A Shortest Path-based Approach to the Multileaf Collimator Sequencing Problem

Hadrien Cambazard, Eoin O'Mahony, and Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{h.cambazard|e.omahony|b.osullivan}@4c.ucc.ie

Abstract. The multileaf collimator sequencing problem is an important component in effective cancer treatment delivery. The problem can be formulated as finding a decomposition of an integer matrix into a weighted sequence of binary matrices whose rows satisfy a consecutive ones property. Minimising the cardinality of the decomposition is an important objective and has been shown to be strongly NP-Hard, even for a matrix restricted to a single row. We show that in this latter case it can be solved efficiently as a shortest path problem, giving a simple proof that the one line problem is fixed-parameter tractable in the maximum intensity. This result was obtained recently by [9] with a complex construction. We develop new linear and constraint programming models exploiting this idea. Our approaches significantly improve the best known for the problem, bringing real-world sized problem instances within reach of complete methods.

1 Introduction

Radiation therapy is a treatment modality that uses ionising radiation in the treatment of patients diagnosed with cancer (and occasionally benign disease). Radiation therapy represents one of the main treatments against cancer, with an estimated 60% of cancer patients requiring radiation therapy as a component of their treatment. The aim of radiation therapy is to deliver a precisely measured dose of radiation to a well-defined tumour volume whilst sparing the surrounding normal tissue, achieving an optimum therapeutic ratio. Recent progress in technology and computing science have allowed significant improvement in the planning and delivery of all radiation therapy techniques.

Our primary *objective* is to apply recent advances in constraint programming to multileaf collimator sequencing in intensity-modulated radiotherapy (IMRT). At the core of advanced radiotherapy treatments are hard combinatorial optimisation problems, which are typically computationally intractable (Section 2 and 3). The *contributions* of this paper rely on the insight that the multileaf collimator sequencing problem restricted to a single row can be solved as a shortest path problem. A similar but more general result was obtained recently by [9]. We give a simple proof that the single row problem is fixed-parameter tractable in the maximum intensity of the row (Section 4) and exploit this insight to develop novel linear and constraint programming models (Section 5). These approaches significantly out-perform the best known for the problem, and bring real-world sized instances within reach of complete methods (Section 6).

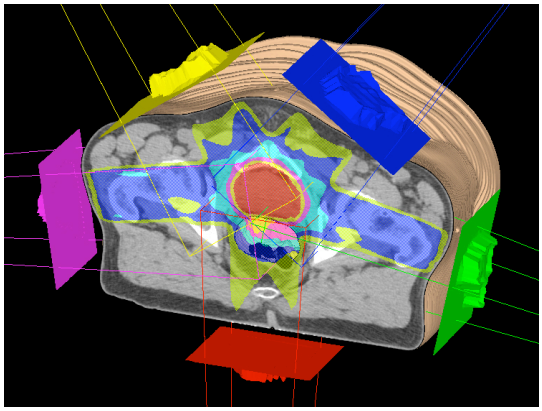
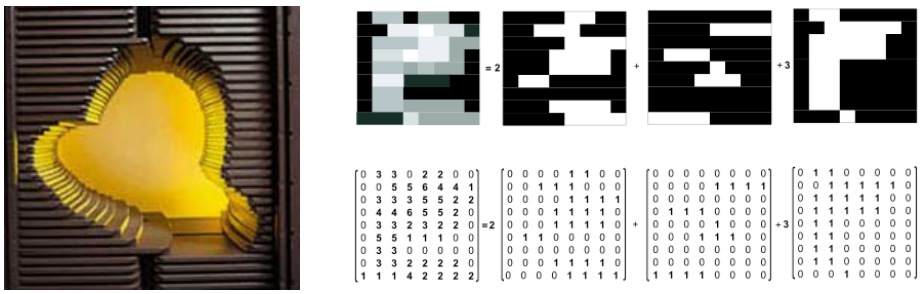


Fig. 1. An example IMRT treatment plan (Courtesy of the Advanced Oncology Center, Inc.).

2 Intensity-Modulated Radiotherapy

IMRT is an advanced mode of high-precision radiotherapy that utilises computer controlled x-ray accelerators to deliver precise radiation doses to a malignant tumour, or specific areas within the tumour. A treatment plan is devised for an individual patient based on the three-dimensional (3D) shape of the patient's tumour. Figure 1 presents an example IMRT treatment plan, clearly showing the location of the tumour in the centre of the image, the positions from which the tumour will be irradiated, and the dosage to be delivered from each position. The treatment plan is carefully developed based on 3D computed tomography images of the patient, in conjunction with computerised dose calculations to determine the dose intensity pattern that will best conform to the tumour shape. There are three optimisation problems relevant to this treatment. Firstly, the *geometry problem* considers the best positions for the beam head from which to irradiate. Secondly, the *intensity problem* is concerned with computing the exact levels of radiation to use in each area of the tumour. Thirdly, the *realisation problem*, tackled in this paper, deals with the delivery of the intensities computed in the intensity problem.

Combinatorial optimisation methods in cancer treatment planning have been reported as early as the 1960s [3]. There is a large literature on the optimisation of IMRT, which has tended to focus on the realisation problem [8]. Most researchers consider the sequencing of multileaf collimators (Figure 2(a)). The typical formulation of this problem considers the dosage plan from a particular position as an integer matrix, in which each integer corresponds to the amount of radiation that must be delivered to a particular region of the tumour. The requisite dosage is built up by focusing the radiation beam using a multileaf collimator, which comprises a double set of metal leaves that close from the outside inwards. Therefore, the collimator constrains the possible set of shapes that can be treated at a particular time. To achieve a desired dosage, a sequence of settings of the multileaf collimator must be used. One such sequence is presented in Figure 2(b). The desired dosage is presented on the left, and it is delivered through a sequence of three settings of the multileaf collimator, which are represented by three



(a) A multileaf collimator. (b) A multileaf sequencing problem.

Fig. 2. A simplified view of the optimisation problem associated with sequencing multileaf collimators in IMRT, Figure 2(b) has been adapted from [1].

matrices. Each matrix is exposed for a specific amount of time, corresponding to the weight associated with the matrix, thus delivering the requisite dosage.

Formally, this problem can be formulated as the decomposition of an integer matrix into a weighted sum of 0/1 matrices, in which each row has the "consecutive ones property" [2, 6]. The state-of-the-art approach is based on constraint programming [1].

3 Formulation of the Multileaf Collimator Sequencing Problem

We present a direct formulation of the multileaf collimator sequencing problem. Let I represent the dosage intensity matrix to be delivered. We represent this as an $m \times n$ (rows $\times$ columns) matrix of non-negative integers. We assume that the maximum dosage that is delivered to any region of the tumour is M units of radiation. Therefore, we set $I_{ij} \leq M, 1 \leq i \leq m, 1 \leq j \leq n$.

To ensure that each step in the treatment sequence corresponds to a valid setting of the multileaf collimator, we represent each step using a 0/1 matrix over which we specify a row-wise *consecutive ones* property (C1). Informally, the property requires that if any ones appear in a row, they appear together in a single block. A C1 matrix is a binary matrix in which every row satisfies the consecutive ones property. Formally, x is an $m \times n$ C1 matrix if and only if for any line i, $1 \leq a < b < c \leq n$,

$$x_{ia} = 1 \wedge x_{ic} = 1 \rightarrow x_{ib} = 1. \tag{1}$$

A solution to the problem is a sequence of C1 matrices, Ω, in which each x_k is associated with a positive integer b_k such that: $I = \sum_{k \in \Omega}(b_k \cdot x_k)$. Let B and K be the sum of coefficients b_k and the number of matrices x_k used in the decomposition of I, respectively. Then $B = \sum_{k \in \Omega} b_k$ and $K = |\Omega|$. B is referred to as the total *beam-on time* of the plan and K is its *cardinality*. The typical problem is to minimise B or K independently (known as the decomposition time and decomposition cardinality problem, respectively) or a combination of both. The minimisation of B alone is known to be linear [2, 6], while minimizing K alone is strongly NP-Hard [2]. We will tackle the

formulation preferred by practitioners, and mostly used in the literature so far, which is to minimise B first and then K (see Figure 2(b)). The problem is the following: given the optimal value B^* of B, find a treatment plan that minimises K, i.e.

$$\begin{array}{l} Minimise(K) \quad \text{such that} \\ \sum_{k\in\Omega} b_k = B^* \\ I = \sum_{k\in\Omega} b_k x_k \\ \forall k \in \Omega, x_k \text{ is a C1 matrix.} \end{array}$$

We now briefly explain how B^* can be found. The minimum sum of weights needed to have a C1 decomposition of a (single) row matrix $[I_1, \ldots, I_n]$ can be computed as:

$$\sum_{i=0}^{n-1} max(I_{i+1} - I_i, 0) \tag{2}$$

assuming $I_0 = 0$ [2, 6]. The expression $I_{i+1} - I_i$ represents the supplementary sum of weights needed for I_{i+1}, the remainder being reused without breaking the consecutive ones property. We provide a small example to help understand Equation 2.

Example 1 (Computing the Minimum Sum of Weights). Consider a dosage plan $I = [3, 2, 0, 3]$. The minimum sum of weights computed according to the previous formula would be $3 + 0 + 0 + 3 = 6$. The weights used to achieve the first value 3 could be reused for the following 2, but all weights already used before the 0 cannot be re-used for the last 3, since all the corresponding row matrices must have a 0 in this position to satisfy the C1 property. Then, three more unit of weights are needed to make the last 3, giving a decomposition $I = (1, 0, 0, 0) + 2 \cdot (1, 1, 0, 0) + 3 \cdot (0, 0, 0, 1)$. ▲

The B^* corresponding to the whole matrix is the maximum of the B^* values amongst the m rows of the matrix. A number of CP models for the multileaf collimator sequencing problem have been proposed in the literature. We briefly present these below.

3.1 The Direct Model

For a fixed K, the problem specification given above can be almost directly encoded with a variable per coefficient of the decomposition and a variable per cell of the matrices in the decomposition, as follows:

$$\begin{array}{lll} Variables: & \forall k \leq K & b_k \in \{1, \ldots, M\} \\ & \forall k \leq K, i \leq m, j \leq n, & x^k_{ij} \in \{0, 1\} \\ \\ DM_1: & & \sum_{k\leq K} b_k = B^* \\ DM_2: & & b_1 \geq b_2, \ldots \geq b_k \\ DM_3: & \forall k \leq K, i \leq m, & \textsc{ConsecutiveOnes}(\{x^k_{i1}, \ldots, x^k_{in}\}) \\ DM_4: & \forall i \leq m, j \leq n & \sum_{k\leq K} b_k \times x^k_{ij} = I_{ij} \end{array}$$

The CONSECUTIVEONES constraint (DM_3) can be implemented using a contiguity constraint [10] or the REGULAR global constraint [12] with a straightforward deterministic finite automaton. Constraint DM_2 eliminates symmetries amongst the weights. A number of symmetries amongst the x^k variables remain. We quote here the example given in [1] to highlight this important drawback of the Direct Model.

Example 2 (Symmetries of the Direct Model). Consider part of a decomposition with two identical weights of value 2. The two following decompositions are symmetrical.

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

The values in the right bottom corners of the matrices can be swapped without changing the solution since the weights are identical. ▲

Symmetries due to identical weights can be partially avoided by dynamically adding lexicographic constraints on the rows and columns (once the weights are known in the search) but that would not be enough. Rows and columns remain lexicographically ordered in this example. The next model was proposed in [1] to address this issue.

3.2 The Counter Model

The Counter Model is based on N_b variables representing the number occurrences of weight b in the decomposition. Q_{ij}^b variables refine this information by counting the number of times a weight b contributes to the sum of I_{ij}. The model is stated as follows:

$$\begin{array}{lll}
Objective: & Minimise(K) \quad \text{such that:} & \\
Variables: & \forall b \leq M & N_b \in \{0,\ldots,B^*\} \\
& \forall i \leq m, j \leq n, b \leq M & Q_{ij}^b \in \{0,\ldots,M\} \\
\\
CM_1: & \forall i \leq m, j \leq n & \sum_{b=1}^{M} b \times Q_{ij}^b = I_{ij} \\
CM_2: & & \sum_{b=1}^{M} b \times N_b = B^* \\
CM_3: & & \sum_{b=1}^{M} N_b = K \\
CM_4: & \forall b \leq M, i \leq m, & \textsc{SumOfIncrements}(\{Q_{i1}^b,\ldots,Q_{in}^b\}, N_b)
\end{array}$$

Constraint $CM1$ ensures that each element of I is properly decomposed. $CM2$ and $CM3$ relate the N_b variables to B^* and K. The model makes use of an ad-hoc global constraint to enforce the C1 property of this decomposition expressed in term of occurrences of each weight. It is the SUMOFINCREMENTS [4], defined by:

$$\textsc{SumOfIncrements}(\{V_1,\ldots,V_n\},U) \equiv \sum_{i=0}^{n-1} max(V_{i+1}-V_i,0) \leq U \text{ with } V_0 = 0 \quad (3)$$

A C1 decomposition of I can be derived from a C1 decomposition of Q^b for each b. The key intuition behind this model is that it is sufficient to find an unweighted decomposition (only with weights of 1) of each Q^b instead of looking for a decomposition of I (see [1]). Keep in mind that Q^b is the matrix defining the number of times a weight equal to b is used to decompose each element of I, thus a weighted decomposition of Q^b would result in identical matrices. The cardinality would, therefore, be reduced by merging the corresponding matrices and increasing the weight. Thus all matrices have to be different and the decomposition of Q^b is necessarily unweighted in an optimal solution.

As explained earlier in Section 3, the expression $\sum_{i=1}^{n} max(V_{i+1}-V_i,0)$ is a convenient way to compute the minimum sum of weights needed for a C1 decomposition of a row. Obviously, if we seek an unweighted decomposition, the formula returns the minimum *number* of weights needed. Therefore, this formula can be used as a lower bound for N_b to ensure the C1 property (constraint CM_4).

4 The Single Row Problem as a Shortest Path

As mentioned previously, finding the minimum total beam-on time, B, for a given intensity matrix can be solved in linear time. However, minimising the cardinality of the multileaf collimator sequence is NP-hard [5]. More recently, it has been shown that even when restricting the problem to a single row of the intensity matrix, minimising the cardinality is strongly NP-Hard [2]. This result was refined by [9] who showed that not only is the single row problem polynomial when the maximum intensity is bounded, but also the complete problem. In this section we show a simple construction representing the single row problem as a shortest path. This gives a simple proof that the single row problem is fixed-parameter tractable (FPT) in the maximum element in the row I. Although [9] achieves a better complexity, we will develop very efficient algorithms based on our construction outperforming the results of [9] in practice. In general, a problem is FPT with respect to a parameter k if there exists an algorithm for it that has running time $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$, where n is the size of the problem and $f(k)$ is an arbitrary function depending only on k. This result will be the keystone to designing very efficient linear and CP models in the remainder of this paper.

C1 DECOMPOSITION CARDINALITY PROBLEM (DC)
Instance: A row matrix of n integers, $I = \langle I_1, \ldots, I_n \rangle$, a positive integer K.
Question: Find a decomposition of I into at most K C1 row matrices.

In any solution of the DC problem, there must be a subset of the weights of the decomposition that sum to every element I_i of the row. In other words, the decomposition must contain an integer partition of every intensity. To represent these integer partitions the following notation will be used: $P(a)$ is the set of partitions of integer a, $p \in P(a)$ is a particular partition of a, and $|p|$ its number of integer summands in p. We denote by $occ(p, v)$, the number of occurrences of value v in p. For example, $P(5) = \{\langle 5\rangle, \langle 4,1\rangle, \langle 3,2\rangle, \langle 3,1,1\rangle, \langle 2,2,1\rangle, \langle 2,1,1,1\rangle, \langle 1,1,1,1,1\rangle\}$, and if $p = \langle 3,1,1\rangle$ then $|p| = 3$ and $occ(p,1) = 2$.

Observe that the DC problem can be formulated as a shortest path problem in a weighted directed acyclic graph, G, which we refer to a *partition graph*. A partition graph G of a row matrix $I = \langle I_1, \ldots, I_n \rangle$ is a layered graph with $n+2$ layers, the nodes of each layer i corresponding to the set of integer partitions of the row matrix element I_i. A source and sink nodes, denoted p_0 and p_{n+1} respectively, are associated with the empty partition $\emptyset$ for sake of simplicity. Two adjacent layers form a complete bipartite graph and the cost added to an edge, $p_i \rightarrow p_j$, between two partitions, p_i and p_j represents the number of additional weights that need to be added to the decomposition to satisfy the C1 property when decomposing the two consecutive elements with the corresponding partitions. The cost of each edge $p_i \rightarrow p_j$ in the partition graph is:

$$c(p_i, p_j) = \textstyle\sum_{b=1}^{M} c(b, p_i, p_j) \tag{4}$$

where $c(b, p_i, p_j) = max(occ(b, p_j) - occ(b, p_i), 0)$. A shortest path in the partition graph answers DC.

Example 3 (A Partition Graph). Consider a single row intensity matrix $I = [3, 2, 3, 1]$. The partition graph for this row problem is presented in Figure 3. Excluding the source

and sink, there are four levels, one corresponding to each element of I. The costs associated with the edges are computing using Equation 4. For example, the cost associated with the edge between the partition $\langle 1,1,1\rangle$ of element 3 of layer 1 and partition $\langle 2\rangle$ of element 2 represents the extra weight that must be added to decompose element 2 if $\langle 1,1,1\rangle$ is used for element 3. In other words, as one moves along a path in this graph the partition chosen to decompose the element at layer i contains the only weights that can be reused to decompose the element at layer $i+1$ because of the C1 property. ▲

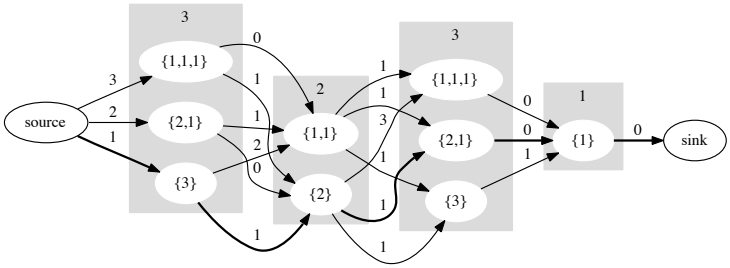


Fig. 3. A partition graph, showing transition weights, for the single row intensity matrix $I = [3,2,3,1]$. A shortest path is indicated in bold.

This formulation is only a refinement over the Counter Model which gives an easy way to show that this algorithm is correct. Consider a row I of n elements and its partition graph G. A path $\Pi = \langle p_0, \ldots, p_{n+1}\rangle$ in G defines a decomposition of I whose cardinality is the length of the path: $K = \sum_{i=0}^{n} c(p_i, p_{i+1})$. From this path Π, we can build a solution to the Counter Model (without the beam-on time constraint CM_2) by setting $N_b = \sum_{i=0}^{n} c(b, p_i, p_{i+1})$ and $Q_i^b = occ(b, p_i)$. Constraint CM_1 is satisfied as p_i is an integer partition of I_i. Constraint CM_2 is ignored as we are in the case of the unconstrained cardinality. Constraint CM_4 is satisfied because $N_b = \sum_{i=0}^{n} c(b, p_i, p_{i+1}) = \sum_{i=1}^{n} max(Q_{i+1}^b - Q_i^b, 0)$ which is the SUMOFINCREMENTS constraint. Finally, one can check the cardinality of the path (constraint CM_3) by computing the sum of the N_b variables: $\sum_{b=1}^{M} N_b = \sum_{b=1}^{M}\sum_{i=0}^{n} c(b, p_i, p_{i+1}) = \sum_{i=0}^{n}\sum_{b=1}^{M} c(b, p_i, p_{i+1}) = \sum_{i=1}^{n} c(p_i, p_{i+1}) = K$. As a path encodes a solution to the Counter Model, and the length of the path is exactly the cardinality, the shortest path gives the optimal K and an answer to DC. We now consider the single row problem when constraining the beam-on time.

C1 DECOMPOSITION-CARDINALITY WITH TIME CONSTRAINT (DCT)
Instance: A row matrix of n integers, $I = [I_1, \ldots, I_n]$, positive integers K and B.
Question: Find a decomposition of I into at most K C1 row matrices such that the sum of its weights is at most B.

To deal with this problem we extend the previous graph with a resource for every edge:

$$r(p_i, p_j) = \sum_{b=1}^{M} b \times c(b, p_i, p_j). \tag{5}$$

Finding a shortest path $\Pi = \langle p_0, \ldots, p_{n+1} \rangle$ in the partition graph whose sum of weights, $\sum_{i=0}^{n} r(p_i, p_{i+1})$, is at most B is a shortest path problem with resource constraints (SPPRC). The two-resource SPPRC is better known as the shortest path problem with time windows (SPPTW), which was studied initially by [11]. A single time window $[0, B]$ can be added to the sink node, capturing constraint CM_2 of the Counter Model. The problem is NP-Hard, but pseudo-polynomial algorithms do exist based on dynamic programming. An algorithm of complexity $O(n^2 B)$ is given in [11], where n is the number of nodes of the graph.

Example 4 (Encoding DCT as a SPPTW). The new partition graph of $I = [3, 2, 3, 1]$, with a cost and resource consumption per edge is given Figure 4. ▲

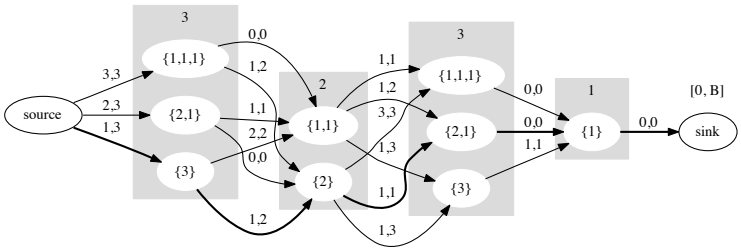


Fig. 4. Encoding an example DCT problem.

This formulation of the single row problem corresponds to a simple FPT result.

Theorem 1 (Fixed-Parameter Tractibility of the Single Row Problem). *Finding an optimal solution to the DC and DCT problems is fixed-parameter tractable in the size of the maximum element of the single-row intensity matrix, I.*

Proof. Let k be the maximum element of the row matrix I. The number of edges in the partition graph is bounded by $|P(k)|^2 \times n$ because the number of nodes of a layer i is the number of integer partitions of the corresponding integer value I_i. In this acyclic graph, solving a simple shortest path problem can be done in $\mathcal{O}(|P(k)|^2 \times n)$. The time complexity can be written as $\mathcal{O}(nf(k))$, where $f(k) = |P(k)|^2$, showing that DC is fixed-parameter tractable in k. Similarly for DCT, using the pseudo-polynomial algorithm of [11] we get a complexity of $\mathcal{O}(n^2 |P(k)|^2 B)$. B can be bounded by nk giving a time complexity of $\mathcal{O}(n^3 f(k))$, with $f(k) = k|P(k)|^2$. ■

In [9] a more sophisticated construction for the shortest path is presented giving an $\mathcal{O}(n)$ complexity for DCT whereas our representation as a SPPTW gives a complexity in $\mathcal{O}(n^3)$. In [4] a global constraint, called the SUMOFINCREMENTS was proposed for maintaining the C1 property and a bounds consistency algorithm in $\mathcal{O}(n)$ was given. An $\mathcal{O}(nd^2)$ arc-consistency algorithm can be obtained based on finding shortest paths.

Corollary 1. *Generalised Arc Consistency on the* SUMOFINCREMENTS *constraint can be achieved in $\mathcal{O}(nd^2)$ where n is the number of variables and d the maximum domain size.*

Proof. We recall that SUMOFINCREMENTS($\{V_1, \ldots, V_n\}, U$) is equivalent to the expression $\sum_{j=0}^{n-1} max(V_{j+1} - V_j, 0) \leq U$ with $V_0 = 0$. Consider a layered graph in which each layer corresponds to a variable V_j and each node of layer j corresponds to the values of $D(V_j)$. The cost associated with two values $a \in D(V_j)$ and $b \in D(V_{j+1})$ is simply $max(b - a, 0)$. Consider an instantiation of all the V_j variables. The value of the expression $\sum_{j=0}^{n-1} max(V_{j+1} - V_j, 0)$ is obviously given by the cost of the corresponding path. Ensuring that the SUMOFINCREMENTS is GAC can be easily done by checking that the value of the shortest path in the layered graph is less than the upper bound of U. The shortest path from the source to all nodes and from all nodes to the sink can be obtained in $\mathcal{O}(e)$ where e is the number of edges of the layered graph. Thus, the filtering process can be done in $\mathcal{O}(nd^2)$ where d is the maximum domain size. ■

5 Shortest Path-based Models

5.1 A Shortest Path Constraint Programming Model

We index, in lexicographic order, the integer partitions of each element I_{ij} of the intensity matrix, and use an integer variable P_{ij} to denote which partition is used to decompose element I_{ij}. For example, $P(5) = \{\langle 5\rangle, \langle 4,1\rangle, \langle 3,2\rangle, \langle 3,1,1\rangle, \langle 2,2,1\rangle, \langle 2,1,1,1\rangle, \langle 1,1,1,1,1\rangle\}$, so $P_{ij} = 4$ means that the weights $3, 1$ and 1 are used to sum to this element in the decomposition. The domain of P_{ij}, denoted $D(P_{ij})$ thus ranges from 1 to $|P(I_{ij})|$. We also have a variable N_b giving the number of occurrences of weight b in the decomposition, similar to the Counter Model presented earlier.

Our CP model makes use of the SHORTESTPATH($G, \{P_1, \ldots, P_n\}, U$) constraint, which enforces U to be greater than the shortest path in a graph G. Our CP model posts the SHORTESTPATH constraint over three different graphs $G_1(i)$, $G_2(i, b)$, $G_3(i)$, which although topologically identical, are weighted using three different costs:

$$\begin{array}{ll} c_1(p_i, p_j) & = \sum_{b=1}^{M} c_2(b, p_i, p_j) \\ c_2(b, p_i, p_j) & = max(occ(b, p_j) - occ(b, p_i), 0) \\ c_3(p_i, p_j) & = \sum_{b=1}^{M} b \times c_2(b, p_i, p_j) \end{array} \quad (6)$$

Example 5 (Example of the Costs G_1, G_2, G_3). Consider $I = [3, 2, 3, 1]$. The three partition graphs are identical in structure, only the costs vary. G_1, G_2 for value $b = 1$ and G_3 are shown in Figure 5, giving the three costs c_1, c_2, c_3, respectively. ▲

Therefore, our CP model is summarised as follows:

$$\begin{array}{lll} Objective: Minimise(K) \text{ with } K \in \{0, \ldots, B^*\} & & \\ & \forall b \leq M & N_b \in \{0, \ldots, B^*\} \\ & \forall i \leq m, j \leq n, & P_{ij} \in \{1, \ldots, |P(I_{ij})|\} \\ & & \\ CP_1: & & \sum_{b=1}^{M} b \times N_b = B^* \\ CP_2: & & \sum_{b=1}^{M} N_b = K \\ CP_3: & \forall i \leq m, & \text{SHORTESTPATH}(G_1(i), \{P_{i1}, \ldots, P_{in}\}, K) \\ CP_4: & \forall i \leq m, b \leq M & \text{SHORTESTPATH}(G_2(i, b), \{P_{i1}, \ldots, P_{in}\}, N_b) \\ CP_5: & \forall i \leq m, & \text{SHORTESTPATH}(G_3(i), \{P_{i1}, \ldots, P_{in}\}, B^*) \\ CP_6: & \forall i \leq m, \forall j < m \text{ s.t } I_{ij} = I_{i,j+1} & P_{ij} = P_{i,j+1} \end{array}$$

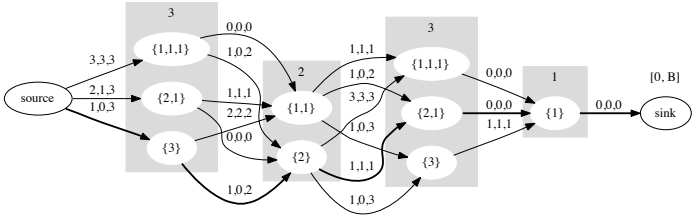


Fig. 5. Example of the three graph costs used in our CP model.

The C1 property of the decomposition is enforced by constraints CP_4. The number of weights of each kind, b, needed so that a C1 decomposition exists for each line i is maintained as a shortest path in $G_2(b, i)$. As those shortest paths are computed independently, maintaining a shortest path in $G_1(i)$ provides a lower bound on the cardinality needed for the decomposition of each line i. This is the purpose of CP_3, which acts as a redundant constraint. Finally CP_5 is a useful redundant shortest path constraint that maintains the minimum value of B associated with each line, which can provide valuable pruning by strengthening CP_1. CP_6 breaks some symmetries by stating that the same partition can be used for two consecutive identical elements in the same row. If the two partitions were different, a solution could be obtained by using any of the two partitions for the two elements. This could not violate the C1 property as the elements are consecutive and any of those two partitions was also satisfying the C1 property.

Filtering the SHORTESTPATH Constraint. The shortest path constraint has already been studied in Constraint Programming [7]. Here, the SHORTESTPATH constraint is simple as the graph is layered and contains only non-negative costs. The constraint SHORTESTPATH$(G, \{P_1, \ldots, P_n\}, U)$ states that U is greater than the shortest path in the partition graph defined by the domains of $\{P_1, \ldots, P_n\}$ and the cost information G. A layer i of the graph corresponds to variable P_i and the nodes of each layer to the domain values of P_i. Our implementation of the constraint maintains for every node α of layer i, the value of the current shortest path from the source, $S^{\alpha}_{\leftarrow}$, and to the sink, $S^{\alpha}_{\rightarrow}$. These two integers are restorable upon backtracking.

If a value is pruned from a layer we proceed with forward (resp. backward) phases to update the $S_{\leftarrow}$ (resp. $S_{\rightarrow}$) values maintaining the simple following equations :

$$\begin{array}{l} S^{\alpha}_{\leftarrow} = \min_{\beta \in D(P_{i-1})}(S^{\beta}_{\leftarrow} + c(\beta, \alpha)) \\ S^{\alpha}_{\rightarrow} = \min_{\beta \in D(P_{i+1})}(S^{\beta}_{\rightarrow} + c(\alpha, \beta)) \end{array} \tag{7}$$

The constraint is partially incremental, so if none of the $S_{\leftarrow}$ (resp. $S_{\rightarrow}$) values of the nodes on layer i have been updated, the process stops and does not examine layer $i+1$ (resp. $i-1$). At each update of a $S_{\leftarrow}$ or $S_{\rightarrow}$, we prune the corresponding value if $S_{\leftarrow} + S_{\rightarrow}$ is greater than the upper bound of U. The time complexity of the forward and backward step including the pruning is $\mathcal{O}(e)$ where e is the number of edges in the graph. $S^{sink}_{\leftarrow}$ (or $S^{source}_{\rightarrow}$) is used to update the lower bound of U. As the upper bound of U is not updated, there is no need to reach a fixed point and arc-consistency is achieved in $\mathcal{O}(e)$. Notice that this constraint could also be decomposed by introducing $S_{\leftarrow}$ and

$S_{\rightarrow}$ as variables, stating Equations 7 as constraints as well as $S^{\alpha}_{\leftarrow} + S^{\alpha}_{\rightarrow} > U \implies P_i \neq \alpha$.

*Example 6 (*SHORTESTPATH *using G_1).* Consider $I = [3, 2, 3, 1]$ and $U = 3$. The graph underlying SHORTESTPATH($G_1, \{P_1, \ldots, P_4\}, U$) is shown in Figure 6. The two restorable integers $S_{\leftarrow}$ and $S_{\rightarrow}$ are given for each node in brackets. A node filled in grey has been pruned because the sum of its two shortest paths is greater than 3. Values $\{1, 1, 1\}, \{1, 1\}$ and $\{1, 1, 1\}$ are pruned respectively from P_1, P_2 and P_3. ▲

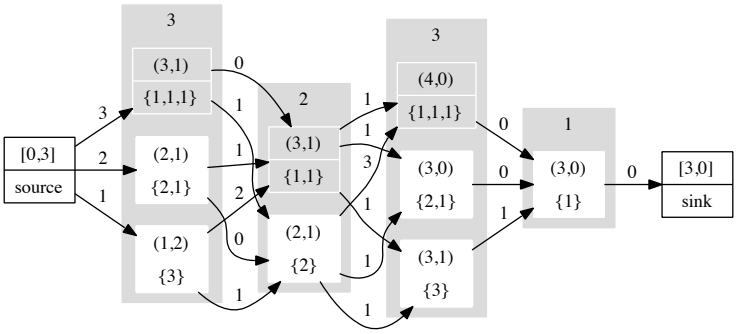


Fig. 6. The SHORTESTPATH using Cost G_1.

Note that our CP model is exponential in space as the implementation of the SHORTESTPATH constraint maintains information for each integer partition of the elements of the matrix. Therefore, the model strongly relies on the fact that the maximum intensity in the matrix is bounded in practice and instances with small intensities remain open.

Search. The branching strategy first assign the K variable in a bottom-up fashion (from its lower bound to its upper bound) until a feasible solution is found (the first feasible solution found is thus an optimal one). The branching then considers the N_b variables and proceeds with 'minimum domain first' variable ordering and lexicographic value ordering (from the lower bound to the upper bound of each N_b). Once the N_b variables are known, the problem is split into m independent sub-problems (one per row). Those problems are solved independently by branching on the P variables, again using minimum domain variable ordering and lexicographic value ordering. The rows are examined in decreasing value of their beam on-time, similar to [1]. Branching on P is mandatory, since the shortest paths on $G_2(i, b)$ are maintained independently for each b. At this stage we are facing a multi-resource constrained shortest path problem as we have a limit N_b of each resource b as well as a limit K on the shortest path in G_1.

5.2 A Shortest Path Linear Programming Model

A simple shortest path formulation in Linear Programming (LP) is unimodular, guaranteeing that the continuous relaxation provides an integral solution. We investigated if

encoding the previous model based on shortest path in LP could lead to a strong lower bound for the whole problem. The linear model simply introduces a boolean variable for each possible integer partition of each element I_{ij} of the intensity matrix.

$$\begin{array}{ll} \forall b \leq M & N_b \in \{0, \ldots, B^*\} \\ \forall i \leq m, j \leq n, p \leq |P(I_{ij})| & x_{ijp} \in \{0, 1\} \end{array}$$

Again, N_b denotes the number of occurrences of value b in the decomposition of B^*, whereas x_{ijp} indicates whether partition p is used or not to sum to I_{ij}. The consecutive ones property is enforced as a shortest path problem on each line in the partition graphs G_1, G_2 and G_3. The nodes of those graphs are mapped to the x_{ijp} variables and the costs are computed using Equations 6. $x_{i,0,0}$ and $x_{i,n+1,0}$ are two nodes acting as the source and sink of the graph of line i, respectively. A linear model called $SP(i)$ encoding the shortest path problem for each line i uses one variable per edge:

$$\forall j \leq n, p_\alpha \leq |P(I_{ij})|, p_\beta \leq |P(I_{i,j+1})| \;\; y_{i,j,p_\alpha,p_\beta} \in \{0, 1\}.$$

The variables y_{i,j,p_α,p_β} indicate whether or not the edge between partition p_α of layer j and partition p_β of layer $j+1$ is used in the solution of line i. The three shortest path constraints introduced in the CP model can be encoded using a simple linear model for shortest path by stating the flow conservation at each node. The following constraints encode the flow conservation, the three costs of the paths and channels the edge variables to the nodes variables, respectively.

$$\begin{array}{ll} \forall j \leq n, p_\alpha \leq |P(I_{ij})|, & \sum_{p_\beta \leq |P(I_{i,j-1})|} y_{i,j-1,p_\beta,p_\alpha} = \sum_{p_\beta \leq |P(I_{i,j+1})|} y_{i,j,p_\alpha,p_\beta} \\ & \sum_{p_\alpha \leq |P(I_{i1})|} y_{i,0,0,p_\alpha} = 1 \\ & \sum_{p_\alpha \leq |P(I_{in})|} y_{i,n,n+1,p_\alpha} = 1 \\ & \sum_{b \leq M} N_b \geq \sum_{j,p_\alpha,p_\beta} c_1(p_\alpha, p_\beta) \times y_{i,j,p_\alpha,p_\beta} \\ \forall b \in [1, \ldots, M] & N_b \geq \sum_{j,p_\alpha,p_\beta} c_2(b, p_\alpha, p_\beta) \times y_{i,j,p_\alpha,p_\beta} \\ & B^* \geq \sum_{j,p_\alpha,p_\beta} c_3(p_\alpha, p_\beta) \times y_{i,j,p_\alpha,p_\beta} \\ \forall j \leq n, p_\alpha \leq |P(I_{ij})|, & x_{ijp_\alpha} = \sum_{p_\beta \leq |P(c_{i,j+1})|} y_{i,j,p_\alpha,p_\beta} \\ \forall j \leq n, p_\alpha \leq |P(I_{ij})|, & x_{ijp_\alpha} = \sum_{p_\beta \leq |P(c_{i,j-1})|} y_{i,j-1,p_\beta,p_\alpha} \end{array}$$

The overall model is written in the following way:

$$\begin{array}{ll} minimise \sum_{b \leq M} N_b & \\ \forall i \leq m & SP(i) \\ & \sum_{b \leq M} b \times N_b = B^* \end{array}$$

The number of variables for this model is exponential as it depends on the number of integer partitions of the maximum element of the matrix, but this is bounded in practice.

6 Experimental Results

We performed a direct comparison between our CP model and the current state-of-the-art [1, 4] which showed our approach to be the fastest by more than two orders-of-magnitude, as well as the most scalable. It solves all 340 instances in the benchmark

suite whereas the best known approach can solve only 259 of them[1]. Secondly, we evaluated the quality of the continuous relaxation of our LP model, showing they typically were extremely close to optimal, and demonstrated that it could sometimes be useful for giving lower bounds to the CP model, providing significant speed-up over the CP model alone on large instances.

Table 1. Comparing the Shortest Path Model CPSP with the Counter Model.

Inst	CPSP					Counter model				
		Time (seconds)					Time (seconds)			
	NS	min	med	avg	max	NS	min	med	avg	max
12-12-10	20	0.11	0.18	0.25	0.66	20	0.32	0.83	1.00	3.62
12-12-11	20	0.14	0.22	0.71	3.32	20	0.67	2.27	2.63	6.23
12-12-12	20	0.23	0.50	0.94	6.94	20	1.04	3.91	4.76	12.30
12-12-13	20	0.28	1.63	1.93	4.77	20	2.26	7.13	8.57	30.50
12-12-14	20	0.35	1.59	3.28	26.36	20	1.19	9.63	11.58	49.76
12-12-15	20	0.61	5.76	12.70	74.59	20	4.37	23.00	40.68	156.23
15-15-10	20	0.13	0.31	0.73	5.67	20	2.51	13.16	14.18	46.14
15-15-12	20	0.41	1.29	3.86	18.20	20	9.02	53.41	105.22	475.95
15-15-15	**20**	1.55	15.45	28.98	102.92	16	111.01	587.73	790.11	3409.69
18-18-10	20	0.24	0.46	1.01	5.88	20	26.22	135.03	183.91	851.34
18-18-12	**20**	0.47	3.07	6.00	19.36	18	121.84	1131.85	1371.88	4534.41
18-18-15	**20**	2.35	20.47	64.85	571.19	6	2553.80	3927.24	3830.12	4776.23
20-20-10	**20**	0.23	0.52	3.99	43.11	19	81.63	660.03	1190.01	3318.89
20-20-12	**20**	0.72	5.10	15.34	83.03	10	666.42	2533.41	3105.34	6139.53
20-20-15	**20**	3.15	61.73	180.70	697.51	0	-	-	-	-
30-30-10	**20**	0.88	2.97	76.77	474.26	0	-	-	-	-
40-40-10	**20**	1.42	11.33	468.19	3533.22	0	-	-	-	-

Evaluation of the CP Model. We compared our CP Shortest Path model (CPSP), from Section 5.1, against the Counter model of [1, 4], which is the best known approach to this problem. The same 340 problem instances and an executable binary from [1] were kindly provided by the authors, facilitating a direct and fair comparison. The benchmarks comprised 17 categories of 20 instances ranging in size from 12×12 to 40×40 with maximum elements between 10 and 15, denoted m-n-M in our results tables. Table 1 reports the number of instances solved in each category (column NS), along with the minimum, median, average and maximum time for each category using a time limit of 2 hours on an iMac[2]. Our CPSP approach clearly outperforms the Counter Model as the size grows. On 20-20-10 instances where the Counter Model fails to solve one instance within two hours, the speed-up is almost two orders-of-magnitude. Our CP model is implemented in Choco[3] and Java, whereas the Counter Model is implemented in Mercury[4] and compiled to C. Results in [9] use 15×15 intensity matrices with a maximum element of 10, requiring up to 10 hours to solve using a 2GHz workstation.

Evaluation of the LP Model. Although the IP shortest-path model is not able to compete with CPSP, the continuous relaxation (LP) is very tight and leads to excellent lower bounds, which are often optimal for large instances. Table 2 reports, for each category,

[1] Benchmark suite available from `http://www.4c.ucc.ie/datasets/imrt`

[2] Mac OS X 10.4.11, 2.33 GHz Intel Core 2 Duo, 3 GB 667MHz DDR2 SDRAM.

[3] `http://choco.sourceforge.net`

[4] `http://nicta.com.au/research/projects/constraint_programming_platform`

Table 2. The quality and time taken to compute the linear programming relaxation.

Inst	%Opt	Avg time	Inst	%Opt	Avg time
12-12-10	95	1.76	18-18-10	100	3.96
12-12-11	85	2.52	18-18-12	95	16.91
12-12-12	95	5.00	18-18-15	100	93.97
12-12-13	95	7.91	20-20-10	100	4.69
12-12-14	95	13.79	20-20-12	90	18.41
12-12-15	60	26.91	20-20-15	95	136.97
15-15-10	95	2.61	30-30-10	95	13.40
15-15-12	85	9.86	40-40-10	100	24.86
15-15-15	85	50.04			

Table 3. Comparing the CP model with and without initial lower bounds from the LP relaxation.

Inst	CPSP						Hybrid = LP + CPSP					
		Time (seconds)				Nodes		Time (seconds)				Nodes
	NS	min	med	avg	max	avg	NS	min	med	avg	max	avg
12-12-10	20	0.05	0.10	0.14	0.60	125.55	20	1.25	1.79	1.86	2.78	108.10
12-12-11	20	0.07	0.12	0.43	2.25	259.25	20	1.82	2.56	2.80	5.26	201.20
12-12-12	20	0.14	0.32	0.66	5.47	194.65	20	3.24	5.11	5.38	8.94	156.35
12-12-13	20	0.18	1.24	1.46	3.70	250.00	20	4.16	8.02	8.64	15.95	171.80
12-12-14	20	0.23	1.17	2.61	21.16	373.25	20	5.39	14.63	15.40	26.07	298.00
12-12-15	20	0.40	4.64	10.65	63.98	611.85	20	16.39	30.36	35.27	85.61	518.05
15-15-10	20	0.07	0.20	0.51	4.07	301.15	20	1.70	2.54	2.76	5.04	177.45
15-15-12	20	0.25	1.05	3.24	15.75	389.15	20	7.61	10.22	11.66	25.50	289.50
15-15-15	20	1.13	13.08	25.37	89.55	938.35	20	31.13	56.63	62.75	138.34	613.65
18-18-10	20	0.16	0.34	0.82	5.30	367.05	20	2.50	3.87	4.24	6.31	296.30
18-18-12	20	0.25	2.66	5.31	18.10	598.95	20	11.73	18.75	18.83	31.84	409.50
18-18-15	20	1.81	17.28	56.03	494.07	1366.20	20	70.40	96.85	105.86	**169.47**	622.35
20-20-10	20	0.14	0.41	3.56	39.83	1313.40	20	2.52	5.31	5.20	9.84	564.15
20-20-12	20	0.45	4.59	13.98	73.91	1329.30	20	12.51	19.25	24.01	81.89	836.75
20-20-15	20	2.44	58.04	159.50	612.43	4435.65	20	92.43	155.19	207.95	635.92	2295.70
30-30-10	20	0.52	2.60	75.52	472.25	15771.05	20	10.73	14.87	**26.55**	**161.09**	7144.85
40-40-10	20	0.91	6.89	466.68	3631.50	130308.80	20	24.27	28.98	**49.07**	**209.02**	23769.35

the percentage of instances for which the optimal value of the relaxation matches the real optimal value, as well as the average time of LP. Table 3 compares the CP model (CPSP) against a hybrid approach in which lower bounds are first computed based on the LP to start the bottom-up approach of CPSP[5]. The LP was solved using the barrier algorithm with CPLEX (version 10.0.0). Although the hybrid model is often slowed down by the continuous relaxation (the minimum times of CPSP are far better than the minimum times of the hybrid), it scales better on the 40-40-10 instances. On 40-40-10, the hybrid approach is on average 9 times faster than CPSP.

7 Conclusion

We have provided a new approach to solving the Multileaf Collimator Sequencing Problem. Although the complexity of the resulting algorithm depends on the number of integer partitions of the maximum intensity, which is exponential, it can be used to design very efficient approaches in practice. We proposed a new CP and Linear models encoding each line as a set of shortest path problems and obtained two orders-of-magnitude

[5] These experiments ran as a single thread on a Dual Quad Core Xeon CPU, 2.66GHz with 12MB of L2 cache per processor and 16GB of RAM overall, running Linux 2.6.25 x64.

improvements compared to the best known method for this problem. The linear model is a very tight formulation giving excellent lower bounds for the cardinality. A simple hybrid approach, using the continuous relaxation at the root node before starting the search with CP, outperforms the CP model alone on large instances.

The resulting approaches strongly rely on the fact that the maximum radiation intensity is often small compared to the size of the matrix. It is, therefore, interesting to determine the complexity of the algorithm by the maximum intensity. [1] explains that in the instances available to them, the maximum intensity does not exceed 20 whereas the collimators can reach 40 rows. This limitation might, thus, not be critical in practice. However many possibilities remain to be investigated to allow better scaling in terms of the maximum element of the intensity matrix. The LP model typically has an exponential number of variables and could certainly be solved more efficiently using column generation techniques. Reasoning on the CP models could be strengthened by solving resource constrained shortest path for each row using dynamic programming and avoiding any branching on the partition variables. Finally, there are other objective functions to consider in this problem, which we will study in the future.

Acknowledgements. This work was supported by Science Foundation Ireland under Grant Number 05/IN/I886. We are indebted to Sebastian Brand for providing his benchmark instances and an executable version of the solver presented in [1].

References

1. D. Baatar, N. Boland, S. Brand, and P.J. Stuckey. Minimum cardinality matrix decomposition into consecutive-ones matrices: CP and IP approaches. In *CPAIOR*, pages 1–15, 2007.
2. D. Baatar, H.W. Hamacher, M. Ehrgott, and G.J. Woeginger. Decomposition of integer matrices and multileaf collimator sequencing. *Discrete Applied Mathematics*, 152(1-3):6–34, 2005.
3. G.K. Bahr, J.G. Kereiakes, H. Horwitz, R. Finney, J. Galvin, and K. Goode. The method of linear programming applied to radiation therapy planning. *Radiology*, 91:686–693, 1968.
4. S. Brand. The sum-of-increments constraints in the consecutive-ones matrix decomposition problem. In *SAC'09: 24th Annual ACM Symposium on Applied Computing*, 2009.
5. R. E. Burkard. Open problem session. In *Oberwolfach Conference on Combinatorial Optimization*, November 2002.
6. K. Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Applied Mathematics*, 152(1-3):35–51, 2005.
7. T. Gellermann, M. Sellmann, and R. Wright. Shorter path constraints for the resource constrained shortest path problem. In *CPAIOR*, pages 201–216, 2005.
8. H.W. Hamacher and M. Ehrgott. Special section: Using discrete mathematics to model multileaf collimators in radiation therapy. *Discrete Applied Mathematics*, 152(1-3):4–5, 2005.
9. T. Kalinowski. The complexity of minimizing the number of shape matrices subject to minimal beam-on time in multileaf collimator field decomposition with bounded fluence. *Discrete Applied Mathematics*, in press.
10. M.J. Maher. Analysis of a global contiguity constraint. In *In Workshop on Rule-Based Constraint Reasoning and Programming*, 2002.
11. D. Martin and S. Francois. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26(3):191–212, 1988.
12. G. Pesant. A regular language membership constraint for finite sequences of variables. In *CP*, pages 482–495, 2004.