Hybrid Methods for 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

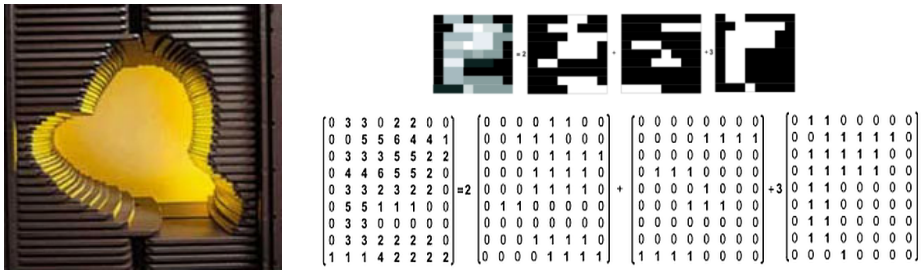
Abstract. The multileaf collimator sequencing problem is an important component of the effective delivery of intensity modulated radiotherapy used in the treatment of cancer. The problem can be formulated as finding a decomposition of an integer matrix into a weighted sequence of binary matrices whose rows satisfy a consecutive ones property. In this paper we extend the state-of-the-art optimisation methods for this problem, which are based on constraint programming and decomposition. Specifically, we propose two alternative hybrid methods: one based on Lagrangian relaxation and the other on column generation. Empirical evaluation on both random and clinical problem instances shows that these approaches can out-perform the state-of-the-art by an order of magnitude in terms of time. Larger problem instances than those within the capability of other approaches can also be solved with the methods proposed.

1 Introduction

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. At the core of advanced radiotherapy treatments are hard combinatorial optimisation problems. In this paper we focus on the multileaf collimator sequencing in intensity-modulated radiotherapy (IMRT).

What is 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. 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

* This work was supported by Science Foundation Ireland under Grant Number 05/IN/I886.



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

Fig. 1. A simplified view of the optimisation problem associated with sequencing multileaf collimators in IMRT, Figure 1(b) has been adapted from [3].

the intensity problem. Combinatorial optimisation methods in cancer treatment planning have been reported as early as the 1960s [5] and a recent interesting survey on the topic can be found in [14]. There is a large literature on the optimisation of IMRT, which has tended to focus on the realisation problem [18]. Most researchers consider the sequencing of multileaf collimators (Figure 1(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 given time. To achieve a desired dosage, a sequence of settings of the collimator must be used. One such sequence is presented in Figure 1(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 matrices. Each matrix is exposed for a specific amount of time, corresponding to the weight associated with the matrix, thus delivering the requisite dosage.

Contribution of this Paper. In our earlier work in this area we presented a novel approach to multileaf collimator sequencing using an approach based on shortest paths [10]. It was shown that such a model significantly out-performed the state-of-the-art and brought clinical-sized instances of the problem within the reach of constraint programming (CP). We now show that the shortest path idea can be exploited to give greater scalability by coupling the CP model with Lagrangian relaxation and column generation techniques. Our shortest-path approach to this problem uniquely provides a basis for benefitting from these techniques. The results presented define the current state-of-the-art for this challenging problem from the domain of cancer treatment planning.

The CP model presented in [10], is briefly introduced in Section 2. We show how to strengthen the CP model with a Lagrangian relaxation in Section 3. An alternative formulation in which the paths are represented explicitly, along with a column generation (CG) model, is presented in Section 4. Section 5 demonstrates that these approaches significantly out-perform the state-of-the-art for this problem.

2 Formulation of the Multileaf Collimator Sequencing Problem

Let I represent the dosage intensity matrix to be delivered. I is 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 a row-wise *consecutive ones* property (C1) must hold. 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, $1 \leq a < b < c \leq n$, $X_{ia} = 1 \wedge X_{ic} = 1 \rightarrow X_{ib} = 1$. 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*; see Figure 1(b) for an example with $K = 3$ and $B = 6$. The overall objective is to minimise the time needed for the complete treatment and the parameters B and K both affect that. Typical problems are to minimise B or K independently (known as the decomposition time and decomposition cardinality problem, respectively) or a linear combination of both: $w_1 K + w_2 B$. We will tackle this general formulation where w_1 accounts for the time needed by the operator to change the settings of the machine and w_2 accounts for the time to deliver one unit of radiation.

The problem of minimising B alone has been widely studied, starting with Bortdeld et al. [7] and Ahuja et al. [2] until a method in linear time was found by Baatar et al. and Engel [4, 15]. Minimising K alone was shown to be strongly NP-Hard [4] even for a single row or column [11] and received a lot of attention [6, 20]. Many heuristics were designed as the problem proved to be very difficult [1, 4]. The problem of minimising K while constraining B (lexicographic objective function) to its optimal value B^* was tackled by Engel and Kalinowski [15, 20]. Exact algorithms were proposed based on dynamic programming, Kalinowski [19], mixed integer linear programming, Langer [21, 26] and Constraint Programming Baatar et al., Ernst et al. and Cambazard et al. [3, 9, 10, 16]. Exact algorithms dealing with a more general objective function as the one used in this paper are designed by Wake et al, Caner Taskin et al [25, 26].

2.1 The Single Row Problem as a Shortest Path

In this section we study a restriction of the minimum cardinality problem DC to a single row. This will help to design efficient inference mechanisms for the general multi-row case. We show a simple construction representing the row problem as a shortest path.

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_j of the row. In other words, the decom-

position must contain an integer partition of every intensity. We will represent integer partitions with the following notation: $P(a)$ is the set of partitions of integer a, $p \in P(a)$ is a particular partition of a, and $|p|$ the number of integer summands in p. We denote by $occ(v, p)$ 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 $occ(1, p) = 2$. Observe that the DC problem can be formulated as a shortest path problem in a weighted directed acyclic graph, G, which we refer to as 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 j corresponding to the set of integer partitions of the row matrix element I_j. The size of this graph is therefore exponential in the maximum intensity. Source and sink nodes, located on layers 0 and $n + 1$ respectively, are associated with the empty partition $\emptyset$. Two adjacent layers form a complete bipartite graph and the cost added to an edge, $p_u \rightarrow p_v$, between two partitions, p_u and p_v of adjacent layers, 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 $p_u \rightarrow p_v$ in the partition graph is: $c(p_u, p_v) = \sum_{b=1}^{M} c(b, p_u, p_v)$ where $c(b, p_u, p_v) = max(occ(b, p_v) - occ(b, p_u), 0)$. Figure 2 shows the partition graph $I = [3, 2, 3, 1]$.

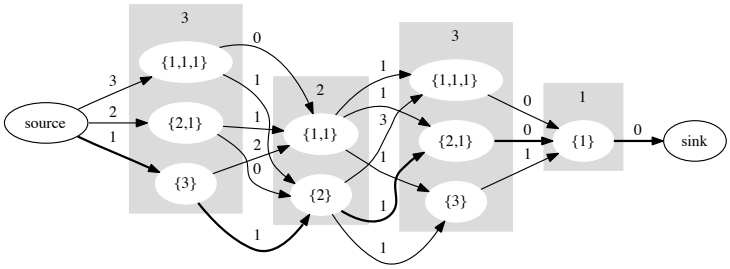


Fig. 2. A partition graph showing transition weights for the single row $I = [3, 2, 3, 1]$.

By following the path $\{\{2, 1\}, \{1, 1\}, \{2, 1\}, \{1\}\}$, we build a decomposition:

$[3, 2, 3, 1] = 2[1, ?, ?, ?] + 1[1, ?, ?, ?]$ (choice of $\{2, 1\}$);
$[3, 2, 3, 1] = 2[1, 0, 0, 0] + 1[1, 1, ?, ?] + 1[0, 1, ?, ?]$ (choice of $\{1, 1\}$);
$[3, 2, 3, 1] = 2[1, 0, 0, 0] + 1[1, 1, 0, 0] + 1[0, 1, 1, ?] + 2[0, 0, 1, 0]$ (choice of $\{2, 1\}$);
$[3, 2, 3, 1] = 2[1, 0, 0, 0] + 1[1, 1, 0, 0] + 1[0, 1, 1, 1] + 2[0, 0, 1, 0]$ (choice of $\{1\}$).

The length of the path represents the cardinality of the decomposition and a shortest path therefore provides a decomposition with minimum cardinality. The key idea is that as one moves along a path in this graph, the partition chosen to decompose the element at layer j contains the only weights that can be reused to decompose the element at layer $j+1$ because of the C1 property. Consider the previous example and the solution given. A coefficient 2 is used by the first partition but not by the second and thus becomes forbidden to decompose any other intensity values. The previous partition alone tells

us the available coefficients to decompose the present intensity value. This is why the *cardinality* cost can be defined between consecutive partitions and the whole problem mapped to a shortest path. We could also restrict the cost to a given weight b to obtain the cardinality of this particular coefficient. We will use this idea in the CP model.

2.2 Shortest Path Constraint Programming Model

We present a CP model for the general multi-row case that takes advantage of the property identified for a single row. 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 the index of the partition used to decompose element I_{ij}. For example, if $I_{ij} = 5$ the domain of P_{ij} is $\{1, ..., 7\}$ corresponding to $\{\langle 5\rangle, \langle 4, 1\rangle, \langle 3, 2\rangle, \langle 3, 1, 1\rangle$ $,\langle 2, 2, 1\rangle, \langle 2, 1, 1, 1\rangle, \langle 1, 1, 1, 1, 1\rangle\}$. Thus, $P_{ij} = 4$ means that the coefficients $3, 1$ and 1 are used to sum to 5 in the decomposition. We also have a variable N_b giving the number of occurrences of weight b in the decomposition.

Our CP model uses the constraint SHORTESTPATH($G, \{P_1, \ldots, P_n\}, U$) [10]. Once instantiated $\{P_1, \ldots, P_n\}$ defines a path in the original partition graph. This constraint states that U must be greater than or equal to the length of this path using the cost information G. We refer to it as SHORTESTPATH because it does not enforce U to be equal to the length of the path but rather greater than or equal to it, and the support for the lower bound on U is a shortest path. A layer j of the graph corresponds to variable P_j and the nodes of each layer to the domain values of P_j. Our CP model posts the SHORTESTPATH constraint over three different cost definitions $G_1(i)$, $G_2(i, b)$, $G_3(i)$ (the partition graphs of a line i are topologically identical). Denoting p_u the partition corresponding to value u of P_{ij} and p_v the partition corresponding to value v of $P_{i,j+1}$, the transition costs are as follows: $c_1(p_u, p_v) = \sum_{b=1}^{M} c_2(b, p_u, p_v)$, $c_2(b, p_u, p_v) = max(occ(b, p_v) - occ(b, p_u), 0)$ and $c_3(p_u, p_v) = \sum_{b=1}^{M} b \times c_2(b, p_u, p_v)$. Therefore, our CP model is as follows:

$$
\begin{array}{lll}
minimise\ \ w_1K + w_2B \ \text{ with} & & K \in \{0, \ldots, ub\}, B \in \{B^*, ..., ub\} \\
& \forall b \leq M & N_b \in \{0, \ldots, ub\} \\
& \forall i \leq m, j \leq n, & P_{ij} \in \{1, \ldots, |P(I_{ij})|\} \\
\\
CP_1: & & \sum_{b=1}^{M} b \times N_b = B \\
CP_2: & & \sum_{b=1}^{M} N_b = K \\
CP_3: & \forall i \leq m, & \text{SHORTESTPATH}(G_1(i), \{P_{i1}, \ldots, P_{in}\}, K) \\
CP_4: & \forall i \leq m, b \leq M & \text{SHORTESTPATH}(G_2(i, b), \{P_{i1}, \ldots, P_{in}\}, N_b) \\
CP_5: & \forall i \leq m, & \text{SHORTESTPATH}(G_3(i), \{P_{i1}, \ldots, P_{in}\}, B) \\
CP_6: & \forall i \leq m, \forall j < m \text{ s.t } I_{ij} = I_{i,j+1} & P_{ij} = P_{i,j+1}
\end{array}
$$

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 $G_2(i, b)$. CP_3 acts as a redundant constraint and provides a lower bound on the cardinality needed for the decomposition of each line i. CP_5 is another 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. Finally CP_6 breaks some symmetries. We refer the reader to [10] for more details in particular related to the SHORTESTPATH constraint.

3 A Hybrid Model based on Lagrangian Relaxation

Once the partition variables of a given line i are instantiated, they define a path in the original partition graph of the line. Constraints CP_3, CP_4, CP_5 constrain the length of this path, each with a different transition cost structure for the edges. The $M + 2$ path problems stated by constraints CP_3, CP_4, CP_5 on a given line define together a resource-constrained path problem. In this section we design a propagator to consider these paths simultaneously in order to achieve a higher degree of consistency for a single line. The underlying optimisation problem is the Resource Constrained Shortest Path Problem (RCSPP). The problem is to find a shortest path between a given source and sink so that the quantity of resources accumulated on each arc for each resource do not exceed some limits. Two approaches are often used to solve this problem: dynamic programming and Lagrangian relaxation. We base our propagator on the RCSPP and the multicost-regular constraint [23, 24].

We present one possible mapping of the problem stated by constraints CP_3, CP_4, CP_5 for line i to a RCSPP. We state it as a binary linear formulation where x^j_{uv} is a 0/1 variable denoting whether the edge between partition p_u and p_v of P_{ij} and $P_{i,j+1}$ is used. Layer 0 denotes the layer of the source and $n+1$ the one of the sink ($P_{i0} = P_{i,n+1} = \emptyset$). The problem formulation is as follows:

$$
\begin{array}{lll}
z = min & \sum_{j \leq n} \sum_{u,v \in P_{ij} \times P_{i,j+1}} c_3(p_u, p_v) \times x^j_{uv} & \\
\forall\, 1 \leq b \leq M & \sum_{j \leq n} \sum_{u,v \in P_{ij} \times P_{i,j+1}} c_2(b, p_u, p_v) \times x^j_{uv} & \leq \overline{N_b} \\
 & \sum_{j \leq n} \sum_{u,v \in P_{ij} \times P_{i,j+1}} c_1(p_u, p_v) \times x^j_{uv} & \leq \overline{K} \\
\forall\, 1 \leq j \leq n, u \in P_{ij} & \sum_{v \in P_{i,j-1}} x^{j-1}_{vu} - \sum_{v \in P_{i,j+1}} x^j_{uv} & = 0 \\
 & \sum_{v \in P_{i,1}} x^0_{1v} & = 1 \\
 & \sum_{u \in P_{i,n}} x^n_{u1} & = 1 \\
 & x^j_{uv} \in \{0, 1\} &
\end{array}
\tag{1}
$$

If the optimal value of the RCSPP, z^*, is less than or equal to $\overline{B}$, then there is a solution to constraints CP_3, CP_4, and CP_5, otherwise there is an inconsistency. The first two constraints in the formulation are resource constraints and the last three are the flow conservation constraints enforcing that the x variables define a path.

Lagrangian relaxation is a technique that moves the "complicating constraints" into the objective function with a multiplier, $\lambda \geq 0$, to penalise their violation. For a given value of λ, the resulting problem is the Lagrangian subproblem and, in the context of minimisation, provides a lower bound on the objective of the original problem. The typical approach is to relax the resource constraints, so the Lagrangian function is:

$$
\begin{array}{l}
f(x, \lambda) = \sum_j \sum_{u,v} c_3(p_u, p_v) \times x^j_{uv} \;+\; \lambda_0 (\sum_j \sum_{u,v} c_1(p_u, p_v) \times x^j_{uv} - \overline{K}) \\
\qquad\qquad + \sum_{1 \leq b \leq M} \lambda_b (\sum_j \sum_{u,v} c_2(b, p_u, p_v) \times x^j_{uv} - \overline{N_b})
\end{array}
\tag{2}
$$

The Lagrangian subproblem in this setting is, therefore, a shortest path problem $w(\lambda) = min_x f(x, \lambda)$ and the Lagrangian dual is to find the set of multipliers that provide the best possible lower bound by maximising $w(\lambda)$ over λ. A central result in Lagrangian relaxation is that $w(\lambda)$ is a piecewise linear concave function, and various algorithms can be used to optimise it efficiently.

Solving the Lagrangian Dual. We followed the approach from [23] and used a subgradient method [8]. The algorithm iteratively solves $w(\lambda)$ for different values of λ, initialised to 0 at the first iteration. The values of λ are updated by following the direction of a supergradient of w at the current value λ for a given step length μ. The step lengths have to be chosen to guarantee convergence (see [8]). We refer the reader to [23] for more details. At each iteration t, we solved the shortest path problem with the penalised costs on the edges:

$$c(p_u, p_v) = c_3(p_u, p_v) + \lambda_0^t c_1(p_u, p_v) + \sum_{1 \leq b \leq M} \lambda_b^t c_2(b, p_u, p_v).$$

This is performed by a traversal of the partition graph; as a byproduct we obtain the values of all shortest paths SO_a from the source to any node a. We can update the lower bound on B using:

$$SO_{sink} - \lambda_0^t \overline{K} - \sum_{1 \leq b \leq M} \lambda_b^t \overline{N_b}.$$

Then we perform a reversed traversal from the sink to the source to get the values of the shortest path SD_a from all nodes a to the sink. At the end of the iteration we mark all the nodes (partitions) that are infeasible in the current Lagrangian subproblem, i.e.:

$$SO_a + SD_a > \overline{B} + \lambda_0^t \overline{K} + \sum_{1 \leq b \leq M} \lambda_b^t \overline{N_b}.$$

At the end of the process, all nodes marked during the iterations are pruned from the domains. This is Lagrangian relaxation-based filtering [24]: if a value is proven inconsistent in at least one Lagrangian subproblem, then it is inconsistent in the original problem. The Lagrangian relaxation is incorporated into the constraint model as a global constraint for each line. The independent path constraints are kept and propagated first, whereas the propagation of the resource constrained path constraint is delayed since it is an expensive constraint to propagate.

4 A Column Generation Approach

Numerous linear models have been designed for this problem, see e.g. [14], but the shortest path approach [10] opens the door for a totally new formulation of the problem to be considered. In [10] we designed a linear model representing every integer partition. We now consider an alternative formulation that, rather than representing the partition graph, explicitly encodes the set of possible paths in the partition graph of each line. The resulting formulation is very large, but such models are typical in many settings, e.g. vehicle routing problems. The optimisation of these models can be performed using *column generation* [12]. The key idea is that the Simplex algorithm does not need to have access to all variables (columns) to find a pivot point towards an improving solution. The Simplex algorithm proceeds by iterating from one basic solution to another while improving the value of the objective function. At each iteration, the

algorithm looks for a non-basic variable to enter the basis. This is the *pricing problem*. Typically, for a linear minimisation problem written

$$min \sum_i c_i x_i \mid \forall j \sum_i a_{ij} x_i \geq b_j, x_i \geq 0,$$

the pricing problem is to find the i (a variable or column) that minimises $c_i - \sum_j \pi_j a_{ij}$ where π_j is the dual variable associated with constraint j. The explicit enumeration of all i is impossible when the number of variables is exponential. Therefore, the column generation works with a restricted set of variables, which define the *restricted master problem* (RMP) and evaluates reduced costs by implicit enumeration e.g., by solving a combinatorial problem. We now apply these concepts to our shortest path model.

4.1 Column Generation for the Shortest Path Model

We denote by pt_i^k the k^{th} path in the partition graph of line i. A path is a sequence of partitions $\langle p_0, \ldots, p_{n+1} \rangle$ characterised by three costs: the cardinality cost $c_{i1}^k = \sum_{j=0}^{j=n} c_1(p_j, p_{j+1})$, the beam-on time cost $c_{i3}^k = \sum_{j=0}^{j=n} c_3(p_j, p_{j+1})$ and the beam-on time cost restricted to a given coefficient b, $c_{ib2}^k = \sum_{j=0}^{j=n} c_2(b, p_j, p_{j+1})$. The restricted master problem where a subset Ω of the columns are present is denoted RMP(Ω), and can be formulated as follows:

$RMP(\Omega):$		$minimise$	$w_1 K + w_2 B$
C_0			$\sum_{b \leq M} N_b = K$
C_1			$\sum_{b \leq M} b \times N_b = B$
C_2	$\forall i,$		$\sum_{k \in \Omega_i} pt_i^k = 1$
C_3	$\forall i,$		$\sum_{k \in \Omega_i} c_{i1}^k \times pt_i^k \leq K$
C_4	$\forall i, \forall b$		$\sum_{k \in \Omega_i} c_{ib2}^k \times pt_i^k \leq N_b$
C_5	$\forall i,$		$\sum_{k \in \Omega_i} c_{i3}^k \times pt_i^k \leq B$
			$K \geq 0, B \geq 0, \ \forall b \, N_b \geq 0$
	$\forall i, k \in \Omega_i$		$pt_i^k \in \{0, 1\}$

This master problem optimises over a set of paths Ω_i per line i. The task of generating improving columns or paths is delegated to the sub-problem which is partitioned into m problems. The reduced cost of a path in a given line does not affect the computation of the reduced cost on another line. This is a typical structure for Danzig-Wolfe decomposition, and the constraints of the RMP involving the N_b variables are the coupling, or complicating, constraints. An improving column for line i is a path of negative reduced cost where the reduced cost is defined by $c_i - \sum_j \pi_j a_{ij}$. This translates as follows in our context. The M different costs on each edge are modified by a multiplier corresponding to the dual variables of constraints C_3 – C_5. We denote by δ_i, π_{i1}, π_{ib2}, and π_{i3} the dual variables associated with constraints C_2 to C_5, respectively. The reduced cost of a path variable is equal to $0 - (\pi_{i1} \times (-c_{i1}^k) + \sum_{b \leq M} \pi_{ib2} \times (-c_{ib2}^k) + \pi_{i3} \times (-c_{i3}^k))$. The subproblem of line i, $PP(i)$ is a shortest path problem where the cost of an edge $c(p_u, p_v)$ is computed as follows:

$$c(p_u, p_v) = \pi_{i1} \times c_1(p_u, p_v) + \sum_b \pi_{ib2} \times c_2(b, p_u, p_v) + \pi_{i3} \times c_3(p_u, p_v).$$

Algorithm 1: ColumnGeneration

Data: Intensity Matrix – A matrix of positive integers

Result: A lower bound on the optimal decomposition.

1 $\Omega = \emptyset$, $DB = -\infty$, $UB = +\infty$, $\epsilon = 10^{-6}$;
2 **for** $i \leq m$ **do**
3 add the path made of $\{1,\ldots,1\}$ partitions for each integer of line i to Ω;
4 set $\pi_{i1} = \pi_{i3} = \pi_{ib2} = -1$ for all b, solve $PP(i)$ and add the shortest path to Ω
5 **repeat**
6 add the paths in Ω to the restricted master problem, RMP;
7 solve RPM, set UB to the corresponding optimal value and record the dual values $(\delta_i, \pi_{i1}, \pi_{i3}, \pi_{ib2})$;
8 $\Omega = \emptyset$;
9 **for** $i \leq m$ **do**
10 solve the pricing problem $PP(i)$ and record its optimal value γ_i;
11 **if** $(\gamma_i - \delta_i) < -\epsilon$ **then**
12 add the optimal path to Ω
13 $DB = max(DB, \Sigma_{i \leq m} \gamma_i)$);
until $\lceil DB - \epsilon \rceil = \lceil UB \rceil$ *or* $\Omega = \emptyset$;
14 **return** $\lceil UB - \epsilon \rceil$

The column generation procedure is summarised in Algorithm 1. Notice that the bound provided by column generation is no better than the one given by the compact linear model because the pricing problem has the *integrality property*. The utility of this formulation is to give better scaling in terms of memory as we can achieve a tradeoff in the subproblem. We briefly explain the main phases of the algorithm.

Main Process. The algorithm must start with an initial set of columns that contain a feasible solution to obtain valid dual values. Lines 1 to 4 define the initialisation step where two paths, the *unit* path and the shortest path, are computed per line. Lines 6 to 14 specify the main column generation process. First, the new columns are added to the RMP, which is a continuous linear problem, and solved to optimality. UB denotes the upper bound provided by the optimal value of the RMP at each iteration. The pricing problem is then solved for each line using the dual values that are recorded (Line 7). Line 11 checks if a path of negative reduced cost has been found; $\gamma_i - \delta_i$ is the reduced cost of the path solution of $PP(i)$. Then, a lower bound on the original problem, the dual bound DB, is computed. The algorithm stops as soon as no path of negative reduced cost can be found ($\Omega = \emptyset$), or the lower and upper bounds have met ($\lceil DB - \epsilon \rceil = \lceil UB \rceil$).

Dual Lower Bound. The dual solution of the RMP, completed by the best reduced cost, forms a feasible solution of the dual of the original problem and, therefore, provides a lower bound. This dual bound DB is computed on Line 13 and we have:

$$DB = \sum_{i \leq m} \delta_i + \sum_{i \leq m} (\gamma_i - \delta_i) = \sum_{i \leq m} \gamma_i,$$

i.e., the sum of the dual objective function and the best reduced cost (see [12]). The dual bound provides a lower bound on the original problem and can be used for termination.

Typically, we can stop as soon as the optimal value is known to be in the interval $]a, a+1]$, in which case one can immediately return $a+1$ as the integer lower bound on the original problem (Condition $\lceil DB - \epsilon \rceil = \lceil UB \rceil$). This last condition is useful to avoid a convergence problem and saves many calls to the subproblems. The use of an ϵ is to avoid rounding issues arising from the use of continuous values.

Solving the Pricing Problem. The pricing problem involves solving a shortest path in a graph whose size is exponential in the maximum element, M. Storing the partition graph explicitly requires $O(n \times P^2)$ space, where P is the (exponentially large) number of partitions of M. Memory remains an issue for the column generation if we solve the pricing problems by explicitly representing the complete graph. To save memory as M increases, the column generation procedure can avoid representing the whole graph by only storing the nodes, thus consuming only $O(nP)$ space. In this case the costs on each edge must be computed on demand as they cannot be stored. In practice, the previous compromise with $O(nP)$ space consumption is perfectly acceptable as instances become very hard before the space again becomes an issue. In our implementation we use a combined approach whereby we store the edges when the two consecutive layers are small enough and only recompute the cost on the fly if it is not recorded.

Speeding up the Column Generation Procedure. The column generation process is known to suffer from convergence problems [12]. In our case, an increase in the value of M implies more time-consuming pricing problems, and the bottleneck lies entirely in this task in practice. We obtained some improvement with a simple stabilisation technique [13, 22] to reduce degeneracy. We added surplus variables, y, to each constraint (except for the convexity constraints) so that constraints C_3 to C_5 read as:

$$\sum_{k \in \Omega_i} c^k_{i1} \times pt^k_i - y_{3i} \leq K; \qquad \sum_{k \in \Omega_i} c^k_{ib2} \times pt^k_i - y_{4ib} \leq N_b; \qquad \sum_{k \in \Omega_i} c^k_{i3} \times pt^k_i - y_{5i} \leq B.$$

We also added slack variables, z, to constraints C_0 and C_1 which now read $\sum_{b \leq M} N_b - y_0 + z_0 = K$ and $\sum_{b \leq M} b \times N_b - y_1 + z_1 = B$. The slack and surplus variables are constrained in *a box* : $y \leq \psi$, $z \leq \psi$ and they are penalised in the objective function by a coefficient ρ. The objective function then reads as:

$$w_1 K + w_2 B + \sum_a \rho y_a + \rho z_0 + \rho z_1.$$

This tries to avoid the dual solutions jumping from one extreme to another by restraining the dual variables in a box as long as no relevant dual information is known. ρ and ψ are updated during the process and must end with $\rho = \infty$ or $\psi = 0$ to ensure the sound termination of the algorithm. We simply fix the value of ψ to a small constant (10% of the upper bound given by the heuristic [15]) and we update ρ when the column generation algorithm stalls, using a predefined sequence of values: $[0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1, \infty]$.

4.2 Branch and Price

We went a step further and designed a Branch and Price algorithm by coupling the CP algorithm with the Column Generation approach. The column generation procedure

provides a valuable lower bound at the root node which can be often optimal in practice. To benefit from this bound during the search, we will now briefly describe a branch and price algorithm where column generation is called at each node of the CP search tree. Branching on N_b raises an issue from the column generation perspective: the subproblem becomes a shortest path with resource constraints, one resource per $b \leq M$ limited by the current upper bound on the N_b variables of the CP model. This also means that finding a feasible set of columns to initialise the master problem becomes difficult.

Interaction with CP. Solving the shortest path problem with multi-resource constraints is far too costly. Recall that the original CP model is relaxing the multi-resource path into a set of independent paths. The propagation obtained from this relaxation removes partitions in the partition graph. We can therefore take advantage of this information to prune the graph used by the subproblem of the column generation and solve a shortest path in a restricted graph. We therefore solve a relaxation of the real subproblem that we obtained from the CP propagation. The current bounds on the domains of the N_b variables are also enforced in the master problem RMP. Propagation allows us to strengthen both the master and the subproblems of the column generation.

Initialisation. The initialisation issue can be easily solved by adding slack variables for constraints C_0, C_1, C_3, C_4, and C_5 of the RMP and adding them to the objective function with a sufficiently large coefficient to ensure they will be set to 0 in an optimal solution. Then one simply needs to independently find a path in the current filtered partition graph of each line to obtain a feasible solution.

Column Management. From one node of the search tree to another, we simply keep the columns that are still feasible based on the domains of the N_b and P_{ij} variables and remove all the others. In addition to these removals, if the number of columns increases beyond a threshold (set to 10000 columns in practice), we delete half of the pool starting with the oldest columns to prevent the linear solver from stalling due to the accumulation of too many variables.

Reduced cost propagation. The CG provides a lower bound on the objective function but also the set of optimal reduced costs for the N_b variables. Propagation based on these reduced costs can be performed in the CP model following [17]. At a given node, once the RMP has been solved optimally, we denote by ub and lb the currents bounds on the objective variable. $ub + 1$ corresponds to the value of the best solution found so far and lb is the optimal value of the RMP at the corresponding node. We denote by rc_b, the reduced cost of variable N_b at the optimal solution of the RMP. rc_b represents the increase in the objective function for an increase of one unit of N_b. The upper bound on each N_b in the CP model can be adjusted to $lb(N_b) + \lfloor \frac{ub-lb}{rc_b} \rfloor$.

5 Experimental Results

We evaluated our methods using both randomly generated and clinical problem instances.[1] We used the *randomly generated instances* of [3, 9], which comprise 17 categories of 20 instances ranging in size from 12×12 to 40×40 with an M between 10 and

[1] All the benchmarks are available from `http://www.4c.ucc.ie/datasets/imrt`

Table 1. Comparing quality and time of LP/CG/CG-STAB on the Lex objective function

Inst	Gap (%)	LP	CG				Stabilised CG			
		Time	Time	NbPath	NbIter	Gain	Time	NbPath	NbIter	Gain
mean	0.64	109.08	40.28	849.10	147.54	10.83	**20.42**	579.53	62.47	14.97
median	0.30	14.73	1.44	660.18	123.30	10.58	**1.13**	434.35	54.48	14.32
min	0.00	0.81	**0.12**	262.75	65.95	6.45	**0.12**	198.95	38.85	6.72
max	5.00	1196.51	762.31	1958.10	297.60	21.96	**368.90**	1404.80	104.60	34.46

15, which we denote as m-n-M in our results tables. We added 9 additional categories with matrix sizes reaching 80×80 and a maximum intensity value, M, of 25, giving 520 instances in total. The suite of 25 *clinical instances* we used are those from [25]. The 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. A time limit of two hours and a memory limit of 3GB was used for each run.

Experiment 1: Evaluation of the LP Model. Firstly, we examine the quality and speed of the linear models (solved with CPLEX 10.0.0). We use a lexicographic objective function to perform this comparison, i.e. seek a minimum cardinality decomposition for the given minimum beam on-time. In the result tables LP refers to the continuous relaxation of the linear model representing every partition [10], CG to the model based on paths and CG-STAB to its stabilised version. Table 1 reports the average gap (in percentage terms) to the optimal value, the average times for the three algorithms as well as the number of iterations and paths for CG and CG-STAB. The improvement in time over LP is also given (column Gain). The mean, median, min and max across all categories are finally reported as well. The linear relaxation leads to excellent lower bounds but LP becomes quite slow as M grows and could not solve the instances with $M = 25$ due to memory errors. CG improves the resolution time significantly and offers better scalability in terms of memory. Its stabilised version clearly performs fewer iterations and proves to be approximately twice as fast on average.

Experiment 2: Evaluation of the Lagrangian Model. We consider the Lagrangian relaxation and its effect on the CP model.[2] We use a lexicographic objective function. Table 2 reports for the hardest categories the number of instances solved (column NS) within the time limit, along with the median, average and maximum time as well as the average number of nodes. The Lagrangian relaxation reduces the search space by an order-of-magnitude but turns out to be very slow when M grows.

Experiment 3: Evaluation of the Branch and Price Model. We evaluate the Branch and Price algorithm against the previous CP models with the lexicographic objective function and also using the more general objective function to perform a direct comparison with [25] on clinical instances. Following [25] we set $w_1 = 7$ and $w_2 = 1$. The upper bound of the algorithm is initialised using the heuristic given in [15] whose running time is always well below a second. Table 3 compares the shortest path CP model (CPSP) with two versions of the Branch and Price using the lex objective function. The first version referred to as *Branch and Price (light)* only solves the CG during the first branching phase on the N_b variables whereas the other version solves the CG

[2] All CP models were implemented in Choco 2.1 – `http://choco.emn.fr`

Table 2. Comparing the effect of the Lagrangian filtering on the Shortest Path Model CPSP .

Inst	CPSP					CP + Lagrangian relaxation				
		Time (seconds)			Nodes		Time (seconds)			Nodes
	NS	med	avg	max	avg	NS	med	avg	max	avg
12-12-20	20	**35.30**	**64.38**	**395.18**	816	20	405.22	796.33	4,532.17	**481**
12-12-25	**18**	1,460.39	2,242.19	6,705.01	8,966	0	-	-	-	-
15-15-15	20	**14.49**	**28.39**	**94.74**	938	20	80.76	120.57	399.37	**389**
18-18-15	20	**19.79**	**65.97**	**586.65**	1,366	20	80.36	180.69	807.63	**413**
20-20-15	20	**66.13**	**192.72**	**725.90**	4,436	20	353.09	559.73	2,328.94	**762**
20-20-20	**18**	1,379.72	1,876.12	6,186.78	7,628	6	1,190.96	1,605.77	5,041.88	572
30-30-15	**14**	115.83	698.37	2,638.54	691,318	12	308.04	839.37	3,942.70	937
40-40-10	20	**6.89**	495.90	3,848.14	130,309	20	19.21	**410.94**	**2,706.02**	**1,517**
40-40-15	**10**	512.88	1,555.49	5,687.44	488,133	8	1,003.10	1,645.86	5,029.01	1,189
50-50-10	15	82.04	888.52	5,275.96	4,022,156	**16**	85.36	784.76	5,216.68	10,534
60-60-10	11	1,100.92	1,967.51	6,079.23	8,020,209	**15**	426.73	1,378.31	5,084.95	34,552
70-70-10	7	2,374.97	2,503.82	3,980.76	11,102,664	**9**	2,534.44	2,894.94	5,970.91	131,494
80-80-10	2	464.57	464.57	737.78	14,274,026	**5**	1,877.76	2,193.92	4,147.88	118,408

at each node of the search tree, including when the branching is made on the partition variables. The Branch and Price significantly improves the CP model and is able to optimally solve the integrality of the benchmark whereas the CPSP solves 455 out of the 520 instances. The light version is often much faster but does not scale to the last two larger sets of instances (70×70 and 80×80 matrices). Both branch and price algorithms outperform CPSP on hard instances by orders of magnitude in search space reduction.

Table 3. Comparing the CP and Branch and Price

Inst	CPSP			Branch and Price (light)					Branch and Price				
		Time (seconds)			Time (seconds)			Nodes		Time (seconds)			Nodes
	NS	med	avg	NS	med	avg	max	avg	NS	med	avg	max	avg
12-12-20	20	35.30	64.38	20	**33.32**	**41.01**	**88.46**	90	20	67.99	75.68	176.40	**83**
12-12-25	18	1,460.39	2,242.19	20	**1,353.34**	**1,684.32**	**4,826.52**	157	20	2,767.06	2,748.22	5,112.66	**141**
15-15-15	20	14.49	28.39	20	**7.86**	**8.95**	**24.96**	142	20	20.44	21.86	38.18	**127**
18-18-15	20	19.79	65.97	20	**10.34**	**10.23**	**16.12**	202	20	34.03	33.64	45.31	**167**
20-20-15	20	66.13	192.72	20	**14.76**	**15.61**	**27.25**	283	20	47.82	52.74	115.09	**218**
20-20-20	18	1,379.72	1,876.12	20	**235.42**	**230.21**	**387.14**	325	20	823.16	855.54	1,433.07	**221**
30-30-15	14	115.83	698.37	20	**38.13**	**42.85**	**108.88**	2,420	20	322.95	335.43	683.56	**492**
40-40-10	20	6.89	495.90	20	**5.10**	**6.04**	**18.44**	5,932	20	97.56	95.28	128.47	**753**
40-40-15	10	512.88	1,555.49	20	**85.49**	**97.53**	**224.71**	23,755	20	1,101.48	1,172.91	2,354.53	**818**
50-50-10	15	82.04	888.52	20	**14.80**	**27.12**	**178.79**	48,216	20	280.11	265.71	393.71	**1,194**
60-60-10	11	1,100.92	1,967.51	20	**67.06**	**252.60**	**3,337.60**	638,157	20	471.39	492.44	705.08	**1,724**
70-70-10	7	2,374.97	2,503.82	17	686.33	1,443.46	7,118.74	5,778,692	**20**	1,153.71	1,147.24	2,243.58	**2,408**
80-80-10	2	464.57	464.57	8	812.35	1,983.79	6,671.28	11,546,885	**20**	1,854.04	2,069.52	3,830.13	**3,059**

Finally, we evaluate the CPSP and the light Branch and Price on 25 clinical instances with the general objective function. Table 4 reports the resolution time, the number of nodes explored (Nodes) and the value of the objective function (Obj). The times reported in [25] are quoted in the table and were obtained on a Pentium 4, 3 Ghz.[3]. The CP model alone already brings significant improvements over the algorithm of

[3] Two optimal values reported in [25] (for `c3b5` and `c4b5`) are incorrect. The corresponding solutions are pruned by their algorithms during search although it accepts them as valid solutions if enforced as hard constraints

Table 4. Comparing the shortest path CP model, the Branch and Price algorithm against [25].

Inst				Caner et al.	CPSP			Branch and Price light		
	m	n	M	Time	Time	Nodes	Obj	Time	Nodes	Obj
c1b1	15	14	20	1.10	8.85	1,144	111	5.26	144	111
c1b2	11	15	20	0.80	0.38	222	104	1.36	77	104
c1b3	15	15	20	11.40	5.90	534	108	3.06	70	108
c1b4	15	15	20	37.00	7.87	389	110	7.10	77	110
c1b5	11	15	20	4.30	0.23	46	104	1.11	37	104
c2b1	18	20	20	26.50	29.68	3,304	132	11.08	665	132
c2b2	17	19	20	20.10	75.30	3,822	132	9.31	255	132
c2b3	18	18	20	14.70	1.86	116	140	6.69	101	140
c2b4	18	18	20	87.30	559.16	42,177	149	64.23	373	149
c2b5	17	18	20	395.60	16.74	911	132	23.39	402	132
c3b1	22	17	20	310.00	21.00	888	132	25.16	322	132
c3b2	15	19	20	4,759.80	48.30	1,527	144	25.82	178	144
c3b3	20	17	20	10,373.90	570.25	12,353	140	617.23	1,228	140
c3b4	19	17	20	524.90	2.18	136	127	13.18	96	127
c3b5	15	19	20	3.30	1.05	0	125	3.24	0	125
c4b1	19	22	20	34.90	0.47	367	152	1.43	356	152
c4b2	13	24	20	20,901.00	42.87	1,183	181	50.83	391	181
c4b3	18	23	20	44.70	17.35	4,059	139	4.99	131	139
c4b4	17	23	20	164.30	13.57	1,069	142	13.85	285	142
c4b5	12	24	20	14,511.40	2,003.76	75,284	192	533.36	1,455	192
c5b1	15	16	20	0.50	0.10	83	96	0.33	60	96
c5b2	13	17	20	14.30	13.33	4,420	125	18.82	248	125
c5b3	14	16	20	3.10	0.56	106	104	2.43	52	104
c5b4	14	16	20	2.20	168.18	19,747	124	37.95	636	124
c5b5	12	17	20	51.90	1.77	547	130	2.27	49	130
Mean				2,091.96	144.43	6,977.36	131.00	**59.34**	307.52	131.00
Median				34.90	13.33	911.00	132.00	**9.31**	178.00	132.00
Min				0.50	**0.10**	0.00	96.00	0.33	0.00	96.00
Max				20,901.00	2,003.76	75,284.00	192.00	**617.23**	1,455.00	192.00

[25]. The Branch and Price algorithm shows even more robustness by decreasing the average, median and maximum resolution times.

6 Conclusion

We have provided new approaches to solving the Multileaf Collimator Sequencing Problem. Although the complexity of the resulting algorithms 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 as shown on both random and clinical instances. The hybrid methods proposed in this paper offer performance significantly beyond the current state-of-the-art and rely on a rich exchange of information between OR and CP approaches.

References

1. Nzhde Agazaryan and Timothy D. Solberg. Segmental and dynamic intensity-modulated radiotherapy delivery techniques for micro-multileaf collimator. *Medical Physics*, 30(7):1758–1767, 2003.
2. Ravindra K. Ahuja and Horst W. Hamacher. A network flow algorithm to minimize beam-on time for unconstrained multileaf collimator problems in cancer radiation therapy. *Netw.*, 45(1):36–41, 2005.

3. 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.
4. 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.
5. 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.
6. 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.
7. T. R. Bortfeld, D. L. Kahler, T. J. Waldron, and A. L. Boyer. X-ray field compensation with multileaf collimators. *International Journal of Radiation Oncology Biology Physics*, 28(3):723–730, 1994.
8. Stephen Boyd, Lin Xiao, and Almir Mutapic. Subgradient methods. In *Notes for EE392o, Standford University*, 2003.
9. 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.
10. Hadrien Cambazard, Eoin O'Mahony, and Barry O'Sullivan. A shortest path-based approach to the multileaf collimator sequencing problem. In *Procs of CPAIOR*, pages 41–55, 2009.
11. Michael J. Collins, David Kempe, Jared Saia, and Maxwell Young. Nonnegative integral subset representations of integer sets. *Inf. Process. Lett.*, 101(3):129–133, 2007.
12. Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column Generation*. Springer, 2005.
13. Olivier du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Math.*, 194(1-3):229–237, 1999.
14. Matthias Ehrgott, Çigdem Güler, Horst W. Hamacher, and Lizhen Shao. Mathematical optimization in intensity modulated radiation therapy. *4OR*, 6(3):199–262, 2008.
15. K. Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Applied Mathematics*, 152(1-3):35–51, 2005.
16. A. T. Ernst, V. H. Mak, and L. A. Mason. An exact method for the minimum cardinality problem in the planning of imrt. *INFORMS Journal of Computing (to appear)*, 2009.
17. Filippo Focacci, Andrea Lodi, and Michela Milano. Cost-based domain filtering. In Joxan Jaffar, editor, *CP*, volume 1713 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 1999.
18. 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.
19. 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.
20. Thomas Kalinowski. A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Applied Mathematics*, 152(1-3):52–88, 2005.
21. M. Langer, V. Thai, and L. Papiez. Improved leaf sequencing reduces segments or monitor units needed to deliver imrt using multileaf collimators. *Medical Physics*, 28(12):2450–2458, 2001.
22. Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.
23. Julien Menana and Sophie Demassey. Sequencing and counting with the multicost-regular constraint. In *Procs of CPAIOR*, pages 178–192, 2009.
24. Meinolf Sellmann. Theoretical foundations of cp-based lagrangian relaxation. In Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, pages 634–647. Springer, 2004.

25. Z. Caner Taskin, J. Cole Smith, H. Edwin Romeijn, and James F. Dempsey. Collimator leaf sequencing in imrt treatment planning. *submitted to Operations Research*, 119, 2009.
26. Giulia M. G. H. Wake, Natashia Boland, and Les S. Jennings. Mixed integer programming approaches to exact minimization of total treatment time in cancer radiotherapy using multi-leaf collimators. *Comput. Oper. Res.*, 36(3):795–810, 2009.