
Decomposition of matrices and static multileaf collimators: a survey

Matthias Ehrgott^{1*}, Horst W. Hamacher^{2†}, and Marc Nußbaum²

¹ Department of Engineering Science, The University of Auckland, Auckland, New Zealand. m.ehrgott@auckland.ac.nz

² Fachbereich Mathematik, Technische Universität Kaiserslautern, Kaiserslautern, Germany. hamacher@mathematik.uni-kl.de

Summary. Multileaf Collimators (MLC) consist of (currently 20-100) pairs of movable metal leaves which are used to block radiation in Intensity Modulated Radiation Therapy (IMRT). The leaves modulate a uniform source of radiation to achieve given intensity profiles. The modulation process is modeled by the decomposition of a given non-negative integer matrix into a non-negative linear combination of matrices with the (strict) consecutive ones property.

In this paper we review some results and algorithms which can be used to minimize the time a patient is exposed to radiation (corresponding to the sum of coefficients in the linear combination), the set-up time (corresponding to the number of matrices used in the linear combination), and other objectives which contribute to an improved radiation therapy.

Keywords: Intensity modulated radiation therapy, multileaf collimator, intensity map segmentation, complexity, multi objective optimization.

1 Introduction

Intensity modulated radiation therapy (IMRT) is a form of cancer therapy which has been used since the beginning of the 1990s. Its success in fighting cancer is based on the fact that it can modulate radiation, taking specific patient data into consideration. Mathematical optimization has contributed considerably since the end of the 1990s (see, for instance, [31]) concentrating mainly on three areas,

* The research has been partially supported by University of Auckland Researcher's Strategic Support Initiative grant 360875/9275.

† The research has been partially supported by Deutsche Forschungsgemeinschaft (DFG) grant HA 1737/7 "Algorithmik großer und komplexer Netzwerke" and by New Zealand's Julius von Haast Award.

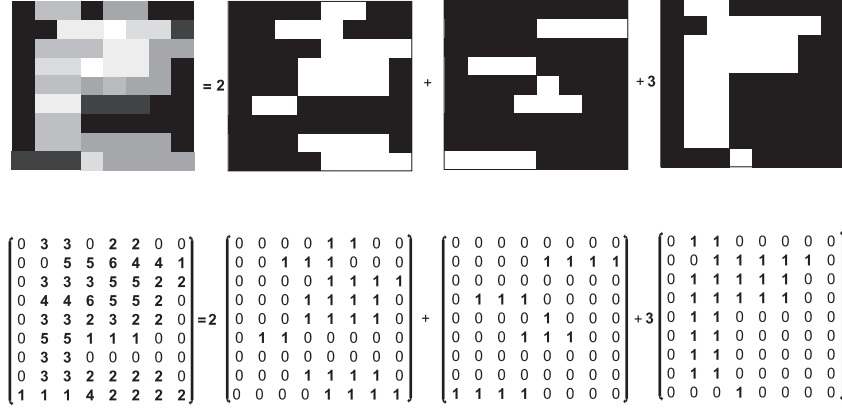


Fig. 1. Realization of an intensity matrix by overlaying radiation fields with different MLC segments.

- the geometry problem,
- the intensity problem, and
- the realization problem.

The first of these problems finds the best selection of radiation angles, i.e., the angles from which radiation is delivered. A recent paper with the most up to date list of references for this problem can be found in [17]. Once a solution of the geometry problem has been found, an *intensity profile* is determined for each of the angles. These intensity profiles can be found, for instance, with the multicriteria approach of [20] or many other intensity optimization methods (see [30] for more references). In Figure 1, an intensity profile is shown as greyscale coded grid. We assume that the intensity profile has been discretized such that the different shades in this grid can be represented by non-negative integers, where black corresponds to 0 and larger integers are used for lighter colors. In the following we will therefore think of intensity profiles and $N \times M$ *intensity matrices* A as one and the same.

In this paper, we assume that solutions for the geometry and intensity problems have been found and focus on the problem of realizing the intensity matrix A using so-called (static) *multileaf collimators* (MLC). Radiation is blocked by M (left, right) pairs of metal leaves, each of which can be positioned between the cells of the corresponding intensity profile. The opening corresponding to a cell of the segment is referred to as a *bixel* or *beamlet*. On the right-hand-side of Figure 1, three possible *segments* for the intensity profile on the left of Figure 1 are shown, where the black areas in the three rectangles correspond to the left and right leaves. Radiation passes (perpendicular to the plane represented by the segments) through the opening between the leaves (white areas). The goal is to find a set of MLC segments such that the intensity matrix A is realized by irradiating each of these segments for a certain amount of time (2, 1, and 3 in Figure 1).

In the same way as intensity profiles and integer matrices correspond to each other, each segment in Figure 1 can be represented by a binary $M \times N$ matrix $Y = (y_{mn})$, where $y_{mn} = 1$ if and only if radiation can pass through bixel (m, n) . Since the area left open by each pair of leaves is contiguous, the matrix Y possesses the (strict) *consecutive-ones (C1)* property in its rows, i.e., for all $m \in \mathcal{M} := \{1, \dots, M\}$ and $n \in \mathcal{N} := \{1, \dots, N\}$ there exists a pair $l_m \in \mathcal{N}, r_m \in \mathcal{N} \cup \{N+1\}$ such that

$$y_{mn} = 1 \iff l_m \leq n < r_m. \quad (1)$$

Hence the realization problem can be formulated as the following *C1 decomposition problem*. Let \mathcal{K} be the index set of all $M \times N$ consecutive-ones matrices and let $\mathcal{K}' \subseteq \mathcal{K}$. A *C1 decomposition* (with respect to \mathcal{K}') is defined by non-negative integers $\alpha_k, k \in \mathcal{K}'$ and $M \times N$ C1 matrices $Y^k, k \in \mathcal{K}'$ such that

$$A = \sum_{k \in \mathcal{K}'} \alpha_k Y^k. \quad (2)$$

The coefficients α_k are often called the *monitor units, MU*, of Y^k . In order to evaluate the quality of a C1 decomposition various objective functions have been used in the literature.

The beam-on-time (BOT), total number of monitor units, or *decomposition time (DT)* objective

$$DT(\alpha) := \sum_{k \in \mathcal{K}'} \alpha_k \quad (3)$$

is a measure for the time a patient is exposed to radiation. Since every change from one segment of the MLC to another takes time, the number of segments or *decomposition cardinality (DC)*

$$DC(\alpha) := |\{\alpha_k : \alpha_k > 0\}| \quad (4)$$

is used to evaluate the (constant) *set-up time*

$$SU_{const}(\alpha) := \tau DC(\alpha) \quad (5)$$

for the MLC. Here we assume that it takes constant time τ to move from one segment to the next. If, on the other hand, τ_{kl} is a variable time to move from Y^k to Y^l and Y^1, \dots, Y^K are the C1 matrices used in a decomposition, then one can also consider the *variable set-up time*

$$SU_{var}(\alpha) = \sum_{k=1}^{K-1} \tau_{\pi(k), \pi(k+1)} \cdot \alpha_{\pi(k)}. \quad (6)$$

Obviously, this objective depends on the sequence $\pi(1), \dots, \pi(K)$ of these C1 matrices. The *treatment time* is finally defined for each radiation angle by

$$TT(\alpha) := DT(\alpha) + SU(\alpha), \quad (7)$$

where $SU(\alpha) \in \{SU_{var}(\alpha), SU_{const}(\alpha)\}$. Since the set-up time $SU(\alpha)$ can be of the constant or variable kind, two different definitions of treatment time are possible.

For therapeutic and economic reasons, it is desirable to find decompositions with small beam-on, set-up, and treatment times. These will be the optimization problems considered in the subsequent sections.

In this paper we will summarize some basic results and present the ideas of algorithms to solve the decomposition time (Section 2) and the decomposition cardinality (Section 3) problem. In Section 4 we will deal with combined objective functions and mention some current research questions.

2 Algorithms for the decomposition time problem

In this section we consider a given $M \times N$ non-negative integer matrix A corresponding to an intensity profile and look for the decomposition (2) of A into a non-negative linear combination $A = \sum_{k \in \mathcal{K}'} \alpha_k Y^k$ of C1 matrices such that the decomposition time (3) $DT(\alpha) := \sum_{k \in \mathcal{K}'} \alpha_k$ is minimized. First, we review results of the *unconstrained DT problem* in which all C1 matrices can be used, i.e., $\mathcal{K}' = \mathcal{K}$. Then we discuss the *constrained DT problem*, where technical requirements exclude certain C1 matrices, i.e., $\mathcal{K}' \subsetneq \mathcal{K}$.

2.1 Unconstrained DT problem

The most important argument in the unconstrained case is the fact that it suffices to solve the DT problem for single row matrices.

Lemma 1. $A = \sum_{k \in \mathcal{K}} \alpha_k Y^k$ is a decomposition with decomposition time $DT(\alpha) := \sum_{k \in \mathcal{K}} \alpha_k$ if and only if each row A_m of A has a decomposition $A_m = \sum_{k \in \mathcal{K}} \alpha_{k_m} Y_m^k$ into C1 row matrices with decomposition time $DT(\alpha_m) := \sum_{k \in \mathcal{K}} \alpha_{k_m}$, such that

$$DT(\alpha) := \max_{m=1}^M DT(\alpha_m). \quad (8)$$

The proof of this result follows from the fact that in the unconstrained DT problem, the complete set of all C1 matrices can be used. Hence, the decomposition of the row with largest $DT(\alpha_m)$ can be extended in an arbitrary fashion by decompositions of the other rows to yield a decomposition of the matrix A with $DT(\alpha) = DT(\alpha_m)$.

The most prominent reference in which the insight of Lemma 1 is used is [8], which introduces the *sweep algorithm*. Each row is considered independently and then checked from left to right, if a position of a left or right leaf needs to be changed in order to realize given intensities a_{mn} . While most practitioners agree that the sweep algorithm provides decompositions with short

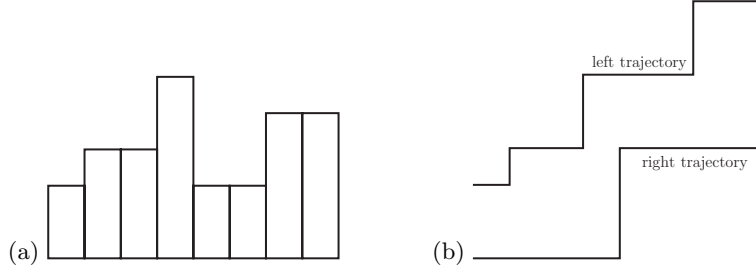


Fig. 2. Representation of intensity row $A_m = (2, 3, 3, 5, 2, 2, 4, 4)$ by rods (a) and the corresponding left and right trajectories (b).

$DT(\alpha)$, the optimality of the algorithm was only proved several years later. We will review some of the papers containing proofs below.

An algorithm which is quoted very often in the MLC optimization literature is that of [32]. Each entry a_{mn} of the intensity map is assigned to a *rod*, the length of which represents the value a_{mn} (see Figure 2). The standard step-and-shoot approach, which is shared by all static MLC algorithms, is implemented in two parts, the *rod pushing* and the *extraction*. While the objective in [32] is to minimize total treatment time TT_{var} , the proposed algorithm is only guaranteed to find a solution that minimizes $DT(\alpha)$.

The authors in [1] prove the optimality of the sweep algorithm by transforming the DT problem into a linear program. The decomposition of a row A_m into C1 row-matrices is first reformulated in a transposed form, i.e., the column vector A_m^T is decomposed into C1 column-matrices (columns with 1s in a single block). This yields a linear system of equations, where the columns of the coefficient matrix are all possible $N(N-1)/2$ C1 column-matrices, the variables are the (unknown) decomposition times and the right-hand-side vector is the transpose A_m^T of row A_m . The objective of the linear program is the sum of the MUs. Such a linear program is well known (see [2]) to be equivalent to a network flow problem in a network with N nodes and $N(N-1)/2$ arcs. The authors in [1] use the special structure of the network and present a shortest augmenting path algorithm which saturates at least one of the nodes in each iteration. Since each of the paths can be constructed in constant time, the complexity for computing $DT(\alpha_m)$ is $\mathcal{O}(N)$. This algorithm is applied to each of the rows of A , such that Lemma 1 implies the following result.

Theorem 1 ([1]). *The unconstrained decomposition time problem for a given non-negative integer $M \times N$ matrix A can be solved in $\mathcal{O}(NM)$ time.*

It is important to notice that the identification of the flow augmenting path and the determination of the flow value which is sent along this path can be interpreted as the two phases of the step-and-shoot process in the sweep algorithm of [8], thus establishing its optimality.

An alternative optimality proof of the sweep algorithm can be found in [23]. Their methodology is based on analyzing the *left and right leaf trajectories* for

each row $A_m, m \in \mathcal{M}$. These trajectory functions are at the focus of research in dynamic MLC models. For static MLC in which each leaf moves from left to right, they are monotonously non-decreasing step functions with an increase of $|a_{m,n+1} - a_{m,n}|$ in the left or right trajectory at position n if $a_{m,n+1} - a_{m,n}$ increases or decreases, respectively. Figure 2 illustrates an example with row $A_m = (2, 3, 3, 5, 2, 2, 4, 4)$, the representation of each entry a_{mn} as *rod*, and the corresponding trajectories. By proving that the step size of the left leaf trajectory in any position n is an upper bound on the number of MUs of any other feasible decompositions, the authors in [23] establish the optimality of the decomposition delivered by their algorithm SINGLEPAIR for the case of single row DT problems. In combination with Lemma 1, this yields the optimality of their solution algorithm MULTIPAIR for the unconstrained DT problem, which is, again, a validity proof of the sweep algorithm.

The same bounding argument as in [23] is used by the author in [18] in his TNMU algorithm (*total number of monitor units*). Instead of using trajectories, he bases his work directly on the $M \times (N + 1)$ *difference matrix*

$$D = (d_{mn}) \text{ with } d_{mn} := a_{mn} - a_{m(n-1)} \\ \text{for all } m = 1, \dots, M, n = 1, \dots, N + 1. \quad (9)$$

Here, $a_{m0} := a_{m(n+1)} := 0$. In each iteration, the TNMU algorithm reduces the *TNMU complexity of A*

$$C(A) := \max_{m \in \mathcal{M}} C_m(A), \quad (10)$$

where $C_m(A) := \sum_{n=1}^{N+1} \max\{0, d_{m,n}\}$ is the *row complexity* of row A_m . More precisely, in each iteration the algorithm identifies some integer $p > 0$ and some C1 matrix Y such that $A' = A - pY$ has non-negative entries and its TNMU complexity satisfies $C(A') = C(A) - p$. Various strategies are recommended to find suitable p and Y , one version of which results in an $\mathcal{O}(N^2 M^2)$ algorithm. As a consequence of its proof, the following closed form expression for the optimal objective value of the DT problem in terms of the TNMU complexity is attained.

Theorem 2 ([18]). *The unconstrained decomposition time problem for a given non-negative integer $M \times N$ matrix A has optimal objective value $DT(\alpha) = C(A)$.*

As will be seen in Section 3.2, this idea also leads to algorithms for the decomposition cardinality problem.

2.2 Constrained DT problem

Depending on the type of MLC, several restrictions may apply to the choice of C1 matrices Y^k which are used in decomposition (2), i.e. $\mathcal{K}' \subsetneq \mathcal{K}$. For example, the mechanics of the multileaf collimator may require that left and right leaf

pairs (l_{m-1}, r_{m-1}) and (l_m, r_m) in adjacent rows Y_{m-1} and Y_m of any C1 matrix Y must not overlap (*interleaf motion constraints*). More specifically, we call a C1 matrix Y *shape matrix* if

$$l_{m-1} \leq r_m \text{ and } r_{m-1} \geq l_m \quad (11)$$

holds for all $m = 2, \dots, M$. The matrix

$$Y = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

is, for instance, a C1 matrix, but not a shape matrix, since there are two violations of (11), namely $r_1 = 4 < 5 = l_2$ and $l_3 = 3 > 2 = r_4$. By drawing the left and right leaves corresponding to the left and right sets of zeros in each row of Y , it is easy to understand why the constraints (11) are called interleaf motion constraints.

Another important restriction is the *width* or *innerleaf motion* constraint

$$r_m - l_m \geq \delta \text{ for all } m \in \mathcal{M}, \quad (12)$$

where $\delta > 0$ is a given (integer) constant.

A final constraint may be enforced to control *tongue-and-groove* ($T\&G$) *error* which often makes the decomposition model (2) inaccurate. Since several MLC types have T&G joints between adjacent leaf pairs, the thinner material in the tongue and the groove causes a smaller or larger radiation than predicted in model 2 if a leaf covers bixel m, n (i.e., $y_{mn} = 0$), but not $m+1, n$ (i.e., $y_{m+1,n} = 1$), or vice versa. Some of this error is unavoidable, but a decomposition with $y_{mn}^k = 1, y_{m+1,n}^k = 0$ and $y_{mn}^{k'} = 0, y_{m+1,n}^{k'} = 1$ can often be avoided by swapping the m^{th} rows of Y^k and $Y^{k'}$.

The authors in [7] present a polynomial algorithm for the DT problem with interleaf motion and width constraints by reducing it to a network flow problem with side constraints. They first construct a layered graph $G = (V, E)$, the *shape matrix graph* which has M layers of nodes. The nodes in each layer represent left-right leaf set-ups in an MLC satisfying the width constraint or — equivalently — a feasible row in a shape matrix (see Figure 3). More precisely, node (m, l, r) stands for a possible row m in a C1 matrix with left leaf in position l and right leaf in position r , where the width constraint is modeled by allowing only nodes (m, l, r) with $r - l \geq \delta$. Hence, in each layer there are $\mathcal{O}(N(N-1))$ nodes, and the network has $\mathcal{O}(MN^2)$ nodes. Interleaf motion constraints are modeled by the definition of the arc set E according to $((m, l, r), (m+1, l', r')) \in E$ if and only if $r' - l \geq \delta$ and $r - l' \geq \delta$.

It should be noted that the definition of the arcs can also be adapted to include the *extended interleaf motion constraint*

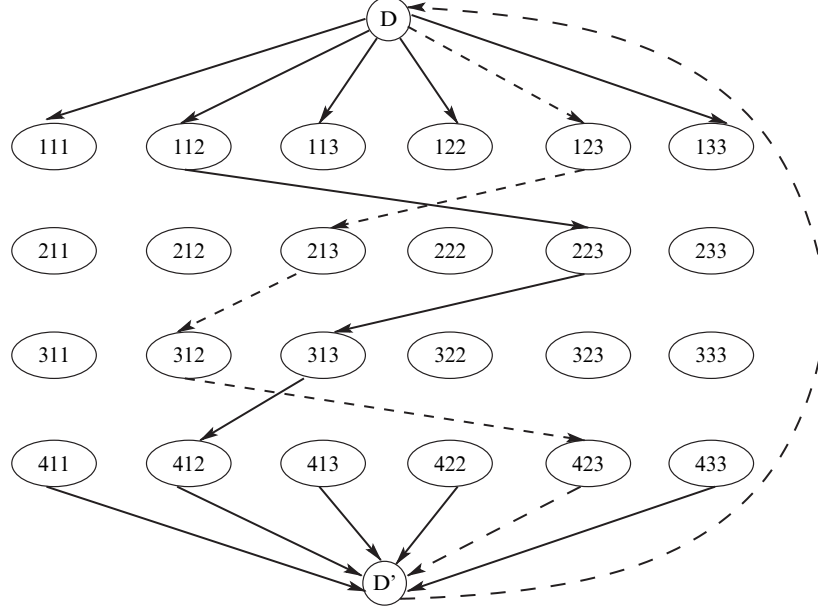


Fig. 3. Shape matrix graph with two paths corresponding to two shape matrices. (Both paths are extended by the return arc (D', D) .)

$$r_m - l_{m-1} \geq \gamma \text{ and } l_m - r_{m-1} \geq \gamma \text{ for all } m \in \mathcal{M}, \quad (13)$$

where $\gamma > 0$ is a given (integer) constant. Also, T&G constraints can be modeled by the network structure. If we add a supersource D and a supersink D' connected to all nodes $(1, l, r)$ of the first layer and from all nodes (M, l, r) of the last layer, respectively (see Figure 3), the following result is easy to show.

Lemma 2 ([7]). *Matrix Y with rows y_1, \dots, y_M is a shape matrix satisfying width (with respect to given δ) and extended interleaf motion (with respect to given γ) constraints if and only if $P(Y)$ is a path from D to D' in G where node (m, l, r) in layer m corresponds to row m of matrix Y .*

In the example of Figure 3 the two paths correspond to the two shape matrices

$$Y^k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } Y^{k'} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Since paths in the shape matrix graph are in one-to-one correspondence with shape matrices, the scalar multiplication $\alpha_k Y^k$ in decomposition (2) is equivalent to sending α_k units of flow along path P_{Y^k} from D to D' . Hence, the DT problem is equivalent to a network flow problem.

Theorem 3 ([7]). *The decomposition time problem with respect to a given non-negative integer valued matrix A is equivalent to the decomposition network flow problem: Minimize the flow value from source D to sink D' subject to the constraints that for all $m \in \mathcal{M}$ and $n \in \mathcal{N}$, the sum of the flow through nodes (m, l, r) with $l \leq n < r$ equals the entry $a_{m,n}$. In particular, the DT problem is solvable in polynomial time.*

The polynomiality of the decomposition network flow algorithm follows, since it is a special case of a linear program. Its computation times are very short, but it generally produces a non-integer set of decomposition times as solution, while integrality is for various practical reasons a highly desirable feature in any decomposition. The authors in [7] show that there always exists an alternative integer solution, which can, in fact, be obtained by a modification of the shape matrix graph. This version of the network flow approach is, however, not numerically competitive.

An improved network flow formulation is given by [4]. A smaller network is used with $\mathcal{O}(MN)$ nodes instead of the shape matrix graph G with $\mathcal{O}(MN^2)$ nodes. This is achieved by replacing each layer of G by two sets of nodes, representing a potential left and right leaf position, respectively. An arc between two of these nodes represents a row of a C1 matrix. The resulting linear programming formulation has a coefficient matrix which can be shown to be totally unimodular, such that the linear program yields an integer solution. Numerical experiments show that this double layer approach improves the running time of the algorithm considerably.

In [5] a further step is taken by formulating a sequence of integer programs, each of which can be solved by a combinatorial algorithm, i.e., does not require any linear programming solver. The variables in these integer programs correspond to the incremental increases in decomposition time which are caused by the interleaf motion constraint. Using arguments from multicriteria optimization, the following complexity result shows that compared with the unconstrained case of Theorem 1, the complexity only worsens by a factor of M .

Theorem 4 ([5]). *The constrained decomposition time problem with (extended) interleaf and width constraint can be solved in $\mathcal{O}(NM^2)$ time.*

While the preceding approaches maintain the constraints throughout the course of the algorithm, [23, 24] solve the constrained decomposition time problem by starting with a solution of the unconstrained problem. If this solution satisfies all constraints it is obviously optimal. If the optimal solution violates the width constraint, there does not exist a solution which does. Violations of interleaf motion and tongue-and-groove constraints are eliminated by a bounded number of modification steps. A similar correction approach is taken by [32] starting from his rod-pushing and extraction algorithm for the unconstrained case.

In the paper of [22] the idea of the unconstrained algorithm of [18] is carried over to the case of interleaf motion constraints. First, a linear program (LP) is formulated with constraints (2). Hence, the LP has an exponential number of variables. Its dual is solved by a maximal path problem in an acyclic graph. The optimal dual objective value is proved to correspond to a feasible C1 decomposition, i.e., a primally feasible solution of the LP, thus establishing the optimality of the decomposition using the strong LP duality theorem.

3 Algorithms for the decomposition cardinality problem

3.1 Complexity of the DC problem

In contrast to the decomposition time problem, we cannot expect an efficient algorithm which solves the decomposition cardinality problem exactly.

Theorem 5. *The decomposition cardinality problem is strongly NP-hard even in the unconstrained case. In particular, the following results hold.*

1. [5] *The DC problem is strongly NP-hard for matrices with a single row.*
2. [14] *The DC problem is strongly NP-hard for matrices with a single column.*

The first NP-hardness proof for the DC problem is due to [9], who shows that the subset sum problem can be reduced to the DC problem. His proof applies to the case of matrices A with at least two rows. Independently, the authors in [12] use the knapsack problem to prove the (non-strong) NP-hardness in the single-row case. The stronger result of Theorem 5 uses a reduction from the 3-partition problem for the single row case. The result for single column matrices uses a reduction from a variant of the satisfiability problem, NAE-3SAT(5).

A special case, for which the DC problem can be solved in polynomial time, is considered in the next result.

Theorem 6 ([5]). *If $A = pB$ is a positive integer multiple of a binary matrix B , then the C1 decomposition cardinality problem can be solved in polynomial time for the constrained and unconstrained case.*

If A is a binary matrix, this result follows from the polynomial solvability of $DT(\alpha)$, since α_k is binary for all $k \in \mathcal{K}'$ and thus $DT(\alpha) = DC(\alpha)$. If $A = pB$ with $p > 1$, it can be shown that the DC problem for A can be reduced to the solution of the DT problem for B .

Theorem 6 is also important in the analysis of the algorithm of [33]. The main idea is to group the decomposition into phases where in phase k , only matrix elements with values $a_{mn} \geq 2^{R-k}$ are considered, i.e., the matrix elements can be represented by ones and zeros depending on whether $a_{mn} \geq 2^k$

or not ($R = \log_2(\max a_{mn})$). By Theorem 6 each of the decomposition cardinality problems can be solved in polynomial time using a DT algorithm. Hence, the Xia-Verhey algorithm runs in polynomial time and gives the best decomposition cardinality, but only among all decompositions with the same separation into phases.

In view of Theorem 5, most of the algorithms in the literature are heuristic or approximative (with performance guarantee). Most often, they guarantee minimal $DT(\alpha)$ and minimize $DC(\alpha)$ heuristically or exactly subject to DT optimality. The few algorithms that are able to solve the problem exactly have exponential running time and are limited to small instances, as evident in Section 5.

3.2 Algorithms for the unconstrained DC problem

The author in [18] applies a greedy idea to his TNMU algorithm. In each of his extraction steps $A' = A - pY$, p is computed as maximal possible value such that the pair (p, Y) is *admissible*, i.e., $a'_{mn} \geq 0$ for all m, n and $C(A') = C(A) - p$. Since the algorithm is a specialized version of Engel's decomposition time algorithm, it will only find good decomposition cardinalities among all optimal solutions of the DT problem. Note, however (see Example 1), that none of the optimal solutions of the DT problem may be optimal for the DC problem.

The author in [21] shows the validity of an algorithm which solves the lexicographic problem of finding among all optimizers of DT one with smallest decomposition cardinality DC. The complexity of this algorithm is $\mathcal{O}(MN^{2L+2})$, i.e., it is polynomial in M and N , but exponential in L (where L is a bound for the entries a_{mn} of the matrix A). It should be noted that this algorithm does not, in general, solve DC . This is due to the fact that among the optimal solutions for DT there may not be an optimal solution for DC (see Sections 4 and 5).

The idea of Kalinowski's algorithm can, however, be extended to solve DC. The main idea of this approach is to treat the decomposition time as a parameter c and to solve the problem of finding a decomposition with smallest cardinality such that its decomposition time is bounded by c . For $c = \min DT(\alpha)$, this can be done by Kalinowski's algorithm in $\mathcal{O}(MN^{2L+2})$. For $c = 1, \dots, MNL$, the author in [28] shows that the complexity increases to $\mathcal{O}((MN)^{2L+2})$. We thus have the following result.

Theorem 7 ([28]). *The problem of minimizing the decomposition cardinality $DC(\alpha)$ in an unconstrained problem can be solved in $\mathcal{O}((MN)^{2L+3})$.*

The authors in [27] present approximation algorithms for the unconstrained DC problem. They define matrices P_k whose elements are the k^{th} digits in the binary representation of the entries in A . The (easy) segmentation of P_k for $k = 1, \dots, \log L$ then results in a $\mathcal{O}(MN \log(L))$ time $(\log[L] + 1)$ -approximation algorithm for DC. They show that the performance guarantee

can be improved to $\lfloor \log D \rfloor + 1$ by choosing D as the maximum of a set of numbers containing all absolute differences between any two consecutive row entries over all rows and the first and last entries of each row. In the context of approximation algorithms we finally mention the following result by [6].

Theorem 8. *The DC problem is APX-hard even for matrices with a single row with entries polynomially bounded in N .*

3.3 Algorithms for the constrained DC problem

A similar idea as in [18] is used in [5] for the constrained decomposition cardinality problem. Data from the solution of the DT problem (see Section 2) is used as input for a greedy extraction procedure. The author in [22] also generalizes the idea of Engel to the case of DC problems with interleaf motion constraints.

The authors in [10–13] consider the decomposition cardinality problem with interleaf motion, width, and tongue-and-groove constraints. The first two groups of constraints are considered by a geometric argumentation. The given matrix A is — similar to [32] — interpreted as a 3-dimensional set of rods, or as they call it a *3D-mountain*, where the height of each rod is determined by the value of its corresponding matrix entry a_{mn} . The decomposition is done by a *mountain reduction technique*, where tongue-and-groove constraints are taken into consideration using a graph model. The underlying graph is complete with its node set corresponding to all feasible C1 matrices. The weight of the edges is determined by the tongue-and-groove error occurring if both matrices are used in a decomposition. Matching algorithms are used to minimize the tongue-and-groove error. In order to speed up the algorithm, smaller graphs are used and the optimal matchings are computed using a network flow algorithm in a sparse graph.

The authors in [19] propose a difference-matrix metaheuristic to obtain solutions with small DC as well as small DT values. The metaheuristic uses a multiple start local search with a heuristic that sequentially extracts segments Y_k based on results of [18]. They consider multiple constraints on the segments, including interleaf and innerleaf motion constraints. Reported results clearly outperform the heuristics implemented in the Elekta MLC system.

4 Combined objective functions

A first combination of decomposition time and cardinality problems is the treatment time problem with constant set-up times $TT(\alpha) := DT(\alpha) + SU(\alpha) = DT(\alpha) + \tau DC(\alpha)$. For τ suitably large, it is clear that the DC problem is a special case of the TT problem. Thus the latter is strongly NP-hard due to Theorem 5.

The most versatile approach to deal with the TT problem including different kinds of constraints, is by integer programming as done by [25]. They first formulate the decomposition time problem as an integer linear program (IP), where interleaf motion, width, or tongue-and-groove constraints can easily be written as linear constraints. The optimal objective $z = DT(\alpha)$ can then be used in a modified IP as upper bound for the decomposition time which is now treated as variable (rather than objective) and in which the number of C1 matrices is to be minimized. This approach can be considered as an ε -constraint method to solve bicriteria optimization problems (see, for instance, [16]). The solutions in [25] can thus be interpreted as Pareto optimal solutions with respect to the two objective functions $DT(\alpha)$ and $DC(\alpha)$. Due to the large number of variables, the algorithm presented in [25] is, however, not usable for realistic problem instances.

The importance of conflict between the DT and DC objectives has not been investigated to a great extent. The author in [3] showed that for matrices with a single row there is always a decomposition that minimizes both $DC(\alpha)$ and $DT(\alpha)$. The following examples show that the optimal solutions of the (unconstrained) DT , DC and TT_{var} problems are in general attained in different decompositions. As a consequence, it is not enough to find the best possible decomposition cardinality among all decompositions with minimal decomposition time as is done in most papers on the DC problem (see Section 3). We will present next an example which is the smallest possible one for different optimal solutions of the DT and DC problems.

Example 1. Let

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

Since the entries $1, \dots, 6$ can only be uniquely represented by the numbers $1, 2$ and 4 , the unique optimal decomposition of the DC problem is given by $A = 1Y^1 + 2Y^2 + 4Y^3$ where

$$Y^1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}, Y^2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \text{and} \quad Y^3 = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Hence, the optimal value of the DC problem is 3, with $DT = 7$. Since the optimal solution of the DT problem has $DT = 6$, we conclude that $DC \geq 4$.

It is not clear whether this example is of practical value. In Section 5 we see that in our tests the optimal solution of the DC problem examples was not among the DT optimal solutions in only 5 out of 32 examples. In these cases the difference in the DC objective was only 1. This is also emphasized by [26] who confirm that the conflict between DT and DC is often small in practice.

Another possible combination of objective functions is the treatment time problem with variable set-up time $TT_{var}(\alpha) := DT(\alpha) + SU_{var}(\alpha) = DT(\alpha) + \sum_{k=1}^{K-1} \tau_{\pi(k), \pi(k+1)}$ (see (6)). Minimizing $TT_{var}(\alpha)$ is strongly NP-hard when looking at the special case $\tau_{kl} = \tau$ for all k, l , which yields the objective function of $TT_{const}(\alpha)$. Here, we consider

$$\tau_{\pi(k), \pi(k+1)} = \max_{m \in \mathcal{M}} \max \left\{ |l_m^{\pi(k)} - l_m^{\pi(k+1)}|, |r_m^{\pi(k)} - r_m^{\pi(k+1)}| \right\}, \quad (14)$$

i.e., the maximal number of positions any leave moves between two consecutive matrices $Y^{\pi(k)}$ and $Y^{\pi(k+1)}$ in the sequence.

Extending Example 1, the following example shows that the three objective functions $DT(\alpha)$, $DC(\alpha)$, and $TT_{var}(\alpha)$ yield, in general, different optimal solutions.

Example 2. Let

$$A = \begin{pmatrix} 8 & 5 & 6 \\ 5 & 3 & 6 \end{pmatrix}.$$

The optimal decomposition for DC is

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

This decomposition yields $DT = 14$, $DC = 3$ and $TT_{var} = DT + SU_{var} = 14 + 3 = 17$, where $SU_{var} = 1 + 2 = 3$. The optimal decomposition for DT is

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

Here we obtain $DT = 9$, $DC = 4$, $SU_{var} = 2 + 2 + 2 = 6$ and thus $TT_{var} = 15$. The optimal decomposition for TT_{var} is

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

We get $DT = 10$, $DC = 4$ and $SU_{var} = 2 + 1 + 1 = 4$, leading to $TT_{var} = 14$.

If the set of C1 matrices Y^1, \dots, Y^K in the formulation $TT_{var}(\alpha)$ is given, one can apply a traveling salesman algorithm to minimize $SU_{var}(\alpha)$. Since the number L of C1 matrices is in general rather small, the TSP can be solved exactly in reasonable time. If the set of C1 matrices is not given, the problem becomes a simultaneous decomposition and sequencing problem which is currently under research.

5 Numerical results

Very few numerical comparisons are available in the literature. The author in [29] compares in his numerical investigations eight different heuristics for the DC problem. He concludes that the Algorithm of Xia and Verhey [33] outperforms its competitors. With new algorithms developed since the appearance of Que's paper, the dominance of the Xia-Verhey algorithm is no longer true, as observed by [15] and seen below.

In this section we present results obtained with the majority of algorithms mentioned in this paper for constrained and unconstrained problems. We consider only interleaf motion constraints, since these are the most common and incorporated in most algorithms. As seen in Section 2 the unconstrained and constrained DT problems can be solved in $\mathcal{O}(NM)$, respectively $\mathcal{O}(NM^2)$ time. Moreover, we found that algorithms that guarantee minimal $DT(\alpha)$ and include a heuristic to reduce $DC(\alpha)$ do not require significantly higher CPU time. Therefore we exclude algorithms that simply minimize $DT(\alpha)$ without control over $DC(\alpha)$. Table 1 shows the references for the algorithms, and some remarks on their properties.

We used 47 clinical examples varying in size from 5 to 23 rows and 6 to 30 columns, with L varying between 9 and 40. In addition, we used 15 instances of size 10×10 with entries randomly generated between 1 and 14. In all experiments we have applied an (exact) TSP algorithm to the resulting matrices to minimize the total treatment time for the given decomposition. Table 2 presents the results for the unconstrained and Table 3 presents those for the constrained problems. All experiments were run on a Pentium 4 PC with 2.4 GHz and 512 MB RAM. In both tables we first show the number of instances for which the algorithms gave the best values for DT , DC and TT_{var} after application of the TSP to the matrices produced by the algorithms. Next, we list the maximal CPU time (in seconds) the algorithm took for any of the instances. The next four rows show the minimum, maximum, median, and average relative deviation from the best DC value found by any of the algorithms. The next four rows show the same for TT_{var} . Finally, we list the improvement of variable setup time according to (14) obtained by applying the TSP to the matrices found by the algorithms.

Table 1. List of algorithms tested.

Algorithm	Problem	Remarks
Baatar et al. [5]	unconstrained	guarantees min DT , heuristic for DC
Engel [18]	unconstrained	guarantees min DT , heuristic for DC
Xia and Verhey [33]	unconstrained	heuristic for DC
Baatar et al. [5]	constrained	guarantees min DT , heuristic for DC
Kalinowski [22]	constrained	guarantees min DT , heuristic for DC
Siochi [32]	constrained	guarantees min DT , heuristic for TT
Xia and Verhey [33]	constrained	heuristic for DC

Table 2. Numerical results for the unconstrained algorithms.

		Baatar et al. [5]	Engel [18]	Xia and Verhey [33]
Best DT		62	62	0
Best DC		7	62	1
Best TT_{var}		38	17	9
Best CPU		0	21	45
Max CPU		0.1157	0.0820	0.0344
ΔDC	Min	0.00%	0.00%	0.00%
	Max	33.33%	0.00%	86.67%
	Median	18.18%	0.00%	36.93%
	Mean	17.08%	0.00%	37.82%
ΔTT	Min	0.00%	0.00%	0.00%
	Max	21.30%	42.38%	83.82%
	Median	0.00%	5.66%	14.51%
	Mean	3.14%	8.74%	17.23%
ΔSU	Min	0.83%	1.43%	7.89%
	Max	37.50%	27.27%	43.40%
	Median	14.01%	10.46%	25.41%
	Mean	13.91%	12.15%	25.74%

Table 3. Numerical results for the constrained algorithms.

		Baatar et al. [5]	Kalinowski [22]	Siochi [32]	Xia and Verhey [33]
Best DT		62	62	62	0
Best DC		1	62	1	0
Best TT_{var}		12	43	11	0
Best CPU		0	0	0	62
Max CPU		0.2828	0.8071	1.4188	0.0539
ΔDC	Min	0.00%	0.00%	0.00%	11.11%
	Max	160.00%	0.00%	191.67%	355.56%
	Median	70.71%	0.00%	108.12%	70.71%
	Mean	71.37%	0.00%	102.39%	86.58%
ΔTT	Min	0.00%	0.00%	0.00%	10.66%
	Max	50.74%	45.28%	26.47%	226.42%
	Median	5.23%	0.00%	8.49%	51.03%
	Mean	7.97%	4.95%	8.26%	61.56%
ΔSU	Min	0.00%	2.27%	0.00%	5.00%
	Max	18.18%	35.25%	20.00%	24.05%
	Median	4.45%	22.45%	2.11%	14.20%
	Mean	5.34%	21.66%	3.24%	14.42%

Table 4. Comparison of Kalinowski [21] and Nußbaum [28]. A * next to the *DC* value indicates a difference between the algorithms.

Data Sets	Kalinowski [21]				Nußbaum [28]			
	DT	DC	TT	CPU	DT	DC	TT	CPU
Clinical 1	27	7	49	0	27	7	49	0
Clinical 2	27	6	43	1	27	6	43	1
Clinical 3	24	8	59	2	28	7*	56	213
Clinical 4	33	6	48	1	33	6	48	1
Clinical 5	41	9	76	41	44	8*	73	134
Clinical 6	13	8	125	8	13	8	125	8
Clinical 7	12	9	134	27	12	9	134	27
Clinical 8	12	8	153	15	12	8	153	15
Clinical 9	12	9	118	174	12	9	118	174
Clinical 10	11	9	108	133	11	9	108	133
Clinical 11	11	6	97	0	11	6	97	0
Clinical 12	10	7	99	0	10	7	99	0
Clinical 14	17	8	48	0	17	8	48	0
Clinical 15	19	7	54	0	19	7	54	0
Clinical 16	15	7	46	0	15	7	46	0
Clinical 17	16	7	48	0	16	7	48	0
Clinical 18	20	8	50	4	20	8	50	9
Clinical 19	16	7	51	0	16	7	51	0
Clinical 20	18	7	47	0	18	7	47	0
Clinical 21	22	8	65	1	22	8	65	1
Clinical 22	22	10	74	10	25	9*	81	23
Clinical 23	26	9	76	24	26	9	76	24
Clinical 24	23	9	63	6	23	9	63	7
Clinical 25	23	9	75	12	23	9	75	13
Clinical 26	22	9	68	2	22	9	68	2
Clinical 39	28	10	88	149	28	10	88	149
Clinical 40	26	8	60	2	27	7*	55	3
Clinical 41	20	7	46	1	20	7	46	1
Clinical 42	23	8	55	0	23	8	55	0
Clinical 45	21	6	42	0	21	6	42	0
Clinical 46	19	9	65	10	21	8*	59	40
Clinical 47	24	10	85	1	24	10	85	1

Table 2 shows that Xia and Verhey [33] is the fastest algorithm. However, it never found the optimal *DT* value and found the best *DC* value for only one instance. Since the largest CPU time is 0.116 seconds, computation time is not an issue. Thus we conclude that Xia and Verhey [33] is inferior to the other algorithms. Baatar et al. [5] and Engel [18] are roughly equal in speed. Both guarantee optimal *DT*, but the latter performs better in terms of *DC*, finding the best value for all instances. However, the slightly greater amount of matrices used by the former method appears to enable better TT_{var} values

and a slightly bigger improvement of the variable setup time by reordering the segments. We observe that applying a TSP algorithm is clearly worthwhile, reducing the variable setup time by up to 40%.

The results for the constrained problems underline that the algorithm of [33], despite being the fastest for all instances, is not competitive. It did not find the best DT , DC , or TT_{var} values for any example. The other three algorithms guarantee DT optimality. The algorithm of [22] performs best, finding the best DC value in all cases, and the best TT_{var} value in 43 of the 62 tests. Baatar et al. [5] and Siochi [32] are comparable, with the former being slightly better in terms of DC , TT_{var} and CPU time. Again, the application of a TSP algorithm is well worth the effort to reduce the variable setup time.

Finally, the results of comparing the algorithm of [21] with its new iterative version of [28] on a subset of the clinical instances are given in Table 4. These tests were performed on a PC with Dual Xeon Processor, 3.2 GHz and 4 GB RAM. In the comparison of 32 clinical cases there were only five cases (3, 5, 22, 40, 46) where the optimal solution of the DC problem was not among the optimal solutions of the DT problem — and thus found by the algorithm of [21]. In these five cases, the DC objective was only reduced by a value of 1. Since the iterative algorithm performs at most $NML - DT$ applications of Kalinowski-like procedures, the CPU time is obviously considerably larger.

Acknowledgements

The authors thank David Craigie, Zhenzhen Mu, and Dong Zhang, who implemented most of the algorithms, and Thomas Kalinowski for providing the source code of his algorithms.

References

1. R. K. Ahuja and H. W. Hamacher. A network flow algorithm to minimize beam-on-time for unconstrained multileaf collimator problems in cancer radiation therapy. *Networks*, 45:36–41, 2004.
2. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.
3. D. Baatar. *Matrix Decomposition with Time and Cardinality Objectives: Theory, Algorithms, and Application to Multileaf Collimator Sequencing*. PhD thesis, Department of Mathematics, Technical University of Kaiserslautern, 2005.
4. D. Baatar and H. W. Hamacher. New LP model for multileaf collimators in radiation therapy planning. In *Proceedings of the Operations Research Peripatetic Postgraduate Programme Conference ORP³, Lambrecht, Germany*, pages 11–29, 2003.
5. D. Baatar, H. W. Hamacher, M. Ehrgott, and G. J. Woeginger. Decomposition of integer matrices and multileaf collimator sequencing. *Discrete Applied Mathematics*, 152:6–34, 2005.

6. Nikhil Bansal, Don Coppersmith, and Baruch Schieber. Minimizing setup and beam-on times in radiation therapy. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, volume 4110 of *Lecture Notes in Computer Science*, pages 27–38. Springer Verlag, Berlin, 2006.
7. N. Boland, H. W. Hamacher, and F. Lenzen. Minimizing beam-on-time in cancer radiation treatment using multileaf collimators. *Networks*, 43:226–240, 2004.
8. T. R. Bortfeld, A. L. Boyer, D. L. Kahler, and T. J. Waldron. X-ray field compensation with multileaf collimators. *International Journal of Radiation Oncology, Biology, Physics*, 28:723–730, 1994.
9. R. E. Burkard. Open Problem Session, Oberwolfach Conference on Combinatorial Optimization, November 24–29, 2002.
10. D. Z. Chen, X. S. Hu, S. Luan, C. Wang, S. A. Naqvi, and C. X. Yu. Generalized geometric approaches for leaf sequencing problems in radiation therapy. In *Proceedings of the 15th Annual International Symposium on Algorithms and Computation (ISAAC), Hong Kong, December 2004*, volume 3341 of *Lecture Notes in Computer Science*, pages 271–281. Springer Verlag, Berlin, 2004.
11. D. Z. Chen, X. S. Hu, S. Luan, C. Wang, S. A. Naqvi, and C. X. Yu. Generalized geometric approaches for leaf sequencing problems in radiation therapy. *International Journal of Computational Geometry and Applications*, 16(2-3):175–204, 2006.
12. D. Z. Chen, X. S. Hu, S. Luan, C. Wang, and X. Wu. Geometric algorithms for static leaf sequencing problems in radiation therapy. *International Journal of Computational Geometry and Applications*, 14:311–339, 2004.
13. D. Z. Chen, X. S. Hu, S. Luan, X. Wu, and C. X. Yu. Optimal terrain construction problems and applications in intensity-modulated radiation therapy. *Algorithmica*, 42:265–288, 2005.
14. M. J. Collins, D. Kempe, J. Saia, and M. Young. Nonnegative integral subset representations of integer sets. *Information Processing Letters*, 101:129–133, 2007.
15. S. M. Crooks, L. F. McAven, D. F. Robinson, and L. Xing. Minimizing delivery time and monitor units in static IMRT by leaf-sequencing. *Physics in Medicine and Biology*, 47:3105–3116, 2002.
16. M. Ehrgott. *Multicriteria Optimization*. Springer Verlag, Berlin, 2nd edition, 2005.
17. M. Ehrgott, A. Holder, and J. Reese. Beam selection in radiotherapy design. *Linear Algebra and its Applications*, doi: 10.1016/j.laa.2007.05.039, 2007.
18. K. Engel. A new algorithm for optimal MLC field segmentation. *Discrete Applied Mathematics*, 152:35–51, 2005.
19. A. D. A. Gunawardena, W. D’Souza, L. D. Goadrick, R. R. Meyer, K. J. Sorensen, S. A. Naqvi, and L. Shi. A difference-matrix metaheuristic for intensity map segmentation in step-and-shoot imrt delivery. *Physics in Medicine and Biology*, 51:2517–2536, 2006.
20. H. W. Hamacher and K.-H. Küfer. Inverse radiation therapy planing – A multiple objective optimization approach. *Discrete Applied Mathematics*, 118:145–161, 2002.

21. T. Kalinowski. Algorithmic complexity of the minimization of the number of segments in multileaf collimator field segmentation. Technical report, Department of Mathematics, University of Rostock, 2004. Preprint 2004/1.
22. T. Kalinowski. A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Applied Mathematics*, 152:52–88, 2005.
23. S. Kamath, S. Sahni, J. Li, J. Palta, and S. Ranka. Leaf sequencing algorithms for segmented multileaf collimation. *Physics in Medicine and Biology*, 48:307–324, 2003.
24. S. Kamath, S. Sahni, S. Ranka, J. Li, and J. Palta. A comparison of step-and-shoot leaf sequencing algorithms that eliminate tongue-and-groove effects. *Physics in Medicine and Biology*, 49:3137–3143, 2004.
25. M. Langer, V. Thai, and L. Papiez. Improved leaf sequencing reduces segments of monitor units needed to deliver IMRT using MLC. *Medical Physics*, 28:2450–58, 2001.
26. M. P. Langer, V. Thai, and L. Papiez. Tradeoffs between segments and monitor units are not required for static field IMRT delivery. *International Journal of Radiation Oncology, Biology, Physics*, 51:75, 2001.
27. S. Luan, J. Saia, and M. Young. Approximation algorithms for minimizing segments in radiation therapy. *Information Processin Letters*, 101:239–244, 2007.
28. M. Nußbaum. Min cardinality $c1$ -decomposition of integer matrices. Master’s thesis, Department of Mathematics, Technical University of Kaiserslautern, 2006.
29. W. Que. Comparison of algorithms for multileaf collimator field segmentation. *Medical Physics*, 26:2390–2396, 1999.
30. L. Shao. A survey of beam intensity optimization in imrt. In T. Halliburton, editor, *Proceedings of the 40th Annual Conference of the Operational Research Society of New Zealand, Wellington, 2-3 December 2005*, pages 255–264, 2005. Available online at <http://secure.orsnz.org.nz/conf40/content/paper/Shao.pdf>.
31. D. M. Shepard, M. C. Ferris, G. H. Olivera, and T. R. Mackie. Optimizing the delivery of radiation therapy to cancer patients. *SIAM Review*, 41:721–744, 1999.
32. R. A. C. Siochi. Minimizing static intensity modulation delivery time using an intensity solid paradigm. *International Journal of Radiation Oncology, Biology, Physics*, 43:671–689, 1999.
33. P. Xia and L. Verhey. Multileaf collimator leaf sequencing algorithm for intensity modulated beams with multiple segments. *Medical Physics*, 25:1424–1434, 1998.

Appendix: The instances

Tables 5 and 6 show the size (N, M, L) of the instances, the optimal value of $DT(\alpha)$ in the constrained and unconstrained problems, and the best $DC(\alpha)$ and $TT_{var}(\alpha)$ values found by any of the tested algorithms, with a * indicating proven optimality for DC in the unconstrained case.

Table 5. The 15 random instances.

Data Set	Size			Unconstrained			Constrained		
	M	N	L	DT	DC	TT	DT	DC	TT
Random 1	10	10	14	37	11	107	39	16	110
Random 2	10	10	14	30	11	102	33	13	100
Random 3	10	10	14	36	11	103	37	16	106
Random 4	10	10	14	37	11	114	37	12	99
Random 5	10	10	14	46	12	120	46	16	107
Random 6	10	10	14	45	12	123	45	14	112
Random 7	10	10	14	41	11	117	47	16	122
Random 8	10	10	14	41	12	119	41	15	106
Random 9	10	10	14	33	11	102	33	13	98
Random 10	10	10	14	34	10	94	40	15	102
Random 11	10	10	14	41	11	113	41	14	102
Random 12	10	10	14	35	11	106	37	15	102
Random 13	10	10	14	32	11	105	32	13	99
Random 14	10	10	14	43	11	114	43	18	112
Random 15	10	10	14	36	10	109	37	14	107

Table 6. The 47 clinical instances.

Data Set	Size			Unconstrained			Constrained		
	M	N	L	DT	DC	TT	DT	DC	TT
Clinical 1	5	6	23	27	7*	49	27	8	51
Clinical 2	5	7	27	27	6*	43	27	8	48
Clinical 3	5	8	18	24	7*	54	24	8	53
Clinical 4	5	7	30	33	6*	48	33	8	51
Clinical 5	5	8	25	41	8*	73	41	10	73
Clinical 6	16	29	10	13	8*	125	13	9	132
Clinical 7	16	27	10	12	9*	122	12	9	138
Clinical 8	16	30	10	12	8*	135	15	11	163
Clinical 9	15	28	9	12	9*	118	12	9	151
Clinical 10	16	28	10	11	9*	108	11	10	106
Clinical 11	20	23	10	11	6*	96	11	7	136
Clinical 12	16	28	10	10	7*	99	13	10	130
Clinical 13	20	25	9	17	11	151	17	12	130
Clinical 14	9	9	10	17	8*	38	17	9	50
Clinical 15	9	10	10	19	7*	53	19	11	62
Clinical 16	10	9	10	15	7*	44	18	9	50
Clinical 17	10	9	10	16	7*	45	16	8	47
Clinical 18	9	9	10	20	8*	50	20	10	49
Clinical 19	10	10	10	16	7*	50	16	8	57
Clinical 20	10	9	10	18	7*	47	18	9	56

(continued)

Table 6. Continued.

Data Set	Size			Unconstrained			Constrained		
	M	N	L	DT	DC	TT	DT	DC	TT
Clinical 21	14	10	10	22	8*	65	23	13	73
Clinical 22	14	10	10	22	9*	74	22	11	79
Clinical 23	14	10	10	26	9*	76	30	15	89
Clinical 24	14	10	10	23	9*	63	24	11	79
Clinical 25	14	10	10	23	9*	74	23	12	84
Clinical 26	14	10	10	22	9*	68	22	10	70
Clinical 27	22	23	24	33	14	165	34	17	158
Clinical 28	23	17	27	46	15	184	46	17	186
Clinical 29	23	16	33	35	12	155	48	17	174
Clinical 30	22	21	31	50	15	197	58	21	196
Clinical 31	22	22	22	47	15	205	58	21	201
Clinical 32	22	15	26	33	11	134	42	16	142
Clinical 33	22	18	24	41	13	186	41	18	175
Clinical 34	9	13	29	45	13	147	45	15	129
Clinical 35	9	10	40	59	11	103	69	14	124
Clinical 36	9	12	26	45	12	131	45	14	111
Clinical 37	9	10	35	46	11	115	46	12	116
Clinical 38	11	11	19	35	9	68	35	10	79
Clinical 39	11	11	22	28	10*	84	33	14	91
Clinical 40	11	12	19	26	7*	55	27	10	75
Clinical 41	11	9	16	20	7*	46	22	8	52
Clinical 42	11	9	14	23	8*	55	23	10	58
Clinical 43	11	12	26	43	11	101	49	16	119
Clinical 44	10	15	26	49	13	137	54	16	114
Clinical 45	11	8	21	21	6*	48	21	9	49
Clinical 46	11	12	16	19	8*	42	19	10	66
Clinical 47	11	14	22	24	10*	59	38	15	105

Optimization in Medicine

Alves, C.J.S.; Pardalos, P.; Vicente, L.N. (Eds.)

2008, X, 195 p. 60 illus., Hardcover

ISBN: 978-0-387-73298-5