In this paper, we study the treatment planning optimization problem for the Volumetric-Modulated Arc Therapy (VMAT). We first present an Integer Programming (IP) formulation for this problem, and linearize all nonlinear terms in it. Then we present some valid inequalities to strengthen this formulation, and conduct polyhedral analysis on an important subproblem, showing that some of these inequalities are facet-defining under certain conditions. Next, we propose four methodologies for solving these problems: two heuristics based on Lagrangian relaxation, one heuristic based on a reformulation of the problem, and a metaheuristic based on the Guided Variable Neighborhood Scheme (GVNS). Finally, we present extensive computational results and discuss various aspects of our methods, including their strengths and challenges ahead. Numerical experiments show that our GVNS was able to tackle randomly generated problem instances of size comparable to those from real clinical data.

Key words: radiation treatment planning; integer programming; heuristics; lagrangian relaxation; metaheuristics.

1. Introduction

Radiation therapy or radiotherapy has become one of the most common treatment methods for cancer, besides chemotherapy and surgery, with almost two-thirds of all cancer patients expected to have radiotherapy at some stage in their treatment plan [6]. In this treatment, high-energy radiation is used to shrink tumors and kill cancer cells, where radiation damages the DNA of cancer cells [9]. The radiation can be delivered either by sources placed inside the patient’s body, called internal radiation therapy; or by a machine outside the body,
called *external radiation therapy*, the most common form of radiotherapy. Since our focus is external radiotherapy, we will refer to that simply as radiotherapy in the remainder of the paper.

In radiotherapy, patients sit or lie on a treatment couch while radiation is directed at the tumor, usually from a *gantry* that rotates around the patient by $360^\circ$ (see Figure 1). The modern forms of radiotherapy, such as the widely-used *intensity modulated radiation therapy* (IMRT) and the recent technology of *volumetric-modulated arc therapy* (VMAT), employ a *multi-leaf collimator* (MLC) mounted on the head of the gantry (see Figure 2). The MLC has many rows of thick metal leaves that can block the radiation from passing through; each row has a left- and a right-leaf, and the leaves move horizontally. When there is an “opening” created by the leaves, radiation passes through and is absorbed by the cells in the exposed area. These openings are referred to as *apertures* or *shapes*, and the modulation of the radiation intensity is achieved by varying the shapes and the beam-on time for each shape.

In radiotherapy, the *treatment planning optimization* problem is about finding the best dose delivery regime such that the amount of radiation delivered to the tumors is as high as possible while sparing the nearby critical organs and sensitive tissues. Each patient’s case
is different and requires a detailed image of the patient’s body obtained through various methods such as CT scans. The 3-D treatment field includes different tissues, organs and tumors, and in practice it is discretized into voxels i.e., cubes of usually 4mm by 4mm by 2.5–4mm, for efficient handling. The criteria to optimize vary from minimizing treatment time to maximizing radiation delivered to tumors, depending on the practice of the health provider.

This is a very challenging mathematical problem in practice, mainly due to following reasons: (i) Increasing radiation delivered to tumors while decreasing radiation delivered to healthy tissue are conflicting aims. (ii) The problem has a hard combinatorial structure, including complicated MLC machinery restrictions commonly known as the interleaf constraints, i.e., the left-leaf in a row cannot collide with the right-leaf in the adjacent rows. Although some recent machinery allows interdigitation, i.e., no interleaf constraints required, this makes in general the optimization problem easier to solve, and we believe addressing a harder mathematical problem is more useful. (iii) The computational time is limited to ensure minimum change of patient’s internal body structure. (iv) The practical problem sizes are very big.

1.1. IMRT vs. VMAT: An Overview and Literature Review

Due to increasing demand for radiotherapy as well as developments both in medical and computational technologies, there have been significant improvements in machinery over the last few decades. The currently well-established technology of intensity modulated radiation therapy (IMRT) has been improving over the years, recently resulting in the new technology of volumetric-modulated arc therapy (VMAT). Essentially, both IMRT and VMAT are very similar and use almost identical machinery. However, the underlying optimization problems are very different, and VMAT is significantly faster, delivering a treatment in about 1-2 minutes that would have taken 20 minutes by IMRT. In IMRT, although it is possible for the gantry to rotate by 360° and to deliver radiation from any angle, in practice, usually only a few beam angles will be chosen for the delivery of radiation. These angles are either predetermined by an experienced radiotherapist or calculated by solving some optimization problems (see, e.g., [15, 21]). Radiation beam is only turned on when the gantry reaches these beam-angles, remains stationary while delivering radiation, and is switched off when the planned dose for this beam angle is delivered and until the gantry moves to the next beam-angle. At each beam angle, radiation is delivered either in a dynamic manner (sliding
windows method, i.e., the radiation beam is turned on whilst the leaves in each row move in a continuous manner, possibly with different speed of movement, see, e.g., [19]), or in a static manner (the step-and-shoot delivery method, i.e., radiation beam is on while the MLC leaves are stationary, and off while the leaves move to form the next shape, see, e.g., [10]). The unit of radiation delivered while the beam is on is referred to as the monitor unit (MU).

In VMAT, on the other hand, the gantry moves continuously for one or more 360° rotations, and the radiation beam is turned on throughout the process while the gantry and the MLC leaves move [16]. This is the reason why this technology is much faster and more efficient than IMRT.

We clarify here important terminology used in the rest of the paper: We will call a solution to the treatment planning optimization problem feasible if it is one that satisfies all of the dose requirements as well as the machinery constraints. Meaning, a solution that has all the voxels receiving a dose that is within their lower and upper bounds prescribed by a radiation oncologist, and that the treatment plan can be carried out by the machinery. An infeasible solution is one that has some of the dose and/or machinery constraints violated.

Previously, the Operations Research/Medical Physics community has conducted a substantial amount of research on the treatment planning optimization of the IMRT, see, e.g., [5]. The field of minimizing treatment times for the step-and-shoot MLC radiotherapy practically began with the work of [22]. The most recent advances in the minimizing of total treatment time can be found in [3, 11, 18]. We also note that there is limited work using exact techniques (see, e.g., [4]), and to our knowledge, the majority of previous work consists of heuristics based on generating infeasible treatment plans that minimize infeasibility and then aim to reduce the infeasibility of such plans (see, e.g., [16]). Therefore, feasibility is not even expected when plans are generated in such frameworks.

Since the underlying mathematical problem for the VMAT is very different, methods developed for the IMRT cannot be directly applied to the VMAT. Aside from the interleaf constraints, there are other major machinery constraints that are unique to the VMAT. First, in order for the gantry to move continuously throughout delivery, the MLC leaf position changes between consecutive gantry positions are restricted. Second, the number of monitor unit per degree of rotation is restricted. Third, in some make of the machinery, the maximum change in number of monitor units is restricted per degree of rotation.

To our knowledge, the treatment planning optimization of the VMAT has rarely been studied in the field of Mathematical Programming. In the field of Medical Physics, some
recent studies can be found in, e.g., [16, 13, 17]. However, these studies are primarily focused on obtaining quick (and possibly infeasible) solutions. This approach seems to be observed in general in radiation treatment and causes at times issues with infeasible treatment plans, i.e., sensitive tissue exposed to too much radiation or tumor exposed to too little radiation. As noted by Dr. Robert D. Timmerman, vice chair of radiation oncology at the University of Texas Southwestern Medical Center, radiotherapy plans will be significantly improved in the future, “simply because current outcomes are unsatisfactory to patients” [6]. Therefore our focus in this paper will be on ensuring feasible solutions when possible.

1.2. Organization of the Paper

In this paper, we study the problem using mathematical programming approaches and propose heuristic methods. In Section 2, we will present an integer programming formulation of the problem, which initially has non-linear terms but can be linearized with additional variables and constraints, and we will discuss some valid inequalities that strengthen this formulation. In Section 3, we will discuss the polyhedral properties of some key subproblems, and also the strength of the inequalities presented in the previous section. We will present all proposed solution methodologies in detail in Section 4, where we will also briefly discuss possible Lagrangian relaxations of the problem. The empirical strength of these solution methods will be investigated extensively in Section 5. Finally, we will summarize our conclusions and address potential future research areas in Section 6.

2. A Mixed Integer Programming Formulation

In this section, we present a mixed integer programming formulation for the treatment planning optimization of the VMAT. Before we introduce the MIP model, we first present the notation and problem description.

2.1. Notation and Problem Description

When the gantry beams radiation in VMAT, it either delivers a “single arc”, i.e., one 360° rotation around the patient; or in case of multiple arcs, it rotates in the reverse direction after completing each 360° rotation to avoid the wires to be entangled. In our model, we partition the path of the gantry (single- or multiple-360° rotations) into K equal sections. In the field of medical physics, similar partitioning methods are commonly used (e.g., [16]
uses 10°-partitions), and each partition is referred to as a sample or a snapshot. In our numerical experiments, some problem instances have partitions as fine as 2°-partitions. For consistency, in the rest of the paper, we will use the term snapshot for the mid-point of each partition.

As explained earlier, the MLC has many rows, each with a left- and a right-leaf. Let \( m \) be the number of rows of leaves, and assume the width of the MLC is partitioned into \( n \) equal columns, where the leaves can be positioned. Let:

- \( I = \{1, \ldots, m\} \) be the index set of the MLC rows;
- \( J = \{1, \ldots, n\} \) be the index set of the MLC columns;
- \( I \times J \) be set of beamlets or a bixels, each cell \((i, j)\) being a beamlet/bixel;
- \( J' = \{0, n + 1\} \cup J \), with 0 and \( n + 1 \) being the home positions of the MLC left and right-leaves, respectively;
- \( V \) be the index set of all voxels;
- \( V_t \) be the set of voxels in the target volumes, i.e., tumor volumes;
- \( V_o \) be the set of voxels in the organs at risk;
- \( \mathcal{L} = \{(\ell, r) \mid \ell, r \in J', \ell < r\} \) be the feasible left- and right-leaf pairs;
- \( K = \{1, \ldots, \mathfrak{K}\} \) be the index set of snapshots;
- \( D_{ijk} \) be the dose received by voxel \( v \) from beamlet \((i, j)\) of snapshot \( k \) at unit intensity;
- \( L_v \) be the prescription dose for tumor voxel \( v \in V_t \);
- \( U_v \) be the maximum dose allowed for voxel \( v \in V \);
- \( y_{i(\ell, r)}^k \in \{0, 1\} \) be a decision variable with \( y_{i(\ell, r)}^k = 1 \) representing the bixels between, but not including, columns \( \ell \) and \( r \) in row \( i \) in Snapshot \( k \) are open;
- \( x_v \in \{0, 1\} \) be a decision variable with \( x_v = 1 \) if voxel \( v \in V_t \) receives a desired dose of \( \bar{d} \) or above, and \( x_v = 0 \) otherwise;
- \( d_v \) the dose that voxel \( v \) receives; and
- \( z^k \) be a continuous decision variable representing the monitor units for Snapshot \( k \).
2.2. The Formulation

Different makes of the machine involve slightly different machinery constraints, see [4] for some detailed machinery description. We assume that the movements happen between each snapshot in a linear fashion, and include the important constraints in our model, as discussed below. The purpose of any kind of radiotherapy is to deliver as high a dosage as possible to the target volumes so as to eliminate the tumor, while guaranteeing critical organs be spared from harmful doses. Therefore we propose the following formulation:

\[
\begin{align*}
\max & \quad \sum_{v \in V_t} x_v \\
\text{s.t.} & \quad \sum_{(\ell, r) \in L} y_{k, (\ell, r)}^i = 1 \quad \forall i \in I, \forall k \in K \\
& \quad \sum_{\ell=\tilde{\ell}+1}^{n+1} \sum_{r=\tilde{r}+1}^{\tilde{r}+1} y_{k, (\ell, r)}^i \leq 1 \\
& \quad \sum_{\ell=\tilde{\ell}}^{r-1} \sum_{r=\tilde{r}+1}^{n+1} y_{k, (\ell, r)}^i \leq 1 \\
& \quad \sum_{\ell=\tilde{\ell}+1}^{n+1} \sum_{r=\tilde{r}+1}^{\tilde{r}+1} y_{k, (\ell, r)}^i \leq 1 \\
& \quad z^k - z^{k+1} \leq \Delta, \forall k \in K \\
& \quad z^{k+1} - z^k \leq \Delta, \forall k \in K \\
& \quad d_v \geq L_v, v \in V_t \\
& \quad d_v \leq U_v, v \in V \\
& \quad d_v \geq \bar{d} x_v, v \in V_t \\
& \quad x \in \{0, 1\}^{|V_t|}; y \in \{0, 1\}^{|I| \times |L|}; 0 \leq z^k \leq \bar{M}^{|K|} 
\end{align*}
\]

We maximize the number of voxels in the target volume that receive a minimum radiation of \(\bar{d}\) by using the objective function (1). By maximizing this counter, we are able to provide
an equal weight for each voxel in the target area to receive a good dose, which will favor dose homogeneity as advised in [8], as opposed to an objective function such as \( \sum_{v \in V_t} d_v \) maximizing the sum of doses, where skewed results of dosages are often observed for voxels in the target area.

Constraint (2) ensures that only one shape is used for each beam angle. Constraints (3) and (4) are the inter-leaf constraints that ensure the right (left) leaf in Row \( i + 1 \) cannot overlap with the left (right) leaf in Row \( i \). Here we note that although some newer generation machinery allows interdigitation (i.e., no interleaf constraints needed), the interleaf constraints in general make the problem computationally harder and they cover a more general set of problems; hence we keep them in the formulation. The constraints (5) and (6) require that MLC do not move faster than \( \delta \) columns between successive snapshots. In a similar fashion, the constraints (7) and (8) limit the difference between monitor units applied on successive snapshots to be at most \( \Delta \). Since we assume the beam change from one snapshot to the next happens in a linear fashion, the total dose calculation takes the average of dose between successive angles, as shown in detail in (13). The constraints (9) and (10) ensure the dose lower and upper bounds be satisfied, and (11) determines whether the binary variable \( x_v \) can take a value of 1, i.e., the dose applied on voxel \( v \) is at least the desired minimum of \( \bar{d} \). Finally, (12) indicates bounds and integrality requirements of the variables.

The dose that a voxel receives, denoted by \( d_v \) for each \( v \in V \) and calculated as below, depends on both the MU delivered and the shape of the MLC at each snapshot:

\[
d_v = \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} \frac{1}{2} \left( z^k \times D_{ijv}^k \times \sum_{(\ell, r) \in \mathcal{L}} y_{i, (\ell, r)}^k \right) + \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} \frac{1}{2} \left( z^{k+1} \times D_{ijv}^{k+1} \times \sum_{(\ell, r) \in \mathcal{L}} y_{i, (\ell, r)}^{k+1} \right) \tag{13}
\]

In this form, the dose expression is nonlinear and not convex; therefore it would complicate the problem significantly. Although some MINLP techniques could be used, it would be more beneficial if an efficient linearization can be established. We first note that all the nonlinear terms above are bilinear. Moreover, these terms are special that exactly one of the components of these terms is a binary variable and the other one a continuous variable.
Therefore, we can define a new variable $\bar{z}^k_{ij}$ to indicate the MU amount for snapshot $k$ and beamlet $(i,j)$ of the MLC, and hence redefine $d_v$ for each $v \in V$ linearly as

$$d_v = \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} \frac{1}{2} (\bar{z}^k_{ij} \times D^k_{ijv} + \bar{z}^{k+1}_{ij} \times D^{k+1}_{ijv}) \quad (14)$$

To finalize the linearization, we add to the formulation the following four constraints for each $(k, i, j)$ combination such that $k \in K, i \in I, j \in J$:

$$\bar{z}^k_{ij} \leq M \sum_{(\ell,r) \in \mathcal{L}, \ell < j < r} y^k_{i(\ell,r)} \quad (15)$$

$$\bar{z}^k_{ij} \leq z^k \quad (16)$$

$$\bar{z}^k_{ij} \geq M \left(-1 + \sum_{(\ell,r) \in \mathcal{L}, \ell < j < r} y^k_{i(\ell,r)}\right) + z^k \quad (17)$$

$$\bar{z}^k_{ij} \geq 0 \quad (18)$$

Since $\sum_{(\ell,r) \in \mathcal{L}, \ell < j < r} y^k_{i(\ell,r)} \in \{0, 1\}$, the variable $\bar{z}^k_{ij}$ will be forced to zero when $\sum_{(\ell,r) \in \mathcal{L}, \ell < j < r} y^k_{i(\ell,r)} = 0$, and to $z^k$ when $\sum_{(\ell,r) \in \mathcal{L}, \ell < j < r} y^k_{i(\ell,r)} = 1$. Note that this linearization corresponds to McCormick’s envelope [12] and is known to give the convex envelope for general bilinear terms. Substituting the linear dose definition (14) into the constraints (9)-(11) and adding the above constraints, the problem is defined as: $z_{VMAT} = \max\{(1) | (x, y, z, \bar{z}) \in X_{VMAT}\}$, where $X_{VMAT} = \{(2) - (12), (15) - (18)\}$.

### 2.3. Improving the Formulation: Valid Inequalities

In this subsection, we improve the above formulation by considering stronger inequalities that would replace some of the original constraints, and present numerical results to show the strengths of these inequalities. The number of such inequalities is equal to the number of original constraints and hence provide us a benefit without any obvious computational cost. Moreover, as discussed in the upcoming sections, these inequalities are facet-defining for some subproblems and provide computational improvements.

**Proposition 1** The following inequality is valid and dominates (11).

$$d_v - L_v \geq (\bar{d} - L_v)x_v \quad (19)$$
Figure 3: Geometric view of (19) (dashed line) vs. (11) (dotted line starting from the origin)

Note that if \( L_v = 0 \) holds for any \( v \in V \), (19) becomes equivalent to (11). The effect of these strengthened inequalities can be seen in Figure 3 with respect to original constraints (11), when \( L_v > 0 \). The proof of this proposition is straightforward and is hence omitted here. The following notation is used frequently in the rest of this section.

Definition 1 The cumulative demand from column \( \ell \) to column \( r \): \( D_{k,i,\ell,r,v} = \sum_{j=\ell}^{r} D_{k,ij,v} \).

Proposition 2 For all \( k \in K, i \in I, j \in J \), the inequality

\[
 z^k - \bar{z}_{ij}^k \leq \sum_{\substack{(\ell,r) \in \mathcal{L} \\ \ell \geq j \lor r \leq j}} \min_{v \in V} \left\{ \frac{\min_{v \in V} U_v}{\frac{D_{k,\ell+1,r-1,v}}{D_{k,\ell,\ell+1,v}}} \right\} y_{k,ij}^k
\]

(20)

is valid for the VMAT problem and dominates (17).

Proof. First of all, note that due to constraint (2), we have:

\[
 \sum_{\substack{(\ell,r) \in \mathcal{L} \\ \ell \geq j \lor r \leq j}} y_{k,ij}^k = 1 - \sum_{\substack{(\ell,r) \in \mathcal{L} \\ \ell < j < r}} y_{k,ij}^k, \quad \forall k \in K, i \in I, j \in J.
\]

If, for any given \( (\ell,r) \in \mathcal{L} \) such that \( \ell \geq j \) or \( r \leq j \), \( y_{k,ij}^k = 1 \), then \( \bar{z}_{ij}^k = 0 \), and \( z^k \leq \frac{U_v}{D_{k,\ell+1,r-1,v}} \) has to be satisfied due to (10). On the other hand, if for any given \( (\ell,r) \in \mathcal{L} \) such that \( \ell < j < r \), \( y_{k,ij}^k = 1 \), then \( \bar{z}_{ij}^k = z^k \). Therefore, (20) is valid. The dominance of (20) over (17) simply follows that \( \min \left\{ \frac{M}{\frac{U_v}{D_{k,\ell+1,r-1,v}}} \right\} \leq M. \quad \square \)
Corollary 1 For all $k \in K$, $i \in I$, $j \in J$, the inequality

$$\bar{z}_{ij}^k \leq \sum_{(\ell, r) \in \mathcal{L} \text{ s.t. } \ell < j < r} \min_{v \in V} \min \left\{ \bar{M}, \min_{v \in V} \frac{U_v}{D^k_{i, \ell+1, r-1, v}} \right\} y_{ik(\ell, r)}^k \tag{21}$$

is valid for the VMAT problem and dominates (15).

The corollary can be proven similar to Proposition 2 and hence we omit the proof.

Proposition 3 The following are valid inequalities for the VMAT problem:

(a) For $i \in I$, $j, j' \in J$, $k \in K$:

$$\bar{z}_{ij}^k - \bar{z}_{ij'}^k \leq \sum_{(\ell, r) \in \mathcal{L} \text{ s.t. } \ell < j < r} \min_{v \in V} \min \left\{ \bar{M}, \min_{v \in V} \frac{U_v}{D^k_{i, \ell+1, r-1, v}} \right\} y_{ik(\ell, r)}^k \tag{22}$$

(b) For $i, i' \in I$, $j, j' \in J$, $k \in K$:

$$\bar{z}_{ij}^k - \bar{z}_{ij'}^k \geq \left( \sum_{(\ell, r) \in \mathcal{L} \text{ s.t. } \ell < j < r} \min_{v \in V} \min \left\{ \bar{M}, \min_{v \in V} \frac{U_v}{D^k_{i, \ell+1, r-1, v}} \right\} y_{ik(\ell, r)}^k \right) - \bar{M} \tag{23}$$

The proof of the second inequality is similar to the proof presented on p.82 of [20] and hence omitted here. The proof of the first inequality is straight forward due to the fact that $\bar{z}_{ij}^k \leq \min_{v \in V} \min \left\{ \bar{M}, \min_{v \in V} \frac{U_v}{D^k_{i, \ell+1, r-1, v}} \right\}$ and hence omitted here. Note that the first set of inequalities are facet-defining in some of the subproblems we will discuss in the next section.

We conclude this section with a remark: we run preliminary computational tests to see if and how the inequalities discussed in this section would make a difference. We tested the strengths of the inequalities (19)- (23) individually added to the LP relaxation of the original formulation, as well as the effect of adding all these inequalities together. These results indicated that constraints (19) provide by far the most significant improvement to the LP relaxation bound. The improvements made by (21) and (22) were observed to be mild, and ones made by constraint (20) and (23) very insignificant. Therefore, we simply replaced (11) with (19) and will use this formulation in the remainder of the paper for all the methods that are based on MIP.
3. Assumptions and Polyhedral Analysis

In this section, we study the polyhedral structure of the MIP. First, we list our assumptions for the model to be realistic as well as non-trivial:

- \( \sum_{k \in K, i \in I, j \in J} \tilde{M}D_{ijv}^k \geq U_v, \forall v \in V. \) Otherwise, we can set \( U_v = \sum_{k \in K, i \in I, j \in J} \tilde{M}D_{ijv}^k. \)
  
  Note also that from the practical point of view (with realistic number of snapshots and size of MLC discretisation matrix) \( \sum_{k \in K, i \in I, j \in J} \tilde{M}D_{ijv}^k >> U_v. \)

- \( \tilde{d} \leq U_v, \forall v \in V_t. \) Otherwise, \( x_v = 0 \) and the variable can be eliminated from the problem. To keep problem instances more interesting, we further assume \( \tilde{d} < U_v. \)

- \( L_v < \tilde{d}, \forall v \in V_t. \) Otherwise, with the current objective function, \( x_v = 1 \) will hold in any feasible solution.

- All parameters strictly positive. Otherwise variables can be eliminated.

Next, we will look at subproblems that can be analytically studied but can also be extended to other crucial subproblems and hence provide important insight. We use the notation \( P_{i \times j \times k \times v} \) to indicate the convex hull of a problem with \( i \) rows, \( j \) columns, \( k \) angles, and \( v \) voxels.

3.1. A Study of the \( P_{1 \times n \times |K| \times 1} \) Polytope

From an analytical point of view, a model with a single voxel captures the basic VMAT setting, where multiple angles with decisions for MLC and dose amounts to be made, and this is an extension over IMRT. Performing polyhedral analysis for a more realistic problem, e.g., multiple voxels on a single row, the most straightforward assumption would be to allow an unified \( D_{ijv}^k \) value for all voxels. This assumption, however, does not necessarily make the problem mathematically more challenging, as the analysis will be no more complex than that of a single voxel. Hence, we study here the \( P_{1 \times n \times |K| \times 1} \) polytope. Using our notation, \( P_{1 \times n \times |K| \times 1} \) represents the convex hull of the VMAT problem with a single row, \( n \) columns, \( |K| \) snapshots, and a single voxel. The notation \( D_{k, r} = \sum_{j = r}^{r} D_j^k \) for \( k \in K \) will be used throughout this section. Also, note that \( |\mathcal{L}| = \frac{(n+1)(n+2)}{2}. \)

**Proposition 4** \( \text{dim}(P_{1 \times n \times |K| \times 1}) = |K||\mathcal{L}| + n|K| + 1, \) when \( \tilde{M}D_{1,n}^k \geq L \) for each angle \( k \in K, \) and \( |K| \geq 4. \)
Proposition 5

The following are the trivial facets of $P_{1 \times n \times |K| \times 1}$, under the condition that $|K| \geq 4$ and other necessary conditions as indicated next to the constraints:

Proof. First, note that there are $|K||\mathcal{L}| + n|K| + |K| + 1$ variables (the $y/\bar{z}/z/x$ variables, respectively) and $|K|$ equations, hence $\dim(P_{1 \times n \times |K| \times 1}) \leq |K||\mathcal{L}| + n|K| + 1$. In order to show $\dim(P_{1 \times n \times |K| \times 1}) \geq |K||\mathcal{L}| + n|K| + 2$ affinely independent points, where $\epsilon > 0$ is a sufficiently small number and $\exists \bar{k}, \hat{k} \in K$ with $\bar{k} \neq \hat{k}$.

We choose an arbitrary $\tau \in \{0, \ldots, n\}$. We have the following three cases.

C1 We have altogether $|K| + 2$ points with: $y^k_{(r,\tau+1)} = 1$, for all $k \in K \setminus \{\hat{k}\}$; and $y^k_{(0,n+1)} = 1;

C1a 1 solution with: $z^k = z^k_{j'} = \frac{L}{D^k_{(1,n)}}$, for all $j' = 1, \ldots, n$; and 0 otherwise;

C1b 1 solution with: $z^k = z^k_{j'} = \frac{U}{D^k_{(1,n)}}$, for all $j' = 1, \ldots, n$; $x = 1$, and 0 otherwise;

C1c 1 solution with: $z^k = z^k_{j'} = \frac{L+x}{D^k_{(1,n)}}$, for all $j' = 1, \ldots, n$ and 0 otherwise;

C1d $|K| - 1$ solutions, one for each $k \in K \setminus \{\hat{k}\}$, with: $z^k = z^k_{j'} = \frac{L}{D^k_{(1,n)}}$, for all $j' = 1, \ldots, n$; $z^k = \kappa$, for some $0 < \kappa \leq M$, and 0 otherwise.

C2 There are $K(|\mathcal{L}| - 1)$ points, one for each $k \in K$ and for each $(\ell, r) \in \mathcal{L}_k$

$\mathcal{L}_k = \left\{ \begin{array}{ll} \mathcal{L} \setminus \{(0,n+1)\} & \text{if } k = \hat{k}; \text{ and} \\
\mathcal{L} \setminus \{(\tau,\tau+1)\} & \text{otherwise}, \end{array} \right.$

$y^k_{(\ell,r)} = 1$ holds for $(\ell, r) \in \mathcal{L}_k$; $y^k_{(r,\tau+1)} = 1$, for all $k' \in K \setminus \{k, \beta\}$, for $\beta = \bar{k}$ if $k = \hat{k}$, otherwise $\beta = \hat{k}$; $y^\beta_{(0,n+1)} = 1$, $z^\beta_{j'} = \frac{L}{D^\beta_{(1,n)}}$, for all $j' = 1, \ldots, n$; $z^\beta = \frac{L}{D^\beta_{(1,n)}}$, where $\beta = \bar{k}$ if $k = \hat{k}$, otherwise $\beta = \hat{k}$; and all other variables equal zero.

C3 We have $n|K|$ points, one for each $k \in K$ and each $j = 1, \ldots, n$, given as:

$y^k_{(j-1,j+1)} = 1$, $z^k = z^k_{j} = \frac{L}{nD^k_{j}}$; $y^\beta_{(0,n+1)} = 1$, for $\beta = \bar{k}$ if $k = \hat{k}$, and $\beta = \hat{k}$ otherwise;

$y^k_{(\tau,\tau+1)} = 1$, for all $k' \in K \setminus \{k, \beta\}$ for $\beta = \bar{k}$ if $k = \hat{k}$, and $\beta = \hat{k}$ otherwise;

$z^\beta_{j'} = \frac{(n-1)L}{nD^\beta_{j'}(1,n)}$, for all $j' = 1, \ldots, n$; $z^\beta = \frac{(n-1)L}{nD^\beta_{(1,n)}}$, for $\beta = \bar{k}$ if $k = \hat{k}$, and $\beta = \hat{k}$ otherwise, and all other variables equal zero.

For $k = \bar{k}$, as we have $z^k_j > 0$, the only way to obtain this is through some linear combination of other vectors with $z^k_j > 0$. However, the only vectors with $z^k_j > 0$ have $y^k_{(0,n+1)} = 0$, which will not give us $y^k_{(0,n+1)} = 1$ as needed. Hence these vectors are affinely independent to all of the previously introduced vectors. A similar justification can also be made for the case of $k = \hat{k}$. □

Proposition 5

The following are the trivial facets of $P_{1 \times n \times |K| \times 1}$, under the condition that $|K| \geq 4$ and other necessary conditions as indicated next to the constraints:
(a) $x \geq 0$;

(b) $x \leq 1$ when $\bar{M}D^k_{(1,n)} > \bar{d}$, $\forall k \in K$ and $\bar{d} < U$;

(c) $y^k_{(\ell,r)} \geq 0$, for all $k \in K$, and all $(\ell,r) \in \mathcal{L}$;

(d) $\bar{z}^k_j \geq 0$, for all $k \in K$ and all $j \in \{1,\ldots,n\}$;

(e) $\bar{z}^k_j \leq z^k$, for all $k \in K$ and all $j \in \{1,\ldots,n\}$.

Proof. Here we prove the trivial facets of $P_{1 \times n \times |K| \times 1}$.

(a) This is straightforward, as one can simply remove the last point from the previous proof and obtain sufficient number of affinely independent points with $x_v = 0$.

(b) This is also straightforward, as we can simply re-use all cases used in the proof of Proposition 4 (except for Case (1b)) with the following minor modifications to satisfy $x_v = 1$:

(i) Replace $L$ with $U$ in all instances;

(ii) In Case (1c), $L + \epsilon$ is replaced with $U - \epsilon$, for some $\epsilon$ such that $\epsilon < U - \bar{d}$; and

(iii) $x_v = 0$ is replaced with $x_v = 1$ in all cases.

(c) Here we have two cases: (i) $y^k_{(\ell,r)}$ for all $k \in K$ and $r \neq \ell + 2$; and (ii) $y^k_{(\ell,r)}$ for all $k \in K$ and $r = \ell + 2$. Let $\hat{k}, \bar{k} \in K$ such that $\hat{k} \neq \bar{k}$.

(i) We can use all the vectors constructed in the proof of Proposition 4, except for the vector with $y^k_{(\ell,r)} = 1$, and for the cases where $r = \ell + 1$, $(\tau, \tau + 1)$ will be arbitrarily chosen from $\{(h, h + 1) \mid h = 0, \ldots, n\} \setminus \{(\ell, r)\}$.

(ii) From the vectors constructed in the proof of Proposition 4, for all $y^k_{(j-1,j+1)}$ where $k \in K$ and $j \in \{0,\ldots,n\}$, there are exactly two vectors with $y^k_{(j-1,j+1)} = 1$. We remove both of these vectors, use all of the rest of the vectors, and then add the following vector.

\[ y^k_{(j-1,j+2)} = 1, \text{ when } j \leq n - 1, \quad z^k = \bar{z}^k_j = \bar{z}^k_{j+1} = \frac{L}{nD^k_{(j,j+1)}}, \]

(or, $y^k_{(j-2,j+1)} = 1$, when $j = n$, $z^k = \bar{z}^k_{j-1} = \bar{z}^k_j = \frac{L}{nD^k_{(j-1,j)}}$)
\[ y_{(0,n+1)}^\beta = 1, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ z_{j'}^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for all } j' = 1, \ldots, n; \]
\[ z^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ y_{(\tau,\tau+1)}^{k'} = 1, \text{ for all } k' \in K \setminus \{k, \beta\} \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise, and all other variables equal zero. (Note this implies that an additional condition is given by } |K| = 4.\]

(d) This is very similar to the proof of Case c(i). We reuse all the affinely independent vectors used in the proof of Proposition 4, except the vector with \( z_j^k = 0 \), with \( \hat{k} \neq k \) and \( \bar{k} \neq k \).

(e) In all the affinely independent vectors used in the proof of Proposition 4, there are precisely \( n \) vectors where \( z_j^k - z^k \neq 0 \). One of such case is in Case 1b in the proof of Proposition 4, and we can simply remove this point, as the dimension of the facet is one less than that of the polytope. The rest of the \( n - 1 \) vectors where \( z_j^k - z^k \neq 0 \) appear in Case 3 of the proof of Proposition 4, when \( \alpha \in \{1, \ldots, n\} \setminus \{j\} \). These vectors are as follows:
\[ y_{(\alpha-1,\alpha+1)}^k = 1, z^k = \bar{z}_\alpha^k = \frac{L}{nD_{(1,n)}^k}, \text{ (whilst } \bar{z}_j^k = 0, \text{ hence } z^k \neq \bar{z}_j^k); \]
\[ y_{(0,n+1)}^\beta = 1, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ z_{j'}^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for all } j' = 1, \ldots, n; \]
\[ z^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ y_{(\tau,\tau+1)}^{k'} = 1, \text{ for all } k' \in K \setminus \{k, \beta\} \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise, and all other variables equal zero.} \]

We remove all these vectors, and replace them with the following \((n-1)\) vectors, where we set \( y_k^{(j-1,j+\gamma)} \) variables for \( \gamma = 2, \ldots, n+1-j \), (there are \( n-j \) of them for each \( k \)), and \( y_k^{(j+\sigma,j+1)} \) variables for \( \sigma = 0, \ldots, j-2 \) (there are \( j-1 \) of them for each \( k \)) to a value of 1 instead. We can define these vectors formally as below:
\[ y_{(j-1,j+\gamma)}^k = 1, z^k = \bar{z}_j^k = \ldots = \bar{z}_{j+\gamma-1}^k = \frac{L}{nD_{(j+\gamma-1)}^k}; \]
\[ (\text{or, } y_{(j-\sigma,j+1)}^k = 1, z^k = \bar{z}_{j-\sigma+1}^k = \ldots = \bar{z}_j^k = \frac{L}{nD_{(j-\sigma+1,j)}^k}); \]
\[ y_{(0,n+1)}^\beta = 1, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ z_{j'}^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for all } j' = 1, \ldots, n; \]
\[ z^\beta = \frac{(n-1)L}{nD_{(1,n)}^\beta}, \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise}; \]
\[ y_{(\tau,\tau+1)}^{k'} = 1, \text{ for all } k' \in K \setminus \{k, \beta\} \text{ for } \beta = \bar{k} \text{ if } k = \hat{k}, \text{ and } \beta = \hat{k} \text{ otherwise, and all} \]
other variables equal zero.

Next, we present key results regarding the inequalities presented in Section 2.3.

**Proposition 6** The following are non-trivial facets of \( P_{1 \times n \times |K| \times 1} \):

(a) \( d_v - L_v \geq (\bar{d} - L_v) x_v \); and

(b) \( z^k - \bar{z}_j^k \leq \sum_{(\ell, r) \in \mathcal{L}} \min\{ \bar{M}, \frac{U}{D^k_{\ell+1, r-1}} \} y^k_{(\ell, r)} \).

**Proof.** The non-trivial facets of \( P_{1 \times n \times |K| \times 1} \).

(a) As we need to satisfy that \( d_v = L_v \) when \( x_v = 0 \), and \( d_v = \bar{d} \) when \( x_v = 1 \), we can reuse the vectors presented in the proof of Proposition 4 with the following modifications: We remove the vector presented in Case 1(c), and in Case 1(b), we replace \( U \) by \( \bar{d} \).

(b) We use again the vectors presented in the proof of Proposition 4 as a starting point. First, note that we need to construct vectors that satisfy

\[
\begin{align*}
\bar{z}_j^k &= \sum_{(\ell, r) \in \mathcal{L}} \min\{ \bar{M}, \frac{U}{D^k_{\ell+1, r-1}} \} y^k_{(\ell, r)}.
\end{align*}
\]

Most of the vectors presented in Proposition 4 satisfies the equality, unless noted in the list of modifications listed below. This condition is satisfied under three cases, which we call as Case (\( \alpha \)), Case (\( \beta \)), and Case (\( \gamma \)).

Case (\( \alpha \)): We have \( y^k_{(\ell, r)} = 1 \), for \( \ell < j < r \), in which case the right hand side equals zero. This means that either have \( z^k = \bar{z}_j^k = 0 \) (these cases appear in Case 2 of the proof of Proposition 4), or \( z^k = \bar{z}_j^k = \lambda \), for \( \lambda \neq 0 \), (the only such case is in Case 3 with \( y^k_{j-1, j+1} = 1 \), and we have \( z^k = \bar{z}_j^k = \frac{L}{nD_j} \)).

Case (\( \beta \)): We have \( y^k_{(\ell, r)} = 1 \), for \( j \leq \ell \) or \( j \geq r \), and \( r = \ell + 1 \). In this case, \( \bar{z}_j^k = 0 \), and the right hand side also equals zero. In all but one case presented in the proof of Proposition 4 that concern such \( y^k_{(\ell, r)} \), we have \( z^k = \bar{z}_j^k = 0 \). The only vector with such \( y^k_{(\ell, r)} \) and with \( z^k \neq \bar{z}_j^k \) appears in Case (1d) of the proof, where \( (\ell, r) = (\tau, \tau + 1) \), \( z^k \neq 0 \), but \( \bar{z}_j^k = 0 \). We simply remove this vector.

Case (\( \gamma \)): We have \( y^k_{(\ell, r)} = 1 \), for \( j \leq \ell \) or \( j \geq r \), and \( r > \ell + 1 \). Again, \( \bar{z}_j^k = 0 \). In the proof of Proposition 4, these cases appeared twice, once in Case 2, and once in Case 3 in the proof of Proposition 4. The vectors all have \( \bar{z}_j^k = 0 \). We simply change
the value of \( z^k \) from their original value assigned in the respective cases in the proof of Proposition 4 to \( z^k = \min\{\bar{M}, \frac{U}{D_{\ell+1,r-1}}\} \).

### 3.2. The Special Case of \( P_{1 \times n \times 1 \times 1} \)

We now present our results on a number of non-trivial facet-defining constraints for the special case of single row, a single angle and a single voxel. This very special case is unlikely to be an insightful subproblem for VMAT, but it is more insightful for IMRT, and is therefore included in the paper. For ensuring feasibility, we simply exclude closed leaf positions from the following discussion, i.e., \( \mathcal{L} := \mathcal{L}\backslash\{(j, j + 1) | j = 0, \ldots, n\} \). Let \( D_{\ell,r} = \sum_{j=\ell}^{r} D_j \). Note that the results presented in the previous section are still valid and hence not repeated here.

**Proposition 7** The inequality

\[
z - \bar{z}_j \geq \sum_{(\ell,r)\in \mathcal{L}} \frac{L}{D_{\ell+1,r-1}} y_{\ell,r}
\]

is valid for \( P_{1 \times n \times 1 \times 1} \) and dominates (16). Moreover, it is facet-defining for \( P_{1 \times n \times 1 \times 1} \) under the general condition of \( \bar{M} D_{j,\ell''} > L, \forall j'' \in J \).

**Proof.** First of all, note that due to constraint (2), we have:

\[
\sum_{(\ell,r)\in \mathcal{L}} y_{\ell,r} = 1 - \sum_{(\ell,r)\in \mathcal{L}} y_{\ell,r}
\]

If for any given \( (\ell, r) \in \mathcal{L} \) such that \( \ell \geq j \) or \( r \leq j \), \( y_{\ell,r} = 1 \), then \( \bar{z}_j = 0 \), and \( z \geq \frac{L}{D_{\ell+1,r-1}} \) has to be satisfied due to (9). On the other hand, if for any given \( (\ell, r) \in \mathcal{L} \) such that \( \ell < j < r \), \( y_{\ell,r} = 1 \), then (24) reduces to \( \bar{z}_j \leq z \). Therefore, (24) is valid. The dominance of (24) over (16) follows simply that \( L \geq 0 \).

To prove its facet-defining property, first let \( S_j = \{(\ell, r) \in \mathcal{L} | \ell \leq j \leq r\} \). Also let \( d' = \min\{\bar{d}, \bar{M} D_j\} \). Consider the following \( \frac{n(n+1)}{2} + n + 1 \) affinely independent points:

- \( \frac{n(n+1)}{2} \) points, where \( y_{\ell,r} = 1 \) holds for one \( (\ell, r) \), \( \bar{z}_j = L/D_{\ell+1,r-1} \) where \( \ell < \hat{j} < r \), \( z = L/D_{\ell+1,r-1} \), and all other variables zero.

- \( n \) points, where \( y_{\ell,r} = 1 \) holds for one \( (\ell, r) \in S_j \), all \( \bar{z}_j = d'/D_{\ell+1,r-1} \) where \( \ell < \hat{j} < r \), \( z = d'/D_{\ell+1,r-1} \), and all other variables zero.
• 1 point, where \( y_{0n+1} = 1, \bar{z}_j = \bar{d}/D_{1,n}, z = \bar{d}/D_{1,n}, x = 1, \) and all other variables zero. □

Proposition 8 2-column inequalities for \( P_{1 \times n \times 1 \times 1} \):

1. The 2-column inequality with \( j < \bar{j} \),

\[
\bar{z}_j + \bar{z}_j - z \leq \sum_{(\ell, r) \in \mathcal{L}} \min\{\bar{M}, \frac{U}{D_{\ell+1,r-1}}\} y_{\ell, r} - \sum_{(\ell, r) \in \mathcal{L}} \overline{L} y_{\ell, r} \tag{25}
\]

is valid for \( P_{1 \times n \times 1 \times 1} \), and is facet-defining under the general condition \( \overline{MD}_{j''} > L, \forall j'' \in J \).

2. The 2-column inequality with \( j < \bar{j} \),

\[
\bar{z}_j + \bar{z}_j - z \geq \sum_{(\ell, r) \in \mathcal{L}} \frac{L}{D_{\ell+1,r-1}} y_{\ell, r} - \sum_{(\ell, r) \in \mathcal{L}} \min\{\bar{M}, \frac{U}{D_{\ell+1,r-1}}\} y_{\ell, r} \tag{26}
\]

is valid for \( P_{1 \times n \times 1 \times 1} \), and is facet-defining under the conditions that \( \overline{MD}_{j''} > L, \forall j'' \in J \) and \( \exists (\ell, r) \in \mathcal{L}\backslash\{(0, n+1)\} \) such that \( \ell \geq j \) or \( \bar{j} \geq r \) and \( \overline{MD}_{\ell+1,r-1} \geq U \).

Proof. First, note that due to constraint (2), we have one of the following possibilities:

(a) \( y_{\ell, r} = 1 \) for a given \( (\ell, r) \in \mathcal{L} \) such that \( \ell < j \) and \( \bar{j} < r \).

(b) \( y_{\ell, r} = 1 \) for a given \( (\ell, r) \in \mathcal{L} \) such that either \( \ell \geq \bar{j} \) or \( r \leq j \) or \( \ell \geq j \) and \( r \leq \bar{j} \).

(c) \( y_{\ell, r} = 1 \) for a given \( (\ell, r) \in \mathcal{L} \) such that either \( \ell < j \) and \( j < r \leq \bar{j} \) or \( \bar{j} > \ell \geq j \) and \( r > \bar{j} \).

If (a) is the case, then we know that \( \bar{z}_j = \bar{z}_j = z \) and hence the left-hand side of both inequalities reduce to \( z \), whereas the right-hand side simply reduces to \( \min\{\bar{M}, \frac{U}{D_{\ell+1,r-1}}\} \) in (25) and to \( \frac{L}{D_{\ell+1,r-1}} \) in (26). The first condition holds due to (10) and \( z \leq \bar{M} \), and the second condition holds due to (9).

If (b) is the case, then we know that \( \bar{z}_j = 0 = \bar{z}_j \) and hence the left-hand side of both inequalities reduce to \( -z \), whereas the right-hand side simply reduces to \( \frac{L}{D_{\ell+1,r-1}} \) in (25) and to \( \min\{\bar{M}, \frac{U}{D_{\ell+1,r-1}}\} \) in (26). The first condition holds due to (9), and the second condition holds due to (10) and \( z \leq \bar{M} \).
If (c) is the case, then right-hand side of both constraints becomes simply 0, and either \( \bar{z}_j = 0 \) or \( \hat{z}_j = 0 \) holds. This concludes the validity proof.

To prove facet-defining property of (25), first let \( S_j = \{(\ell, r) \in L \mid (\ell < j \land j < r \leq \hat{j}) \lor (j \geq j \land r > \hat{j})\} \), i.e., the case (c). Consider the following \( \frac{n(n+1)}{2} + n + 1 \) affinely independent points:

- \( n \) points, where \( y_{\ell r} = 1 \) holds for one \((\ell, r)\), and
  - if \( \ell < j < \hat{j} < r \), then \( \bar{z}_j'' = \min\{\bar{M}, U/D_{\ell+1,r-1}\} \) where \( \ell < j'' < r \), \( z = \min\{\bar{M}, U/D_{\ell+1,r-1}\} \), and all other variables zero.
  - else \( \bar{z}_j'' = L/D_{\ell+1,r-1} \) where \( \ell < j'' < r \), \( z = L/D_{\ell+1,r-1} \), and all other variables zero.

- \( n \) points, where \( y_{\ell r} = 1 \) holds for one \((\ell, r)\) in \( S_j \), all \( \bar{z}_j'' = d'/D_{\ell+1,r-1} \) where \( \ell < j'' < r \) and \( d' = \min\{\bar{d}, \min_{\ell < j'' < r} \bar{M}D_{j''}\} \), \( z = d'/D_{\ell+1,r-1} \), and all other variables zero.

- 1 point, where \( y_{0,n+1} = 1 \), \( \bar{z}_j'' = U/D_{1,n} \), \( z = U/D_{1,n} \), \( x = 1 \), and all other variables zero.

To prove facet-defining property of (26), first let \( S_j = \{(\ell, r) \in L \mid (\ell < j \land j < r \leq \hat{j}) \lor (j \geq j \land r > \hat{j})\} \), i.e., the case (c). Consider the following \( \frac{n(n+1)}{2} + n + 1 \) affinely independent points:

- \( \frac{n(n+1)}{2} \) points, where \( y_{\ell r} = 1 \) holds for one \((\ell, r)\), and
  - if \( \ell < j < \hat{j} < r \), then \( \bar{z}_j'' = L/D_{\ell+1,r-1} \) where \( \ell < j'' < r \), \( z = L/D_{\ell+1,r-1} \), and all other variables zero.
  - else \( \bar{z}_j'' = \min\{\bar{M}, U/D_{\ell+1,r-1}\} \) where \( \ell < j'' < r \), \( z = \min\{\bar{M}, U/D_{\ell+1,r-1}\} \), and all other variables zero.

- \( n \) points, where \( y_{\ell r} = 1 \) holds for one \((\ell, r)\) in \( S_j \), all \( \bar{z}_j'' = d'/D_{\ell+1,r-1} \) where \( \ell < j'' < r \) and \( d' = \min\{\bar{d}, \min_{\ell < j'' < r} \bar{M}D_{j''}\} \), \( z = d'/D_{\ell+1,r-1} \), and all other variables zero.

- 1 point, where \( y_{\ell r} = 1 \) for \((\ell, r)\) such that \( \ell \geq j \) or \( \hat{j} \geq r \) and \( \bar{M}D_{\ell+1,r-1} \geq \bar{d} \); \( \bar{z}_j'' = U/D_{\ell+1,r-1} \), \( z = U/D_{\ell+1,r-1} \), \( x = 1 \), and all other variables zero. \( \square \)
4. Solution Methodologies

In this section, we introduce four heuristic solution methods. We first discuss three natural ways to obtain Lagrangian relaxations to the original MIP. We then explain why only two of them are implemented in Sections 4.2 and 4.3. The third heuristic is based on a different IP-formulation of the model, and the last heuristic is based on the metaheuristic idea of Varying Neighborhood Search, where we derive some problem-specific features to guide the search.

4.1. Lagrangian Relaxation (LR) for Upper Bounds

Here, we consider three variants:

1. **Relax constraints (3),(4), (9)-(10), (15)-(18)**: As proven by Proposition 9, the subproblem is guaranteed to be solved in polynomial time.

2. **Relax constraints (5)-(11)**: This creates $K$ separate subproblems, one for each snapshot $k$ (each subproblem being an IP problem).

3. **Relax constraints (15) and (17)**: This relaxation generates two subproblems, one for $y$ variables and one for $z$ and $x$ variables.

Let $LR_i$ indicate the Lagrangian Relaxation and $LD_i$ be the optimal solution of the Lagrangian dual for $i = 1, 2, 3$, i.e.,

$$LD_i = \min_u \max_{x,y,z} \ LR_i(u, x, y, z, z),$$

where $u$ is the vector of Lagrangian multipliers.

**Proposition 9** $LD_1 = \max\{ (1)\|(2) - (11), (15) - (18), 0 \leq x \leq 1, 0 \leq y \leq 1].$

In words, optimizing the Lagrangian dual for the first relaxation will generate a bound equal to LP relaxation of the original problem.

**Proof.** For this subproblem, consider the following network: For each $i \in I$, draw a network with $K + 2$ levels as follows: Level 0 has a single, dummy source node with supply of 1, level $K + 1$ has a single dummy sink node with demand 1, and each of the other $K$ levels have $L$ nodes (representing all $(\ell, r) \in \mathcal{L}$) with a demand of zero. First, from dummy source node, draw arcs to all $L$ nodes in level 1. From level 1, draw arcs to level 2 to such
that MLC shape \((\ell, r)\) in level 1 can be changed to \((\ell', r')\) in level 2. Repeat this for all
levels until level \(K\) is reached. Then, connect all the nodes in that level to the dummy
sink node. It is easy to observe that solving this network problem will generate integral
solutions to \(y\) variables that satisfy the constraints \((2), (5), (6)\). Finally, note that constraints
\((11)\) are independent from previous constraints and there is only a single binary variable on
each constraint related to linear variables; therefore all extreme point solutions will be in-
tegral. Therefore, \(\text{conv}(\{(x, y, z)\mid (2), (5), (6), (7), (8), (11), x \in \{0, 1\}^{|\mathcal{I}|}, y \in \{0, 1\}^{|\mathcal{I}| x |\mathcal{L}|}) = \text{conv}(\{(x, y, z)\mid (2), (5), (6), (7), (8), (11), 0 \leq x \leq 1^{|\mathcal{I}|}, 0 \leq y \leq 1^{|\mathcal{I}| x |\mathcal{L}|})\). □

Since the Lagrangian dual for this relaxation does not provide a bound better than the
LP relaxation bound of the original problem, we do not investigate this further. However,
we make this technical comment for the sake of completeness.

**Corollary 2** \(LD_3 \leq LD_1\).

Note that for \(LD_2\) and \(LD_3\), the subproblems do not have integrality property in general
and hence they will generate bounds at least as strong as (or probably lower than) the LP
relaxation, however with possibly more computational effort. As we will see in numerical
results, \(LR_2\) is computationally cheap, whereas \(LR_3\) requires significant effort.

### 4.2. LR-Based Heuristic I

Our first heuristic is based on the snapshot independence feature of \(LR_2\). We relax machinery
constraints that link the adjacent snapshots, i.e., \((5)-(11)\), and are thus able to solve each
snapshot subproblem independently. Instead of penalizing all the violated constraints in the
objective function like the usual LR methods do, we penalize only the dose violations, but
“fix” the neighboring snapshot machinery constraints as follows. Once a single-snapshot
subproblem is solved and its MLC shape is determined, i.e., \(y\)-variables are fixed, we can
impose the machinery constraints for the neighboring snapshots. This is done iteratively
until all MLC shapes for all snapshots are obtained. Then, the problem reduces to a simple
IP without machinery constraints (it would have been simply an LP if constraint \((11)\) was
not included). Algorithm 1 provides an overview of the heuristic, where \((a)^+ = \max(a, 0)\).

When we minimize the objective function in the inner loop, we involve all snapshots
that are fixed so far, in addition to \(k'\), in order to minimize the error function \(\sum_{v \in V_1}(L_v - d_v)^+ + \sum_{v \in V}(d_v - U_v)^+\) more accurately and to obtain more involved fixing decisions. Also
Algorithm 1 A heuristic that exploits snapshots consecutively

Ensure: $|K|$ candidate solutions.

1: for $k \in K$ do
2:   Solve the problem $P1$: $\min \{ \sum_{v \in V} (L_v - d_v)^+ + \sum_{v \in V} (d_v - U_v)^+ | (y, z, \bar{z}) \in X_{rel} \}$
3:   where $X_{rel} = \{ (2) - (4), (12), (15) - (18) \}$, for only $k$;
4:   Solutions are fixed $y$-variables $\Rightarrow$ MLC-shape for Snapshot $k$;
5:   for $k' = k + 1$ to $k' = |K|$ and $k' = k - 1$ to $k' = 1$ do
6:     Solve the problem $P1'$: $\min \{ \sum_{v \in V} (L_v - d_v)^+ + \sum_{v \in V} (d_v - U_v)^+ | (y, z, \bar{z}) \in X_{rel}' \}$
7:     where $X_{rel}' = \{ (2) - (8), (12), (15) - (18) \}$, for only $k'$;
8:     Fix shapes for $k'$;
9:   end for
10: Now all shapes fixed, solve the original problem;
11: end for

Note that one can replace this objective function with a discrete function that counts infeasibilities, i.e., $\min \sum_{v \in V} w_v^- + \sum_{v \in V} w_v^+$ such that $w_v^-, w_v^+ \in \{0, 1\}$ and $L_v w_v^- \geq L_v - d_v$; $(\bar{M} \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} D_{ij} K) w_v^+ \geq d_v - U_v$. This will simply provide an equal weight for each voxel to satisfy their dosage lower and upper bound constraints; rather than the “error”. We believe that in theory, such a counting objective is appropriate, at least mathematically, and is based on a similar logic like the objective function of our problem. However, in a number of preliminary tests we run, we have observed that this discrete counting function does not behave well computationally possibly due to symmetry, and therefore we use the original error function in the final algorithm.

In the current implementation, once the problem $P1$ is solved for $k$, first the forward (i.e., for $k' = k + 1$ to $k' = |K|$) and then the backward (i.e., for $k' = k - 1$ to $k' = 1$) subproblems are solved. Forward and backward subproblems can be solved separately in parallel at the same time if needed for a faster result, although we consider in this paper the sequential time for fair comparison purposes and leave parallelization to future research. We also note that one can apply different running orders of subproblems in the inner loop to obtain different solutions, e.g., in the alternating order (i.e., $k' = k + 1, k' = k - 1, \ldots, k' = K, k' = 1$) or in a randomized order (i.e., pick with probability $p$ forward or probability $1 - p$ backward subproblem). We run some limited experiments on this aspect and did not observe any significant difference. Hence we implemented the simple sequential order in this paper, although we plan more extensive computational tests as part of future research. The computational performance of Algorithm 1 (in particular how often feasible solutions can be found, solution qualities, and computation times) will be discussed in Section 5.
4.3. LR-Based Heuristic II

Our second LR-based heuristic is a combination of subgradient optimization and an IP-heuristic applied to the third Lagrangian relaxation (LR₃). If we relax constraints (15) and (17) and assign multipliers $\alpha^k_{ij}$ and $\beta^k_{ij}$ respectively to these two sets of constraints, we obtain two separate subproblems as follows:

\[(Pr.1): \max_{x,z,\bar{z}} \sum_{v \in V} x_v + \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} (\beta^k_{ij} - \alpha^k_{ij}) \bar{z}_{ij}^k - \beta^k_{ij} z^k \]

s.t. (7) - (12), (16), (18)

\[(Pr.2): \max_{y} \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} \bar{M} (\alpha^k_{ij} - \beta^k_{ij}) \sum_{(\ell,r) \in L, \ell<j<r} y^k_{i(\ell,r)} \]

s.t. (2) - (6)

Once these subproblems are solved individually, we perform subgradient optimization by updating the multipliers using a stepsize $\theta$ described as follows:

\[\alpha^k_{ij} \leftarrow \alpha^k_{ij} + \theta \left( \bar{z}_{ij}^k - \bar{M} \sum_{(\ell,r) \in L, \ell<j<r} y^k_{i(\ell,r)} \right)\]

\[\beta^k_{ij} \leftarrow \beta^k_{ij} + \theta \left( (\bar{M}(-1 + \sum_{(\ell,r) \in L, \ell<j<r} y^k_{i(\ell,r)}) + z^k) - \bar{z}_{ij}^k \right)\]

Every time the multipliers are updated, the solution to (Pr.2) is used to fix all $y$-variables and then a simplified version of the original problem is optimized to obtain a heuristic solution. The computational performance (in particular how often feasible solutions can be found, solution qualities, and computation times) will be discussed in Section 5.

4.4. A Centering-Based Heuristic

The challenge with the original formulation is the vast number of the $y$ variables, as these are defined for any possible left-right leaves combination. If one knew in advance where the “center” of an opening in a row lies, then one could simply define $n$ binary variables for this row instead of $n^2$ binary variables (hence a significant reduction of problem dimension and computational complexity), where these binary variables either indicate the left-leaf position.
(if column position is smaller than center) or the right-leaf position (if column position is bigger than center). In a heuristic fashion, one can extend this idea to the “center of an opening” for a given snapshot, i.e., a column being a center column for all rows, and then, solve this (probably easy) problem iteratively multiple times, e.g., by using different centering schemes. The pseudocode for the proposed heuristic is presented in Algorithm 2.

First, we define our notation and reformulate the problem for snapshot $k$. Let $y_{ij}^\ell$ ($y_{ij}^r$) be binary variables for row $i$ and column $j$, where $y_{ij}^\ell = 1$ ($y_{ij}^r = 1$) holds when the left (right) leaf position is on column $j$. Let $\hat{c}_k$ represent the predefined center column, and $\hat{\ell}_i$ ($\hat{r}_i$) indicate the left (right) leaf position fixed for row $i$ of the neighboring snapshot (if $k$ is not the first processed snapshot). The reformulation is then as follows:

$$\sum_{j=0}^{n+1} y_{ij}^\ell = 1 \quad \forall i \in I$$  \hspace{1cm} (27)

$$\sum_{j=\hat{c}_k+1}^{n+1} y_{ij}^r = 1 \quad \forall i \in I$$  \hspace{1cm} (28)

$$d_v = \sum_{i \in I} \sum_{j \in J} (\bar{z}_{ij}^k \times D_{ijv}^k) \quad \forall v \in V$$  \hspace{1cm} (29)

$$\bar{z}_{ij}^k \leq M \sum_{j'=0}^{j-1} y_{ij'}^\ell \quad \forall i \in I, \forall j \in [1, \hat{c}_k]$$  \hspace{1cm} (30)

$$\bar{z}_{ij}^k \leq M \sum_{j'=j+1}^{n+1} y_{ij'}^r \quad \forall i \in I, \forall j \in [\hat{c}_k+1, n]$$  \hspace{1cm} (31)

$$\bar{z}_{ij}^k \leq z^k \quad \forall k \in K, \forall i \in I, \forall j \in J$$  \hspace{1cm} (32)

$$\bar{z}_{ij}^k \geq M \left( -1 + \sum_{j'=0}^{j-1} y_{ij'}^\ell \right) + z^k \quad \forall i \in I, \forall j \in [1, \hat{c}_k]$$  \hspace{1cm} (33)

$$\bar{z}_{ij}^k \geq M \left( -1 + \sum_{j'=j+1}^{n+1} y_{ij'}^r \right) + z^k \quad \forall i \in I, \forall j \in [\hat{c}_k+1, n]$$  \hspace{1cm} (34)

$$\bar{z}_{ij}^k \geq 0 \quad \forall i \in I, \forall j \in J$$  \hspace{1cm} (35)

$$y_{ij}^\ell = 0 \quad \forall i \in I, \forall j \in J^\prime \setminus \{\hat{\ell}_i - \delta, \hat{\ell}_i + \delta\}$$  \hspace{1cm} (36)

$$y_{ij}^r = 0 \quad \forall i \in I, \forall j \in J^\prime \setminus \{\hat{r}_i - \delta, \hat{r}_i + \delta\}$$  \hspace{1cm} (37)

One important aspect that needs further elaboration is the selection of $\hat{c}_k$. We propose
Algorithm 2 A centering-based heuristic

Ensure: \(|K|\) candidate solutions.
1: Pre-solve to obtain \(c^k\) for each snapshot \(k \in K\);
2: for \(k \in K\) do
3: Using \(c^k\), solve the problem: \(\min \{\sum_{v \in V_t} (L_v - d_v)^+ + \sum_{v \in V} (d_v - U_v)^+ | (y^l, y^r, z, \bar{z}) \in X_{ref}\}\)
4: where \(X_{ref} = \{(27) - (35)\}\), for only Snapshot \(k\);
5: Fix shapes for \(k\), set \(\hat{\ell}_i\) and \(\hat{r}_i\) values;
6: for \(k' = k + 1\) to \(k' = |K|\) and \(k' = k - 1\) to \(k' = 1\) do
7: Using \(c^{k'}\), solve the problem: \(\min \{\sum_{v \in V_t} (L_v - d_v)^+ + \sum_{v \in V} (d_v - U_v)^+ | (y^l, y^r, z, \bar{z}) \in X'_{ref}\}\)
8: where \(X'_{ref} = \{(27) - (37)\}\), for only \(k'\);
9: Fix shapes for \(k'\), set \(\hat{\ell}_i\) and \(\hat{r}_i\) values;
10: end for
11: Now all shapes fixed, solve the original problem;
12: end for

the following intuitive possibilities:

\[
\hat{c}^k_1 = \frac{\sum_{v \in V_t} L_v \left( \sum_{i \in I} \sum_{j \in J} (j - 0.5) D_{ijv}^k \right)}{\sum_{v \in V_t} L_v}
\]

\[
\hat{c}^k_2 = \frac{\sum_{v \in V_o} \frac{1}{U_v} \left( \sum_{i \in I} \sum_{j \in J} (j - 0.5)(1/D_{ijv}^k) \right)}{\sum_{v \in V_o} \frac{1}{U_v}}
\]

\[
\hat{c}^k_3 = w_1 \hat{c}^k_1 + w_2 \hat{c}^k_2
\]

\(\hat{c}^k_1\) emphasizes higher dose and lower \(L_v\), and depends on only tumor voxel parameters whereas \(\hat{c}^k_2\) emphasizes lower dose and higher \(U_v\), and depends only on sensitive tissue parameters. \(\hat{c}^k_3\) simply combines these two with weights \(w_1, w_2\). We observed minimal differences on some preliminary computational tests using different weights, hence we use \(\hat{c}^k_3\) with \(w_1 = 0.5 = w_2\) in the final algorithm.

The main advantage of this framework is not only a significant reduction in the number of variables but also in the elimination of inter-leaf constraints. On the other hand, this
method has the disadvantage that it limits the candidate opening patterns for a snapshot to only “centered” patterns (e.g., the pattern cannot be diagonally-shaped) and infeasibilities are more probable. We will discuss these aspects in detail in Section 5. We also note that this heuristic is related to the first \(LR\)-based heuristic, where single snapshot problems are solved consecutively for each snapshot after the previous snapshots are fixed. The main difference here is that the subproblem is even further simplified, since allowing only one center for a whole snapshot reduces the solution space significantly. We will compare computational results for both of these methods in the next section.

Finally, we also note that one might use this approach in an exact fashion, where the parameters \(\hat{c}_k\) will need to be redefined as variables. This will require a sophisticated approach using a specialized branch-and-bound and column generation scheme, which is discussed in detail in a companion paper [2].

### 4.5. A Guided Varying Neighborhood Scheme (GVNS)

Our Guided Varying Neighborhood Scheme (GVNS) aims to tackle problems of large scale, and therefore, feasibility is the primary objective; the first attempt is to obtain solutions that satisfy all machinery and dose constraints. When such a feasible solution is found, a predetermined number of attempts to improve the solution will then be triggered. This is done so by keeping track of the current best feasible solution with the highest original objective value. If there are no improvements to the original objective value after a predetermined number of dose and machinery feasible solutions are found, we return the current feasible solution with the best original objective value as an output.

The method randomly generates monitor units and MLC shapes that satisfy the machinery constraints for each snapshot. We define two main neighborhoods:

- \(N^\mu\) is the neighborhood of solutions obtained by modifying the monitor units. This is further divided into \(N^\mu_+\) (increase MU by \(\Delta_\mu\)) and \(N^\mu_-\) (decrease MU by \(\Delta_\mu\)).

- \(N^s\) is the neighborhood of solutions obtained by modifying the MLC shapes.

For the former, we further define \(N^\mu_{\pm 1} \subset N^\mu_{\pm 2} \subset \cdots \subset N^\mu_{\pm VNS_{\text{max}}},\) i.e., \(N^\mu_{\pm i}\) is the neighborhood defined by having \(i\) MUs increased (or decreased) while satisfying the machinery constraints. (In our experiments, we used \(\Delta_\mu = 1\). \(VNS_{\text{max}}\) denotes the maximum number of MUs that are allowed to be changed, calculated as a fraction of \(|K|\).
Similarly, we also have $N_1^s \subset N_2^s \subset \cdots \subset N_{VNS_{max}}^s$, with $N_p^s$ representing the random selection of $p$ snapshots and modifying the associated MLC shapes by moving each leaf by a random selection from the feasible moves of the leaves. Our preliminary experiments indicated that the result of implementing two different shape change schemes (enlarge_shape and reduce_shape) does not differ significantly from that of the random shape change. With the former, while fixing violations in under dosed voxels, overdose is often induced in other voxels, and vice versa. We implemented four variations of our heuristic method:

- **Method 1 - Random Decent Local Search**
  
  In this method, the neighbor search alternating between MU change $(N^\mu)$ and shape change $(N^s)$. The method randomly selects a snapshot and modifies its MU or shape.

- **Method 2 - A Guided Search**
  
  This is done in a way that the neighborhood search is “guided” by the current solution to improve machinery constraint satisfaction. Let $oCNT$ be the number of voxels overdosed and $uCNT$ be the number of voxels under-dosed. Then:
  
  - if $\frac{oCNT - uCNT}{oCNT} \geq 0.1$, we search the neighborhood of $N_{-1}^\mu$ to decrease the MU;
  - if $\frac{uCNT - oCNT}{uCNT} \geq 0.1$, we search the neighborhood of $N_{+1}^\mu$ to increase the MU;
  - otherwise we search $N^S$ to change the shape.

- **Method 3 - A Varying neighborhood Search**
  
  In this method, we also alternate between MU change and shape change. First, we calculate $VNS_{max} = \lceil\gamma|K|\rceil$, for $0 < \gamma < 1$ a user-determined value (we used $\gamma = 0.3$ in our experiments). Initially, we set $size_{VNS} = \lceil\frac{|oCNT - uCNT|}{|V|} \times (VNS_{max} - 1)\rceil$. If there is no improvement to the objective value, we increase $size_{VNS}$ by 1, until $VNS_{max}$ is reached.

Method 3 is executed in two different ways. In Method 3a, shape change is carried out when the iteration count is even with $size_{VNS}$ snapshots changed at a time, and MU change is carried out otherwise. In Method 3b, in each iteration of the neighborhood search, MU change and shape change are executed iteratively.

- **Method 4 - A Guided Varying neighborhood Search**
Our preliminary results showed that Method 4 outperformed Methods 1–3, hence we will describe only this method in detail. The VNS and its variations has been around for many years (for general literature overview, see [7, 14]). In our problem-specific implementation, we modified the VNS by integrating the “guided search” idea of Method 2.

We begin with a randomly generated machinery feasible solution with leaf positions and monitor units that satisfy the machinery constraints. We then calculate the dose delivered to each voxel to find out the dose violations, and perform a search guided in the manner of promoting dose satisfaction.

The objective function is given by $vCNT = oCNT + uCNT$. If $vCNT > 0$, the GVNS will proceed to search for “neighboring” solutions by using one of the two neighborhood search schemes $N^\mu$ and $N^s$. Our method attempts to increase or decrease MUs on first instance. We also perform MLC shape change on a regular basis, or if no consecutive-snapshot-feasible MU change is possible. Should there be no machinery feasible leaf movements available, we modify our shape change procedure by first selecting $i$ snapshots randomly, select the left- and right-leaf positions for the first row of the MLC of each of these $i$ snapshots, and generate leaf positions for subsequent rows that satisfy the inter-leaf constraints. It is possible that some consecutive snapshot leaf restrictions are violated, and these violations are added into the cost function. Once in a while, a new MLC-feasible solution will be generated for all $|K|$ snapshots, allowing random starts for the search. See Algorithm 3 for the pseudocode.
Algorithm 3 GVNS for VMAT

Ensure: Feasible solution (or solution with least violations).

1: Set $ItCNT = \kappa = \omega = 0$;
2: Randomly generate a machinery feasible initial solution $x'$;
3: Set objective cost $z' = oCNT + uCNT$;
4: Set best soln $x^* \leftarrow$ initial soln $x'$, best cost $z^* \leftarrow$ initial cost $z'$;
5: if $z' = 0$ then set $\omega' = \omega(x')$, $\kappa \leftarrow \kappa + 1$;
6: end if
7: while $(\kappa < \kappa_{\text{max}}) \wedge (ItCNT < ItMAX)$ do
8:     if $(ItCNT \mod ItNEW = 0)$ then Generate a new $x'$;
9:     end if
10:    if $(oCNT > uCNT) \lor (uCNT > oCNT)$ then
11:        Set $\text{sizeVNS} = \left\lceil \frac{(\kappa - oCNT)}{(VNS_{\text{max}} - 1)} \right\rceil$;
12:    else if $oCNT > uCNT$ then
13:        Reduce MUs;
14:        Obtain $x''$ by random from $N_{-\text{sizeVNS}}(x')$, calculate $oCNT$, $uCNT$; and set $z'' \leftarrow oCNT + uCNT$;
15:        if $z'' = 0$ then
16:            if $\omega(x'') \geq \omega^*$ then
17:                Set $\omega^* \leftarrow \omega(x'')$ and Set $\kappa = 0$;
18:            else
19:                Set $\kappa \leftarrow \kappa + 1$;
20:            end if
21:        else if $z'' > z^*$ then
22:            Set $\text{sizeVNS} \leftarrow \min\{VNS_{\text{max}}, \text{sizeVNS} + 1\}$;
23:        else if $z'' < z^*$ then
24:            Set $z^* \leftarrow z''$ and $x^* \leftarrow x''$;
25:            Set $\text{sizeVNS} \leftarrow \max\{1, \text{sizeVNS} - 1\}$;
26:        end if
27:    else if $uCNT > oCNT$ then
28:        Increase MUs;
29:        Obtain $x''$ by random from $N_{+\text{sizeVNS}}(x')$, calculate $oCNT$, $uCNT$; and set $z'' \leftarrow oCNT + uCNT$;
30:        if $z'' = 0$ then
31:            if $\omega(x'') \geq \omega^*$ then
32:                Set $\omega^* \leftarrow \omega(x'')$ and Set $\kappa = 0$;
33:            else
34:                Set $\kappa \leftarrow \kappa + 1$;
35:            end if
36:        else if $z'' > z^*$ then
37:            Set $\text{sizeVNS} \leftarrow \min\{VNS_{\text{max}}, \text{sizeVNS} + 1\}$;
38:        else if $z'' < z^*$ then
39:            Set $z^* \leftarrow z''$ and $x^* \leftarrow x''$;
40:            Set $\text{sizeVNS} \leftarrow \max\{1, \text{sizeVNS} - 1\}$;
41:        end if
42:    else
43:        Perform ChangeShape routine;
44:        Obtain $x''$ by random from $N_{+\text{sizeVNS}}(x')$, calculate $oCNT$, $uCNT$; and set $z'' \leftarrow oCNT + uCNT$;
45:        if $z'' = 0$ then
46:            if $\omega(x'') \geq \omega^*$ then
47:                Set $\omega^* \leftarrow \omega(x'')$ and Set $\kappa = 0$;
48:            else
49:                Set $\kappa \leftarrow \kappa + 1$;
50:            end if
51:        else if $z'' > z^*$ then
52:            Set $\text{sizeVNS} \leftarrow \min\{VNS_{\text{max}}, \text{sizeVNS} + 1\}$;
53:        else if $z'' < z^*$ then
54:            Set $z^* \leftarrow z''$ and $x^* \leftarrow x''$; and $\text{sizeVNS} \leftarrow \max\{1, \text{sizeVNS} - 1\}$;
55:        end if
56:    end if
57:    else
58:        Perform ChangeShape routine (as described above);
59:    end if
60:    end while

We note the key notation, as follows:
- \( \text{ItCNT} \) a counter for number of iterations the GVNS has been performed.
- \( \text{ItMAX} \) a predetermined maximum number of iterations to be performed.
- \( \text{ItNEW} \) a new machinery feasible solution is generated every \( \text{ItNEW} \) iterations.
- \( \kappa \) a counter for number of occurrence when a feasible solution is found and it does not improve the current best original objective function value.
- \( \kappa_{\text{max}} \) the maximum value for \( \kappa \) allowed in the GVNS (in our experiments, \( \kappa_{\text{max}} = 5 \)).
- \( \omega(x') \) the value of the original objective function given by solution \( x' \).
- \( \omega^* \) best value of original objective function out of all feasible solutions found.

5. Numerical Results

In this section, we present our numerical results for the various heuristic methods, starting with a discussion of some preliminary aspects. For our experiments, we used randomly generated problem instances that are feasible (30 instances with sizes from small to large, plus 3 very large instances), as described in [1], where detailed documentation and entire data set are available. Note that we also generated 10 extra large problem instances with the number of non-zero \( D^k_{ijv} \) values comparable (in some instances larger) to those reported in the recent work of [17] and tested them only with the GVNS, as the IP-based heuristics were not able to solve problems this size. In Table 1, we provide an overview of the parameters used in the problem generator. This notation is used to name instances in the format of MLC-Voxels/\( |K| - \rho - z_{\text{max}} - \Delta - \delta - d_{\text{max}} \), e.g., 6-6-6-6-6-6/16-0.1-5-2-2-3 is an instance with a 6 × 6 MLC array and 6 × 6 × 6 voxels, 16 snapshots, and so on.
Table 1 Parameters used in the problem generator.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>number of rows in MLC</td>
</tr>
<tr>
<td>$n$</td>
<td>number of columns in MLC</td>
</tr>
<tr>
<td>$w$</td>
<td>width of treatment field (# voxels)</td>
</tr>
<tr>
<td>$h$</td>
<td>height of treatment field (# voxels)</td>
</tr>
<tr>
<td>$d$</td>
<td>depth of treatment field (# voxels)</td>
</tr>
<tr>
<td>$</td>
<td>K</td>
</tr>
<tr>
<td>$\rho$</td>
<td>for generating $U_v$ and $L_v$ by using $d_v(1 \pm \rho)$; tested values $\rho = 0.08$, 0.1, 0.2</td>
</tr>
<tr>
<td>$z_{\text{max}}$</td>
<td>max. value for MU allowed from any snapshot</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>max. MU change in consecutive snapshots</td>
</tr>
<tr>
<td>$\delta$</td>
<td>max. leaf traveling in consecutive snapshots</td>
</tr>
<tr>
<td>$d_{\text{max}}$</td>
<td>max. possible value for $D_{ij}^k$</td>
</tr>
</tbody>
</table>

All computations reported in this section are executed on a PC with i7-2600 processor (3.40 GHz) and 8GB allocated memory. The optimization models, Lagrangian relaxations (Section 4.1), and all the IP-based heuristics (Sections 4.2-4.4) are implemented in OPL modeling language and tested using IBM ILOG Cplex 12.2, whereas the GVNS heuristic (Section 4.5) is implemented in C++. Before discussing results comparing the efficiency of the proposed heuristic methods, we discuss some key preliminary tests.

First, we tested the computational complexity of the test instances. Out of 33 problems, ILOG Cplex with default settings and 1 hour/1% relative gap limitation could find feasible solutions for only 8 problems, proving optimality for only 2 of these. These results are presented in Table 2, where t/gap column indicates time if $< 3600$ sec. and otherwise $\text{gap} = \frac{(UB - LB)}{LB}$. For instances with 15x15 MLC arrays and above (except 15-15-15-15-15/8-0.1-5-2-3-3), Cplex could not even find an upper bound in this time (see online supplement for details).

Table 2 Default Cplex runs with solutions found.

<table>
<thead>
<tr>
<th>Problem instance</th>
<th>LB</th>
<th>UB</th>
<th>t/gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-6-6-6-6/16-0.1-5-2-2-3</td>
<td>17</td>
<td>86</td>
<td>405.88%</td>
</tr>
<tr>
<td>7-7-7-7/8-0.08-5-1-1-3</td>
<td>273</td>
<td>274</td>
<td>801</td>
</tr>
<tr>
<td>7-7-7-7/8-0.1-5-1-1-3</td>
<td>103</td>
<td>118</td>
<td>14.56%</td>
</tr>
<tr>
<td>7-7-7-7/8-0.2-5-1-1-3</td>
<td>6</td>
<td>33</td>
<td>450.00%</td>
</tr>
<tr>
<td>7-7-7-7/12-0.2-5-1-1-3</td>
<td>0</td>
<td>92</td>
<td>$\infty$</td>
</tr>
<tr>
<td>8-8-8-8-8/16-0.1-5-2-2-3</td>
<td>27</td>
<td>426</td>
<td>1477%</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-3-2-3-2</td>
<td>1728</td>
<td>1728</td>
<td>993</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-3-4-3</td>
<td>312</td>
<td>1710</td>
<td>448.08%</td>
</tr>
</tbody>
</table>
Next, we discuss the computational complexity of the Lagrangian relaxations presented in Section 4.1. For 3 groups of test instances with MLC sizes $7 \times 7$, $12 \times 12$ and $15 \times 15$, we run 100 iterations of each Lagrangian relaxation on each test instance using randomly selected objective function parameters. The average computational time over all test instances for a single iteration of each relaxation is given in Table 3, where * indicates that there were iterations that were not finished before a pre-set time limit of 900 seconds. As these results show, $LR_3$ is significantly harder to solve (not even finishing for many iterations of bigger problems when $cplex.tilim = 900$ is used), and as we will see later, this affects the efficiency of the second LR-based heuristic. On the other hand, $LR_2$ can be solved very efficiently, even faster than $LR_1$. Note that one might improve the computational times of $LR_1$ by implementing the network idea discussed in the proof of Proposition 9.

<table>
<thead>
<tr>
<th>MLC size</th>
<th>$LR_1$</th>
<th>$LR_2$</th>
<th>$LR_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$7 \times 7$</td>
<td>0.81</td>
<td>0.51</td>
<td>23.64</td>
</tr>
<tr>
<td>$12 \times 12$</td>
<td>15.48</td>
<td>6.05</td>
<td>511.21*</td>
</tr>
<tr>
<td>$15 \times 15$</td>
<td>64.56</td>
<td>17.04</td>
<td>628.50*</td>
</tr>
</tbody>
</table>

Finally, we present and discuss the computational results for the four proposed methods. We use the abbreviations $LRHeur1$, $LRHeur2$ and $CentHeur$ to refer to the LR-based Heuristics (Sections 4.2 and 4.3) and Centering-based Heuristic (Section 4.4), respectively. To ensure reasonable efficiency, pre-set time limits were used for different routines of all the IP-based heuristics, for which we run extensive preliminary tests to determine these parameters. In order to avoid fine tuning and obtain fair comparisons, we used the same time limit of $cplex.tilim = 90$ for all different subroutines except the “fixed heuristic” executed at the end of each iteration (i.e., the heuristic that uses the fixed $y$ variables to solve the original problem). The fixed heuristic uses a time limit of $cplex.tilim = 120$ to increase the probability of finding feasible solutions. We note that increasing these time limits do not necessarily improve solution quality, as results in the online supplement show for the case of $cplex.tilim = 900$ (and $cplex.tilim = 1200$ for the fixed heuristic). Finally, we note that for $LRHeur2$, we set the step size $\theta = 0.01$ (based on preliminary testing with a range of values) and limit the number of subgradient optimization iterations to 20 (based on the fact that $LR_3$ is computationally hard to solve).

Detailed computational results can be seen in Tables 6 and 5. Table 6 is organized in
the following column order: objective values of the 1st and best solutions, computation
time (in seconds) to the 1st solution, best solution, and total time. Note that since GVNS
heuristic has an element of randomness in it and since this method is much faster than
the other methods, 5 runs are executed for each instance, and details for each of these
runs are presented for the sake of completeness. Table 5 presents the results for the IP-
based heuristics, where for each heuristic, number of iterations (to the 1st solution, to the
best solution, and total number), number of feasible solutions obtained, objective values of
the 1st and best solutions, and computation time (in seconds if < 3600, otherwise in the
hour/minute format) are presented, in this order. As summarized in Table 4, LRHeur2 is
very efficient for small problems but fails to find feasible solution even for most of the medium
problems. It is also important to note that CentHeur and GVNS are quite inefficient for small
problems (they either fail or find poor solutions), which is an expected result as these methods
are prone to infeasibility when the search space is limited. LRHeur1 and CentHeur seem to
be similarly efficient for medium and large problems (especially the difference between their
best solutions is less than 10% for all large instances). Finally, we emphasize that GVNS
is the method of choice when the problem sizes become very large that IP-based heuristics
mostly fail or cannot generate good solutions.

As results indicate, LRHeur1 seems to be the most successful method overall for the
aspect of consistency in finding solutions, having failed to find a feasible solution only in 12
iterations from overall 452 iterations executed (note this excludes the very large instances).
A similar comment can be made for GVNS, unless the problem is small. The method finds a
solution in every run for all instances with MLC size 8 × 8 or above, although the best solution
qualities might vary significantly from one run to another. This variation is encompassed
by a very crucial advantage of GVNS, however: The method runs very fast for the majority
of instances. No medium or large instance takes more than one hour for 5 runs (indeed,
14 of these instances take 10 minutes or less), and this is even true for 2 of the 3 very
large instances. We also note that the variation in solution time is in general acceptable. IP-based heuristics are far slower, with the fastest one (*CentHeur*) achieving computational times of around 1 hour for the majority of medium and large instances (with occasional variation), and around 2 hours for very large instances. However, we note that in case of limited time, one can run these heuristics for a few iterations only, possibly focusing on snapshots/iterations that seem to be most successful. For example, *CentHeur* seems more successful for later snapshots and *LRHeur1* is more successful for earlier snapshots (although it is also important to note for these two heuristics that like in any other heuristics, there is a factor of chance in this as well as the effect of control parameters used such as time and memory limits). Finally, we also note that only *LRHeur2* and GVNS are methods with a user-defined number of iterations, as it is not dependent on the number of snapshots (hence more iterations can be added if computational resources are available) to possibly improve solutions.

To summarize, we observe that the GVNS performs very efficiently for larger problems, finding feasible (and many times, good quality) solutions in very short times, where solution times remain fairly stable for different runs of the same problem instance. This justifies the GVNS to be implemented in real-life applications, compared to other methods. In order to verify this conclusion with more realistic problems, we tested the GVNS on 10 problem instances of very large scales, which are comparable or even larger than the instances presented in [17], where the most difficult problem instance has $114,315,187$ non-zero $D^k_{i,j,v}$. We present the results for 1st and overall solutions in Table 7, including problem characteristics in the same format as that of [17] for easier comparability. For these runs, we used a quad-core iMac with 2.93GHz Intel Core i7 and 32 GB RAM.
Table 5 Detailed results for LRHeur1, LRHeur2 and CentHeur, with \textit{cplex.tilim} = 90 set for subroutines, and \textit{cplex.tilim} = 120 set for the “fixed heuristic” run at the end of each iteration.

<table>
<thead>
<tr>
<th>Problem</th>
<th>LRHeur1 Results</th>
<th>Objective values</th>
<th>LRHeur2 Results</th>
<th>Objective values</th>
<th>CentHeur Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># iterations</td>
<td>1st sol</td>
<td>Best sol</td>
<td>Total</td>
<td># sol’s</td>
</tr>
<tr>
<td>6-6-6-6-6/16-0.1-5-2-2-3</td>
<td>1 4 16 16 3 8</td>
<td>75</td>
<td>1 7 20</td>
<td>18</td>
<td>4 15</td>
</tr>
<tr>
<td>7-7-7-7-7/0.8-5-1-1-3</td>
<td>1 1 8 8 66 66</td>
<td>47</td>
<td>3 19 20</td>
<td>9</td>
<td>54 61</td>
</tr>
<tr>
<td>7-7-7-6-0.5-1-1-1-3</td>
<td>1 4 8 8 0 1</td>
<td>19</td>
<td>2 2</td>
<td>20 12</td>
<td>7</td>
</tr>
<tr>
<td>7-7-7-7-8/0.2-5-1-1-3</td>
<td>1 1 8 8 0 0</td>
<td>812</td>
<td>2 2 20</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.1-5-1-1-3</td>
<td>1 1 12 5 0 0</td>
<td>362</td>
<td>No feasible solution after 20 iterations</td>
<td>No feasible solution after 12 iterations</td>
<td>3 5</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.1-5-1-1-3</td>
<td>1 6 12 12 3 9</td>
<td>311</td>
<td>2 3</td>
<td>20 19</td>
<td>14</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.2-5-1-1-3</td>
<td>1 1 12 12 0 0</td>
<td>880</td>
<td>1</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.1-5-1-1-3</td>
<td>1 8 16 16 4 7</td>
<td>145</td>
<td>3</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.1-5-1-1-3</td>
<td>1 1 16 11 2 2</td>
<td>809</td>
<td>3</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.2-5-1-1-3</td>
<td>1 1 16 16 0 0</td>
<td>1218</td>
<td>1</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>8-8-8-8-8/16-0.1-5-2-2-3</td>
<td>1 3 16 16 35 65</td>
<td>3350</td>
<td>3</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>10-10-10-10/32-0.1-5-3-3-5</td>
<td>1 12 32 32 30 688</td>
<td>2h45n</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12-12-12-12/8-0.1-5-2-3-3</td>
<td>1 2 8 8 149 230</td>
<td>2159</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-3-2-3-2</td>
<td>1 1 16 16 1728 1728</td>
<td>1768</td>
<td>No feasible solution after 20 iterations</td>
<td>No feasible solution after 20 iterations</td>
<td>3</td>
</tr>
<tr>
<td>12-12-12-12/16-0.08-5-2-3-3</td>
<td>1 2 16 16 1154 1344</td>
<td>1h24m</td>
<td>3</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-8-2-3-5</td>
<td>1 4 16 16 336 491</td>
<td>1h26m</td>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-1-2-3-3</td>
<td>1 5 16 16 197 325</td>
<td>2h16m</td>
<td>2</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-2-3-5</td>
<td>1 4 16 16 189 341</td>
<td>2h06m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-3-4-3</td>
<td>1 3 16 16 706 768</td>
<td>1h24m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.2-5-2-3-3</td>
<td>1 1 16 16 567 567</td>
<td>1h25m</td>
<td>2</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/24-1.5-2-3-3</td>
<td>1 1 24 24 1728 1728</td>
<td>1h57m</td>
<td>3</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>15-15-15-15/8-0.1-2-3-3</td>
<td>1 4 8 8 1889 2213</td>
<td>2h31</td>
<td>2</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-2-3-5</td>
<td>1 5 16 16 2527 2906</td>
<td>3h12m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.08-5-2-3-3</td>
<td>1 2 16 16 3324 3352</td>
<td>2h42m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-1-2-3-2</td>
<td>1 3 16 16 3103 3172</td>
<td>2h56m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-2-3-3</td>
<td>1 3 16 16 3374 3375</td>
<td>2h46</td>
<td>2</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-3-5-5</td>
<td>1 1 16 16 3375 3375</td>
<td>1h47m</td>
<td>3</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.2-5-2-3-3</td>
<td>1 5 16 16 3093 3152</td>
<td>2h30m</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/24-1.5-2-3-3</td>
<td>1 1 16 16 3375 3375</td>
<td>3h18m</td>
<td>4</td>
<td>7</td>
<td>24</td>
</tr>
<tr>
<td>18-18-18-18/16-0.1-5-3-4-3</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>2</td>
</tr>
<tr>
<td>20-20-20-20/16-0.1-5-3-4-3</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>No feasible solution after 16 iterations</td>
<td>3</td>
</tr>
<tr>
<td>Problem Instance</td>
<td>Objective Value</td>
<td>Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-6-6-6/16-0.1-5-2-3</td>
<td>1</td>
<td>80</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-7-7-7/6-0.08-5-1-1-3</td>
<td>No feasible solution found in 4 runs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-7-7-7/6-0.1-5-1-1-3</td>
<td>No feasible solution found in 5 runs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-7-7-7/6-0.2-5-1-1-3</td>
<td>No feasible solution found in 5 runs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-8-8-8/1</td>
<td>6</td>
<td>140</td>
<td>266</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-0.1-5-2-3-3</td>
<td>1</td>
<td>15</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>145</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>136</td>
<td>183</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-10-10-10/</td>
<td>68</td>
<td>68</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-12-12-12/</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>376</td>
<td>376</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>240</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7 Problem instances with very large number of snapshots.

<table>
<thead>
<tr>
<th>Problem Instances</th>
<th>Problem characteristics</th>
<th>Solution details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># voxels</td>
<td># beamlets</td>
</tr>
<tr>
<td>$20 \times 20 \times 20 \times 20 \times 20 \times 32$</td>
<td>8,000</td>
<td>12,800</td>
</tr>
<tr>
<td>$10 \times 10 \times 25 \times 25 \times 40 \times 45$</td>
<td>25,000</td>
<td>4,500</td>
</tr>
<tr>
<td>$10 \times 10 \times 15 \times 15 \times 30 \times 180$</td>
<td>6,750</td>
<td>18,000</td>
</tr>
<tr>
<td>$15 \times 15 \times 15 \times 15 \times 15 \times 180$</td>
<td>3,375</td>
<td>40,500</td>
</tr>
<tr>
<td>$20 \times 20 \times 20 \times 20 \times 20 \times 45$</td>
<td>8,000</td>
<td>18,000</td>
</tr>
<tr>
<td>$10 \times 10 \times 25 \times 25 \times 40 \times 60$</td>
<td>25,000</td>
<td>6,000</td>
</tr>
<tr>
<td>$15 \times 15 \times 20 \times 20 \times 20 \times 90$</td>
<td>8,000</td>
<td>20,250</td>
</tr>
<tr>
<td>$10 \times 10 \times 25 \times 25 \times 40 \times 75$</td>
<td>25,000</td>
<td>7,500</td>
</tr>
<tr>
<td>$20 \times 20 \times 20 \times 20 \times 20 \times 60$</td>
<td>8,000</td>
<td>24,000</td>
</tr>
<tr>
<td>$20 \times 20 \times 20 \times 20 \times 20 \times 72$</td>
<td>8,000</td>
<td>28,800</td>
</tr>
</tbody>
</table>

† Best solution with original objective recorded 3 times before 1,000 iterations reached.

6. Conclusions

In this paper, we studied the combinatorial optimization problem underlying the treatment planning optimization of a new form of radiotherapy: Volumetric-Modulated Arc Therapy. Our contribution lies in presenting a formulation for this problem, investigating some theoretical properties of key subproblems of it, and proposing four solution methods that are also extensively tested using a set of problem instances with sizes ranging from small to very large and practical. We also make our set of test problems available to other researchers, in the hope that this will provide a venue for more efficient comparisons and fruitful discussions in the research community.

One of the important aims we stated at the start was to find feasible solutions that satisfy both the machinery and dose constraints. Our computational results indicated that this was achieved in general, in particular for our most efficient method, Guided Variable Neighborhood Search, with success in problems of sizes comparable to real clinical problems. One area we left for future research is to investigate possibilities of mixing these different methods such that parallelization can help practitioners to obtain even better performance. It is also important to note that exact methods and theoretical properties should be investigated further to help us understand these problems better, some of which we address in a companion paper [2].
Acknowledgments

We would like to thank Dr. Thu Tran, the Chief Medical Physicist at Andrew Love Cancer Centre (Geelong, Vic, Australia), for her valuable clinical guidance and advices. Dr. Mak-Hau’s research is supported in part by Deakin University’s CRGS Grant 2007 for the project titled “Novel treatment planning in intensity-modulated radiotherapy”.

References


Appendix: Detailed Computational Results

Table 8 Lower/upper bounds after 1 hour default Cplex run, with `cplex.epochs = 0.01`. Time/Gap column indicates time if < 3600 sec., otherwise \( gap = \frac{(UB-LB)}{LB} \). Note that no upper bounds can be found for any other instances with MLC size of 15 × 15.

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th>LB</th>
<th>UB</th>
<th>Time/Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-6-6-6-6/16-0.1-5-2-2-3</td>
<td>17</td>
<td>86</td>
<td>405.88%</td>
</tr>
<tr>
<td>7-7-7-7-7/8-0.08-5-1-1-3</td>
<td>273</td>
<td>274</td>
<td>801</td>
</tr>
<tr>
<td>7-7-7-7-7/8-0.1-5-1-1-3</td>
<td>103</td>
<td>118</td>
<td>14.56%</td>
</tr>
<tr>
<td>7-7-7-7-7/8-0.2-5-1-1-3</td>
<td>6</td>
<td>33</td>
<td>450.00%</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.08-5-1-1-3</td>
<td>NA</td>
<td>69</td>
<td>∞</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.1-5-1-1-3</td>
<td>NA</td>
<td>154</td>
<td>∞</td>
</tr>
<tr>
<td>7-7-7-7-7/12-0.2-5-1-1-3</td>
<td>0</td>
<td>92</td>
<td>∞</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.08-5-1-1-3</td>
<td>NA</td>
<td>160</td>
<td>∞</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.1-5-1-1-3</td>
<td>NA</td>
<td>73</td>
<td>∞</td>
</tr>
<tr>
<td>7-7-7-7-7/16-0.2-5-1-1-3</td>
<td>NA</td>
<td>38</td>
<td>∞</td>
</tr>
<tr>
<td>8-8-8-8-8/16-0.1-5-2-2-3</td>
<td>27</td>
<td>426</td>
<td>1477%</td>
</tr>
<tr>
<td>10-10-10-10-10/32-0.3-10-3-3-3-5</td>
<td>NA</td>
<td>995</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-12/8-0.1-5-2-3-3</td>
<td>NA</td>
<td>1611</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-16-0.1-3-2-3-2</td>
<td>1728</td>
<td>1728</td>
<td>993</td>
</tr>
<tr>
<td>12-12-12-12-16-0.08-5-2-3-3</td>
<td>NA</td>
<td>1719</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-16-0.1-8-2-3-5</td>
<td>NA</td>
<td>1642</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-16-0.1-5-1-2-3</td>
<td>NA</td>
<td>1628</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-16-0.1-5-2-3-3</td>
<td>NA</td>
<td>1609</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-16-0.1-5-3-4-3</td>
<td>312</td>
<td>1710</td>
<td>448.08%</td>
</tr>
<tr>
<td>12-12-12-12-16-0.2-5-2-3-3</td>
<td>NA</td>
<td>1649</td>
<td>∞</td>
</tr>
<tr>
<td>12-12-12-12-24-0.1-5-2-3-3</td>
<td>NA</td>
<td>1728</td>
<td>∞</td>
</tr>
</tbody>
</table>
Table 9 LRHeur1 results for medium (MLC size $12 \times 12$) and large (MLC size $15 \times 15$) problems, with \texttt{cplex.tilim} = 900 for subroutines and \texttt{cplex.tilim} = 1200 for fixed heuristic. * indicate instances where Cplex stopped due to no memory left (hence total number of iterations $< |K|$).

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th># iterations</th>
<th>Objective values</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># sol's</td>
<td>1st sol</td>
<td>Best sol</td>
</tr>
<tr>
<td>12-12-12-12/8-0.1-5-2-3-3</td>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-3-2-3-2</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.08-5-2-3-3</td>
<td>1</td>
<td>2</td>
<td>9*</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-8-2-3-5</td>
<td>1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-1-2-3</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-2-3-3</td>
<td>1</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-3-4-3</td>
<td>1</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.2-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/24-0.1-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>15-15-15-15/8-0.1-5-2-3-3</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-8-2-3-5</td>
<td>1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.08-5-2-3-3</td>
<td>1</td>
<td>2</td>
<td>5*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-3-2-3-2</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-1-2-3</td>
<td>1</td>
<td>3</td>
<td>5*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-2-3-3</td>
<td>1</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-3-5-3</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.2-5-2-3-3</td>
<td>1</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/24-0.1-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 10 LRHeur2 results for medium (MLC size $12 \times 12$) problems, with \texttt{cplex.tilim} = 900 for subroutines and \texttt{cplex.tilim} = 1200 for fixed heuristic. No feasible solution for any of the large (MLC size $15 \times 15$) problems could be found after 20 iterations. * indicate instances where Cplex stopped due to no memory left (hence total number of iterations $< 20$).

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th># iterations</th>
<th>Objective values</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># sol's</td>
<td>1st sol</td>
<td>Best sol</td>
</tr>
<tr>
<td>12-12-12-12/8-0.1-5-2-3-3</td>
<td>1</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-3-2-3-2</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12-12-12-12/16-0.08-5-2-3-3</td>
<td>1</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-8-2-3-5</td>
<td>1</td>
<td>1</td>
<td>2*</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-1-2-3</td>
<td>2</td>
<td>3</td>
<td>5*</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>2*</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-3-4-3</td>
<td>1</td>
<td>1</td>
<td>19*</td>
</tr>
<tr>
<td>12-12-12-12/16-0.2-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>2*</td>
</tr>
<tr>
<td>12-12-12-12/24-0.1-5-2-3-3</td>
<td>1</td>
<td>1</td>
<td>10*</td>
</tr>
</tbody>
</table>
Table 11: CentHeur results for medium (MLC size $12 \times 12$) and large (MLC size $15 \times 15$) problems, with `cplex.tilim = 900` for subroutines and `cplex.tilim = 1200` for fixed heuristic. * indicate instances where Cplex stopped due to no memory left (hence total number of iterations < $|K|$).

<table>
<thead>
<tr>
<th>Problem Instance</th>
<th># iterations</th>
<th>Objective values</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># iterations</td>
<td>1st sol</td>
<td>Best sol</td>
</tr>
<tr>
<td>12-12-12-12/8-0.1-5-2-3-3</td>
<td>2</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-3-2-3-2</td>
<td>3</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.08-5-2-3-3</td>
<td>3</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-8-2-3-5</td>
<td>3</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-1-2-3</td>
<td>2</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-2-3-3</td>
<td>2</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.1-5-3-4-3</td>
<td>2</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/16-0.2-5-2-3-3</td>
<td>2</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>12-12-12-12/24-0.1-5-2-3-3</td>
<td>3</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>15-15-15-15/8-0.1-5-2-3-3</td>
<td>2</td>
<td>6</td>
<td>7*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-8-2-3-5</td>
<td>2</td>
<td>6</td>
<td>7*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.08-5-2-3-3</td>
<td>2</td>
<td>9</td>
<td>10*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-3-2-3-2</td>
<td>2</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-1-2-3</td>
<td>2</td>
<td>12</td>
<td>14*</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-3-3-3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>15-15-15-15/16-0.1-5-3-5-3</td>
<td>3</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/16-0.2-5-2-3-3</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>15-15-15-15/24-0.1-5-2-3-3</td>
<td>4</td>
<td>7</td>
<td>24</td>
</tr>
</tbody>
</table>