Mixed-Integer Programming Techniques for Decomposing IMRT Fluence Maps Using Rectangular Apertures

Z. Caner Taşkın[*] J. Cole Smith[†] H. Edwin Romeijn[‡]

March 31, 2008

Abstract

We consider a matrix decomposition problem arising in Intensity Modulated Radiation Therapy (IMRT). The problem input is a matrix of intensity values that are to be delivered to a patient via IMRT from some given angle, under the condition that the IMRT device can only deliver radiation in rectangular shapes. This paper studies the problem of minimizing the number of rectangles (and their associated intensities) necessary to decompose such a matrix. We propose an integer programming-based methodology for providing lower and upper bounds on the optimal solution, and demonstrate the efficacy of our approach on actual clinical data.

[*]Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, P.O. Box 116595, Gainesville, Florida 32611-6595; e-mail: `taskin@ufl.edu`.

[†]Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, P.O. Box 116595, Gainesville, Florida 32611-6595; e-mail: `cole@ise.ufl.edu`.

[‡]Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, P.O. Box 116595, Gainesville, Florida 32611-6595; e-mail: `romeijn@ise.ufl.edu`. The work of this author was supported by the National Science Foundation under grant no. DMI-0457394.

1 Introduction and Literature Survey

Over the past decade, Intensity Modulated Radiation Therapy (IMRT) has developed into the most successful external-beam radiation therapy delivery technique for many forms of cancer. This is due to its ability to deliver highly complex dose distributions to cancer patients that enable the eradication of cancerous cells while limiting damage to nearby healthy organs and tissues. Patients treated with IMRT therefore often experience a higher chance of cure, suffer from fewer side effects of the treatment, or both. In this paper, we will study an optimization problem that is related to the efficient clinical implementation of IMRT using a simpler technology than currently used, which, if successful, will reduce the cost as well as the complexity of delivering IMRT, and thereby make such superior treatments accessible to significantly more patients worldwide.

External-beam radiation therapy is delivered from multiple angles by a device that can rotate around a patient. The use of multiple (typically 3–9) angles is one of the tools that allow for the treatment of deep-seated tumors while limiting the radiation dose to surrounding functioning organs. Conventional conformal radiation therapy then further uses blocks and wedges to shape the beams (see, e.g., Lim [14] and Lim et al. [15, 16]). IMRT is a more powerful therapy that instead modulates beam intensity. The most common technique for achieving this modulation is to dynamically shape beams with the help of a multileaf collimator (MLC) system. Such systems can dynamically form many complex apertures by independently moving leaf pairs that block part of the radiation beam. Unfortunately, MLC systems are very costly and technologically advanced, and are therefore difficult and expensive to operate and maintain. Moreover, MLC systems are currently only available for use with a so-called linear accelerator that generates high-energy photon beams for treatment. However, the use of radioactive ^{60}Co (Cobalt) sources for radiation therapy is still ubiquitous in many parts of the world, and is poised to experience a revival in the United States and Europe through the Renaissance$^{\text{TM}}$ device that is under development by ViewRay, Inc. based in Cleveland, Ohio. Without a MLC, IMRT delivery may be achieved through the use of compensators: high-density blocks that control the intensity profile of a radiation beam.

Such blocks are custom-made for each individual patient, which makes compensator-based IMRT not only labor and storage space intensive, but it also makes the actual treatment very time-consuming due to the fact that therapists must enter the treatment room to place each individual compensator. In addition, compensators have several undesirable properties that make it difficult to perform accurate dose calculations, thereby reducing the advantages of IMRT (see, e.g., Earl et al. [5]). Recently, researchers have begun to explore the clinical feasibility of delivering IMRT using conventional jaws that are already integrated into radiation delivery devices and can create apertures that are rectangular in shape (see, e.g., Earl et al. [5], Kim et al. [10], and Men et al. [17]). Successful application of this much simpler delivery technique depends critically on the ability to *efficiently* deliver high-quality treatment plans. In this paper, we therefore develop and test new optimization approaches to minimize the treatment time required for a particular treatment plan using rectangular apertures only.

Solving a so-called fluence map optimization problem yields an optimal IMRT treatment plan that resolves different, and conflicting, clinical measures of treatment plan quality related to tumor control and side effects (see, e.g., Shepard et al. [22] for a review; Lee et al. [12, 13] for mixed-integer programming approaches; Romeijn et al. [21] for convex programming models; and Hamacher and Küfer [7] and Küfer et al. [11] for a multicriteria approach). A treatment plan then consists of a collection of nonnegative intensity matrices, often referred to as fluence maps, one corresponding to each beam angle. In order to limit treatment time, each of these matrices is then expressed as a multiple of an integral fluence map in which the maximum element is on the order of 10–20. To allow delivery of the treatment plan, each of these fluence maps should be decomposed into a number of apertures and corresponding intensities, where the collection of apertures that may be used depends on the delivery equipment. For MLC delivery this problem is called the leaf sequencing problem and is very widely studied; for examples, we refer to Ahuja and Hamacher [1], Boland et al. [3], Kamath et al. [9], Engel [6], Kalinowski [8], and Taşkın et al. [23]. (Note that integrated approaches to fluence map optimization, also referred to as aperture modulation, have been proposed as well; we refer to, e.g., Preciado-Walters et al. [19], Romeijn et al. [20], and Men et al. [17].)

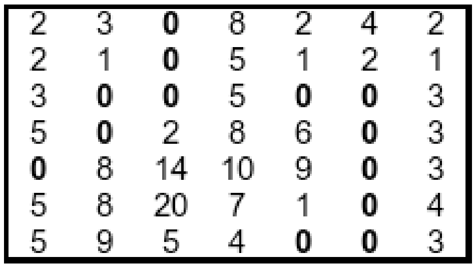
The problem that we study in this paper is the decomposition of an integral fluence map into rectangular apertures and corresponding intensities. While Dai and Hu [4] proposed a straightforward heuristic for a variant of this decomposition problem, we develop the first computationally viable optimization approach to this problem. In Section 2 we consider the core problem of decomposing an (integral) fluence map while minimizing the number of rectangular apertures. In Section 3 we then extend our models to the problems of (i) minimizing total treatment time (as measured by the sum of the required aperture setup times and the beam-on-time, i.e., the actual time that radiation is being delivered); and (ii) minimizing the number of apertures subject to beam-on-time being minimal. Section 4 discusses our computational results on a collection of clinical fluence maps, and we conclude the paper in Section 5.

2 A Mixed-Integer Programming Approach

We begin in Section 2.1 by formally describing the optimization model under investigation, and modeling it with a mixed-integer programming formulation. We next describe several classes of valid inequalities in Section 2.2. Finally, we discuss methods for partitioning the input matrix in Section 2.3, which leads to effective lower and upper bounding techniques.

2.1 Model Development

In this section, we discuss an integer programming approach to decomposing a fluence map into a minimum number of rectangular apertures and corresponding intensities. Throughout this paper, we will denote the fluence map to be delivered by a matrix $B \in \mathbb{N}^{m \times n}$, where the element at row i and column j, (i, j), corresponds to a bixel with required intensity b_{ij}. We call a bixel having an intensity requirement of zero a *zero-bixel*. We also define a *nonzero-bixel* analogously. Figure 1 shows an example fluence map, which we will use throughout this paper.



2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 1: Example fluence map

Let R be the set of all $O(n^2m^2)$ possible rectangular apertures (i.e., submatrices of B having contiguous rows and columns) that can be used to decompose B, excluding those that contain a zero-bixel. For each rectangle $r \in R$ we define a continuous variable x_r that represents the intensity assigned to rectangle r, and a binary variable y_r that equals 1 if rectangle r is used in decomposing B (i.e., if $x_r > 0$), and equals 0 otherwise. Let C_r be the set of bixels that is exposed by rectangle r. We define $M_r = \min_{(i,j)\in C_r}\{b_{ij}\}$ to be the minimum intensity requirement among the bixels covered by rectangle r. Furthermore, we denote the set of rectangles that cover bixel (i, j) by $R(i, j)$. Given these definitions, we can formulate the problem as follows:

$$\textbf{IPR:}\ \text{Minimize} \sum_{r\in R} y_r \tag{1}$$

$$\text{subject to:} \sum_{r\in R(i,j)} x_r = b_{ij} \quad \forall i = 1,\ldots,m,\ j = 1,\ldots,n \tag{2}$$

$$x_r \le M_r y_r \quad \forall r \in R \tag{3}$$

$$x_r \ge 0,\ y_r \text{ binary} \quad \forall r \in R. \tag{4}$$

The objective function (1) minimizes the number of rectangles used in the decomposition. Constraints (2) guarantee that each bixel receives exactly the required dose. Constraints (3) enforce the condition that x_r cannot be positive unless $y_r = 1$. Finally, (4) states bounds and logical restrictions on the variables. Note that the objective (1) guarantees that $y_r = 0$ when $x_r = 0$ in any optimal solution of IPR.

Formulation IPR contains two variables and a constraint for each rectangle, resulting in a large-scale mixed-integer program for problem instances of clinically relevant sizes. Furthermore, the M_r-terms in constraints (3) lead to a weak linear programming relaxation; with no valid inequalities or branching yet performed on the problem, we have that $y_r = x_r/M_r$ at optimality to the linear programming relaxation of IPR. An alternative formulation that does not require M_r-terms employs a decomposition method. Taşkın et al. [23] investigate the problem of decomposing an integer matrix into "row-convex" matrices, where in each decomposed matrix, all nonzero values take the same value, and appear consecutively on each row. Their computational results show that solvability of the problem is significantly improved by applying a bilevel optimization algorithm. A similar approach for the problem considered in this paper would formulate a master problem as:

$$\textbf{MP:}\ \text{Minimize} \sum_{r\in R} y_r \tag{5}$$

$$\text{subject to: } \mathbf{y} \text{ corresponds to a feasible decomposition} \tag{6}$$

$$y_r \text{ binary} \quad \forall r \in R, \tag{7}$$

where we address the form of (6) in the sequel. Given a vector $\hat{\mathbf{y}}$, we can check whether constraint (6) is satisfied by solving the following linear program:

$$\textbf{SP}(\hat{\mathbf{y}}):\ \text{Minimize } 0 \tag{8}$$

$$\text{subject to: } \sum_{r\in R(i,j)} x_r = b_{ij} \quad \forall i = 1,\ldots,m,\ j = 1,\ldots,n \tag{9}$$

$$x_r \le M_r \hat{y}_r \quad \forall r \in R \tag{10}$$

$$x_r \ge 0 \quad \forall r \in R. \tag{11}$$

Associating variables α_{ij} with (9), and β_r with (10), we obtain the dual formulation:

$$\textbf{DSP}(\hat{\mathbf{y}}):\ \text{Maximize} \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}\alpha_{ij} + \sum_{r\in R} M_r \hat{y}_r \beta_r \tag{12}$$

$$\text{subject to: } \sum_{(i,j)\in C_r} \alpha_{ij} + \beta_r \ge 0 \quad \forall r \in R \tag{13}$$

$$\alpha_{ij} \text{ unrestricted} \quad \forall i = 1, \ldots, m,\ j = 1, \ldots, n \tag{14}$$

$$\beta_r \leq 0 \quad \forall r \in R. \tag{15}$$

Our Benders decomposition strategy first solves MP, which yields $\hat{\mathbf{y}}$. If SP($\hat{\mathbf{y}}$) is feasible, then $\hat{\mathbf{y}}$ corresponds to a feasible decomposition, and is optimal. Else, DSP($\hat{\mathbf{y}}$) is unbounded (since the trivial all-zero solution guarantees its feasibility). Let ($\hat{\boldsymbol{\alpha}}$, $\hat{\boldsymbol{\beta}}$) be an extreme dual ray of DSP($\hat{\mathbf{y}}$) such that $\sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij}\hat{\alpha}_{ij} + \sum_{r \in R} M_r \hat{y}_r \hat{\beta}_r > 0$. Then, all $\mathbf{y}$-vectors that are feasible with respect to (6) must satisfy

$$\sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij}\hat{\alpha}_{ij} + \sum_{r \in R} (M_r \hat{\beta}_r) y_r \leq 0. \tag{16}$$

We add (16) in a cutting-plane fashion as necessary.

Remark 1. Even though the number of rectangles that can be used in partitioning the input matrix B is $O(n^2m^2)$, we observe that optimal solutions typically use only a small percentage of the total number of rectangles. This observation suggests that another way to overcome the dimensional complexity associated with solving IPR is to apply a column generation approach. In this approach, we start with a feasible set of columns and rows corresponding to a subset of rectangles, and generate additional columns and rows as necessary within a branch-and-price-and-cut (BCP) algorithm. Even though this approach requires the solution of much smaller linear programming relaxations, several features of the branch-and-cut algorithm such as preprocessing and automatic cutting-plane generation are not applicable. As a result, our implementation of the BCP approach was not computationally competitive with the other algorithms presented in this paper, and further details are therefore omitted.

2.2 Valid Inequalities

In this section we discuss several valid inequalities and optimality conditions for our problem. All inequalities that we describe in this section are applicable to both the integer programming formulation and the master problem of the Benders decomposition approach we described in Section 2.1.

2.2.1 Adjacent Rectangles

We call two non-overlapping rectangles r_1 and r_2 *adjacent* if either of the following conditions is satisfied:

(a) r_1 and r_2 cover an identical range of columns, with r_1 having bottom row i and r_2 having top row $i+1$, or

(b) r_1 and r_2 cover an identical range of rows, with r_1 having right-most column j and r_2 having left-most column $j+1$.

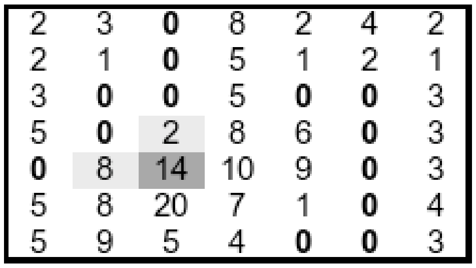
We observe that there exists an optimal solution in which no two adjacent rectangles are used in the decomposition. In order to see this, assume that adjacent rectangles r_1 and r_2 have intensities x_{r_1} and x_{r_2}, respectively, where $x_{r_1} \leq x_{r_2}$ without loss of generality. In this case, an alternative optimal solution can be constructed by extending r_1 into r_2. Specifically, let r' be the rectangle for which $C_{r'} = C_{r_1} \cup C_{r_2}$. An alternative optimal solution that does not contain any adjacent rectangles uses r_2 having intensity $x_{r_2} - x_{r_1}$, and r' having intensity x_{r_1}. This dominance criterion can be written as:

$$y_{r_1} + y_{r_2} \leq 1 \quad \forall \text{ adjacent rectangles } r_1, r_2. \tag{17}$$

2.2.2 Bounding Box Inequalities

We first observe that intensity requirements of adjacent bixels can be used to derive certain necessary conditions that any feasible decomposition of a matrix needs to satisfy. We say that a rectangle *starts* at bixel (i, j) if the upper-left corner of the rectangle is located at (i, j). Consider the bixel $(5, 3)$ marked with dark gray in Figure 2. Since $b_{43} = 2$, the total intensity delivered to $(5, 3)$ by all rectangles that start in rows $i = 1, \ldots, 4$ cannot exceed 2. However, $b_{53} = 14 > 2$, and hence at least one rectangle that starts in row 5 is required to cover bixel $(5, 3)$. Similarly, $b_{53} > b_{52}$ implies that at least one rectangle that starts in column 3 is required to cover the same bixel. These results can be strengthened by considering both

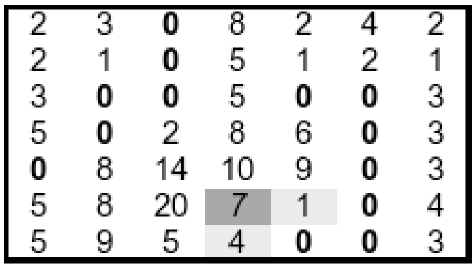
(4, 3) and (5, 2) simultaneously. Since $b_{53} > b_{43} + b_{52}$, we conclude that at least one rectangle that starts at bixel (5, 3) is required in any feasible decomposition of the fluence map. In general, a rectangle must start at (i, j) if $b_{ij} > b_{(i-1)j} + b_{i(j-1)}$ is satisfied. Figure 3 illustrates



2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 2: Example start index

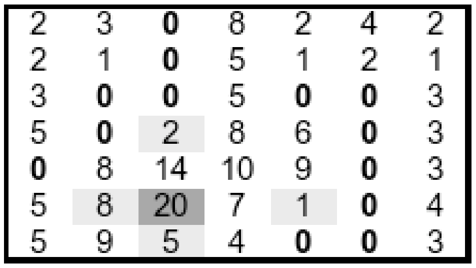
a similar idea, where we compare the intensity requirement of bixel (6, 4) with the bixel below it, and the one on its right. Using arguments similar to the ones regarding starting indices, we conclude that a rectangle must *end* (i.e., have a lower-right corner) at (6, 4) since $b_{64} > b_{74} + b_{65}$.



2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 3: Example end index

Starting and ending index conditions can be generalized further as follows. Assume that there exist integers $u \in [0, i-1]$, $d \in [i+1, m+1]$, $l \in [0, j-1]$ and $r \in [j+1, n+1]$ so that $b_{ij} > b_{il} + b_{uj} + b_{ir} + b_{dj}$, where we define $b_{i0} = b_{0j} = b_{m+1,j} = b_{i,n+1} = 0$ for $i \in \{0, \ldots, m+1\}, j \in \{0, \ldots, n+1\}$. In this case, we say that (l, u, r, d) is a *bounding box* for bixel (i, j). Figure 4 illustrates a bounding box for bixel (6, 3) (marked in dark gray), which corresponds to $(l, u, r, d) = (2, 4, 5, 7)$. The four bixels that represent the borders of a bounding box are marked in light gray. We note that any rectangle that contains bixel (6, 3),



2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

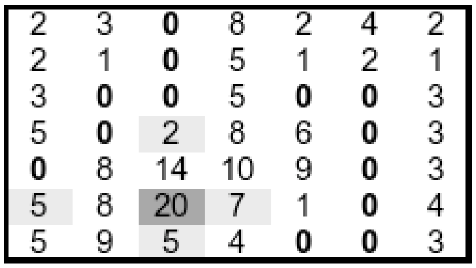
Figure 4: Example bounding box

and does not start inside the bounding box (at (5,3) or (6,3)) or end inside the bounding box (at (6,3) or (6,4)), has to contain at least one of the four bixels on the border. Therefore, the sum of intensities of those rectangles is bounded by the total required intensity of the bixels in light gray. Since the intensity of the dark gray bixel cannot be satisfied by those rectangles alone, it follows that at least one rectangle contained within the bounding box must be used to cover bixel $(6,3)$. Let BB_{ij} represent the interior of a bounding box for bixel (i,j), i.e., given (l,u,r,d) all bixels at the intersection of rows $u+1,\ldots,d-1$ and columns $l+1,\ldots,r-1$. We denote the set of rectangles in $R(i,j)$ that are contained within BB_{ij} by $R(BB_{ij})$. In this case, the following inequality is valid:

$$\sum_{r\in R(BB_{ij})} y_r \geq 1. \tag{18}$$

Note that $(0,0,n+1,m+1)$, which corresponds to the input matrix, is a bounding box for any bixel. Therefore there can be multiple bounding boxes associated with each bixel. Let BB_{ij} and BB'_{ij} be two bounding boxes for bixel (i,j). We say that BB_{ij} *dominates* BB'_{ij} if $R(BB_{ij}) \subset R(BB'_{ij})$. Since the inequality (18) that corresponds to a dominated bounding box is implied by the inequality that is associated with the corresponding dominating bounding box, we are only interested in generating nondominated bounding boxes. Figure 5 displays another nondominated bounding box for the bixel considered in Figure 4.

In order to generate nondominated bounding boxes, we first make the following observation. A nondominated bounding box for bixel (i,j) is minimal in the sense that none of its edges can be shifted closer to (i,j) without violating the bounding box intensity property. We use this observation to design an algorithm that finds several nondominated bounding



2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 5: Another nondominated bounding box seeded at (6,3)

boxes associated with a given bixel. In our algorithm, we start at a bixel (i, j), and first move in a vertical or horizontal direction until we encounter a bixel (i', j') having $b_{i'j'} < b_{ij}$. We mark (i', j') as an edge of the bounding box, reduce b_{ij} by $b_{i'j'}$, and return to (i, j). We then move in the remaining directions one-by-one, updating b_{ij} after each step, in order to find the remaining edges of the bounding box. We repeat the same procedure for all 4! permutations of the directions, and obtain a nondominated bounding box in each iteration. Finally, we eliminate duplicates in order to obtain a set of nondominated bounding boxes, and we generate a constraint of type (18) for each bounding box.

2.2.3 Aggregate Intensity Inequalities

We derive a simple class of valid inequalities by observing that the total intensity that can be delivered to each bixel needs to be greater than or equal to its required intensity. Formally,

$$\sum_{r \in R(i,j)} M_r y_r \geq b_{ij} \quad \forall i = 1, \ldots, m, \; j = 1, \ldots, n. \tag{19}$$

We note that inequalities (19) are implied by (2) and (3) in IPR. However, (19) can be used to tighten the master problem of the Benders decomposition approach discussed in Section 2.1. Furthermore, various tightening procedures can be applied to (19) for use in either the direct solution of IPR or in the Benders master problem. In our implementation, we apply a Chvátal-Gomory rounding procedure (see, e.g., [18]) in which we divide both sides of the inequality by the smallest M_r coefficient on the left-hand-side (unless b_{ij} is divisible by that number), and round up coefficients on both sides of the inequality. If b_{ij} is divisible by the smallest M_r-coefficient on the left-hand-side of (19), then the rounding procedure yields an inequality implied by (19), and hence we do not generate it.

2.2.4 Special Submatrices

An alternative strategy to the one described in Section 2.2.3 divides both sides of (19) by $b_{ij} - 1$, provided that $b_{ij} \geq 2$, and then rounds up all coefficients and the right-hand-side. Noting that all coefficients on the left-hand-side are bounded from above by b_{ij}, this process yields:

$$\sum_{\substack{r \in R(i,j): \\ M_r < b_{ij}}} y_r + 2 \sum_{\substack{r \in R(i,j): \\ M_r = b_{ij}}} y_r \geq 2 \quad \forall i = 1, \dots, m, \ j = 1, \dots, n. \tag{20}$$

Equations (20) imply that bixel (i, j) can either be covered by a single rectangle having a maximum intensity of b_{ij}, or otherwise needs to be covered by at least two rectangles. The idea behind (20) can be extended to other special cases. For instance, consider the following lemma.

Lemma 1. Consider any 1×2 or 2×1 submatrix of B in which both elements equal a common nonzero value, q. Define $A_1^=$ as the set of rectangles that cover exactly one of the two bixels, and have a maximum intensity of q. Let $A_1^<$ be the set of all rectangles that cover exactly one of the two elements, and have a maximum intensity less than q. Define $A_2^=$ and $A_2^<$ analogously for rectangles that cover both elements. The following inequality is valid:

$$4 \sum_{r \in A_2^=} y_r + 2 \sum_{r \in A_2^<} y_r + 2 \sum_{r \in A_1^=} y_r + \sum_{r \in A_1^<} y_r \geq 4. \tag{21}$$

Proof. Consider any feasible solution, and let vector $\mathbf{v}$ denote how many rectangles exist in the solution belonging to $A_2^=, A_2^<, A_1^=$, and $A_1^<$, respectively. We claim (without proof, for brevity) that the following vectors $\mathbf{v}_1, \dots, \mathbf{v}_6$ are minimal, in the sense that $\mathbf{v} \geq \mathbf{v}_i$ for at least one $i = 1, \dots, 6$, for every feasible $\mathbf{v}$: $\mathbf{v}_1 = (1, 0, 0, 0), \mathbf{v}_2 = (0, 1, 0, 2), \mathbf{v}_3 = (0, 2, 0, 0), \mathbf{v}_4 = (0, 0, 1, 2), \mathbf{v}_5 = (0, 0, 2, 0), \mathbf{v}_6 = (0, 0, 0, 4)$. Note that each solution represented by $\mathbf{v}_i$ satisfies (21), and thus all $\mathbf{v}$ corresponding to a feasible solution must also satisfy (21). $\square$

Similarly, consider submatrices of the form

$$\begin{bmatrix} q_L & q_R \end{bmatrix},$$

or its transpose, where we assume $0 < q_L < q_R$ without loss of generality. We define $A_L^=$ and $A_L^<$ to be the sets of rectangles that cover q_L, but not q_R, with maximum intensity q_L, and less than q_L, respectively. Let $A_R^=$ and $A_R^<$ be defined for rectangles that cover q_R but not q_L, with a maximum intensity greater than or equal to $(q_R - q_L)$ and less than $(q_R - q_L)$, respectively. We define $A_2^=$ and $A_2^<$ as before, with a maximum intensity of q_L, and less than q_L, respectively. A similar analysis as in proof of Lemma 1 reveals that the following inequality is valid:

$$2\sum_{r\in A_L^=} y_r + 2\sum_{r\in A_R^=} y_r + 2\sum_{r\in A_2^=} y_r + \sum_{r\in A_L^<} y_r + \sum_{r\in A_R^<} y_r + \sum_{r\in A_2^<} y_r \geq 4. \tag{22}$$

The last special case that we consider is a nonzero submatrix of the form:

$$\begin{bmatrix} q & q \\ q & q \end{bmatrix}.$$

We define $A_i^=$ to be the sets of rectangles having maximum intensity equal to q, and covering exactly i elements of the 2×2 submatrix, for $i = 1$, 2, and 4. Similarly, define $A_i^<$ to be the sets of rectangles having maximum intensity less than q, and covering exactly i elements of the submatrix. Given these definitions, we obtain:

$$8\sum_{r\in A_4^=} y_r + 4\sum_{r\in A_4^<} y_r + 4\sum_{r\in A_2^=} y_r + 2\sum_{r\in A_2^<} y_r + 2\sum_{r\in A_1^=} y_r + \sum_{r\in A_1^<} y_r \geq 8. \tag{23}$$

2.2.5 Submatrix Inequalities

It is possible to generate valid inequalities using arguments similar to the ones discussed in Section 2.2.4 for other submatrices as well. However, this process is very tedious, and there is a large number of possible submatrix combinations. In this section we describe a similar set of inequalities, which are weaker than those described in the previous section, but are easier to generate. We first observe that the formulation IPR can be solved quickly for small input matrices. Let S denote a submatrix of the input matrix, and $R(S)$ represent the set of rectangles that cover at least one bixel in S. Let $LB(S)$ be a lower bound on the number of rectangles required to decompose S. Since $LB(S)$ constitutes a lower bound on the total number of rectangles required, the following inequality is valid for any submatrix S:

$$\sum_{r \in R(S)} y_r \geq \lceil LB(S) \rceil. \tag{24}$$

We can obtain $LB(S)$ by formulating an auxiliary integer programming problem of type IPR for S, and setting a limit on the maximum solution time.

2.3 Partitioning Approach

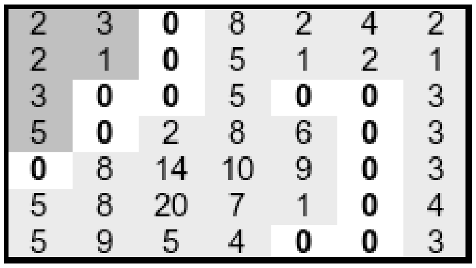
In this section, we propose a partitioning approach for our problem. We first propose an algorithm for detecting completely separable regions of the input matrix, which can be solved independently. Next, we explore methods for partitioning the large components, in order to obtain simultaneous upper and lower bounds, which we use to improve the solvability of our formulation.

2.3.1 Separable Components

Our observations on clinical data sets suggest that input matrices can usually be decomposed into several small components, and one or two large components. The small components can usually be solved to optimality by formulation IPR enhanced with the valid inequalities discussed in Section 2.2.

We observe on clinical data that several regions of the input matrix are completely surrounded by zero-bixels. Since no rectangle can cover a zero-bixel, each of these regions can be solved independently. A *connected* subset of the input matrix obeys the property that a rectilinear path exists between any two nonzero-bixels of the subset, such that each bixel in the path is also a nonzero-bixel that belongs to the subset. We call a connected set of nonzero-bixels a *component* of the input matrix if it is adjacent to zero-bixels across all of its boundaries (i.e., if the subset is not contained within a larger connected subset).

In order to identify the components of the input matrix, we generate a graph G in which each nonzero-bixel has a corresponding node. We add an arc between a pair of nodes if and only if the corresponding bixels are adjacent in the input matrix. We then identify connected components on G by running a standard depth-first-search algorithm. Each connected component on G corresponds to a component of the input matrix, which can be solved independently of other components. Figure 6 depicts the components of the fluence map given in Figure 1.



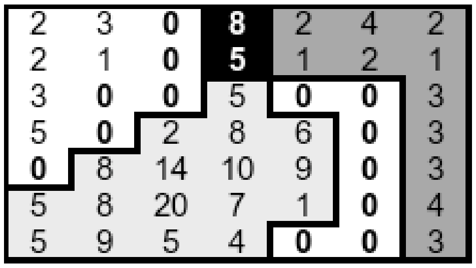
2	3	**0**	8	2	4	2
2	1	**0**	5	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 6: Two components of a fluence map

2.3.2 Independent Regions

After finding separable components of the input matrix, we attempt to further partition each component into smaller *regions*. We say that distinct regions of a component are *independent* if no rectangle intersects two bixels belonging to different regions without also intersecting a zero-bixel. In Figure 7, the regions with light and dark gray background are independent. If we solve IPR separately over all independent regions, the sum of rectangles required to decompose each independent region yields a lower bound on the objective function for the corresponding component.

In general, there are multiple ways of partitioning a component into independent regions, with each yielding possibly different lower bounds. The problem of finding a partition that yields the best lower bound can be thought of as a "dual" of finding the minimum number



2	3	**0**	**8**	2	4	2
2	1	**0**	**5**	1	2	1
3	**0**	**0**	5	**0**	**0**	3
5	**0**	2	8	6	**0**	3
0	8	14	10	9	**0**	3
5	8	20	7	1	**0**	4
5	9	5	4	**0**	**0**	3

Figure 7: Regions of a connected component

of rectangles to decompose a component. In order to solve this dual problem, we need to balance two conflicting criteria:

- The number of bixels assigned to each independent region needs to be small enough so that each region can be solved quickly.
- The number of bixels not assigned to any independent regions needs to be as small as possible in order to obtain a good lower bound.

We use a heuristic procedure to partition a component into independent regions, which employs an auxiliary objective of maximizing the number of component bixels covered by an independent region. Each bixel (i, j) is called "committed" if it either belongs to an independent region, or if (i, j) is contained within some rectangle in R that also covers bixels in an independent region (and hence, (i, j) cannot belong to another independent region). All other bixels are called "uncommitted." We select our independent regions one at a time, until no more uncommitted bixels remain. The procedure's details are described as follows.

Initialization. Labels all nonzero-bixels as "uncommitted."

Step 1. Each candidate independent region (or just "candidate") is seeded from a rectangle $r \in R$ such that rectangle r contains only uncommitted bixels, and such that the number of bixels in the rectangle is no more than some limit L. For each such rectangle r, define ℓ_r to be the (initial) candidate region.

Step 2. For each candidate ℓ_r, if ℓ_r covers exactly L bixels, then go to Step 4. Else, continue to Step 3.

Step 3. For each candidate ℓ_r, determine if there exists an uncommitted bixel (i,j) adjacent to ℓ_r (i.e., a bixel $(i,j) \notin \ell_r$ such that either $(i-1,j)$, $(i+1,j)$, $(i,j-1)$, or $(i,j+1)$ belongs to ℓ_r), such that for every $r' \in R(i,j)$, all bixels in r' either belong to ℓ_r, or would already become committed due to the selection of ℓ_r as an independent region. That is, adding (i,j) to ℓ_r would not increase the number of bixels committed by selecting ℓ_r as a new independent region. If such a bixel exists, then add (i,j) to ℓ_r, and return to Step 2. Else, continue to Step 4.

Step 4. For each candidate ℓ_r, compute κ_r^C = the number of bixels in ℓ_r, and κ_r^D = the number of uncommitted bixels (i,j) such that some rectangle in R includes both (i,j) and a bixel in ℓ_r. If any candidates exist such that $\kappa_r^D = 0$, then choose $\ell_r^\star$ to be any such candidate. Else, choose $\ell_r^\star$ to be any candidate that maximizes κ_r^C/κ_r^D. Go to Step 5.

Step 5. Create an independent region corresponding to $\ell_r^\star$. For each bixel (i,j) that can be covered by a rectangle in R intersecting at least one bixel in $\ell_r^\star$, change the status of (i,j) to "committed." (This includes all bixels in $\ell_r^\star$ itself.) If all bixels are committed, terminate the procedure; else, return to Step 1.

In our algorithm for solving a component, we execute the foregoing heuristic to find a set of independent regions. We formulate IPR for each region, with a limit on the maximum solution time. We then use the lower bound obtained for each region to generate an inequality of type (24). (It is often prudent to skip this step if *only* one region is computed for a component.)

2.3.3 Dependent Regions

In this section, we attempt to improve the lower bound obtained using independent regions by focusing on those bixels not included in the union of independent regions. We define a *dependent region* to be a connected set of bixels in a component that does not overlap with any of the independent regions in that component. In our example, the region with black background in Figure 7 is a dependent region. Let D represent the set of bixels in a dependent region, and let $\mathcal{R}(D)$ represent the set of rectangles that cover only a subset of the bixels in D.

In order to improve our lower bound, we wish to compute the minimum number of rectangles required to cover D; however, we wish to avoid double-counting those rectangles used to cover bixels in independent regions. Accordingly, we seek the minimum number of rectangles in $\mathcal{R}(D)$, perhaps in concert with rectangles outside $\mathcal{R}(D)$, required to cover the bixels in D. Using the x- and y-variables as before, we formulate the following variation of IPR in order to find the minimum number of rectangles in $\mathcal{R}(D)$ required to partition D.

$$\textbf{DPR}\text{: Minimize } \sum_{r \in \mathcal{R}(D)} y_r \tag{25}$$

$$\text{subject to: } \sum_{r \in R(i,j)} x_r = b_{ij} \quad \forall (i,j) \in D \tag{26}$$

$$\sum_{r \in R(i,j)} x_r \leq b_{ij} \quad \forall i = 1, \ldots, m,\ j = 1, \ldots, n,\ (i,j) \notin D \tag{27}$$

$$x_r \leq M_r y_r \quad \forall r \in \mathcal{R}(D) \tag{28}$$

$$x_r \geq 0\ \forall r \in R, \quad y_r \text{ binary } \forall r \in \mathcal{R}(D) \tag{29}$$

Objective (25) minimizes the number of rectangles in $\mathcal{R}(D)$ used in the solution. Constraints (26) ensure that the bixels in D get partitioned exactly, where (27) limit the intensity delivered to the remaining bixels. Constraints (28) relate the x- and y- variables as done in IPR, and finally (29) define variable types. As before, we set a time limit for the solution of DPR, and obtain a lower bound on the objective function value, which we denote by $LB(D)$. Given this value, the following inequality is valid:

$$\sum_{r \in \mathcal{R}(D)} y_r \geq \lceil LB(D) \rceil. \tag{30}$$

In our example, the optimal value of DPR for the black (dependent) region is 1 since the intensity requirement of bixel $(1,4)$ cannot be satisfied completely by rectangles that cover bixels in the gray (independent) regions (in fact, this result can also be seen due to the bounding box constraint implying that one rectangle representing the singleton bixel (1,4) must appear in any feasible solution). We note that the rectangles in $\mathcal{R}(D)$, by definition, do not intersect any other (dependent or independent) regions. Therefore, the lower bounds obtained for all regions can be summed to obtain a lower bound on the minimum number of rectangles required to decompose a component.

2.3.4 Upper Bound Calculation

In this section, we discuss how a related approach leads to a heuristic algorithm to obtain a feasible decomposition of a component. We first note that a feasible decomposition of a component can be obtained by combining feasible solutions obtained for individual regions within a component. Feasible solutions for independent regions are readily available from the integer programming problems solved for obtaining lower bounds on those regions, as discussed in Section 2.3.2. Feasible solutions for dependent regions can be extracted from solutions of the formulation given by DPR. However, since DPR minimizes the number of rectangles that are contained within a dependent region, and not necessarily the total number of rectangles required to decompose a dependent region, the solutions obtained from DPR potentially use an unnecessarily large number of rectangles not contained in $\mathcal{R}(D)$.

A better way of obtaining feasible solutions for dependent regions is to formulate the problem IPR for each dependent region. Since IPR explicitly minimizes the total number of rectangles required, we expect this approach to result in feasible solutions of higher quality. However, this approach does not consider the fact that some of the rectangles that are already used for decomposing independent regions can be extended into dependent regions without increasing the total number of rectangles. In order to permit the use of rectangles that intersect independent and dependent regions, we require a revised integer programming formulation.

In our approach, we solve the integer programming formulations for decomposing the independent regions first, and store the best feasible solutions found within the allowed time limit. Let $\bar{x}_r$ represent the intensity assigned to rectangle r for decomposing independent regions. Next, we generate a feasible solution for each dependent region, one at a time, as follows. We first find the set of rectangles that can be extended into the current dependent region, and determine how those rectangles can be extended. Let $E(D, r)$ represent the set of rectangles in R that extend rectangle r into dependent region D. We also define the parameter $I(r)^e_{ij}$ equal to one if bixel $(i, j) \in D$ is covered by extension e of rectangle r, and zero otherwise. Let z_{re} be a binary variable that equals 1 if and only if extension e of rectangle r is used in the solution. We define the x- and y- variables as before, and formulate the following problem:

$$\textbf{EPR}\text{: Minimize } \sum_{r \in \mathcal{R}(D)} y_r \tag{31}$$

$$\text{subject to: } \sum_{r \in R(i,j)} x_r = b_{ij} - \sum_{r \in R} \sum_{e \in E(D,r)} \left(\bar{x}_r I(r)^e_{ij}\right) z_{re} \quad \forall (i,j) \in D \tag{32}$$

$$\sum_{e \in E(D,r)} z_{re} \leq 1 \quad \forall r \in R \tag{33}$$

$$x_r \leq M_r y_r \quad \forall r \in \mathcal{R}(D) \tag{34}$$

$$x_r \geq 0, \; y_r \text{ binary} \quad \forall r \in \mathcal{R}(D) \tag{35}$$

$$z_{re} \text{ binary} \quad \forall r \in R, \; e \in E(D, r). \tag{36}$$

We generate a feasible solution by combining three types of rectangles: (i) rectangles used to decompose independent regions that are not extended by EPR; (ii) rectangles obtained by extending rectangles from independent regions into dependent regions by EPR; and (iii) rectangles in $\mathcal{R}(D)$ used by EPR.

Note that the optimal value of EPR for the dependent region given in Figure 7 is 1. This can be seen by observing that the rectangle(s) that cover bixel $(3, 4)$ can be extended up to fully satisfy the intensity requirement of bixel $(2, 4)$ without any penalty on the objective function of EPR formulated for the dependent region. Therefore, a single rectangle contained in the dependent region solves EPR optimally. Since the optimal value of DPR for the dependent region is also 1, our partition solves the problem of finding the minimum number of rectangles to optimality.

3 Extensions

In this section, we briefly discuss how our model can be adjusted to tackle the problems of minimizing total treatment time, and lexicographically minimizing beam-on-time and number of apertures.

3.1 Minimize Total Treatment Time

The total time spent delivering a given fluence map is composed of (i) time required to move the jaws to form the next rectangular aperture (setup time), and (ii) time during which radiation is delivered (beam-on-time). Even though the setup time required for switching from one rectangular aperture to the next one depends on the jaw settings corresponding to these apertures, and hence is sequence-dependent, in this paper we make the common assumption that total setup time is proportional to the total number of apertures used. With this assumption, our model can easily be adjusted to explicitly minimize the total treatment time by changing the objective function of IPR to

$$\text{Minimize } w \sum_{r \in R} y_r + \sum_{r \in R} x_r, \tag{37}$$

where w is a parameter that represents the average setup time per aperture relative to the time required to deliver a unit of intensity.

The Benders decomposition procedure discussed in Section 2.1 also needs to be adjusted accordingly. We first add a continuous variable t to MP, which "predicts" the minimum beam-on-time that can be obtained by the set of rectangles chosen by MP. The updated master problem can be written as follows.

$$\textbf{MPTT: } \text{Minimize } w \sum_{r \in R} y_r + t \tag{38}$$

$$\text{subject to: } \mathbf{y} \text{ corresponds to a feasible decomposition} \tag{39}$$

$$t \geq \text{minimum beam-on-time corresponding to } \mathbf{y} \tag{40}$$

$$y_r \text{ binary} \quad \forall r \in R. \tag{41}$$

Given a vector $\hat{\mathbf{y}}$, we can find the minimum beam-on-time for the corresponding decomposition, if one exists, by solving:

$$\textbf{SPTT}(\hat{\mathbf{y}})\text{: Minimize } \sum_{r\in R} x_r \tag{42}$$

$$\text{subject to: } \sum_{r\in R(i,j)} x_r = b_{ij} \quad \forall i=1,\dots,m,\ j=1,\dots,n \tag{43}$$

$$x_r \le M_r\hat{y}_r \quad \forall r\in R \tag{44}$$

$$x_r \ge 0 \quad \forall r\in R. \tag{45}$$

Note that SPTT is obtained by simply changing the objective function of SP. If SPTT($\hat{\mathbf{y}}$) is infeasible, then we add a Benders feasibility cut of type (16) as before, and re-solve MPTT. Otherwise, let the value of t in MPTT be $\hat{t}$, and the optimal objective function value of SPTT be $t^\star$. If $\hat{t} = t^\star$, then $(\hat{\mathbf{y}}, \hat{t})$ is an optimal solution of MPTT that minimizes the total treatment time. However, if $\hat{t} > t^\star$, then we add the following Benders optimality cut

$$t \ge \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}\hat{\alpha}_{ij} + \sum_{r\in R}(M_r\hat{\beta}_r)y_r, \tag{46}$$

where $\hat{\alpha}_{ij}$ and $\hat{\beta}_r$ are optimal dual multipliers associated with constraints (43) and (44), respectively.

3.2 Optimization with Beam-on-Time Restrictions

Another related problem that we consider is finding the minimum number of rectangles that yields the minimum beam-on-time. Note that the minimum beam-on-time required to decompose a fluence map can be found (in polynomial time) by solving SPTT, which is a linear program, for the vector $\hat{y}_r = 1$, $\forall r \in R$. Let $T^\star$ denote the optimal objective function value of SPTT($\vec{1}$), where $\vec{1}$ is the vector of all 1's. Given this value, it is sufficient to add

$$\sum_{r\in R} x_r \le T^\star \tag{47}$$

to minimize the number of rectangles while limiting beam-on-time to $T^\star$.

The modifications required for the Benders decomposition algorithm are also straightforward. In order to enforce the minimum beam-on-time restriction, we add (47) to SP, which checks whether a given set of rectangles can decompose the fluence map. The updated feasibility cut is given by

$$\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}\hat{\alpha}_{ij} + \sum_{r\in R}(M_r\hat{\beta}_r)y_r + T^{\star}\hat{\theta} \leq 0, \tag{48}$$

where θ is the dual variable associated with (47) in SP. Finally, we need to check whether the solution generated by our heuristic discussed in Section 2.3.4 satisfies constraint (47); if so, then it can be used as an initial upper bound.

4 Computational Results

We have implemented our algorithms using CPLEX 11 running on a Windows XP PC with a 3.4 GHz CPU and 2 GB RAM. Our base set of test problem instances consists of 25 clinical problem instances ("case1beam1", ..., "case5beam5"). These instances were obtained from treatment plans for five patients treated using five beam angles each. We report problem characteristics in terms of the number of rows m, the number of columns n, and the maximum intensity value L. We imposed a time limit of 1800 seconds (30 minutes) in all of our tests. For problem instances that were not solved to optimality within the imposed time limit, we report the best upper and lower bounds obtained, where we round lower bounds up for the cases in which the objective function is guaranteed to have an integral value.

Our preliminary computational tests showed that the naive implementation of our Benders decomposition approach, in which we add a cut and re-solve the master problem in each iteration, was not computationally competitive with solving the explicit integer programming formulation. This is due to the fact that repetitively solving the master problem, which is an integer programming problem, is computationally very expensive. We instead used callback functions of CPLEX to generate a single branch-and-bound tree in which we solve SP($\hat{\mathbf{y}}$) (or (SPTT($\hat{\mathbf{y}}$)) corresponding to each integer solution found in the branch-and-bound tree, and add cuts to tighten the master problem as necessary. While this approach produced better results than the naive implementation, it still yielded inferior bounds than those obtained from the explicit formulation. Therefore, we omit further Benders-based computational results in this paper.

Our first experiment quantifies the effects of the valid inequalities discussed in Section 2.2, and the partitioning approach discussed in Section 2.3 on solution quality and execution time. In Table 1, the set of columns labeled "Default CPLEX" shows the results we obtained by solving the formulation IPR on each problem instance using default CPLEX options. The "+ Valid Inequalities" columns represent the IPR formulation enhanced with the adjacent rectangle inequalities (17), bounding box inequalities (18), strengthened aggregate intensity inequalities (19) and (20), and 1×2 submatrix inequalities (21) and (22). (Additional computational results showed that the 2×2 submatrix inequalities (23) and the arbitrary submatrix inequalities (24) did not improve the solvability of the model.) The set of columns labeled "+ Partitions" shows the results we obtained by partitioning the problem into separable components (Section 2.3.1), further partitioning each component into independent and dependent regions (Sections 2.3.2 and 2.3.3), and using our upper bounding heuristic (Section 2.3.4) in addition to the valid inequalities used for the tests in the previous set of columns. We refer to the latter settings as our *base algorithm* in the remaining computational tests.

Each set of columns in Table 1 displays the time spent for each problem instance ("CPU"), and upper bound ("UB"), lower bound ("LB"), and optimality gap ("GAP") obtained. We also report the average and maximum gaps over all problem instances. We observe that none of the problem instances were solved to optimality using the default CPLEX options, whereas case1beam2 and case5beam2 were solved to optimality after adding the valid inequalities of Section 2.2. An additional instance (case5beam5) was solved using the partitioning strategy described in Section 2.3. We note that even though our approach was not able to provide provably optimal solutions for most instances, it significantly improved both lower and upper bounds for several instances.

Our next experiment tests our base algorithm under the extensions discussed in Section

Instance				Default CPLEX				+ Valid Inequalities				+ Partitions			
Name	**m**	**n**	**L**	**CPU**	**UB**	**LB**	**GAP**	**CPU**	**UB**	**LB**	**GAP**	**CPU**	**UB**	**LB**	**GAP**
case1beam1	15	14	20	1800	66	60	0.09	1800	63	62	**0.02**	1800	64	62	0.03
case1beam2	11	15	20	1800	48	47	0.02	138.23	48	48	**0**	1009.92	48	48	**0**
case1beam3	15	15	20	1800	57	54	0.05	1800	57	54	**0.05**	1800	58	54	0.07
case1beam4	15	15	20	1800	61	52	0.15	1800	61	53	0.13	1800	59	55	**0.07**
case1beam5	11	15	20	1800	47	45	0.04	1800	46	45	**0.02**	1800	47	45	0.04
case2beam1	18	20	20	1800	114	79	0.31	1800	119	85	0.29	1800	103	87	**0.16**
case2beam2	17	19	20	1800	95	69	0.27	1800	96	81	0.16	1800	94	82	**0.13**
case2beam3	18	18	20	1800	98	73	0.26	1800	103	77	0.25	1800	94	77	**0.18**
case2beam4	18	18	20	1800	114	80	0.3	1800	115	84	0.27	1800	105	88	**0.16**
case2beam5	17	18	20	1800	94	64	0.32	1800	98	72	0.27	1800	91	72	**0.21**
case3beam1	22	17	20	1800	121	69	0.43	1800	134	79	0.41	1800	119	79	**0.34**
case3beam2	15	19	20	1800	73	46	0.37	1800	71	52	0.27	1800	70	52	**0.26**
case3beam3	20	17	20	1800	119	69	0.42	1800	119	75	0.37	1800	107	77	**0.28**
case3beam4	19	17	20	1800	103	69	0.33	1800	106	73	0.31	1800	99	78	**0.21**
case3beam5	15	19	20	1800	73	55	0.25	1800	71	58	**0.18**	1800	73	58	0.21
case4beam1	19	22	20	1800	106	79	0.25	1800	107	89	**0.17**	1800	109	89	0.18
case4beam2	13	24	20	1800	88	54	0.39	1800	99	58	0.41	1800	91	58	**0.36**
case4beam3	18	23	20	1800	95	71	0.25	1800	99	75	0.24	1800	93	77	**0.17**
case4beam4	17	23	20	1800	103	78	0.24	1800	102	81	0.21	1800	98	83	**0.15**
case4beam5	18	24	20	1800	93	62	0.33	1800	93	66	0.29	1800	87	67	**0.23**
case5beam1	15	16	20	1800	66	64	0.03	1800	66	65	**0.02**	1800	66	65	**0.02**
case5beam2	13	17	20	1800	58	57	0.02	102.09	58	58	**0**	213.6	58	58	**0**
case5beam3	14	16	20	1800	63	54	0.14	1800	68	56	0.18	1800	65	57	**0.12**
case5beam4	14	16	20	1800	63	57	0.1	1800	64	59	0.08	1800	62	59	**0.05**
case5beam5	12	17	20	1800	53	47	0.11	1800	51	48	0.06	36.25	49	49	**0**
						Max	0.43			**Max**	0.41			**Max**	0.36
						Avg	0.22			**Avg**	0.19			**Avg**	0.14

Table 1: Effect of valid inequalities and the partitioning strategy, with best gap figures reported in bold

3. The set of columns labeled as "Total Time" in Table 2 presents the extension in which the objective function is defined as a linear combination of the beam-on-time and the number of rectangles. The actual value of w depends on the particular treatment delivery equipment used in the clinic, where values of w in the range 1–10 are typical (see, e.g., Dai and Hu [4], and Taşkın et al. [23]). In our experiments, we therefore used $w = 7$ as a representative value. The next set of columns ("Lexicographic") is dedicated to the extension in which we first minimize beam-on-time, $T^\star$, and then find the minimum number of rectangles that yields the minimum beam-on-time. The column "BOT" represents the value of $T^\star$, and "Total Time" represents the total treatment time associated with the solution found, where we again use $w = 7$ as the average setup time per rectangle. We observe that our algorithm could solve more problem instances to optimality for both extensions compared to the problem of finding the minimum number of rectangles. In order to understand why this is the case, we first note that the difficulty of the matrix decomposition problem varies greatly based on the objective function used. On one hand, minimizing the number of rectangles is strongly NP-Hard, even for fluence maps having a single row (see Baatar et al. [2]). On the other hand, minimizing the beam-on-time is a polynomially solvable problem (see Section 3). Therefore, we expect that the problem should become easier as the weight of the beam-on-time term in the objective function increases. The reason the lexicographic minimization problem is easier to solve than the other two variations is because the additional beam-on-time constraint considerably shrinks the feasible solution space.

Another way of looking at the problem of balancing the number of apertures and the beam-on-time is to view the problem as a multicriteria optimization problem. In this setting, we are interested in constructing the Pareto efficient frontier of solutions with the property that neither of the two criteria can be improved without deteriorating the other. Note that the lexicographic approach that we considered above determines a particular Pareto optimal solution to the multicriteria problem. In order to generate other non-dominated solutions for the multicriteria version of the problem, we sequentially impose different upper bounds on the number of apertures allowed, say γ, and find the corresponding minimum beam-on-

Instance				**Total Time**				**Lexicographic**					
Name	**m**	**n**	**L**	**CPU**	**UB**	**LB**	**GAP**	**CPU**	**UB**	**LB**	**GAP**	**BOT**	**Total Time**
case1beam1	15	14	20	255.95	621	621	0	36.48	66	66	0	176	638
case1beam2	11	15	20	330.31	459	459	0	132.58	50	50	0	121	471
case1beam3	15	15	20	1800	548	542.72	0.01	130.39	62	62	0	147	581
case1beam4	15	15	20	1800	557	542.49	0.03	186.88	62	62	0	136	570
case1beam5	11	15	20	1800	451	443.63	0.02	30.95	53	53	0	115	486
case2beam1	18	20	20	1800	962	814.24	0.15	1800	107	104	0.03	194	943
case2beam2	17	19	20	1800	883	797.74	0.1	1800	96	92	0.04	207	879
case2beam3	18	18	20	1800	918	797.6	0.13	1800	96	88	0.08	237	909
case2beam4	18	18	20	1800	1028	889.36	0.13	1800	111	106	0.05	258	1035
case2beam5	17	18	20	1800	890	721.13	0.19	1800	92	83	0.1	207	851
case3beam1	22	17	20	1800	1161	858.9	0.26	1800	116	103	0.11	266	1078
case3beam2	15	19	20	1800	668	533.24	0.2	1800	70	64	0.09	151	641
case3beam3	20	17	20	1800	1066	847.09	0.21	1800	111	95	0.14	278	1055
case3beam4	19	17	20	1800	1023	857.91	0.16	1800	103	95	0.08	287	1008
case3beam5	15	19	20	1800	722	610.01	0.16	204.44	76	76	0	182	714
case4beam1	19	22	20	1800	1044	918.57	0.12	1800	108	105	0.03	275	1031
case4beam2	13	24	20	1800	895	656.15	0.27	1800	95	76	0.2	232	897
case4beam3	18	23	20	1800	858	743.62	0.13	1800	92	89	0.03	189	833
case4beam4	17	23	20	1800	943	834.32	0.12	1800	101	96	0.05	235	942
case4beam5	18	24	20	1800	913	740.19	0.19	1800	86	77	0.1	260	862
case5beam1	15	16	20	271.48	626	626	0	5.5	71	71	0	158	655
case5beam2	13	17	20	33.39	597	597	0	19.97	63	63	0	156	597
case5beam3	14	16	20	1800	623	597.96	0.04	869.22	68	68	0	180	656
case5beam4	14	16	20	1800	584	571.15	0.02	192.36	66	66	0	145	607
case5beam5	12	17	20	90.41	503	503	0	37.2	57	57	0	147	546
						Max	0.27			**Max**	0.2		
						Avg	0.11			**Avg**	0.05		

Table 2: Computational results on model extensions

time for these values of γ. As an example, we considered the problem instance case5beam5. For this instance, we note that the minimum number of apertures is 49 (see Table 1) with a corresponding beam-on-time of 160, while the minimum beam-on-time for this problem instance is 147 (see Table 2) which requires 57 apertures. Figure 8 then depicts (i) the non-dominated solutions; (ii) the Pareto efficient frontier for values of $\gamma \in [49, 57]$, and (iii) the (boundary of the) convex hull of the Pareto set. The solutions on the latter are the optimal solutions to the problem of minimizing total treatment time that can be obtained with different values of w.

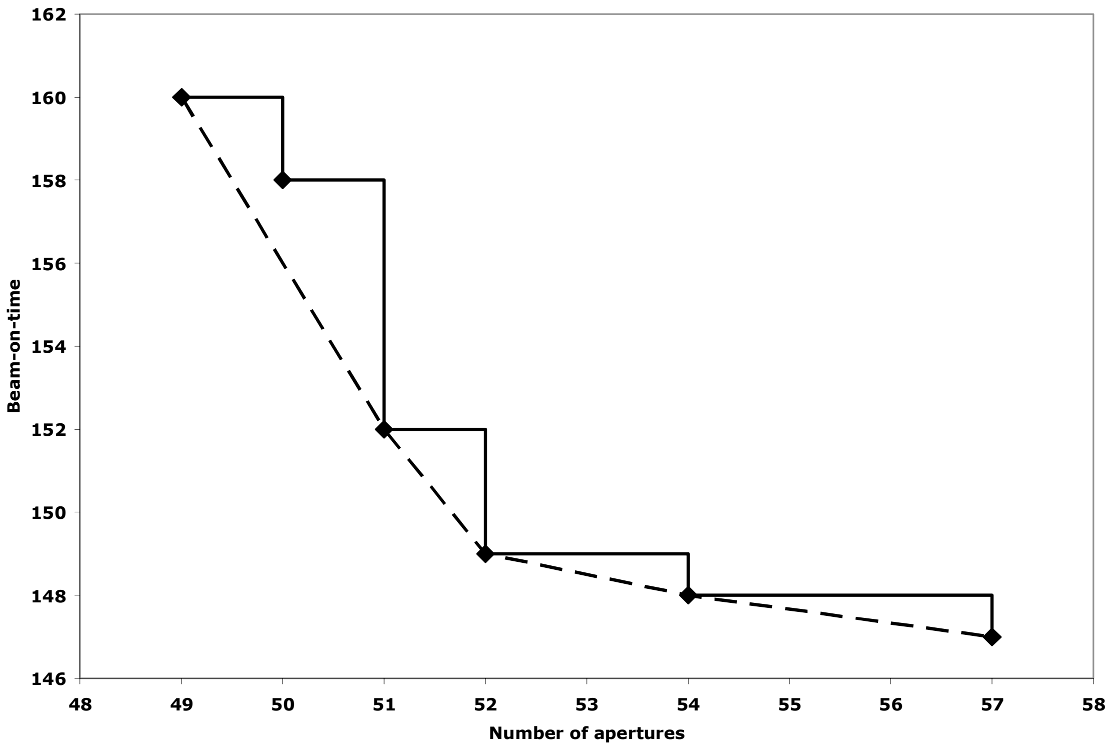


Figure 8: Efficient frontier for number of apertures and beam-on-time

Our final experiment analyzes the effect of the maximum intensity value L. Usually fluence maps are obtained by solving a nonlinear optimization problem for each beam angle in order to determine an intensity profile for each beam angle, which is represented by a matrix of real numbers. Later, these matrices are rounded to integer matrices in order to limit the delivery time. In order to analyze the trade-off between round-off errors and treatment time, we started from clinical treatment plans, and applied rounding with different levels of granularity. Specifically, we generated problem instances from the same fluence maps with $L \in \{5, 10, 15, 20\}$, and used our algorithm to find the minimum total treatment time required as a measure of delivery efficiency. Table 3 shows the results of our experiments. We observe that our algorithm produces smaller optimality gaps as L decreases, which is not surprising since IPR becomes tighter as the M_r-coefficients (which are bounded by L) decrease. Furthermore, delivery efficiency is also higher for small values of L. The average treatment time (calculated over the lower bounds) for all problem instances increases from 366.05 for $L = 5$ to 513.79 for $L = 10$, 609.79 for $L = 15$, and 684.96 for $L = 20$, which is calculated using the set of columns labeled "Total Time" in Table 2. Our results show that the choice of granularity chosen for rounding has a significant effect on the treatment time. For each individual patient, the risks associated with the deterioration in treatment plan quality due to the rounding of intensities needs to be weighed against the disadvantages of a longer treatment time by the physician or clinician.

5 Conclusions

In this paper, we have investigated a problem encountered regarding efficient delivery of a fluence map using rectangular apertures in IMRT treatment. Rectangular apertures can be formed by using conventional jaws already integrated into IMRT treatment devices, and do not need an advanced MLC system, which is costly to manufacture and operate. In this paper, we designed an exact optimization algorithm that can be used to analyze whether a jaws-only treatment system can deliver fluence maps efficiently. Our algorithm is based on an integer programming formulation, which we enhance using several valid inequalities and by partitioning the problem into simpler problems. We next showed how our approach can be extended in order to optimize other efficiency criteria such as beam-on-time, total treatment time, and beam-on-time-constrained number of rectangular apertures. Our tests on clinical data showed that a few problem instances were solved to optimality within a half-hour, and we were able to obtain good optimality bounds for most other instances that could not be

Instance			$L = 5$				$L = 10$				$L = 15$			
Name	**m**	**n**	**CPU**	**UB**	**LB**	**GAP**	**CPU**	**UB**	**LB**	**GAP**	**CPU**	**UB**	**LB**	**GAP**
case1beam1	15	14	4.42	305	305	0	22.95	441	441	0	1800	539	536.99	0
case1beam2	11	15	1.44	238	238	0	4.39	320	320	0	498.47	394	394	0
case1beam3	15	15	9.64	287	287	0	228.61	377	377	0	1800	495	487.67	0.01
case1beam4	15	15	5.8	269	269	0	1800	393	377.42	0.04	1800	513	493.13	0.04
case1beam5	11	15	2.41	216	216	0	28.97	326	326	0	316.7	411	411	0
case2beam1	18	20	31.31	440	440	0	1800	648	635.1	0.02	1800	826	732.68	0.11
case2beam2	17	19	51.48	448	448	0	1800	625	599.84	0.04	1800	759	690.67	0.09
case2beam3	18	18	144.45	428	428	0	1800	645	615.64	0.05	1800	800	704.61	0.12
case2beam4	18	18	1593.03	487	487	0	1800	755	678.49	0.1	1800	919	796.13	0.13
case2beam5	17	18	197.89	429	429	0	1800	606	538.62	0.11	1800	728	621.9	0.15
case3beam1	22	17	1359.36	480	480	0	1800	747	662.47	0.11	1800	1080	779.16	0.28
case3beam2	15	19	1800	280	274.97	0.02	1800	414	376.01	0.09	1800	532	461.6	0.13
case3beam3	20	17	1800	461	446.77	0.03	1800	731	641.2	0.12	1800	908	755.25	0.17
case3beam4	19	17	1800	463	456.42	0.01	1800	713	634.26	0.11	1800	900	762.02	0.15
case3beam5	15	19	1800	332	325.87	0.02	1800	481	466.6	0.03	1800	582	532.2	0.09
case4beam1	19	22	39.89	529	529	0	1800	758	719.53	0.05	1800	899	827.73	0.08
case4beam2	13	24	1800	422	408.14	0.03	1800	595	503.92	0.15	1800	764	582.69	0.24
case4beam3	18	23	126.28	409	409	0	1800	579	564.5	0.03	1800	695	666.23	0.04
case4beam4	17	23	321.42	444	444	0	1800	662	649.1	0.02	1800	815	742.81	0.09
case4beam5	18	24	1194.69	414	414	0	1800	636	573.02	0.1	1800	794	662.74	0.17
case5beam1	15	16	5.89	342	342	0	5.58	442	442	0	28.91	584	584	0
case5beam2	13	17	4.28	289	289	0	5.5	439	439	0	49.3	516	516	0
case5beam3	14	16	1574.41	294	294	0	1800	453	441.44	0.03	1800	566	538.22	0.05
case5beam4	14	16	3.64	239	239	0	1800	473	468.49	0.01	1800	534	524.31	0.02
case5beam5	12	17	2.14	252	252	0	9.31	354	354	0	63.91	441	441	0
					Max	0.03			**Max**	0.15			**Max**	0.28
					Avg	0			**Avg**	0.05			**Avg**	0.09

Table 3: Effect of maximum intensity value on solvability

solved within the imposed time limits. We also examined the trade-off between minimizing the beam-on-time and the number of apertures computationally, and illustrated the efficient frontier in a multicriteria setting. Finally, we analyzed the relationship between delivery efficiency and the desired level of granularity in delivering a given fluence map. Our results indicate that our model is flexible enough to analyze several aspects of delivery efficiency in a jaws-only treatment device.

References

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

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

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

[4] J. Dai and Y. Hu. Intensity-modulation radiotherapy using independent collimators: an algorithm study. *Medical Physics*, 26(12):2562–2570, 1999.

[5] M. A. Earl, M. K. N. Afghan, C. X. Yu, Z. Jiang, and D. M. Shepard. Jaws-only IMRT using direct aperture optimization. *Medical Physics*, 34(1):307–314, 2007.

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

[7] H. W. Hamacher and K.-H. Küfer. Inverse radiation therapy planning – a multiple objective optimization approach. *Discrete Applied Mathematics*, 118(1):145–161, 2002.

[8] T. Kalinowski. A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Applied Mathematics*, 152(1):52–88, 2005.

[9] S. Kamath, S. Sahni, J. Li, J. Palta, and S. Ranka. Leaf sequencing algorithms for segmented multileaf collimation. *Physics in Medicine and Biology*, 48(3):307–324, 2003.

[10] Y. Kim, L. J. Verhey, and P. Xia. A feasibility study of using conventional jaws to deliver IMRT plans in the treatment of prostate cancer. *Physics in Medicine and Biology*, 52(8):2147–2156, 2007.

[11] K.-H. Küfer, M. Monz, A. Scherrer, H. L. Trinkaus, T. Bortfeld, and C. Thieke. Intensity Modulated Radiotherapy – a large scale multi-criteria programming problem. *OR Spektrum*, 25:223–249, 2003.

[12] E. K. Lee, T. Fox, and I. Crocker. Optimization of radiosurgery treatment planning via mixed integer programming. *Medical Physics*, 27(5):995–1004, 2000.

[13] E. K. Lee, T. Fox, and I. Crocker. Integer programming applied to intensity-modulated radiation treatment planning. *Annals of Operations Research*, 119:165–181, 2003.

[14] G. J. Lim. *Optimization in Radiation Treatment Planning.* PhD thesis, University of Wisconsin, Madison, Wisconsin, 2002.

[15] G. J. Lim, M. C. Ferris, and D. M. Shepard. Optimization tools for radiation treatment planning in Matlab. In M. L. Brandeau, F. Sainfort, and W. P. Pierskalla, editors, *Operations Research and Health Care: A Handbook of Methods and Applications*, pages 775–806. Kluwer Academic Publishers, Boston, MA, 2004.

[16] G. J. Lim, M. C. Ferris, S. J. Wright, D. M. Shepard, and M. A. Earl. An optimization framework for conformal radiation treatment planning. *INFORMS Journal on Computing*, 19(3):366–380, 2007.

[17] C. Men, H. E. Romeijn, Z. C. Taşkın, and J. F. Dempsey. An exact approach to direct aperture optimization in IMRT treatment planning. *Physics in Medicine and Biology*, 52(24):7333–7352, 2007.

[18] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, NY, 1999.

[19] F. Preciado-Walters, R. Rardin, M. Langer, and V. Thai. A coupled column generation, mixed integer approach to optimal planning of intensity modulated radiation therapy for cancer. *Mathematical Programming*, 101(2):319–338, 2004.

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

[21] H. E. Romeijn, R. K. Ahuja, J. F. Dempsey, and A. Kumar. A new linear programming approach to radiation therapy treatment planning problems. *Operations Research*, 54(2):201–216, 2006.

[22] 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(4):721–744, 1999.

[23] Z. C. Taşkın, J. C. Smith, H. E. Romeijn, and J. F. Dempsey. Optimal multileaf collimator leaf sequencing in IMRT treatment planning. Technical report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 2007.