A Note on Improving the Performance of Approximation Algorithms for Radiation Therapy

Therese Biedl* Stephane Durocher† Holger H. Hoos‡
Shuang Luan§ Jared Saia¶ Maxwell Young†

Abstract

The segment minimization problem consists of representing an integer matrix as the sum of the fewest number of integer matrices each of which have the property that the non-zeroes in each row are consecutive. This has direct applications to an effective form of cancer treatment. Using several insights, we extend previous results to obtain constant-factor improvements in the approximation guarantees. We show that these improvements yield better performance by providing an experimental evaluation of all known approximation algorithms using both synthetic and real-world clinical data. Our algorithms are superior for 76% of instances and we argue for their utility alongside the heuristic approaches used in practice.

1 Introduction

Intensity-modulated radiation therapy (IMRT) is an effective form of cancer treatment in which the region to be treated is discretized into a grid. A treatment plan specifies the amount of radiation to be delivered to the area corresponding to each grid cell. A device called a multileaf collimator (MLC) is used to administer treatment in several steps. In each step, two banks of metal leaves in the MLC are positioned to cover certain portions of the body surface, while the exposed portions are subjected to a specific amount of radiation.

A treatment plan is represented as an $m \times n$ *intensity matrix* T of non-negative integer values, whose entries represent the amount of radiation to be delivered to the corresponding grid cells. The MLC leaves partially cover rows of T; for each row i there are two leaves, one which slides inward from the left and one which slides inward from the right. After each step, the amount of radiation applied in that step (this can differ per step) is subtracted from each entry of T that is exposed. The treatment is complete when all entries are 0. *In many cases, the number of segments does not have significant bearing on whether overdosing/underdosing occurs; therefore, approximation algorithms are suitable.* Setting leaf positions in each step requires time. We aim to minimize the number of steps as this increases patient throughput and reduces the procedure cost.

Formally, a *segment* is a matrix S such that non-zeroes in each row of S are consecutive, and all non-zero entries of S are the same integer, which we call the *segment-value*. A *segmentation* of T is a set of segment matrices that sum to T, and we call the cardinality of such a set the *size* of that segmentation. The *segmentation problem* is, given an intensity matrix T, to find a minimum-size segmentation of T.

Related Work: The segmentation problem is known to be NP-complete, even for a single row [3], and APX-complete [4]. A number of heuristics are known (see [2, 3, 11, 16] and references therein). Approaches for obtaining optimal solutions also exist (see [8, 12] and references therein); these approaches do not necessarily terminate in polynomial time. Bansal *et al.* [4] provide a $24/13$-approximation algorithm for the single-row problem. Most relevant to our current work, Luan *et al.* [13] give two approximation algorithms for the full $m \times n$ problem; however, they do not confirm the performance of their algorithms with experiments. Finally, we note that other important metrics for treatment planning exist, such as total irradiation time (see [1, 7, 12])

Our Contributions: Luan et al. [13] made two observations: (1) T can be decomposed into a particular set $\mathcal{P}$ of 0/1 matrices where the segmentation size of each $p \in \mathcal{P}$ can be related to the optimal segmentation size of T, and (2) segmentations for the single-row problem can be used to obtain good segmentations for the full-matrix problem. By exploiting these properties, they obtained two algorithms with respective approximation factors of $1 + \log_2 h$ and $2(1 + \log_2 D)$ where h is the largest value in T, and D is roughly the largest difference between consecutive row elements. Throughout, $\log_b x$ denotes $\lceil \log_b x \rceil$. Our first contribution is:

*David R. Cheriton School of Computer Science, University of Waterloo, ON, Canada,{biedl, m22young}@uwaterloo.ca
†Department of Computer Science, University of Manitoba, MB, Canada, durocher@cs.umanitoba.ca
‡Department of Computer Science, University of British Columbia, BC, Canada, hoos@cs.ubc.ca
§Department of Computer Science, University of New Mexico, NM, USA,{sluan, saia}@cs.unm.edu

- We extend the ideas of [13] to achieve three fast algorithms with approximation factors of (roughly) $\frac{3}{2} \cdot (1 + \log_3 h)$, $\frac{11}{6} \cdot (1 + \log_4 h)$ and $(24/13) \log D$. Since $\frac{11}{6} \log_4(h) < \frac{3}{2} \log_3(h) < \log_2(h)$, for sufficiently large OPT and h, our first two algorithms improve on previous work by a factor of ≈ 1.057 and $\frac{12}{11}$, respectively, while our third algorithm improves by a factor of $13/12$.

While admittedly these improvements are not large, the hope is that they translate into improved performance in practice. Previous approximation algorithms have not been tested; therefore, our second contribution is:

- We provide the first experimental evaluation of known approximation algorithms for the full segmentation problem, using both synthetic and real-world clinical data. Our approximation improvements yield significant performance gains. Together, our new algorithms are superior for 76% of test instances.

We remark that our experimental evaluation has practical value. While newer approaches in radiation therapy exist [14], the delivery method as described in this paper is the mainstream in current clinics and will likely stay because of its simplicity and less machine wear and tear. In current MLCs, segment minimization is performed by heuristics available in commercial software such as the CORVUS system manufactured by the NOMOS Corporation [10]. However, heuristics do not offer solution-quality guarantees and the run-time for exact methods can be prohibitively high. Instead, fast approximation algorithms can be used in parallel with heuristics to catch poor-quality solutions. Finally, we expect that as intensity matrixes become larger, approximation algorithms will become increasingly useful due to the high running time of exact methods.

2 Improved Approximation Algorithms

Let $T = (T[i, j])$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ be the target-matrix. Define a *marker* as an index j for which $T[i, j-1] \neq T[i, j]$, or $j = 1$ and $T[i, 1] \neq 0$, or $j = n + 1$ and $T[i, n] \neq 0$ (alternatively, one can imagine an additional column of 0s on the left and the right of T). Let ρ^i be the number of markers in row i of T, and let $\rho = \max_{\text{All rows } i}\{\rho^i\}$, i.e. the number of markers in the row of T having the most markers over all rows. We reiterate the following observation noted in [13]: $\rho \leq 2 \cdot OPT$ where OPT is the size of a minimal segmentation of T. The first approximation algorithm given by Luan et al. [13] works as follows. Split the intensity matrix T into matrices $P_0, \ldots, P_k$ such that $T = \sum_{\ell=0}^{k} 2^\ell \cdot P_\ell$ where $k = \log_2 h$ and each P_ℓ is a $0/1$-matrix. A segmentation for T can then be obtained by taking segmentations of each P_ℓ, multiplying their values by 2^ℓ, and taking their union. Since each P_ℓ is a 0/1-matrix, an optimal segmentation of it can be found easily, and an approximation bound of $1 + \log_2 h$ holds.

FIRST IMPROVED ALGORITHM: We extend this approach by increasing the base to $b = 3, 4$, i.e. writing $T = \sum_{\ell=0}^{k} b^\ell \cdot P_\ell$. But this raises two crucial questions: Can we solve the segmentation problem in a matrix with values in $\{0, 1, \ldots, b-1\}$? And is the resulting segmentation a good approximation of the optimal segmentation? Resolving these questions is non-trivial and requires new techniques over those used in [13].

For $b = 3$, we wish to segment an intensity matrix P_ℓ that has all entries in $\{0, 1, 2\}$; we call this a $0/1/2$-*matrix*. Let ρ_ℓ^i denote the number of markers in the ith row of P_ℓ.

Lemma 1. *There exists a segmentation of row i of a $0/1/2$-matrix P_ℓ such that the number of 1-segments is at most $\frac{1}{2} \cdot \rho_\ell^i$, and the number of 2-segments is at most $\frac{1}{4} \cdot \rho_\ell^i + \frac{1}{2}$.*

Proof. We use induction on ρ_ℓ^i. The base case is where none of the cases for the induction can be applied; we treat this last. For the induction, we identify a subsequence of the row for which we can add segments, resulting in the removal of many markers. We detail this for the first of the cases in the induction step and illustrate them all in Figure 1:

1. Assume that the row contains a subsequence of the form 12^+1. We use regular expression notation: 12^+1 denotes an entry 1, followed by ≥ 1 entries 2, followed by an entry 1. Let s be a 1-segment that covers exactly the subsequence of 2s, and consider $P' = P - s$. Then P' has two fewer markers in the ith row (at the endpoints of s), and so by induction the ith row can be segmented using at most $\frac{1}{2} \cdot (\rho_\ell^i - 2)$ 1-segments, and $\frac{1}{4} \cdot (\rho_\ell^i - 2) + \frac{1}{2}$ 2-segments. Adding the 1-segment s yields the result.
2. If there exists a subsequence of the form 01^+0, then apply a 1-segment to the subsequence of 1s. This removes 2 markers, and adds a 1-segment, and no 2-segment to the inductively obtained segmentation.
3. If there exists a subsequence of the form $02^+1^+2^+0$, then similarly apply a 2-segment at the first subsequence of 2s, then two 1-segments to remove the remaining 1^+2^+. This removes 4 markers, and adds two 1-segments, and one 2-segment to the inductively obtained segmentation.
4. If there exist two subsequences of the form 02^+1^+0 or 01^+2^+0, then similarly apply one 1-segment to one subsequence of 2s, and one 2-segment to the other subsequence of 2s, then apply two 1-segments to the two remaining sequences of 1s. This removes 6 markers, and adds three 1-segments and one 2-segment to the inductively obtained segmentation.

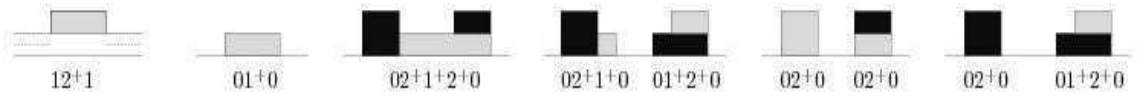

Figure 1: An illustration of cases (1) through (6) of the proof of Lemma 1.

5. If there exist two subsequences of the form 02^+0, then similarly apply one 2-segment to one of them, and two 1-segments to the other. This removes 4 markers, and adds two 1-segments and one 2-segment to the inductively obtained segmentation.
6. If there exists one subsequence of the form 02^+1^+0 or 01^+2^+0, and one subsequence of the form 02^+0, then apply one 2-segment to the subsequence 02^+0, and two one 1-segments to the other. This removes 5 markers, and adds two 1-segments and one 2-segment to the inductively obtained segmentation.

In all the above cases, we have removed at least 2 markers per 1-segment and at least 4 markers per 2-segment. Thus, counting only segments created and markers removed 1 thus far, we have at most $(1/2) \cdot p_\ell^i$ 1-segments and $(1/4) \cdot p_\ell^i$ 2-segments. All that remains to do is to consider any markers that are remaining.

Assume that none of the above cases can be applied (i.e., the base case) - we argue that now at most three markers are left. Let $0(1+2)^+0$ be a subsequence that has markers in it where $(1+2)$ denotes the presence of a 1 or a 2. Assume first the leftmost non-zero is a 1. Then the subsequence must contain a 2 somewhere (otherwise we're in case (2)), so it has the form $01^+2^+(1+2)^+0$. But after the 2s, no 1 can follow (otherwise we're in case (1)), so this subsequence has the form 01^+2^+0. Likewise, if the last non-zero is 1, then the subsequence has the form 02^+1^+0. If the first and last non-zero are 2, then the subsequence has the form 02^+0 (otherwise we're in case (1) or (3)).

If we had two subsequences $0(1+2)^+0$, then each would have the form 01^+2^+0 or 02^+1^+0 or 02^+0, and this is case (4), (5) or (6). So there is only one of them, and it has at most three markers. We can now eliminate either three remaining markers with a 1-segment and a 2-segment, or two remaining markers with a 2-segment. In either case, the bound on the number on the number of 1-segments used is still $(1/2) \cdot p_\ell^i$ and the 2-segments is used is $(1/4) \cdot p_\ell^i + 1/2$ (tight for the case of 02^+0). □

There exists a simple algorithm GREEDYROWPACKING for combining segmentations of *rows* of a matrix P_ℓ with values in $1, \ldots, b-1$ into a segmentation of the *whole* matrix P_ℓ. For each value $v \in \{1, \ldots, b-1\}$, check whether any segment in any row has value v. If so, remove a segment of value v from each row that has one. Combine these segments into one segment-matrix (also with value v), and add it to $\mathcal{S}$. Continue until all segments in all rows have been used in a segment-matrix. Clearly, if each row has at most n_i *i-segments* (i.e., segments with value i), this gives a segmentation of P_ℓ with at most n_i i-segments and $n_1 + \cdots + n_{b-1}$ segments in total. Using the segmentations of each row obtained with Lemma 1, and combining them with GREEDYROWPACKING, gives a segmentation $\mathcal{S}_\ell$ of each 0/1/2-matrix P_ℓ.

Theorem 1. *Assume $T = \sum_{\ell=0}^{k} 3^\ell P_\ell$, where $k = 1 + \log_3 h$ and h is the largest value in T, and each P_ℓ is a 0/1/2-matrix. Combining the above segmentations $\mathcal{S}_0, \ldots, \mathcal{S}_k$ for matrices $P_0, \ldots, P_k$ gives a segmentation $\mathcal{S}$ for T of size at most $\frac{3}{2} \cdot k \cdot OPT + \frac{1}{2} \cdot k$. This segmentation requires $O(m \cdot n \cdot \log h)$ time to find.*

Proof. Recall that the segmentation of row i of P_ℓ has at most $\frac{1}{2} \cdot \rho_\ell^i$ 1-segments and at most $\frac{1}{4} \cdot \rho_\ell^i + \frac{1}{2}$ 2-segments (Lemma 1). Let $\rho_\ell = \max_i \rho_\ell^i$ be the maximum number of markers within any row of P_ℓ. By algorithm GREEDYPACKING segmentation $\mathcal{S}_\ell$ of P_ℓ then has at most $\frac{1}{2} \cdot \rho_\ell$ 1-segments and at most $\frac{1}{4} \cdot \rho_\ell + \frac{1}{2}$ 2-segments. So $|\mathcal{S}_\ell| \leq \frac{3}{4} \cdot \rho_\ell + \frac{1}{2}$. Matrix P_ℓ can have a marker only if matrix T has a marker in the same location, so $\rho_\ell \leq \rho$ by [13]. Since $\rho \leq 2 \cdot OPT$, we put it all together to get:

$$|\mathcal{S}| = \sum_{\ell=0}^{k} |\mathcal{S}_\ell| \leq \sum_{\ell=0}^{k} \left(\frac{3}{4} \cdot \rho_\ell + \frac{1}{2} \right) \leq \sum_{\ell=0}^{k} \left(\frac{3}{4} \cdot 2 \cdot OPT + \frac{1}{2} \right) = \left(\frac{3}{2} \cdot OPT + \frac{1}{2} \right) \cdot (1 + \log_3 h) .$$

For each P_ℓ, this requires $O(m \cdot n)$ time; thus, the entire algorithm runs in time $O(m \cdot n \cdot \log h)$. □

By extending these ideas, we can prove a $\frac{11}{6} \cdot (1 + \log_4 h)$ approximation when $b = 4$. In theory, our approach could be taken further; however, the case-by-case analysis is involved (see [6] for details for $b = 4$).

Example 1. *Let* $T = \begin{pmatrix} 1 & 7 & 2 & 6 & 0 \\ 6 & 2 & 4 & 8 & 2 \\ 2 & 1 & 3 & 7 & 2 \end{pmatrix} = 3^0 \cdot \begin{pmatrix} 1 & 1 & 2 & 1 & 0 \\ 0 & 2 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 2 \end{pmatrix} + 3^1 \cdot \begin{pmatrix} 0 & 2 & 0 & 2 & 0 \\ 2 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 \end{pmatrix}$

By Lemma 1, the row segmentations for P_0 are as follows: row 1 belongs to Case 1 and uses segments $[0\ 0\ 1\ 0\ 0]$, $[1\ 1\ 1\ 1\ 0]$, *row 2 belongs to Case 3 and uses segments* $[0\ 0\ 1\ 1\ 1]$, $[0\ 0\ 0\ 1\ 1]$, $[0\ 2\ 0\ 0\ 0]$,

and row 3 belongs to Case 4 and uses segments $[1\ 0\ 0\ 0\ 0]$, $[0\ 0\ 0\ 0\ 2]$, $[1\ 1\ 0\ 0\ 0]$, $[0\ 0\ 0\ 1\ 0]$. *Applying* GREEDYROWPACKING *first to the value 1 and then to the value 2 gives the following segmentation for* P_0*:*

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

For P_1*, the row segments are: row 1 belongs to Case 5 uses* $[0\ 1\ 0\ 0\ 0]$, $[0\ 1\ 0\ 0\ 0]$, $[0\ 0\ 0\ 2\ 0]$*, row 2 belongs to Case 6 and uses* $[2\ 0\ 0\ 0\ 0]$, $[0\ 0\ 1\ 1\ 0]$, $[0\ 0\ 0\ 1\ 0]$*, and row 3 belongs to one of the remaining cases and uses* $[0\ 0\ 1\ 0\ 0]$, $[0\ 0\ 0\ 2\ 0]$*. Applying* GREEDYROWPACKING *first to the value 1 and then to the value 2 gives the following segmentation for* P_1*:*

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{pmatrix}$$

SECOND IMPROVED ALGORITHM: Luan et al. showed that for a single-row problem, if there is an α-approximate solution where all segment-values are at most M, then an $\alpha(\log M + 1)$-approximate segmentation of T can be found in polynomial time. Using this property, the authors gave a $2(\log_2 D + 1)$ approximation since any single-row problem has a 2-approximate solution. Here, the *row-difference* D is the maximum difference between consecutive row elements, or the maximum of the first and last entries in the row, whichever is larger. We extend this approach with two observations. First, *any* segmentation can be converted into a segmentation of the same size with values at most D. Secondly, values $\alpha < 2$ are known.

Theorem 2. *There exists an algorithm that, for an intensity matrix* T *with maximum row-difference* D*, finds a segmentation of size at most* $\frac{24}{13} \cdot (\log D + 1)OPT$ *and runs in* $O(m \cdot n^2 \cdot h \cdot \log D)$ *time.*

Proof. Let $\mathcal{S}$ be any segmentation of a single-row intensity matrix T with row-difference D. Modify $\mathcal{S}$ such that no two segments meet, i.e., if some segment ends at index i, then no segment starts at $i + 1$. This can always be done without increasing the number of segments (see [3]). Any segment S must then have value $v \leq D$, for if S ends at i, then $T[i + 1] = T[i] - v$ since no segment starts at $i + 1$. Therefore, we have a segmentation of T of size at most $|\mathcal{S}|$ where all segments have value at most D. Given that $M = D$, it follows from the observation of Luan *et al.*, that we have an $\alpha(\log D + 1)$-approximate segmentation. Since $\alpha \leq \frac{24}{13}$ is shown in [4], our result follows. The $\alpha \leq \frac{24}{13}$ algorithm of [4] runs in $O(n^2 \cdot h)$ time. Since we run this over all m rows of each of the $O(\log D)$ matrices specified in [13], the running time of our algorithm is $O(m \cdot n^2 \cdot h \cdot \log D)$. □

3 Performance Evaluation

In many areas, heuristics outperform approximation algorithms in practice. However, heuristics can become trapped in local optima and yield low-quality solutions. On the other hand, as demonstrated by previous work [12], and by our experiments, computing the optimum is computationally intensive and only possible with matrices of limited size and h values (hence the need for heuristics). Therefore, approximation algorithms can play an important role by providing a fast method for checking solution quality.

Preliminary work [5] shows that our algorithms frequently catch poor quality solutions yielded by the popular heuristic of Xia and Verhey [16] which is extensively used as a benchmark in the literature (see [6] for details). That said, we expect more recent and sophisticated heuristics to outperform our algorithms most of the time. *We stress that we do not aim to beat heuristics, only to provide an efficient safeguard against poor quality solutions*. There is a vast literature on segmentation heuristics and, while a comprehensive comparison involving these approaches would be valuable, such an undertaking is outside the scope of this work.

We implemented four algorithms in Java using roughly 3600 lines of code: (i) $\mathrm{A}_{b=2}$, the $(\log_2 h + 1)$ approximation algorithm of [13], (ii) $\mathrm{A}_{b=3}$, our $3/2 \cdot (\log_3 h + 1)$ approximation algorithm, (iii) $\mathrm{A}_{\alpha=2}$, the $2(\log D + 1)$ approximation algorithm of [13], (iv) $\mathrm{A}_{\alpha=24/13}$, our $24/13 \cdot (\log D + 1)$ which utilizes our implementations of algorithms from [4]. Experimentation with the $\frac{11}{6} \cdot (1 + \log_4 h)$ approximation algorithm did not yield improved performance over $\mathrm{A}_{b=3}$, so we do not consider it further. Finally, we compare against OPT using the recent state-of-the-art exact algorithms by Cambazard *et al.* [9] which are shown to compare favourably with another recent exact algorithm [15]. Cambazard *et al.* provide two exact algorithms: the shortest path constraint programming algorithm (CPSP) and the Branch-and-Price (BP) algorithm. We use the following test data:

- *Data Set I:* a real-world data set of 70 clinical intensity matrices from the Department of Radiation Oncology at the University of California at the San Francisco School of Medicine.

- *Data Set II:* a real-world data set of 22 clinical intensity matrices from prostate, brain, and head-neck cases from the Department of Radiation Oncology at the University of Maryland School of Medicine.
- *Data Set III:* a synthetic data set of 20 intensity matrices. Each matrix is obtained as follows: compute the sum of the probability density functions of 2-4 bivariate Gaussians generated from two independent standard univariate Gaussian distributions with amplitude h and the centers of the distributions are sampled uniformly at random. The distributions are discretized by adding as the value in the $m \times n$-grid the integer part of the corresponding function value.

First, we note that all three data sets are scaled so $h \leq 23$ - *this is necessary so that the exact algorithm of [9] (1) completes within a reasonable amount of time and (2) does not exceed the allotted memory*. Second, Data Set III allows for testing on intensity matrices where D values are relatively small compared to h. This allows us to investigate the effect of small D values on the performance of our approximation algorithms. Testing on matrices with small D values is also pertinent assuming higher precision MLCs can allow for more fine-grained intensity matrices.

ANALYSIS OF EXPERIMENTS: Tables 3-8 contain the results for each instance of our evaluation. All experiments were conducted on a $i7$ 2.8 GHz Intel CPU machine running a 64-bit version of Linux 10.04. At most 2 GB of RAM was utilized in any trial with the approximation algorithms while for OPT, 5 GB of RAM (Tripple Channel DDR3 2000) was allotted to the program. The time to compute the optimal solutions in Data Set I and II was negligible (under 0.5 CPU seconds using BP); however, only CPSP was able to solve the Data Set II instances and the run-time values are included in Table 8 since they were significant. Table 1 summarizes performance by enumerating the number of instances in which each algorithm outperformed all others (excluding OPT) with ties included.

	# Instances	$A_{b=2}$	$A_{b=3}$	$A_{\alpha=2}$	$A_{\alpha=24/13}$
Data Set I	70	24 (34.3%)	**55** (78.6%)	14 (20.0%)	18 (25.7%)
Data Set II	22	3 (13.6%)	9 (40.9%)	11 (50.0%)	**12** (54.5%)
Data Set III	20	0 (0.0%)	0 (0.0%)	11 (55.0%)	**18** (90.0%)

Table 1: The number of instances where each of approximation algorithms achieves the smallest segmentation with ties included. The largest value in each row is bolded.

Our Questions: In analyzing our results, we focus on three questions: (1) How do our improved algorithms compare against their older counterparts by Luan *et al.*? (2) How do the algorithms with an $O(\log h)$ approximation guarantee compare to those with an $O(\log D)$ approximation guarantee? (3) How do these approximation algorithms compare against the optimum solution?

Question 1: Table 1 shows that $A_{b=3}$ and $A_{\alpha=24/13}$ outperform on a larger number of instances than the algorithms of [13] in all three data sets for a total of 85 out of 112 instances (75.9%). In particular, $A_{b=3}$ ties or outperforms all other approximation algorithms in 55 out of the 70 instances (78.6%) in Data Set I while $A_{\alpha=24/13}$ ties or outperforms all other approximation algorithms in 12 out of the 22 instances (54.5%) in Data Set II and in 18 out of the 20 instances (90.0%) in Data Set III.

Given these positive results, we wish to know by *how much* we improve. We examine the number of segments required by an algorithm per instance and calculate the ratio of these two values; the median (Med.), minimum (Min.) and maximum (Max.) ratios over all instances is reported in Table 3. $A_{b=3}$ performs substantially better than $A_{b=2}$ overall judging by both the median values. In the case of $A_{\alpha=24/13}$ and $A_{\alpha=2}$, our gains are smaller, yet there is still an overall improvement on average.

	$A_{b=3}$ outperforms $A_{\alpha=\frac{24}{13}}$	$A_{\alpha=\frac{24}{13}}$ outperforms $A_{b=3}$	Ties
Data Set I	47 (67.1%)	6 (8.6%)	17 (24.3%)
Data Set II	7 (31.8%)	9 (40.9%)	6 (27.3%)
Data Set III	0 (0.0%)	20 (100.0%)	0 (0.0%)

Table 2: An instance-by-instance comparison of $A_{b=3}$ and $A_{\alpha=24/13}$.

Question 2: We contrast the performance of the $O(\log h)$ and $O(\log D)$ approximation algorithms. We restrict ourselves to a comparison of $A_{b=3}$ and $A_{\alpha=24/13}$ given the previous discussion. Table 2 provides the results of our comparison. We also calculate the average, median, minimum and maximum ratios on a per-instance basis of $A_{\alpha=24/13}$ over $A_{b=3}$ in Table 4.

We can tentatively draw some conclusions. When h and D are relatively equal, $A_{b=3}$ approximation generally yields superior performance in practice; this is certainly the case for Data Set I. However, as Data Set II illustrates, there are exceptions; neither algorithm is clearly superior here. For the case where D is significantly smaller than h, all statistics suggest that $A_{\alpha=24/13}$ yields substantially better solutions.

	Ratio of $A_{b=3}$ over $A_{b=2}$				Ratio of $A_{\alpha=\frac{24}{13}}$ over $A_{\alpha=2}$			
	Ave.	Med.	Min.	Max.	Ave.	Med.	Min.	Max.
Data Set I	0.9262	0.9161	0.6250	1.2000	0.9860	1.0000	0.7000	1.1667
Data Set II	0.9074	0.8990	0.5714	1.1429	0.9878	1.0000	0.8333	1.1818
Data Set III	0.9644	0.9687	1.0333	0.8958	0.9451	1.0000	1.0357	0.7407

Table 3: Average, median, minimum and maximum ratios measuring the extent of our improvements.

	Ave.	Med.	Min.	Max.
Data Set I	1.1650	1.1111	0.4444	1.8889
Data Set II	0.9810	1.0000	0.6250	1.2500
Data Set III	0.6289	0.6189	0.7333	0.5400

Table 4: Average, median, minimum and maximum ratios of $A_{\alpha=24/13}$ over $A_{b=3}$.

	Ave.	Med.	Worst	Best
Data Set I	1.1893	1.2000	1.5000	1.0000
Data Set II	1.3636	1.3636	1.6000	1.1250
Data Set III	1.1847	1.1425	1.5000	1.0385

Table 5: Statistics using the *best* approximations achieved by running all four algorithms.

Question 3: Finally, we compare against OPT. For each data set, all approximation algorithms are run on each instance and we take the best solution. Using these best solutions, average, median, worst and best values are reported in Table 5. We see that our heuristics are not far from the optimal solution in most cases.

Running Time: All approximation algorithms completed each instance within 0.01 CPU seconds on Data Set I, 0.02 CPU seconds on Data Set II, and 0.240 CPU seconds on Data Set III. In contrast, the running time for computing OPT can be significant. For Data Sets I & II, the algorithm of [9] performs superbly. However, for Data Set II, this is due to the fact that the h values are scaled to be small (clinical values were truncated at a single decimal point). By incorporating even another decimal place of clinical data, we found OPT (using CPSP) did not terminate within 24 hours. In fact, for any of our attempts with $h \geq 25$, the CPSP algorithm of [9] did not complete within 6 hours. Furthermore, attempts with the Branch-and-Price algorithm (both the light and normal versions) quickly terminated due to memory errors [9] for $h \geq 20$. These limitations are a concern for present-day real-world instances. From a forward-looking perspective, larger intensity matrices may become feasible as technology advances and this will greatly increase the running time and memory usage of exact algorithms. The impact of h-values and size is apparent in Data Set III where computing OPT for several cases required hundreds, or even thousands, of CPU seconds.

In conclusion, our theoretical and experimental results show that there are fast algorithms that yield segmentations that are provably not too far from the optimum and perform well in experiments. While more sophisticated heuristics likely outperform them, these algorithms are fast enough that they could be run in parallel with heuristics, thus providing a check on solution quality without significant overhead.

References

[1] R. Ahuja and H. Hamacher. A Network Flow Algorithm to Minimize Beam-On Time for Unconstrained Multileaf Collimator Problems in Cancer Radiation Therapy. Networks, 45, 36-41, 2005.

[2] D. Baatar, N. Boland and R. Johnston. A New Sequential Extraction Heuristic for Optimizing the Delivery of Cancer Radiation Treatment Using Multileaf Collimators. INFORMS Journal on Computing, 21(2), 224-241, 2009.

[3] D. Baatar, W. Hamacher, M. Ehrgott and G. J. Woeginger. Decomposition of Integer Matrices and Multileaf Collimator Sequencing. Discrete Applied Mathematics, 152(1-3), 6-34, 2005.

[4] N. Bansal, D. Coppersmith and B. Schieber. Minimizing Setup and Beam-On Times in Radiation Therapy. Proc. of APPROX, Lecture Notes in Computer Science, 4110, 27-38, 2006.

[5] T. Biedl, S. Durocher, H. H. Hoos, S. Luan, J. Saia and M. Young. Fixed-Parameter Tractability and Improved Approximations for Segment Minimization. Technical Report CS-2009-03, University of Waterloo, January, 2009.

[6] T. Biedl, S. Durocher, H. H. Hoos, S. Luan, J. Saia and M. Young. Fixed-Parameter Tractability and Improved Approximations for Segment Minimization. Preprint arXiv:0905.4930, 2009.

[7] N. Boland, H. Hamacher and F. Lenzen. Minimizing Beam-On Time in Cancer Radiation Treatment Using Multileaf Collimators. Networks, 43(4), 226240, 2003.

[8] S. Brand. The Sum-of-Increments Constraint in the Consecutive-Ones Matrix Decomposition Problem. Proc. of the Symposium on Applied Computing, 1417-1418, 2009.

[9] H. Cambazard, E. O'Mahony and B. O'Sullivan. Hybrid Models for the Multileaf Collimator Sequencing Problem. Proc. of the Intl. Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimiz. Problems, 56-70, 2010.

[10] CORVUS Treatment Planning System. `http://www.nomos.com/products_Cor.html`

[11] J. Dai and Y. Zhu. Minimizing the Number of Segments in a Delivery Sequence for Intensity-Modulated Radiation Therapy with a Multileaf Collimator. Medical Physics, 28(10), 2113-2120, 2001.

[12] T. Kalinowski. The Complexity of Minimizing the Number of Shape Matrices Subject to Minimal Beam-On Time in Multileaf Collimator Field Decomposition with Bounded Fluence. Discrete Applied Mathematics, 157(9), 2089-2104, 2009.

[13] S. Luan, J. Saia and M. Young. Approximation Algorithms for Minimizing Segments in Radiation Therapy. Information Processing Letters, 101, 239-244, 2007.

[14] K. Otto. Volumetric Modulated Arc Therapy: IMRT in a Single Gantry Arc Medical Physics 35(1), 310-317, 2008.

[15] Z. C. Taskin, J. C. Smith, H. E. Romeijn and J. F. Dempsey. Optimal Multileaf Collimator Leaf Sequencing in IMRT Treatment Planning. Operations Research, 58(3), 674-690, 2010.

[16] P. Xia and L. Verhey. Multileaf Collimator Leaf Sequencing Algorithm for Intensity Modulated Beams with Multiple Static Segments. Medical Physics, 25, 1424-1434, 1998.

Experimental Results

Instance	m	n	h	D	OPT	$A_{b=2}$	$A_{b=3}$	$A_{\alpha=2}$	$A_{\alpha=\frac{24}{13}}$
1	20	19	5	5	7	10	8	12	12
2	19	18	5	5	7	11	9	11	11
3	19	14	5	5	9	11	10	15	15
4	19	14	5	5	8	10	10	13	15
5	19	16	5	5	8	12	9	14	13
6	20	16	5	5	8	11	9	12	12
7	20	16	5	5	9	12	9	14	15
8	20	16	5	5	8	12	10	13	13
9	20	11	5	5	7	8	8	12	12
10	27	21	5	5	10	13	14	13	14
11	27	20	5	5	10	12	13	11	11
12	26	18	5	5	8	9	10	12	12
13	26	15	5	5	7	9	9	10	10
14	26	18	5	5	8	11	12	12	14
15	26	17	5	5	8	11	11	10	10
16	26	13	5	5	7	10	9	10	10
17	26	18	5	5	8	11	11	11	11
18	27	20	5	5	8	11	10	10	10
19	21	19	5	5	11	15	12	13	13
20	21	17	5	5	7	9	10	12	12
21	21	15	5	5	8	11	8	11	11
22	20	18	5	5	9	12	9	14	14
23	21	18	5	5	8	11	10	12	12
24	21	15	5	5	6	8	7	9	9
25	21	17	5	5	9	12	9	15	14
26	21	19	5	5	9	13	10	14	12
27	21	21	5	5	11	14	14	13	13
28	21	19	5	5	10	14	13	13	13
29	22	16	5	5	8	11	9	11	11
30	21	11	5	5	5	6	7	7	7
31	20	20	5	5	10	14	13	14	14
32	20	19	5	5	9	11	11	12	13
33	22	15	5	5	7	11	10	10	10
34	21	20	5	5	10	13	12	14	14
35	21	16	5	5	8	9	9	10	10
36	21	14	5	5	8	11	11	12	12
37	25	18	5	5	7	10	10	11	10
38	25	21	5	5	11	14	13	14	13
39	25	18	5	5	8	11	10	13	12
40	26	19	5	5	11	12	14	20	14
41	26	21	5	5	13	16	15	19	17
42	26	18	5	5	9	11	11	12	12
43	25	18	5	5	8	10	10	11	9
44	25	17	5	5	8	11	10	12	12
45	25	21	5	5	10	15	12	15	15
46	7	7	5	5	5	7	6	7	7
47	7	8	5	5	4	6	4	7	7
48	8	9	5	5	5	8	7	7	7
49	8	8	5	5	5	7	6	7	7
50	8	9	5	5	5	7	6	7	6
51	8	9	5	5	6	9	7	11	11
52	8	9	5	5	5	8	5	6	6
53	8	7	5	5	5	7	5	7	7
54	8	9	5	5	6	8	7	8	8
55	21	17	5	5	8	10	10	10	10
56	20	19	5	5	7	9	8	9	9
57	19	14	5	5	5	7	8	6	6
58	20	18	5	5	6	7	8	9	9
59	20	17	5	5	6	7	7	8	8
60	19	15	5	5	3	5	6	4	4
61	20	18	5	5	7	9	10	10	10
62	21	18	5	5	8	10	10	12	12
63	21	20	5	5	7	10	10	10	10
64	23	19	5	5	11	15	12	16	16
65	23	16	5	5	6	10	8	8	8
66	23	12	5	5	4	6	6	7	7
67	23	18	5	5	8	12	10	13	11
68	23	17	5	5	8	11	9	11	11
69	22	14	5	5	5	7	7	8	7
70	22	16	5	5	7	8	9	9	9

Table 6: Experimental results for Data Set II with the best result provided by the approximation algorithms underscored. The running time was negligible.

Instance	m	n	h	D	OPT	$A_{b=2}$	$A_{b=3}$	$A_{\alpha=2}$	$A_{\alpha=\frac{24}{13}}$
1	15	16	10	8	8	18	15	12	12
2	15	16	10	8	11	16	15	15	15
3	15	15	10	9	8	15	16	10	10
4	16	13	10	9	7	14	8	10	10
5	16	16	10	9	9	14	14	14	14
6	16	16	10	8	9	21	13	17	15
7	15	13	10	10	5	8	9	10	9
8	23	27	10	9	14	24	21	25	25
9	24	24	10	7	14	21	18	17	19
10	23	32	10	10	15	24	23	23	20
11	23	24	10	8	14	22	20	19	19
12	23	26	10	8	12	25	17	17	18
13	23	33	10	7	16	23	19	19	18
14	23	36	10	10	17	27	24	22	20
15	20	23	10	9	9	14	14	13	14
16	20	19	9	8	10	14	16	12	13
17	20	22	10	10	10	15	13	13	13
18	20	22	10	9	10	15	17	16	15
19	20	21	10	7	10	16	14	15	14
20	20	19	10	6	9	14	12	11	13
21	20	23	10	10	11	17	16	19	19
22	21	20	10	10	10	17	17	18	15

Table 7: Experimental results for Data Set II with the best result provided by the approximation algorithms underscored. The running time in CPU seconds for OPT is provided in parentheses.

Instance	m	n	h	D	OPT	$A_{b=2}$	$A_{b=3}$	$A_{\alpha=2}$	$A_{\alpha=\frac{24}{13}}$
1	58	49	20	2	20 (45)	37	34	27	22
2	50	68	20	2	29 (59)	54	54	31	31
3	54	61	20	2	25 (53)	47	46	28	28
4	51	39	20	2	17 (44)	32	29	20	20
5	44	71	20	2	30 (63)	55	55	33	33
6	56	43	21	2	18 (66)	33	30	24	22
7	40	67	21	2	30 (74)	57	54	37	32
8	52	59	21	2	25 (2359)	48	43	29	30
9	44	39	21	2	18 (61)	33	31	23	21
10	68	63	21	2	25 (4214)	48	43	28	29
11	68	47	22	2	18 (123)	32	31	20	20
12	44	69	22	2	32 (147)	59	59	36	32
13	59	37	22	2	18 (115)	30	31	19	19
14	54	59	22	2	26 (124)	49	50	27	27
15	65	60	22	2	18 (103)	32	31	19	19
16	41	50	23	2	25 (12691)	45	45	32	27
17	64	62	23	2	18 (192)	32	32	27	20
18	62	58	23	2	19 (208)	37	34	25	23
19	36	47	23	2	23 (1898)	43	41	28	26
20	59	38	23	2	18 (126)	31	32	19	19

Table 8: The experimental instances using Data Set III with the best result provided by the approximation algorithms underscored. The running time in CPU seconds (rounded to the nearest integer) for OPT using the CPSP algorithm of [9] is provided in parentheses. In several cases, the running time is significant.