm is the observation that in the two–column case every $(0,1)$–matrix is a segment. So for a two–column matrix $A$ the segment corresponding to the coefficient $u$ may be defined by

$$s_{i,j} = \begin{cases} 1 \text{ if } a_{i,j} \geq u, \\ 0 \text{ otherwise.} \end{cases}$$

In the whole segmentation process every power of 2 between 1 and $2^{\lceil \log L \rceil - 1}$ appears at most once as a coefficient, and thus the NS is at most $\lceil \log L \rceil$. The straightforward generalization of this method to an $n$–column matrix $A$ is to divide $A$ into two–column submatrices, and apply the algorithm to these submatrices. (If $n$ is odd one has to add a dummy $(n+1)$–th column with all entries equal to 0.) This yields $\lceil \frac{n}{2} \rceil \lceil \log L \rceil$ as an upper bound for the NS. Actually, this bound can be replaced by

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} \lceil \log L_k \rceil,$$

where $L_k$ is the maximal entry of the submatrix which consists of the columns $2k-1$ and $2k$. Obviously, it is not very efficient to treat the two–column submatrices independently, because it may be possible to combine some segments for different two–column submatrices to obtain a single segment for the whole matrix. The authors of [24] propose two ways of doing this. The sliding window technique determines the coefficient always according to the leftmost nonzero two–column submatrix, say columns $j$ and $j+1$. Then the leaves are set to obtain the largest possible extension of a leaf setting for columns $j$ and $j+1$. The reducing level technique determines the coefficient according to the maximal entry of the whole matrix $A$ and sets the leaves such that the irradiated area, i.e. the number of 1's in the segment $S$, is maximal.

**Example 6.** *The segmentation of the benchmark matrix using the sliding window technique is*

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

$$+ 2 \begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&0 \end{pmatrix} + 1 \begin{pmatrix} 0&0&0&1&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&1 \end{pmatrix} + 1 \begin{pmatrix} 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix} + 1 \begin{pmatrix} 0&0&0&0&0&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix},$$

*and with the reducing level technique we obtain*

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

$$+ 2\begin{pmatrix} 0&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix} + 1\begin{pmatrix} 0&1&0&0&0&0\\ 0&0&1&1&1&0\\ 0&1&0&0&0&0\\ 1&1&1&0&0&0 \end{pmatrix} + 1\begin{pmatrix} 0&0&0&1&0&0\\ 0&0&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&0&0&1&1 \end{pmatrix} + 1\begin{pmatrix} 0&0&0&0&0&1\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0 \end{pmatrix}.$$

In [17] four variations of the Xia–Verhey–algorithm are compared to the algorithm of Galvin, Chen and Smith and the algorithm of Bortfeld *et al.* The three alternative versions of the Xia–Verhey–algorithm differ in the choice of the coefficient $u$. In the first one it is $u = \lceil \frac{L}{2} \rceil$, in the second one it is the nearest integer to the average of the nonzero entries of $A$, and in the third one it is the median of the nonzero entries of $A$. The essential result of the comparison is that none of these variants is most efficient in all cases (neither for random test matrices nor for clinical examples), but the original version of Xia and Verhey yields on average the smallest NS. The NS can be reduced by a factor of 2 compared to Bortfeld's algorithm at the cost of an increase of the TNMU by about 50%.

### 2.8 The Algorithm of Siochi

In [18] a segmentation algorithm is described which is the basis of the Siemens IMFAST algorithm, as implemented in the commercial IMRT planning system CORVUS. This algorithm minimizes a more realistic measure for the total treatment time which takes into account both the irradiation time and the leaf travel time. For the segmentation $A = \sum_{t=1}^{k} u_t S_t$ we put

$$\tau = \sum_{t=1}^{k} \frac{u_t}{D} + \sum_{t=2}^{k} \max\{T_{VR}, \delta_t\}, \tag{17}$$

where $D$ is the dose rate (in MU/min), $T_{VR}$ is the verification and record (V&R) overhead and

$$\delta_t = \max_{1 \le i \le m} \max \left\{ \frac{|l_i^{(t)} - l_i^{(t-1)}|}{v}, \frac{|r_i^{(t)} - r_i^{(t-1)}|}{v} \right\}$$

is the leaf travel time between segments $t-1$ and $t$. Here $v$ is the leaf speed, and $l_i^{(t-1)}, r_i^{(t-1)}$ $(i \in [m])$ and $l_i^{(t)}, r_i^{(t)}$ $(i \in [m])$ are the parameters of the segments $t-1$ and $t$, respectively. The second sum starts at $t = 2$ since it is assumed that the leaves are set to the first position before the treatment starts. The motivation for taking the maximum in the second sum in (17) instead of the sum of the two values is that we can already start the V&R–cycle in rows where the leaves have already stopped while in others they are still moving, and according to [18] the V&R of the last leaf pair is negligible compared to that of all the others combined. Now the proposed algorithm is a combination of two

parts called *extraction* and *rod pushing*. The extraction part is closely related to the algorithm of Galvin, Chen and Smith, but formulated in a way that allows to include ICC and tongue and groove constraints. The rod pushing part is essentially a reformulation of the algorithm of Bortfeld *et al.* in a geometric setting, but also adjustable to additional constraints. First we describe the basic algorithm without additional constraints, and after that we show how the two parts have to be modified to include the constraints.

### The basic algorithm

**Rod pushing:** The matrix $A$ is visualized as a rectangular $m \times n$–array of rods, where the rod at position $(i, j)$ consists of $a_{i,j}$ cubes. In the beginning all the rods stand on a plane $\pi$. Fig. 8 illustrates this for the matrix

$$\begin{pmatrix} 1 & 4 & 2 \\ 3 & 2 & 1 \\ 1 & 2 & 2 \end{pmatrix}.$$

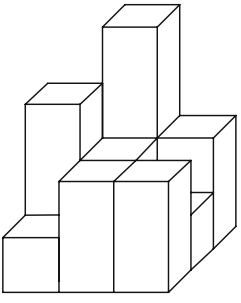

**Fig. 8.** Visualization of an intensity matrix as an array of rods

Now we push some of the rods up in order to achieve a situation where, for all $h > 0$, the positions of the cubes at height $h$ (above $\pi$) can be used to describe a segment. The position of any rod $(i, j)$ is uniquely determined by the height of its lowest cube (the base of the rod), which we call $b(i, j)$. That is, the rod $(i, j)$ occupies the cubes

$$(i, j, b(i, j)), (i, j, b(i, j) + 1), \ldots, (i, j, t(i, j)),$$

where $t(i, j) := b(i, j) + a_{i,j} - 1$ is the height of the highest cube (the top) of the rod. Now the rod pushing procedure can be described as follows:

**for** $i = 1$ to $m$ **do**
  $b(i, 1) := 1$, $t(i, 1) = a_{i,1}$
  **for** $j = 2$ to $n$ **do**
    **if** $a_{i,j} > a_{i,j-1}$ **then**
      $b(i, j) := b(i, j - 1)$; $t(i, j) = b(i, j) + a_{i,j} - 1$
    **else**
      $t(i, j) := t(i, j - 1)$; $b(i, j) = t(i, j) - a_{i,j} + 1$

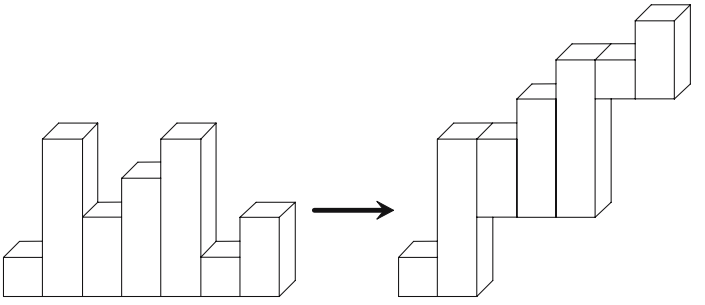

**Fig. 9.** The rod pushing process for one row

Fig. 9 illustrates the rod pushing process corresponding to the segmentation

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

By construction, for every $h$, the cubes at height $h$ describe a segment, the sum of these segments is $A$ and the maximal height of a cube in row $i$ is $\sum_{j=1}^{n} \max\{0, a_{i,j} - a_{i,j-1}\}$. So by Theorem 2 the result is optimal with respect to the TNMU, and one can check that the same segmentation is obtained by the algorithm of Bortfeld *et al.* using the sweep technique.

**Extraction:** This step consists of the determination of a sequence of coefficients $u_1, u_2, \ldots, u_{k_0}$ and corresponding segments $S_1, \ldots, S_{k_0}$ such that the residual matrix

$$A' = A - \sum_{i=1}^{k} u_i S_i$$

is nonnegative. The optimization algorithm does an exhaustive search on a certain set of pairs $(k_0, \mathbf{u})$ where $k_0$ is a positive integer and $\mathbf{u}$ is a $k_0$–tuple $\mathbf{u} = (u_1, \ldots, u_{k_0})$ of positive integers (to be defined below). For each of these pairs, a sequence of segments $(S_1, \ldots, S_{k_0})$ is determined as follows.

$A_0 := A$
**for** $i = 1$ to $k_0$ **do**
determine a segment $S_i$ with respect to the matrix $A_{i-1}$ and the coefficient $u_i$ as in the algorithm of Galvin *et al.*
$A_i := A_{i-1} - u_i S_i$

Then the rod pushing procedure is applied to $A_{k_0}$ and the pairs $(k_0, \mathbf{u})$ are evaluated according to the total treatment time $\tau$ for the segmentation that results from the combination of the two parts. Finally, the $(u_1, \ldots, u_{k_0})$ yielding the smallest $\tau$ is chosen for the segmentation together with the corresponding $(S_1, \ldots, S_{k_0})$ and the subsequent rod pushing segments. The search space is restricted by the following conditions, which have been found to be strong enough to make the search computationally feasible but weak enough to give good solutions ([18]).

1. $u_1 \geq u_2 \geq \ldots \geq u_{k_0}$.
2. $u_1 \leq \max\{\lfloor L/2 \rfloor, \hat{u}\}$, where $\hat{u}$ is the extract value yielding the best result if we extract only one segment before applying the rod pushing, i.e. if $k_0 = 1$.
3. $$\sum_{i=1}^{k_0} u_i \leq \max_{1 \leq i \leq m} \sum_{j=1}^{n} \max\{0, a_{i,j} - a_{i,j-1}\}.$$

**Example 7.** *To illustrate the algorithm we assume that the size of the cells in our benchmark matrix is* $1\,cm \times 1\,cm$*, the leaf speed* $v = 1\,cm/sec$*, the verification and record overhead* $T_{VR} = 2\,sec$ *and the dose rate* $D = 60\,MU/min$*. The best solution if only one segment is extracted is obtained with* $\hat{u} = 3$ *and so the extraction sequences* $(u_1, \ldots, u_{k_0})$ *with* $3 \geq u_1 \geq u_2 \geq \ldots \geq u_{k_0} > 0$ *and* $\sum_{i=1}^{k_0} u_i \leq 10$ *have to be checked. The result is that the segmentation with only one extraction is already the optimal one, namely*

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

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

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

*where we have*

$$\tau = (10 + 2 + 3 + 4 + 3 + 2 + 2 + 2)\,sec = 28\,sec.$$

**Interleaf collision constraint:** The ICC forbids the overlapping of opposite leaves in adjacent rows, that is we must have

$$l_i \leq r_{i+1} + 1 \text{ and } r_i \geq l_{i+1} - 1 \quad (1 \leq i \leq m-1).$$

**Extraction:** In the extraction step with coefficient $u$ we have to find, in each row $i$, an interval of entries greater or equal to $u$. If we fix two adjacent columns $j$ and $j+1$ and require that every nonempty of the intervals intersects at least one of these columns, then the ICC is automatically satisfied, since then $l_i \leq j+1$ and $r_i \geq j$ for all $i$ with $l_i \leq r_i$, and for the zero–rows of the segment it is obvious how to choose $(l_i, r_i)$ with $l_i = r_i + 1$ in order to satisfy the ICC. Now we can do this for all possible pairs of adjacent columns $j$, $j+1$ and finally choose the segment with the largest irradiated area.

**Rod pushing:** A violation of the ICC can only occur, if for some $(i, j)$ we have $b(i, j) > t(i+1, j) + 1$ or $t(i, j) < b(i+1, j) - 1$ (see Fig. 10, where the segment corresponding to height 3 is $\left(\begin{smallmatrix} 1&0&0\\ 0&0&1 \end{smallmatrix}\right)$ and thus violates the ICC).

If this is the case we may push up the rod $(i+1, j)$ (resp. $(i, j)$) (see Algorithm 3 for the details).

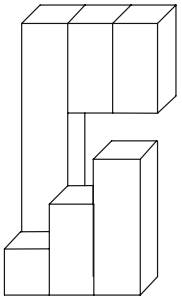

**Fig. 10.** The segment corresponding to height 3 violates the ICC

**Tongue and groove effect:** We have to make sure that it does not occur that in one segment the bixel $(i, j)$ is exposed and the bixel $(i \pm 1, j)$ is covered, while in some later step $(i, j)$ is covered and $(i \pm 1, j)$ is exposed.

**Extraction:** A sufficient condition to avoid the tongue and groove effect between different extract matrices is that, for every extract matrix $S^{(t)} = \left(s_{i,j}^{(t)}\right)$ with coefficient $u_t$, we have

$$a_{i,j} - u_t \geq a_{i+1,j} \quad \text{if } s_{i,j}^{(t)} = 1 \text{ and } s_{i+1,j}^{(t)} = 0 \tag{18}$$

$$a_{i,j} - u_t \geq a_{i-1,j} \quad \text{if } s_{i,j}^{(t)} = 1 \text{ and } s_{i-1,j}^{(t)} = 0. \tag{19}$$

This implies that if in some later step $t'$ the bixel $(i \pm 1, j)$ is exposed (i.e. $s_{i\pm1,j}^{(t')} = 1$) then it is also possible to expose bixel $(i, j)$ and so the tongue and groove underdosage is avoided. In order to achieve the validity of (18) and (19) one proceeds as follows

construct a segment $S = (s_{i,j})$ as above
**repeat**
**for** $(i, j)$ with $s_{i,j} = 1$ and (18) or (19) is violated **do**
$s_{i,j} := 0$
change entries from 1 to 0 so that a segment satisfying the ICC results
**until** no entry has to be changed

**Rod pushing:** In the rod pushing process the tongue and groove effect can be avoided using a modification of the basic method similar to Algorithm 3. Instead of the corrections in lines 14 to 19 and 22 to 27 of this algorithm one has to use

$$\left.\begin{aligned} t(i+1,j) &:= t(i,j) \\ b(i+1,j) &:= t(i+1,j) - a_{i+1,j} + 1 \end{aligned}\right\} \begin{array}{l}\text{if } a_{i,j} < a_{i+1,j} \\ \text{and } t(i,j) > t(i+1,j),\end{array}$$

$$\left.\begin{aligned} b(i,j) &:= b(i+1,j) \\ t(i,j) &:= b(i,j) + a_{i,j} - 1 \end{aligned}\right\} \begin{array}{l}\text{if } a_{i,j} < a_{i+1,j} \\ \text{and } b(i,j) < b(i+1,j),\end{array}$$

$$\left.\begin{aligned} t(i,j) &:= t(i+1,j) \\ b(i,j) &:= t(i,j) - a_{i,j} + 1 \end{aligned}\right\} \begin{array}{l}\text{if } a_{i,j} \geq a_{i+1,j} \\ \text{and } t(i,j) < t(i+1,j),\end{array}$$

**Algorithm 3.** Rod pushing with ICC

```
    for i = 1 to m do
      b(i,1) = 1; t(i,1) = a_{i,1}
    for j = 2 to n do
      for i = 1 to m do
5:      if a_{i,j} > a_{i,j-1} then
          b(i,j) := b(i,j-1); t(i,j) = b(i,j) + a_{i,j} - 1
        else
          t(i,j) := t(i,j-1); b(i,j) = t(i,j) - a_{i,j} + 1
      Choose i_0 ∈ [m] with t(i_0,j) ≥ t(i,j) for all i
10:   for i = i_0 - 1 downto 1 do
        if t(i,j) < b(i+1,j) - 1 then
          t(i,j) := b(i+1,j) - 1; b(i,j) := t(i,j) - a_{i,j} + 1
        if b(i,j) > t(i+1,j) + 1 then
          t(i+1,j) := b(i,j) - 1; b(i+1,j) := t(i+1,j) - a_{i+1,j} + 1
15:   for i = i_0 + 1 to m do
        if t(i,j) < b(i-1,j) - 1 then
          t(i,j) := b(i-1,j) - 1; b(i,j) := t(i,j) - a_{i,j} + 1
        if b(i,j) > t(i-1,j) + 1 then
          t(i-1,j) := b(i,j) - 1; b(i-1,j) := t(i-1,j) - a_{i+1,j} + 1
```

$$\left.\begin{array}{l} b(i+1,j) := b(i,j) \\ t(i+1,j) := b(i+1,j) + a_{i+1,j} - 1 \end{array}\right\} \begin{array}{l} \text{if } a_{i,j} \geq a_{i+1,j} \\ \text{and } b(i,j) > b(i+1,j). \end{array}$$

These corrections make sure that, for two rods that are adjacent along a column, the shorter one has its base above or at the same level as the longer one and has its top below or at the same level as the longer one. Thus the resulting segments also satisfy the ICC.

### 2.9 The Algorithm of Kamath *et al.*

Another segmentation algorithm is described in [13]. Here the authors consider more general constraints which are motivated by the design of some MLCs:

**Minimum separation constraint (MSC):** The distance between the left and the right leaf in every row can not be smaller than a minimum distance $\delta_0 \geq 0$. In our terminology this means

$$r_i - l_i \geq \delta_0 - 1 \qquad (i \in [m]).$$

**Leaf interdigitation constraint (LIC):** The distance between opposite leaves in adjacent rows is at least $\delta_1$ for some $\delta_1 \geq 0$, i.e.

$$r_{i+1} - l_i \geq \delta_1 - 1,\ r_i - l_{i+1} \geq \delta_1 - 1 \qquad (i \in [m-1]).$$

For $\delta_1 = 0$ this is just the ICC.

The proposed algorithm constructs segmentations in which the leaves always move from left to right. So the segmentation can be described by

$$I_L^{(i)}(1) \le I_L^{(i)}(2) \le \ldots \le I_L^{(i)}(n),\ I_R^{(i)}(1) \le I_R^{(i)}(2) \le \ldots \le I_R^{(i)}(n) \qquad (i \in [m]),$$

where $I_L^{(i)}(j)$ and $I_R^{(i)}(j)$ denote the numbers of monitor units that have been delivered when the left and the right leaf, respectively, in row $i$ passes column $j$. These numbers can be translated into segments as follows. Let

$$S^{(t)} = \left(s_{i,j}^{(t)}\right)$$

denote the segment corresponding to the leaf position when the $t-$th monitor unit is delivered. Then

$$s_{i,j}^{(t)} = \begin{cases} 1 \text{ if } I_R^{(i)} < t \le I_L^{(i)}(j) \\ 0 \text{ otherwise.} \end{cases}$$

The condition that must be satisfied in order to generate the matrix $A$ is

$$I_L^{(i)}(j) - I_R^{(i)}(j) = a_{i,j} \qquad (i \in [m],\ j \in [n]).$$

First neglecting the leaf interdigitation constraint a segmentation is build up from segmentations of the single rows as described in Algorithm 4. Observe that

**Algorithm 4.** Basic segmentation

**for** $i = 1$ to $m$ **do**
  $I_L^{(i)}(1) = a_{i,1}$; $I_R^{(i)} = 0$
  **for** $j = 2$ to $n$ **do**
    $I_L^{(i)}(j) = I_L^{(i)}(j-1) + \max\{0, a_{i,j} - a_{i,j-1}\}$
    $I_R^{(i)}(j) = I_R^{(i)}(j-1) + \max\{0, a_{i,j-1} - a_{i,j}\}$

this is another formulation of the rod pushing part of Siochi's algorithm: $I_L^{(i)}(j)$ and $I_R^{(i)}(j)+1$ correspond to $t(i,j)$ and $b(i,j)$, respectively. The essential result on Algorithm 4 is

**Theorem 4.**

1. *Algorithm 4 is optimal with respect to the TNMU even when bidirectional leaf movement is permitted. (Theorem 3 in [13])*
2. *If there exists a segmentation of A satisfying the MSC then the segmentation constructed using Algorithm 4 satisfies the MSC. (Theorem 5 in [13])*

In order to construct a segmentation satisfying the LIC it is proposed to modify the $I_L^{(i)}(j)$, $I_R^{(i)}(j)$ obtained by Algorithm 4 until the LIC is satisfied. If, as a result of Algorithm 4, $I_R^{(i)}(j) > 0$ for some $i \in [m]$, $j \le \delta_1$ there is no segmentation

satisfying the LIC. So w.l.o.g. we may assume $I_R^{(i)}(j) = 0$ for all $i \in [m]$, $1 \le j \le \delta_m$. An LIC–violation occurs iff $I_L^{(k)}(j - \delta_1) < I_R^{(i)}(j)$ for some $i \in [m]$, $j \in [n]$, $k \in \{i+1, i-1\}$. Among all violations we determine one with minimal $j$ and eliminate it by putting

$$I_L^{(k)}(j - \delta_1) := I_R^{(i)}(j),$$

and modifying the $I_L^{(k)}(j')$ $(j' > j - \delta_1)$ appropriately (see Algorithm 5 for the details). The main result on Algorithm 5 is

**Theorem 5 (Theorem 6 in [13]).**

1. *Algorithm 5 terminates.*
2. *If Algorithm 5 terminates with a violation of the MSC, then there is no segmentation satisfying MSC and LIC.*
3. *Otherwise the algorithm yields a segmentation satisfying MSC and LIC and having minimal TNMU under these conditions.*

**Algorithm 5.** : Elimination of LIC violations

**while** The MSC is satisfied and the LIC is violated **do**
  $j_0 := \min\{j \in [n] : \exists i \in [m]$ with $I_L^{(k)}(j-\delta_1) < I_R^{(i)}(j)$ for some $k \in \{i+1, i-1\}\}$
  choose $i$ and $k \in \{i+1, i-1\}$ with $I_L^{(k)}(j_0 - \delta_1) < I_R^{(i)}(j_0)$
  $I_L^{(k)}(j_0 - \delta_1) := I_R^{(i)}(j_0)$
5: $I_R^{(k)}(j_0 - \delta_1) := I_L^{(k)}(j_0 - \delta_1) - a_{k,j_0-\delta_1}$
  **for** $j = j_0 - \delta_1 + 1$ to $n$ **do**
    $I_L^{(k)}(j) := \max\left\{I_L^{(k)}(j), I_L^{(k)}(j-1) + \max\{0, a_{i,j} - a_{i,j-1}\}\right\}$
    $I_R^{(k)}(j) := I_L^{(k)}(j) - a_{k,j}$

If $\delta_1 = 0$ lines 4–9 of Algorithm 5 can be replaced by

$\Delta := I_R^{(i)}(j_0) - I_L^{(k)}(j_0)$
**for** $j = j_0$ to $n$ **do**
  $I_L^{(k)}(j) := I_L^{(k)}(j) + \Delta$
  $I_R^{(k)}(j) := I_R^{(k)}(j) + \Delta$

In this case the algorithm coincides with the ICC–version of Siochi's rod pushing method, no MSC–violation can occur and thus always a TNMU–optimal segmentation is obtained.

### 2.10 The Algorithm of Boland, Hamacher and Lenzen

In [2] is given a network flow formulation of the TNMU–minimization which also includes the ICC. The set of segments is identified with the set of paths from $D$ to $D'$ in the layered 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:

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

Between two vertices $(i, l, r)$ and $(i+1, l', r')$ there is an arc if the corresponding leaf positions are consistent with the ICC, i.e. if $l' \leq r+1$ and $r' \geq l-1$. 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

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

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

There is a bijection between the possible leaf positions and the cycles in $G$. This is illustrated in Fig. 11 which shows two cycles in $G$ for $m = 4$, $n = 2$, corresponding to the segments

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

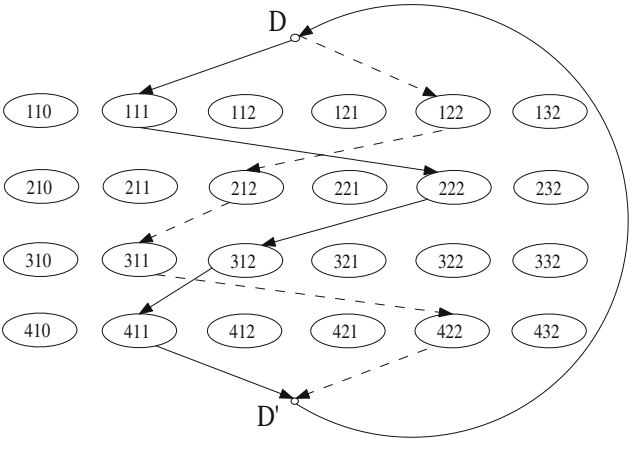


**Fig. 11.** The vertices of $G$ for $m = 4$, $n = 2$ and two cycles

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

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

Then any positive combination of segments defines a circulation $\phi : E \to \mathcal{R}_+$ 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 $(D,(1,1,1),(2,2,2),(3,1,2),(4,1,1),D')$, 2 units of flow on $(D,(1,2,2),(2,1,2),(3,1,1),(4,2,2),D')$ and 5 units of flow on $(D',D)$. The amount of radiation that is released at bixel $(i,j)$ equals the sum of the flows going through the vertices $(i,l,r)$ with $l \le j \le r$, hence the conditions that must be satisfied by the circulation in order to correspond to a segmentation of $A$ are

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

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

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

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

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

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 segmentation of $A$.

$$\begin{aligned}\hat{V} = \{&(i,l,r)^1,(i,l,r)^2 \ :\ 1 \le i \le m,\ 1 \le l \le r+1 \le n+1\} \\ &\cup \{(i,j) \ :\ 1 \le i \le m,\ 0 \le j \le n\} \cup \{D,D'\}.\end{aligned}$$

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

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

Now a segment with parameters $l_i, r_i$ $(i \in [m])$ corresponds to the cycle

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

Figure 12 shows the cycles in $\hat{G}$ corresponding to the cycles in Figure 11.

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

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

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

Now in order to obtain another reformulation of the TNMU–minimization problem one just has to make sure that the flow on the edge $((i, l, r)^1, (i, l-1))$ equals the flow on the edge $((i, r), (i, l, r)^2)$, since both of these correspond to the amount of radiation that is released while $l_i = l$ and $r_i = r$.

**Theorem 6 ([2]).** *The TNMU–minimization problem is equivalent to the network flow problem*

$$\phi(D', D) \;\rightarrow\; \min$$

*subject to $\phi$ 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 $(i, l, r)^{1,2} \in \hat{V}$,*

$$\phi((i, l, r)^1, (i, l-1)) = \phi((i, r), (i, l, r)^2). \tag{24}$$

This formulation is quite close to a pure Min–Cost–Network–Flow problem. But the standard algorithms for this problem type have to be adjusted in order to include the side constraint (24). Doing this one obtains a polynomial time algorithm for the TNMU–minimization with ICC (see [2] and [15]).

**Example 8.** *A segmentation of the benchmark matrix with ICC that is optimal with respect to the TNMU, and thus corresponds to an optimal flow on the appropriate $\hat{G}$ is the one given in Example 5.*

### 2.11 The Algorithm of Baatar and Hamacher

Another TNMU–optimal segmentation algorithm was proposed in [1]. For this a digraph $G = (V, E)$ is constructed as follows: The vertex set consists of $2m$ layers $L_1, R_1, L_2, R_2, \ldots, L_m, R_m$ and two additional vertices $D$ and $D'$. Here, for $i = 1, 2, \ldots m$,

$$L_i = \{(i, 1, 1), (i, 1, 2), \ldots, (i, 1, n+1)\},\ R_i = \{(i, 2, 0), (i, 2, 1), \ldots, (i, 2, n)\}$$

Now arcs between $L_i$ and $R_i$ correspond to possible leaf positions in row $i$, that is

$$E_1 = \{((i, 1, l), (i, 2, r)) \;:\; 1 \le i \le m,\ 1 \le l \le r+1 \le n+1\},$$

and arcs between $R_i$ and $L_{i+1}$ correspond to leaf positions satisfying the ICC, that is

$$E_2 = \{((i, 2, r), (i+1, 1, l)) \;:\; 1 \le i \le m-1,\ 1 \le l \le r+1 \le n+1\},$$

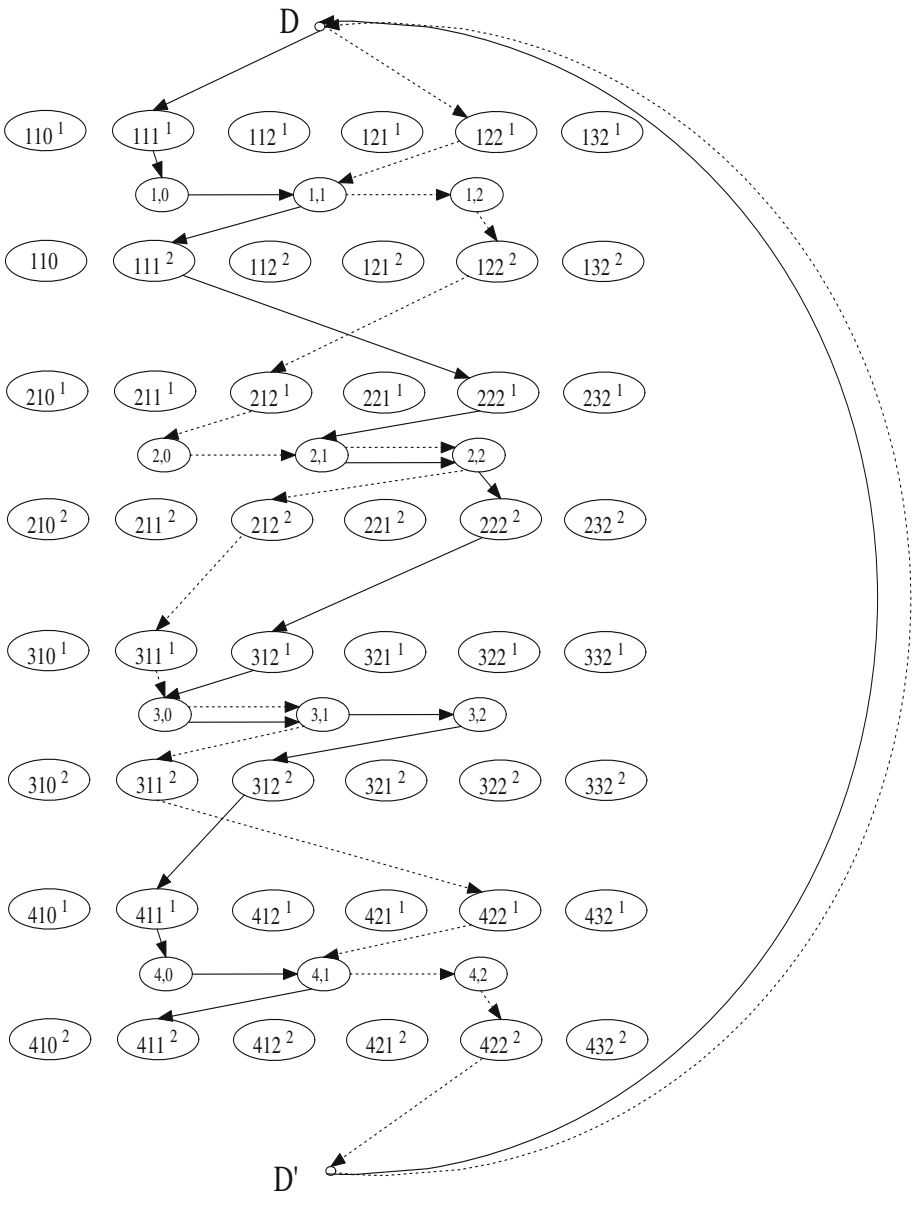


**Fig. 12.** The vertices of $\hat{G}$ for $m = 4$, $n = 2$ and two cycles

and finally all the arcs between $D$ and $L_1$ and between $R_m$ and $D'$ are added, that is

$$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup \{((m, 2, r), D') \ : \ 0 \leq r \leq n\} \cup \{(D', D)\},$$

where

$$E_3 = \{(D,(1,1,l)),\ :\ 1 \le l \le n+1\} \text{ and } E_4 = \{((m,2,r),D')\ :\ 0 \le r \le n\}.$$

Figure 13 shows $G$ for $m = 3$ and $n = 4$.

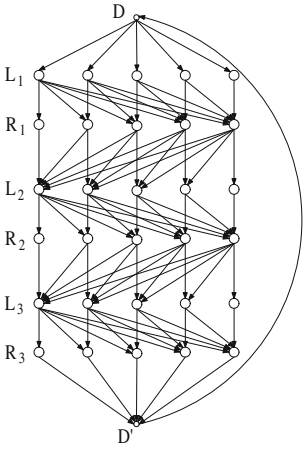


**Fig. 13.** The digraph $G$ for $m = 3$ and $n = 4$

As in the description of Hamacher's algorithm we associate to a segment with parameters $l_1, r_1, l_2, r_2, \ldots, l_m, r_m$ a unit flow on the cycle

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

So every segmentation corresponds to a circulation on $G$ (but not conversely, since the ICC between the left leaf of row $i$ and the right leaf of row $i+1$ is not reflected in the structure of the digraph). In the circulation corresponding to a segmentation the total flow going through vertex $(i,1,l)$ equals the number of monitor units for which the left leaf is positioned at $j-1$, i.e. for which $l_i = j$, and similarly, the total flow going through $(i,2,r)$ equals the number of monitor units for which $r_i = j$. Let $\phi$ be a circulation on $G$, and denote by $\phi(v)$ the total flow going through $v \in V$. In [1] it is shown that in order to find a circulation corresponding to a segmentation with minimal TNMU, we may assume that, for all $i \in [m]$,

$$\phi(i,1,1) = a_{i,1},\ \phi(i,2,n) = a_{i,n} \text{ and } \phi(i,2,0) = \phi(i,1,n+1) = 0.$$

Fixing these values, necessary and sufficient conditions for the $\phi(v)$ to correspond to a segmentation of $A$ are (see [1])

$$\phi(i,1,j) - l^*_{i,j} = \phi(i,2,j) - r^*_{i,j} \geq 0 \qquad (i \in [m], j \in [n]), \tag{25}$$

$$\sum_{j=1}^{k} \phi(i,1,j) \geq \sum_{j=1}^{k-1} \phi(i+1,2,j) \qquad (1 \leq i \leq m-1, 1 \leq k \leq n), \tag{26}$$

$$\sum_{j=1}^{k} \phi(i,1,j) \leq \sum_{j=1}^{k-1} \phi(i-1,2,j) \qquad (2 \leq i \leq m, 1 \leq k \leq n), \tag{27}$$

$$\phi(i,1,l) \in \mathcal{Z} \qquad (1 \leq i \leq m,\ 2 \leq l \leq n), \tag{28}$$

$$\phi(i,2,r) \in \mathcal{Z} \qquad (1 \leq i \leq m,\ 1 \leq r \leq n-1). \tag{29}$$

where

$$l^*_{i,j} = \max\{0, a_{i,j} - a_{i,j-1}\} \quad \text{and} \quad r^*_{i,j} = \max\{0, a_{i,j} - a_{i,j+1}\}.$$

So the task to determine $\phi(v)$ corresponding to a segmentation with minimal TNMU leads to the mixed integer program

$$\left.\begin{array}{ll} T \ \rightarrow \ \min & \text{subject to} \\ T \ = a_{i,1} + \sum\limits_{j=2}^{n} \phi(i,1,j) & \text{and (25)} - \text{(29)}. \end{array}\right\} \tag{30}$$

The constraint matrix of this problem is totally unimodular, as is shown in [1] using the theorem of Ghouila–Houri, and so in order to determine the $\phi(v)$ it is sufficient to solve the LP–relaxation of (30). When the $\phi(v)$ are determined a segmentation with minimal TNMU can be constructed by iteratively extracting unit flows along cycles in $G$ taking in each layer the leftmost vertex with positive throughput. In [1] it is also proposed to reduce the NS by a greedy strategy: an integer program is solved to determine the maximal $u$ such that there is a cycle in $G$ along which flow $u$ can be extracted and the residual network still satisfies (25)–(29).

### 2.12 The Algorithm of Langer, Thai and Papiez

In [14] the authors give a mixed integer linear program formulation of the segmentation problem which can be used to find a segmentation with minimal NS among those with minimal TNMU. The model also allows to include additional constraints. In order to describe segmentations binary variables $l^{(t)}_{i,j}$ and $r^{(t)}_{i,j}$ are introduced for $i \in [m]$, $j \in [n]$ and $t \in [T]$, where $T$ is an upper bound for the TNMU, for instance

$$T = \sum_{(i,j) \in [m] \times [n]} a_{i,j}.$$

$l^{(t)}_{i,j}$ takes value 1 if bixel $(i,j)$ is covered by the left leaf while the $t$–th MU is delivered, but takes value 0 otherwise. Similarly, $r^{(t)}_{i,j}$ takes value 1 if bixel $(i,j)$ is covered by the right leaf. Then the segment delivering the $t$–th MU is given by

$$s_{i,j}^{(t)} = 1 - l_{i,j}^{(t)} - r_{i,j}^{(t)}. \tag{31}$$

Observe that this equation, together with $s_{i,j}^{(t)}, l_{i,j}^{(t)}, r_{i,j}^{(t)} \in \{0,1\}$, also implies that the opposite leaves in row $i$ do not overlap, i.e. that for no $(i,j,t)$ both $l_{i,j}^{(t)}$ and $r_{i,j}^{(t)}$ equal 1. The geometric properties of the leaves are modelled by the following constraints:

$$l_{i,j+1}^{(t)} \le l_{i,j}^{(t)} \qquad (i \in [m],\ j \in [n-1],\ t \in [T]), \tag{32}$$

$$r_{i,j}^{(t)} \le r_{i,j+1}^{(t)} \qquad (i \in [m],\ j \in [n-1],\ t \in [T]). \tag{33}$$

Furthermore, for a segmentation of $A$ we obtain the constraints

$$\sum_{t=1}^{T} s_{i,j}^{(t)} = a_{i,j} \qquad (i \in [m], j \in [n]). \tag{34}$$

The MUs can be counted by introducing new binary variables $z^{(t)}$ $(t \in [T])$, where $z^{(t)}$ takes value 1 iff $s_{i,j}^{(t)} = 1$ for at least one pair $(i,j)$, formally

$$\sum_{(i,j)\in[m]\times[n]} s_{i,j}^{(t)} \le mnz^{(t)} \qquad (t \in [T]). \tag{35}$$

Now the TNMU–minimization problem can be formulated as

$$\sum_{t=1}^{T} z^{(t)} \to \min \quad \text{subject to (31)–(35).} \tag{36}$$

Let $T_0$ denote the optimal value of the objective function, i.e. the minimal TNMU. Observe that the determination of $T_0$ as the solution of (36) can be replaced by the calculation of the minimal TNMU according to Theorem 2. The next step is to find, among all the segmentations with $T_0$ MU, one with minimal NS. For this new binary variables $g^{(t)}$ $(t \in [T_0 - 1])$ are introduced, where $g^{(t)}$ takes value 1 if $s_{i,j}^{(t)} \ne s_{i,j}^{(t+1)}$ for some $(i,j) \in [m] \times [n]$. The global variable $g^{(t)}$ is described by local binary variables

$$\sigma_{i,j}^{(t)} = \alpha_{i,j}^{(t)} + \beta_{i,j}^{(t)}, \tag{37}$$

where $\alpha_{i,j}^{(t)}$ and $\beta_{i,j}^{(t)}$ are binary variables satisfying

$$-\alpha_{i,j}^{(t)} \le s_{i,j}^{(t+1)} - s_{i,j}^{(t)} \le \beta_{i,j}^{(t)}. \tag{38}$$

$g^{(t)}$ can take the value 0 only if all the $\sigma_{i,j}^{(t)}$ are zero, and this yields

$$\sum_{(i,j)\in[m]\times[n]} \sigma_{i,j}^{(t)} \le mng^{(t)} \qquad (t \in [T_0]). \tag{39}$$

So the NS–minimization (with minimal TNMU) is

$$\sum_{t=1}^{T_0} g^{(t)} \;\rightarrow\; \min \text{ subject to (31)–(35),(37)–(39).} \tag{40}$$

The authors of [14] suggest to solve this program by standard branch and bound techniques as implemented in commercial packages as CPLEX. Special restrictions can be included by adding constraints to the program. So the ICC corresponds to

$$l_{i,j}^{(t)} + r_{i+1,j}^{(t)} \leq 1 \quad (i \in [m-1],\ j \in [n],\ t \in [T]), \tag{41}$$

$$r_{i,j}^{(t)} + l_{i+1,j}^{(t)} \leq 1 \quad (i \in [m-1],\ j \in [n],\ t \in [T]). \tag{42}$$

The method for the segmentation problem with ICC described in [14] is to increase the number $T'$ of monitor units step by step, starting with $T' = T_0$, and in each step try to find a feasible solution with $T'$ monitor units. This procedure can be shortened by determining the minimal TNMU for a segmentation with ICC according to Theorem 3 and fixing $T'$ at this value.

The tongue and groove condition is described by

$$-1 \leq s_{i+1,j}^{(t)} - s_{i,j}^{(t)} + s_{i,j}^{(t')} - s_{i+1,j}^{(t')} \leq 1$$
$$(i \in [m-1],\ i \in [n],\ 1 \leq t < t' \leq T). \tag{43}$$

The drawback of this method is that it requires to solve integer programs with a huge number of variables, and so it seems to be applicable only to very small problems.

**Example 9.** *For the benchmark matrix the algorithm yields the same result as Engel's algorithm.*

### 2.13 The Algorithm of Dai and Zhu

The algorithm proposed in [6] searches for a segmentation with a small NS. Again this is done by choosing a segment $S$ and a coefficient $u$ such that $A' = A - uS$ is nonnegative and continuing with $A'$. The criterion for the choice of $u$ and $S$ is the complexity of the residual matrix $A'$, where the complexity of a matrix $A$ is the number of segments necessary for a segmentation of $A$ using some other algorithm. Obviously, the result of this method depends on the algorithm that is used to measure the complexity.

Recall that $L$ denotes the maximum entry of $A$. For $u \in [L]$ we determine in each row $i$ maximal intervals of entries greater than or equal to $u$. As in the section on the algorithm of Galvin *et al.* these can be described by numbers

$l_{i,1,u}, \ldots, l_{i,t(i,u),u}$ and $r_{i,1,u}, \ldots, r_{i,t(i,u),u}$, such that

$$
\begin{aligned}
1 \le l_{i,1,u},\ r_{i,t(i,u),u} \le n & \\
l_{i,k,u} \le r_{i,k,u} & \qquad (1 \le k \le t(i,u)) \\
r_{i,k,u} < l_{i,k+1,u} - 1 & \qquad (1 \le k \le t(i,u)-1) \\
a_{i,j} \begin{cases} \ge u \text{ if } l_{i,k,u} \le j \le r_{i,k,u} \text{ for some } k, \\ < u \text{ otherwise.} \end{cases} &
\end{aligned}
$$

Now for all $u \in [L]$ the complexities of $A - uS$ are computed for all the $\prod_{i=1}^{m} t(i,u)$ possible segments, and among all the tested pairs $(u, S)$ one with minimal complexity of $A - uS$ is chosen. If there are several pairs with minimal complexity of the residual matrix, we choose one with maximal irradiated area, i.e. with maximal number of 1's in $S$. The ICC can easily be included into the algorithm by excluding segments that violate the ICC from the complexity checking. The obvious drawback of this algorithm is the time complexity. The number of segments that have to be checked grows exponentially with the size of the matrix, and so the method becomes infeasible for moderate problem sizes.

**Example 10.** *We consider segmentation without ICC and use the algorithm of Bortfeld* et al. *with the sweep technique for the calculation of the complexity. Then our benchmark matrix has complexity 9, and the first extracted matrix is*

$$
3 \begin{pmatrix} 1&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&1 \\ 1&1&1&0&0&0 \end{pmatrix},
$$

*where the residual matrix* $\begin{pmatrix} 1&2&0&1&4&5 \\ 2&1&1&3&1&4 \\ 2&3&2&1&2&1 \\ 2&0&0&2&5&3 \end{pmatrix}$ *has complexity 6. Continuing we obtain the segmentation*

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

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

As indicated by the example, the algorithm yields quite good results compared to other algorithms, in particular it is essentially more NS–efficient than the algorithm used for the calculation of the complexity. This is confirmed in [6] by a number of numerical experiments.

### 2.14 Reduction of Leaf Motion

After the segments and their coefficients have been determined by some algorithm we still have the freedom to choose the order in which the corresponding homogeneous fields are delivered. In order to reduce the total treatment time it is suggestive to choose an order which minimizes the leaf travel time between

consecutive segments. The minimization of the overall leaf travel time is equivalent to the search for a Hamiltonian path of minimal weight on the complete graph which has the segments as vertices and a weight function $\mu$ on the edges, defined as follows: for two segments $S$ and $S'$, given by $l_i, r_i$ $(i \in [m])$ and $l'_i, r'_i$ $(i \in [m])$, respectively, we put

$$\mu(S, S') = \max_{1 \le i \le m} \max\{|l_i - l'_i|, |r_i - r'_i|\}.$$

Clearly, $\mu(S, S') = \mu(S', S)$, $\mu(S, S') \ge 0$ with equality iff $l_i = l'_i$ and $r_i = r'_i$ for all $i \in [m]$ and

$$\begin{aligned}\mu(S, S'') &= \max_{1 \le i \le m} \max\{|l_i - l''_i|, |r_i - r''_i|\} \\ &\le \max_{1 \le i \le m} \max\{|l_i - l'_i| + |l'_i - l''_i|, |r_i - r'_i| + |r'_i - r''_i|\} \\ &\le \mu(S, S') + \mu(S', S'').\end{aligned}$$

Thus $\mu$ is a metric and there are good approximations for a minimal Hamiltonian path ([10]). When the number NS is not relatively small, as is the case for practical problems, it is even possible to solve the Hamiltonian path problem exactly.

**Example 11.** *Using the version of Kalinowski's algorithm that heuristically reduces the NS we obtain the segmentation*

$$\begin{pmatrix} 16 & 10 & 0 & 5 & 4 & 12 \\ 1 & 13 & 16 & 6 & 2 & 14 \\ 6 & 6 & 3 & 3 & 15 & 2 \\ 8 & 15 & 3 & 0 & 13 & 11 \\ 3 & 16 & 13 & 6 & 9 & 3 \\ 7 & 16 & 10 & 11 & 14 & 6 \end{pmatrix} = 8 \begin{pmatrix} 0&0&0&0&0&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&1 \\ 0&0&0&0&1&0 \\ 0&1&1&1&1&0 \end{pmatrix} + 8 \begin{pmatrix} 1&1&0&0&0&0 \\ 0&1&1&0&0&0 \\ 0&0&0&0&0&0 \\ 1&1&0&0&0&0 \\ 0&1&1&0&0&0 \\ 0&1&0&0&0&0 \end{pmatrix} + 4 \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&1&1&0&0 \\ 1&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&1&1&0&0 \\ 1&0&0&0&0&0 \end{pmatrix}$$
$$+3 \begin{pmatrix} 0&0&0&1&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&0 \\ 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&1 \end{pmatrix} + 2 \begin{pmatrix} 1&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 1&1&1&1&1&0 \\ 0&1&1&0&0&0 \\ 1&1&0&0&0&0 \\ 0&0&1&1&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&1&0&0 \\ 0&0&1&1&1&1 \\ 0&0&0&0&1&1 \\ 0&0&0&0&1&0 \\ 0&0&0&1&0&0 \\ 0&0&0&1&1&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&1&1&1 \\ 0&0&0&0&0&1 \\ 0&0&1&1&1&1 \\ 0&0&0&0&1&0 \\ 0&1&1&1&1&0 \\ 1&0&0&0&0&0 \end{pmatrix}$$
$$+ \begin{pmatrix} 1&0&0&0&0&0 \\ 1&1&1&1&1&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&1&0&0&0&0 \\ 1&0&0&0&0&0 \end{pmatrix} + \begin{pmatrix} 1&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&1&1&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \end{pmatrix}.$$

*If we deliver the segments in this order the length of the corresponding Hamiltonian path is* $5 + 1 + 4 + 5 + 4 + 3 + 5 + 1 = 28$. *Using a minimum spanning tree approximation for the Hamiltonian path we obtain the delivery order* $1, 4, 6, 7, 5, 2, 3, 8, 9$ *with a length of* $3 + 3 + 3 + 3 + 1 + 1 + 1 + 1 = 16$.

First numerical results for the reduction of leaf motion when the algorithms of Engel and Kalinowski are used are shown in 1.

For algorithms using a sweep technique, such that the leaves move always in one direction there is nothing to do, since the leaf motion is automatically minimized.

**Lemma 2.** *Let* $A = \sum_{t=1}^{k} u_t S^{(t)}$ *be a segmentation obtained by some algorithm using a sweep–technique, that is if* $l_i^{(t)}$ *and* $r_i^{(t)}$ *are the parameters of* $S^{(t)}$ $(t \in [k])$

**Table 1.** Reduction of leaf motion by a minimum spanning tree approximation of the minimal Hamiltonian path for the algorithms of Kalinowski ([12]) and Engel ([7]). $P_{\text{old}}$ is the Hamiltonian path corresponding to the order in which the segments are constructed by the algorithm and $P_{\text{new}}$ is the approximation of a minimal Hamiltonian path. The results are averaged over 1000 $15 \times 15-$matrices with random entries from $\{0, 1, \ldots, 16\}$.

| | Engel | | Kalinowski | |
|---|---|---|---|---|
| $L$ | $\mu(P_{\text{old}})$ | $\mu(P_{\text{new}})$ | $\mu(P_{\text{old}})$ | $\mu(P_{\text{new}})$ |
| 3 | 112.5 | 106.5 | 64.7 | 49.3 |
| 4 | 126.1 | 119.1 | 81.5 | 59.9 |
| 5 | 136.7 | 128.2 | 128.3 | 85.7 |
| 6 | 145.9 | 136.4 | 141.5 | 93.3 |
| 7 | 152.0 | 141.9 | 152.1 | 99.4 |
| 8 | 157.4 | 146.7 | 163.4 | 105.4 |
| 9 | 163.2 | 151.2 | 172.7 | 111.7 |
| 10 | 166.6 | 154.3 | 179.7 | 116.5 |
| 11 | 170.5 | 158.3 | 187.5 | 121.2 |
| 12 | 174.2 | 161.0 | 193.4 | 124.4 |
| 13 | 178.1 | 165.1 | 199.8 | 128.4 |
| 14 | 180.6 | 167.0 | 206.5 | 131.6 |
| 15 | 183.1 | 169.5 | 211.0 | 134.8 |
| 16 | 185.4 | 171.9 | 217.4 | 138.5 |

*then we have, for all* $i \in [m]$,

$$l_i^{(1)} \leq l_i^{(2)} \leq \ldots \leq l_i^{(k)} \quad \textit{and} \quad r_i^{(1)} \leq r_i^{(2)} \leq \ldots \leq r_i^{(k)}.$$

*Then* $(S^{(1)}, S^{(2)}, \ldots, S^{(k)})$ *is a Hamiltonian path of minimal weight in the complete graph with vertex set* $\{S^{(1)}, S^{(2)}, \ldots, S^{(k)}\}$ *and weight function* $\mu$.

*Proof.* Let $\pi$ be an arbitrary permutation of $[k]$. We have to show that

$$\sum_{t=1}^{k-1} \mu\left(S^{(\pi(t))}, S^{(\pi(t+1))}\right) \geq \sum_{t=1}^{k-1} \mu\left(S^{(t)}, S^{(t+1)}\right).$$

The crucial observation is that, for $1 \leq t \leq t'' \leq t' \leq k$, we have

$$\mu\left(S^{(t)}, S^{(t')}\right) \geq \mu\left(S^{(t)}, S^{(t'')}\right),$$

which follows directly from the definition of $\mu$. If $\pi(t) < \pi(t+1)$ for all $t \in [k-1]$ there is nothing to do. Otherwise put

$$\begin{aligned} t_0 &= \min\{t \ : \ \pi(t) > \pi(t+1)\}, \\ t_1 &= \begin{cases} k & \text{if } \pi(t_0) = k, \\ \min\{t \ : \ \pi(t+1) > \pi(t_0)\} & \text{otherwise,} \end{cases} \\ t_2 &= \min\{t \ : \ \pi(t) > \pi(t_1)\}. \end{aligned}$$

**Table 2.** Test results without ICC. The columns labeled Gal, Bor, X–V and Eng correspond to the algorithm of Galvin *et al.* [8], the algorithm of Bortfeld *et al.* [3], the algorithm of Xia and Verhey [24] and the algorithm of Engel [7], respectively.

| | TNMU | | | | NS | | | |
|---|---|---|---|---|---|---|---|---|
| L | Gal | Bor | X–V | Eng | Gal | Bor | X–V | Eng |
| 3 | 17.0 | 14.0 | 16.6 | 14.0 | 11.3 | 14.0 | 11.1 | 9.7 |
| 4 | 31.3 | 17.9 | 22.4 | 17.9 | 15.4 | 17.9 | 14.1 | 10.9 |
| 5 | 32.7 | 21.8 | 25.0 | 21.8 | 16.2 | 21.8 | 15.1 | 11.7 |
| 6 | 39.5 | 25.6 | 37.7 | 25.6 | 17.7 | 25.6 | 17.9 | 12.5 |
| 7 | 50.9 | 29.5 | 38.8 | 29.5 | 20.1 | 29.5 | 16.2 | 13.1 |
| 8 | 60.5 | 33.3 | 46.3 | 33.3 | 21.3 | 33.3 | 20.2 | 13.7 |
| 9 | 60.3 | 37.1 | 51.0 | 37.1 | 22.1 | 37.1 | 20.2 | 14.2 |
| 10 | 71.0 | 40.9 | 53.9 | 40.9 | 23.1 | 40.9 | 20.5 | 14.7 |
| 11 | 83.5 | 44.8 | 55.7 | 44.8 | 25.1 | 44.8 | 21.6 | 15.1 |
| 12 | 84.5 | 48.6 | 81.1 | 48.6 | 25.7 | 48.6 | 21.8 | 15.5 |
| 13 | 98.2 | 52.4 | 83.3 | 52.4 | 26.5 | 52.4 | 22.4 | 15.8 |
| 14 | 108.7 | 56.2 | 83.5 | 56.2 | 27.2 | 56.2 | 22.8 | 16.2 |
| 15 | 128.2 | 60.1 | 83.5 | 60.1 | 27.9 | 60.1 | 23.5 | 16.5 |
| 16 | 93.6 | 63.8 | 93.6 | 63.8 | 29.4 | 63.8 | 23.9 | 16.8 |

Now we may replace $\pi$ by the permutation $\pi'$ given by

$$\pi(1),\dots,\pi(t_2-1),\pi(t_1),\pi(t_1-1),\dots,\pi(t_2),\pi(t_1+1),\dots,\pi(k).$$

To see this, assume first $t_2 > 1$ and $t_1 < k$. Then $\pi(t_2-1) < \pi(t_1) < \pi(t_2)$ and $\pi(t_1) < \pi(t_2) < \pi(t_1+1)$, hence

$$\begin{aligned}\sum_{t=1}^{k-1}\mu\left(S^{(\pi'(t))},S^{(\pi'(t+1))}\right)&=\sum_{t=1}^{k-1}\mu\left(S^{(\pi(t))},S^{(\pi(t+1))}\right)-\mu\left(S^{(\pi(t_2-1))},S^{(\pi(t_2))}\right)\\&\quad-\mu\left(S^{(\pi(t_1))},S^{(\pi(t_1+1))}\right)+\mu\left(S^{(\pi(t_2-1))},S^{(\pi(t_1))}\right)+\mu\left(S^{(\pi(t_2))},S^{(\pi(t_1+1))}\right)\\&\le\sum_{t=1}^{k-1}\mu\left(S^{(\pi(t))},S^{(\pi(t+1))}\right).\end{aligned}$$

Similarly,

$$\sum_{t=1}^{k-1}\mu\left(S^{(\pi'(t))},S^{(\pi'(t+1))}\right)\le\sum_{t=1}^{k-1}\mu\left(S^{(\pi(t))},S^{(\pi(t+1))}\right)$$

if $t_2 = 1$ or $t_1 = k$. Repeating this replacement if necessary, we obtain the permutation $1,2,\dots,k$, and the lemma is proved.

### 2.15 Numerical Results

In this subsection the performance of some of the algorithms is compared based on the segmentation of $15 \times 15$–matrices. As in [24] for every algorithm we

construct segmentations of 1000 matrices with random entries from $\{0, 1, \ldots, L\}$ ($L = 3, 4, \ldots, 16$) and determine the average TNMU and the average NS. We used the results from [24] for the algorithm of Galvin *et al.*, the algorithm of Bortfeld *et al.* and the algorithm of Xia and Verhey, and we implemented the algorithm of Engel, the algorithm of Kamath and the algorithm of Kalinowski in C++. The results are shown in Tables 2 and 3.

**Table 3.** Test results with ICC. The columns labeled Gal, Bor, X–V, Kam and Kal correspond to the algorithm Galvin *et al.* [8], the algorithm of Bortfeld *et al.* [3], the algorithm of Xia and Verhey [24], the algorithm of Kamath [13] and the algorithm of Kalinowski [12], respectively.

| | TNMU | | | | | NS | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| L | Gal | Bor | X–V | Kam | Kal | Gal | Bor | X–V | Kam | Kal |
| 3 | 19.7 | 17.7 | 19.5 | 15.4 | 15.4 | 13.4 | 17.7 | 13.3 | 15.4 | 12.6 |
| 4 | 40.5 | 22.8 | 29.6 | 19.5 | 19.5 | 20.4 | 22.8 | 18.6 | 19.5 | 14.5 |
| 5 | 40.1 | 27.9 | 30.9 | 23.6 | 23.6 | 20.4 | 27.9 | 19.0 | 23.6 | 16.0 |
| 6 | 44.2 | 32.8 | 46.8 | 27.6 | 27.6 | 21.5 | 32.8 | 20.3 | 27.6 | 17.2 |
| 7 | 67.1 | 37.9 | 45.6 | 31.7 | 31.7 | 27.1 | 37.9 | 20.0 | 31.7 | 18.2 |
| 8 | 72.3 | 42.8 | 63.4 | 35.7 | 35.7 | 28.2 | 42.8 | 24.3 | 35.7 | 19.1 |
| 9 | 72.3 | 47.8 | 67.1 | 39.8 | 39.8 | 28.3 | 47.8 | 24.3 | 39.8 | 19.9 |
| 10 | 76.5 | 52.6 | 68.6 | 43.8 | 43.8 | 28.9 | 52.6 | 25.7 | 43.8 | 20.7 |
| 11 | 81.4 | 57.6 | 68.6 | 47.7 | 47.7 | 30.9 | 57.6 | 25.7 | 47.7 | 21.3 |
| 12 | 106.8 | 62.4 | 101.1 | 51.8 | 51.8 | 34.8 | 62.4 | 27.0 | 51.8 | 21.9 |
| 13 | 101.1 | 67.3 | 100.6 | 55.7 | 55.7 | 35.5 | 67.3 | 26.9 | 55.7 | 22.5 |
| 14 | 112.7 | 72.2 | 100.0 | 59.8 | 59.8 | 35.6 | 72.2 | 26.9 | 59.8 | 23.0 |
| 15 | 116.0 | 77.1 | 98.0 | 63.8 | 63.8 | 35.9 | 77.1 | 26.7 | 63.8 | 23.5 |
| 16 | 154.5 | 82.0 | 124.9 | 67.7 | 67.7 | 41.7 | 82.0 | 30.0 | 67.7 | 24.0 |

With respect to the computation time all of the considered algorithms are acceptable: On an 1.3GHz PC the computation of the whole column for the algorithm of Kalinowski took 40 minutes, and for all the other algorithms the whole column can be computed in a few minutes.

## 3 Dynamic Methods

Another approach to the generation of intensity modulated irradiation fields is to use an MLC in the dynamic mode. That means the beam is always switched on and the modulation is realized by varying the speed of the leaves. In the literature two different variants can be found according to the starting positions of the leaves. For the sweep technique both leaves start at the left end of the field and move always to the right, while for the close–in technique the leaves start at opposite ends of the field and move towards each other. Obviously, with a single run of the close–in technique only profiles with a single maximum can be generated, and profiles where the gradient is too small are also excluded due to

the finiteness of the maximal leaf velocity. In contrast, the sweep technique can be used to generate arbitrary profiles, and several authors have derived equations for the leaf velocities realizing a given profile and minimizing the total irradiation time ([5,16,20,21,4,19]). In the first section we sketch the basic principle that is common to all of these approaches and in the second section we describe how the tongue and groove effect can be avoided using a method introduced in [22].

### 3.1 The Basic Principle

The leaf trajectories are determined independently for each row, and thus we only describe the realization of a single row profile. Let $a_0, a_1, \ldots, a_n$ be the required doses at the equidistant points $x_0, x_1, \ldots, x_n$, where $x_0$ and $x_n$ are the coordinates of the left and the right end of the field, respectively. Denote by $t_L(x)$ ($t_R(x)$) the time when the left (right) leaf passes the point with coordinate $x$. Then the dose delivered at $x$ is proportional to $t_L(x) - t_R(x)$ and by scaling the time appropriately we may assume

$$a_j = t_L(x_j) - t_R(x_j).$$

Denote the maximal leaf velocity by $\hat{v}$ and the velocities of the left and the right leaf by $v_L(x)$ and $v_R(x)$, respectively. Suppose $t_L(x_j)$ and $t_R(x_j)$ are already known. Then in order to minimize the time that is needed to generate the profile over the interval $[x_j, x_{j+1}]$ we put, for $x_j \le x < x_{j+1}$,

$$v_R(x) = \hat{v} \text{ if } a_{j+1} \ge a_j \quad \text{and} \quad v_L(x) = \hat{v} \text{ if } a_{j+1} < a_j. \tag{44}$$

First assume $a_{j+1} \ge a_j$. Then

$$a_{j+1} = t_L(x_{j+1}) - t_R(x_{j+1}) = t_L(x_{j+1}) - \left(t_R(x_j) + \frac{\Delta x}{\hat{v}}\right),$$

where $\Delta x = x_{j+1} - x_j$. We interpolate the profile between $x_j$ and $x_{j+1}$ linearly, so $v_L(x)$ is constant for $x_j \le x < x_{j+1}$, and we obtain

$$v_L(x) = \frac{\Delta x}{t_L(x_{j+1}) - t_L(x_j)} = \frac{\hat{v}}{1 + (a_{j+1} - a_j)\frac{\hat{v}}{\Delta x}}, \tag{45}$$

and analogously, if $a_{j+1} < a_j$,

$$v_R(x) = \frac{\Delta x}{t_R(x_{j+1}) - t_R(x_j)} = \frac{\hat{v}}{1 - (a_{j+1} - a_i)\frac{\hat{v}}{\Delta x}}. \tag{46}$$

The generation of the whole profile is complete when the left leaf reaches $x_n$. The time it takes for the left leaf to cross the interval $[x_j, x_{j+1}]$ is

$$\frac{\Delta x}{\hat{v}} \text{ if } a_{j+1} \le a_j \quad \text{and} \quad \frac{\Delta x}{\hat{v}} + (a_{j+1} - a_j) \text{ if } a_{j+1} \ge a_i.$$

Thus the total irradiation time is

$$t_L(x_n) = \frac{x_n - x_0}{\hat{v}} + \sum_{j=0}^{n-1} \max\{a_{j+1} - a_j, 0\}.$$

To see that this is optimal under the condition that both leaves start at $x_0$, observe that, for $0 \le j \le n-1$,

$$a_{j+1} = t_L(x_{j+1}) - t_R(x_{j+1}) \text{ and } a_j = t_L(x_j) - t_R(x_j),$$

thus, if $a_{j+1} > a_j$,

$$t_L(x_{j+1}) - t_L(x_j) = t_R(x_{j+1}) - t_R(x_j) + a_{j+1} - a_j \ge \frac{\Delta x}{\hat{v}} + a_{j+1} - a_j.$$

Clearly the total irradiation time for a multiple row intensity map is just the maximum of the irradiation time over the rows.

This method can be refined in several ways. So [19] and [20] include a compensation for the transmission through the leaves, and [21] takes into account the finite acceleration of the leaves.

### 3.2 The Tongue and Groove Effect

As in the static mode the tongue and groove design of the MLCs causes underdosage in the border region between adjacent rows, as illustrated in Figure 14.

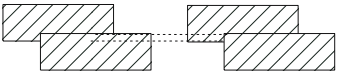

**Fig. 14.** Suppose the method from the previous section yields the same constant velocity for all the depicted leaves. Then the strip between the dotted lines receives only half of the dose that is required in both rows.

In [22] there is proposed a method to avoid this effect in the sense that after the correction the border region always receives the lower of the two relevant doses. The procedure is very similar to the tongue and groove correction of Siochi's rod pushing algorithm.

**Synchronization of two rows:** Consider two adjacent rows $(a_{i,0}, \ldots, a_{i,n})$ and $(a_{i+1,0}, \ldots, a_{i+1,n})$ and denote by $t_L^{(k)}(x)$, $t_R^{(k)}(x)$ $(k \in \{i, i+1\})$ the times when the left (resp. right) leaf of row $k$ passes $x$. We determine inductively leaf velocities $v_L^{(k)}$ and $v_R^{(k)}$ on the intervals $[x_j, x_{j+1}]$ such that the given profile is generated without tongue and groove underdosage. Suppose the leaf motion up to the point $x_j$ is already determined. First we compute the velocities and the corresponding $t_L^{(k)}(x_{j+1})$, $t_R^{(k)}(x_{j+1})$ according to (44)–(45). Tongue and groove underdosage occurs iff

$$t_R^{(i)}(x_{j+1}) > t_R^{(i+1)}(x_{j+1}) \quad \text{and} \quad t_L^{(i)}(x_{j+1}) > t_L^{(i+1)}(x_{j+1}), \tag{47}$$

or the same with the roles of $i$ and $i+1$ interchanged. We call the pair of rows synchronized if

$$t_R^{(i)}(x_{j+1}) = t_R^{(i+1)}(x_{j+1}) \quad \text{or} \quad t_L^{(i)}(x_{j+1}) = t_L^{(i+1)}(x_{j+1}).$$

Then in order to avoid the tongue and groove effect it is sufficient to change the velocities in such a way that the rows are synchronized. By symmetry we may assume that (47) holds. Then we just have to slow down both leaves in row $i+1$. Precisely, if $a_{i,j+1} \le a_{i+1,j+1}$, we put

$$\begin{aligned} t_L^{(i+1)}(x_{j+1}) &:= t_L^{(i)}(x_{j+1}), \\ t_R^{(i+1)}(x_{j+1}) &:= t_L^{(i+1)}(x_{j+1}) - a_{i+1,j+1}, \end{aligned}$$

and if $a_{i,j+1} > a_{i+1,j+1}$ we put

$$\begin{aligned} t_R^{(i+1)}(x_{j+1}) &:= t_R^{(i)}(x_{j+1}), \\ t_L^{(i+1)}(x_{j+1}) &:= t_R^{(i+1)}(x_{j+1}) + a_{i+1,j+1}. \end{aligned}$$

This is illustrated in Figure 15.

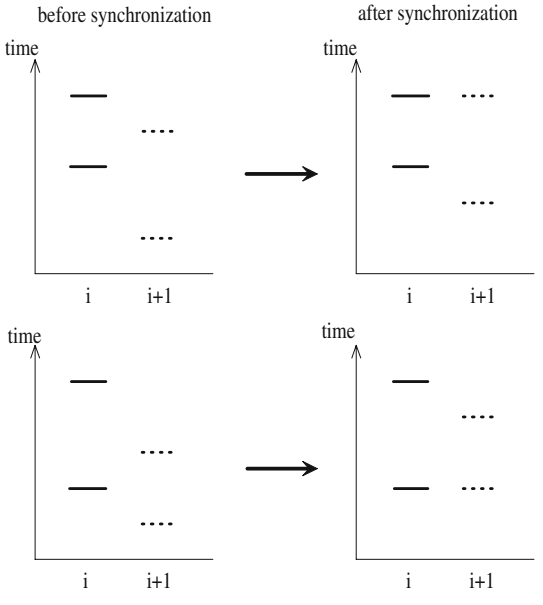


**Fig. 15.** The synchronization for two rows. The straight lines stand for $t_L^{(i)}(x_{j+1})$ and $t_R^{(i)}(x_{j+1})$, the dotted lines stand for $t_L^{(i+1)}(x_{j+1})$ and $t_R^{(i+1)}(x_{j+1})$.

Finally the new velocities for $x_j \leq x < x_{j+1}$ are computed:

$$v_L^{(i+1)}(x) = \frac{\Delta x}{t_L^{(i+1)}(x_{j+1}) - t_L^{(i+1)}(x_j)},$$
$$v_R^{(i+1)}(x) = \frac{\Delta x}{t_R^{(i+1)}(x_{j+1}) - t_R^{(i+1)}(x_j)}.$$

**Synchronization of more than two rows:** For general treatment plans the synchronization of leaf trajectories is based on the iterated synchronization of two rows. To correct the leaf trajectories between the points $x_j$ and $x_{j+1}$ first a row $i_0$ with slowest left leaf is determined, i.e. with $t_L^{(i_0)}(x_{j+1}) \geq t_L^{(i)}(x_{j+1})$ for all $i \in [m]$. Now the whole synchronization is described in Algorithm 6. The algorithm terminates since in every step some leaves are slowed down, but never a left leaf arrives later at $x_{j+1}$ than the one in row $i_0$, and so in the worst case finally $t_L^{(i)}(x_{j+1}) = t_L^{(i_0)}(x_{j+1})$ for all $i$.

**Algorithm 6.** : Synchronization of leaf motion

```
repeat
  finished:=true
  for i = i_0 − 1 downto 1 do
    if rows i and i + 1 are not synchronized then
      finished:=false
      synchronize rows i and i + 1
  for i = i_0 to m − 1 do
    if rows i and i + 1 are not synchronized then
      finished:=false
      synchronize rows i and i + 1
until finished
```

In [23] the authors argue that it might not be necessary to fully synchronize the leaf motion. This is because in the overlap region that is covered by both leaves the depth of each leaf is only half of the full leaf–depth and taking account of the difference between the transmission through the full depth and the half depth they derive a criterion for 'partial synchronization' which assures that the overlap region receives at least the lower of the two relevant doses.

## 4 Summary and Discussion

To realize intensity modulated radiation fields using a multileaf collimator in the static mode it is necessary to determine a sequence of leaf positions and corresponding irradiation times such that the superposition of the homogeneous fields yields the required modulated intensity. This amounts to the problem of representing a nonnegative integer matrix as a positive integer combination of

certain $(0, 1)$–matrices, so called segments. In order to optimize the treatment this segmentation has to be chosen in such a way that the total number of monitor units and the number of segments are small. Ignoring machine–dependent constraints the construction of a segmentation with minimal number of monitor units can be done in polynomial time, for instance by the algorithm of Bortfeld *et al.* [3] or the algorithm of Engel [7]. In contrast, the minimization of the number of segments is NP–complete already for a single row, and thus probably one has to be satisfied with an approximative algorithm for this problem. Our variant of Engel's algorithm seems to be very good in this respect, but there remains the problem to find a theoretical bound for the quality of the approximation. Other algorithms use heuristic principles to reduce the number of segments, but are no longer optimal with respect to the monitor units.

For the segmentation problem with interleaf collision constraint the algorithm of Kamath *et al.* [13], the algorithm of Baatar and Hamacher [1] and the algorithm of Kalinowski [12] minimize the number of monitor units in polynomial time. In addition, [1] and [12] propose greedy heuristics for the reduction of the number of segments, but these algorithms have the drawback that the computation time grows rapidly with the problem size.

The algorithm of Langer *et al.* formulates the segmentation problem as a mixed integer program and finds solutions that are optimal in first instance with respect to the monitor units and in second instance with respect to the segments. Also machine–dependent constraints are easily included. However, as for the algorithm of Dai and Zhu, due to computational complexity the method is not applicable for problem sizes that arise in practice.

One difficulty that comes up when the tongue and groove constraint is taken into account is that the local strategy of most of the published algorithms is no longer applicable since the different segments are not independent of each other. The first question which has to be addressed in this context is if it is necessary to avoid the tongue and groove effect totally, or if it is sufficient to reduce it to some extend. In the second case some quantitative measure for this acceptable underdosage has to be developed, and a corresponding objective function for the minimization has to be constructed.

When the multileaf collimator is used in the dynamic mode the leaf velocities have to be chosen so that the required intensity is generated. It is possible to determine optimal leaf velocities for an unidirectional sweep of the leaves across the field. Also the tongue and groove underdosage can be totally avoided by synchronization of the leaf motion.

## References

1. D. Baatar and H.W. Hamacher, New LP model for multileaf collimators in radiation therapy, contribution to the conference ORP3, University of Kaiserslautern, 2003.
2. N. Boland, H.W. Hamacher, and F. Lenzen, Minimizing beam-on time in cancer radiation treatment using multileaf collimators, Networks, 43, 226, 2004.
3. 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.

4. A.L. Boyer and C.Y. Yu. Intensity–modulated radiation therapy with dynamic multileaf collimators, Semin. Radiat. Oncol., 9, 48–59, 1999.
5. D.J. Convery and M.E. Rosenbloom, The generation of intensity–modulated fields for conformal radiotherapy by dynamic collimation, Phys. Med. Biol., 37, 6, 1359–1374, 1992.
6. J. Dai and Y. Zhu, Minimizing the number of segments in a delivery sequence for intensity–modulated radiation therapy with a multileaf collimator, Med. Phys., 28, 2113–2120, 2001.
7. K. Engel, A new algorithm for optimal multileaf collimator field segmentation, Preprint 03/5, Fachbereich Mathematik, Uni Rostock, 2003.
8. 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.
9. M. Haufschild and U.M. Korn, Mit Mathematik gegen Krebs – Optimale Einstellung eines Gerätes in der Strahlentherapie, contribution to Jugend–forscht, 2003.
10. D. Jungnickel, Graphen, Netzwerke und Algorithmen, BI–Wissenschaftsverlag, Mannheim, 1994.
11. T. Kalinowski, An algorithm for optimal multileaf collimator field segmentation with interleaf collision constraint, Preprint 03/2, Fachbereich Mathematik, Uni Rostock, 2003.
12. T. Kalinowski, An algorithm for optimal multileaf collimator field segmentation with interleaf collision constraint 2, Preprint 03/8, Fachbereich Mathematik, Uni Rostock, 2003.
13. S. Kamath, S. Sahni, J. Li, J. Palta, and S. Ranka, Leaf sequencing algorithms for segmented multileaf collimation, Phys. Med. Biol., 48, 307–324, 2003.
14. M. Langer, V. Thai, and L. Papiez, Improved leaf sequencing reduces segments of monitor units needed to deliver IMRT using multileaf collimators, Med. Phys., 28, 2450–2458, 2001.
15. F. Lenzen, An integer programming approach to the multileaf collimator problem, Master's thesis, University of Kaiserslautern, Dept. of Mathematics, 2000.
16. L. Ma, A.L. Boyer, L. Xing, and C.M. Ma, An optimized leaf setting algorithm for beam intensity modulation using dynamic multileaf collimators, Phys. Med. Biol., 43, 1629–1643, 1998.
17. W. Que, Comparison of algorithms for multileaf collimator field segmentation, Med. Phys., 26, 2390–2396, 1999.
18. 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.
19. S.V. Spirou and C.S. Chui, Generation of arbitrary intensity profiles by dynamic jaws or multileaf collimators, Med. Phys., 21, 1031–1041, 1994.
20. J. Stein, T. Bortfeld, B. Dörschel, and W. Schlegel, Dynamic X–ray compensation for conformal radiotherapy by means of multi–leaf collimation, Radiother. Oncol., 32, 163–173, 1994.
21. R. Svensson, P. Källman, and A. Brahme, An analytical solution for the dynamic control of multileaf collimators, Phys. Med. Biol., 39, 37–61, 1994.
22. J.P.C. van Santvoort and B.J.M. Heijmen, Dynamic multileaf collimation without 'tongue-and-groove' underdosage effects, Phys. Med. Biol., 41, 2091–2105, 1996.
23. S. Webb, T. Bortfeld, J. Stein, and D. Convery, The effect of stair–step leaf transmission on the 'tongue–and–groove problem' in dynamic radiotherapy with a multileaf collimator, Phys. Med. Biol., 42, 595–602, 1996.
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.