**RC23945 (W0604-139) April 28, 2006**
**Computer Science**

# IBM Research Report

## Minimizing Setup and Beam-on Times in Radiation Therapy

**Nikhil Bansal, Don Coppersmith*, Baruch Schieber**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

*currently at IDA Center for Communications Research
85 Bunn Drive
Princeton, NJ 08540

# Minimizing Setup and Beam-On Times in Radiation Therapy

Nikhil Bansal[*] Don Coppersmith[†] Baruch Schieber[*]

April 18, 2006

**Abstract**

Radiation therapy is one of the commonly used cancer therapies. The radiation treatment poses a tuning problem: it needs to be effective enough to destroy the tumor, but it should maintain the functionality of the organs close to the tumor. Towards this goal the design of a radiation treatment has to be customized for each patient. Part of this design are intensity matrices that define the radiation dosage in a discretization of the beam head. To minimize the treatment time of a patient the beam-on time and the setup time need to be minimized. For a given row of the intensity matrix, the minimum beam-on time is equivalent to the minimum number of binary vectors with the consecutive "1"s property that sum to this row, and the minimum setup time is equivalent to the minimum number of *distinct* vectors in a set of binary vectors with the consecutive "1"s property that sum to this row. We give a simple linear time algorithm to compute the minimum beam-on time. We prove that the minimum setup time problem is APX-hard and give approximation algorithms for it using a duality property. For the general case, we give a $\frac{24}{13}$ approximation algorithm. For unimodal rows, we give a $\frac{9}{7}$ approximation algorithm. We also consider other variants for which better approximation ratios exist.

[*]IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, {nikhil,sbar}@us.ibm.com

[†]IDA Center for Communications Research, 805 Bunn Drive, Princeton, NJ 08540, don.coppersmith@idaccr.org. This work was done while at IBM T.J. Watson Research Center.

## 1 Introduction

Radiation therapy is one of the commonly used cancer therapies. It has been shown to be effective, especially in cases where the tumor is localized and metastases have not yet started to form. The radiation treatment poses a tuning problem: the radiation needs to be effective enough to destroy the tumor, but it should maintain the functionality of the organs close to the tumor (organs at risk). Towards this goal the design of a radiation treatment has to be customized for each patient. This design is done using computer tomography that detects the exact location of the tumor and the organs at risk.

The radiation is done using a linear accelerator positioned in a beam head that is positioned in a gantry that can be rotated around the patient. (See Fig. 1(a).) The treatment design specifies the angles of the radiation and its intensity. The angles are specified by a set of positions at which the gantry stops to release radiation. (In most models these are subset of 36 possible positions.) The desired intensity of the beam head at each gantry position is defined using an $m \times n$ *intensity matrix*, denoted $\mathcal{I}$, which corresponds to a discretization of the beam head into an $m \times n$ rectangular grid, with each of its entries called a *bixel*. The intensity matrix contains positive integral entries that determine the desired radiation dosage in each bixel.

The radiation generated by the accelerator is uniform. Thus, in order to achieve the varying intensity this radiation needs to be modulated. An emerging device for modulation is the *multileaf collimator (MLC)*. (See Fig. 1(b).) This device consists of a pair of leafs, a left leaf and a right leaf, for each row of the intensity matrix (often referred to as a *channel*). Consider a time $t$. If the left leaf of channel $i$ is positioned at $\ell$, for $0 \leq \ell \leq n$, and the right leaf is positioned at $r$, for $1 \leq r \leq n+1$, where $\ell \leq r$, then the radiation is blocked in bixels $(i,1),\ldots,(i,\ell)$ and $(i,r),\ldots,(i,n)$, and a uniform amount of radiation is delivered by bixels $(i,\ell+1),\ldots,(i,r-1)$. To achieve the desired intensity $\mathcal{I}(i,j)$, radiation needs to be delivered by bixel $(i,j)$ for $\mathcal{I}(i,j)$ time units.

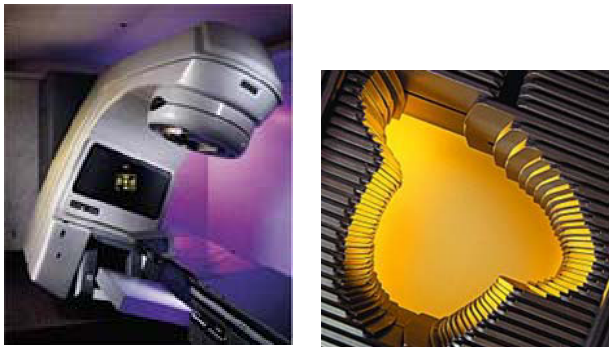

Figure 1: (a) a linear accelerator in a gantry (b) multileaf collimator both made by Varian

The positions of the multileaf collimator in the $m$ channels at time $t$ define a 0-1 $m \times n$ *shape matrix* $S_t$. A “0” entry in $S_t$ indicates a blocked bixel and a “1” entry indicates an active bixel. Note that the multileaf collimator function implies that each row in $S_t$ satisfies the *consecutive 1’s property*, that is, all the 1’s in a row are consecutive.

To achieve the desired radiation intensity in $T$ time units we need to find $T$ shape matrices $S_1,\ldots,S_T$ such that $\sum_{i=1}^{T} S_i = \mathcal{I}$ where the summation is done entry-wise. The *beam-on* time of a treatment is determined by $T$, the number of shape matrices used. The *setup* time is determined by the time it takes to calibrate the multileaf collimator. The setup time varies according to the technology used. In some models the leafs can be calibrated simultaneously and thus the setup time is determined by the number of *distinct* shape matrices (once the shape matrix is set up, it can be used any number of times without any overhead). We call this setup time the *multileaf setup* time. In other models each pair of leafs needs to be calibrated separately, and thus the setup time depends on the number of movements of each pair. We call this setup time the *leaf setup* time. Observe that the leaf setup time is determined by the number of distinct rows in the shape matrices. To minimize the treatment time for a patient a linear combination of the beam-on time and setup time needs to be minimized. (We note that in some models the setup time is not constant but depends on the specific "from" and "to" shape matrices, in which case a more complex term is needed to estimate the setup time accurately.)

In this paper we consider the minimization of beam-on and setup times separately. In many cases one of these terms is dominant and thus minimizing each one separately leads in practice to treatment times that are close to optimal. Our results assume *unconstrained* multileaf collimator whose position in each channel is independent. Note that in the unconstrained case each channel can be considered separately.

From a combinatorial point of view, our problem is that of decomposing a matrix or a vector into a small set of matrices or vectors of a special kind. In the beam-on problem the intensity matrix needs to be decomposed into a small set of 0-1 shape matrices, while in the leaf setup problem, each channel in the intensity matrix needs to be decomposed into a small set of *interval vectors*. An interval vector is a 0-$a$ vector that for some $a > 0$, that satisfies the *consecutive $a$'s property*.

**Results.** First, we note that the beam-on time minimization problem can be solved in linear time using a simple algorithm. Most of the paper deals with the leaf setup time minimization problem. We note that setup time seems to dominate the treatment time in many cases [9]. First, we prove a "duality" relation between the setup time minimization problem and maximum partitioning of certain type of vectors. We use this "duality" to prove that the leaf setup time minimization problem is APX-Hard, and then give several approximation algorithms for this problem depending on the structure of the channels in the intensity matrix. As we shall see, there is a trivial greedy algorithm that achieves an approximation guarantee of 2. Our main contribution is to give algorithms that achieve a guarantee better than 2. For a "unimodal" channel, i.e., a row in the intensity matrix whose only local minima are at its two ends, we give a $\frac{9}{7}$ approximation algorithm. For the general case we give a $\frac{24}{13}$ approximation algorithm. We also give two variants of our general algorithm which are shown to have better performance in specific cases depending on the number of local minima in the channel.

**Prior work.** Radiation treatment design was considered extensively in the medical physics literature. (See, e.g., [6, 5, 9] and references therein.) Previous algorithmic analysis of the setup and beam-on minimization problems is quite limited. Boland, Hamacher and Lenzen [7] gave a polynomial time algorithm for minimizing beam-on time. Their algorithm is based on an integer programming formulation that is shown to be solved using network flow in a graph of quadratic size. An algorithm with improved running time for the beam-on time minimization problem was given by Ahuja and Hamacher [1]. Their algorithm is also based on network flow but in a graph of linear size. Our algorithm is much simpler and more efficient. To the best of our knowledge this paper is the first to algorithmically analyze the setup time minimization problem. A heuristic for a constrained variant of the (multileaf) setup time minimization problem was given in [8].

The rest of the paper is organized as follows. In Section A (that was moved to the Appendix due to space constraints) we consider beam-on time minimization. In Section 2 we show the connection between the leaf setup time minimization problem and maximum partition of *Prefix Positive Zero Sum* vectors (to be defined later). In Section 3 we prove the hardness of the leaf setup time minimization problem. In Section 4 (and the Appendix) we present the approximation algorithms for this problem. Finally, in Section 5 we conclude with some open problems.

## 2 Setup time and vector partition

In this section we consider the setup time minimization problem. To formulate the problem we first define *interval vectors*. For an integer $b > 0$ we call a 0-$b$ vector $V = (v_1, \ldots, v_n)$ an *interval vector of height* $b$ if all the $b$'s are in consecutive positions. We use the triple $(\ell, r, b)$ to denote such a vector, where $v_\ell = \ldots = v_{r-1} = b$ and $v_i = 0$ everywhere else. We say that the vector *begins* at $\ell$ and *ends* at $r$. Recall that the (leaf) setup time minimization problem can be formulated as follows. Given a vector of non-negative integers $A = (a_1, \ldots, a_n)$ find a minimum set of *interval vectors* $(\ell_i, r_i, b_i)$ that sum to the input vector.

Consider an input vector $A = (a_1, \ldots, a_n)$. For notational convenience add one entry $(a_0)$ to the head of $A$ and another entry $(a_{n+1})$ to the tail of $A$ and let $a_0 = a_{n+1} = 0$. Define the *difference vector* of $A$, denoted $\Delta^A$, to be the vector $\Delta^A = (a_1 - a_0, a_2 - a_1, \ldots, a_{n+1} - a_n)$. From the definition of $\Delta^A$ it follows that for $i \in [1..n]$, the prefix sums $\sum_{j=1}^{i} \Delta^A(i) = a_i \geq 0$, and the sum $\sum_{j=1}^{n+1} \Delta^A(i) = a_{n+1} = 0$. We call such a vector *Prefix Positive Zero Sum (PPZS)* vector. We say that two vectors are "disjoint" if the index sets of their nonzero entries are disjoint. Consider the following maximum PPZS vector partitioning problem. Given a PPZS vector $\Delta$ find a maximum set of disjoint PPZS vectors that sum to $\Delta$. In this section we prove the "duality" between the above two problems as follows.

**Theorem 1** *The minimum setup time for a vector $A$ is $t$ if and only if the maximum partition of $\Delta^A$ is of size $z - t$, where $z$ is the number of nonzero entries in $\Delta^A$.*

To prove the theorem we need first to prove some properties of the setup time minimization problem. For an input vector $A$, we say that position $i$ is an *uptick* if $\Delta^A(i) > 0$ (i.e., $a_i - a_{i-1} > 0$). Similarly, position $i$ is a *downtick* if $\Delta^A(i) < 0$.

**Lemma 2** *There exists an optimal solution to the setup time minimization problem that consists only of interval vectors that begin at an uptick and end at a downtick.*

**Proof**: We show that any arbitrary solution can be transformed to one that has the required property without increasing the number of interval vectors used. Given a solution $\mathcal{S}$, suppose that $\mathcal{S}$ contains interval vectors that do not begin at an uptick. Consider such an interval vector $V = (\ell, r, b)$ with the minimum $\ell$. Thus $a_\ell \leq a_{\ell-1}$ and hence there must be at least one other interval vector $V' = (\ell', r', b')$ in $\mathcal{S}$ that ends at $r' = \ell$. As $\ell' < \ell$, by minimality of $\ell$, $V'$ must begin at an uptick. If $b = b'$ we could obtain a better solution by replacing $V$ and $V'$ by $(\ell', r, b)$. If $b < b'$, we replace $V$ and $V'$ by $(\ell', r', b' - b)$ and $(\ell', r, b)$, both of which begin at an uptick. If $b > b'$, we replace $V$ and $V'$ by $(\ell', r, b')$ and $\tilde{V} = (\ell, r, b - b')$. Note that the height of $\tilde{V}$ is strictly less than that of $V$ and hence applying this transformation repeatedly, we get a solution where all interval vectors begin at upticks. An identical argument implies that all interval vectors end at downticks.
$\square$

We call a solution that consists only of interval vectors that begin at upticks and end at downticks a *canonical* solution. We now show how to view any canonical solution to the setup problem as a graph, which will allow us to characterize the value of an optimum solution exactly. Let $I_u$ denote the index set of the upticks and $I_d$ denote the index set of the downticks.

Consider a canonical solution $\mathcal{S} = \{V_1, \dots, V_t\}$, where $V_i = (\ell_i, r_i, b_i)$ are interval vectors. Since all interval vectors begin at upticks and end at downticks if some position $j$ is an uptick, then the sum of heights of all interval vectors that begin at position $j$ is exactly equal to $\Delta^A(j)$. Similarly, if $j$ is a downtick then the sum of heights of all interval vectors that end at $j$ is exactly equal to $|\Delta^A(j)| = -\Delta^A(j)$.

Associate a weighted bipartite graph $G(\mathcal{S}) = (I_u, I_d, E)$ with a canonical solution $\mathcal{S}$ as follows. The vertex sets are $I_u$ and $I_d$, a vertex $i$ (corresponding to position $i$) has a positive weight equal to $|\Delta^A(i)|$. For each interval vector $V_j \in \mathcal{S}$, we have an edge $e_j = (\ell_j, r_j)$ with weight $b_j$.

We will show that each connected component of $G(\mathcal{S})$ corresponds to a PPZS vector. Observe that $G(\mathcal{S})$ has $t$ edges and $z$ vertices. As $\mathcal{S}$ is canonical, the total weight of edges incident at a vertex is equal to the weight of the vertex. Consider a connected component $(C_u, C_d, E_C)$ of $G(\mathcal{S})$. Notice that all the edges that are adjacent to vertices in $C = C_u \cup C_d$ are in $E_C$. It follows that every interval vector that begins at any of the upticks in $C_u$ must end at a downtick in $C_d$, and vice versa.

Consider the vector $\Delta$ defined by the entries of the difference vector $\Delta^A$ in the set of positions $C$; that is, $\Delta(i) = \Delta^A(i)$, for $i \in C$, and $\Delta(i) = 0$ otherwise. We claim that $\Delta$ is a PPZS vector. To see this consider any prefix sum $\sum_{j=1}^{i} \Delta(j)$. It is not difficult to see that the value of this sum is exactly the total height of all interval vectors that begin at an uptick in positions $C_u \cap [1..i]$ and end at a downtick in positions $C_d \cap [i+1..n+1]$. Thus, $\sum_{j=1}^{i} \Delta(j) \geq 0$, for $i \in [1..n]$, and $\sum_{j=1}^{n+1} \Delta(j) = 0$. It follows that a canonical solution $\mathcal{S}$ to the setup problem for vector $A$ induces a set of size $p$ of disjoint PPZS vectors that sum to $\Delta^A$, where $p$ equals the number of connected components in $G(\mathcal{S})$. Since $G(\mathcal{S})$ has $z$ vertices and $p$ connected components, the number of interval vectors in $\mathcal{S}$ (which equals the number of edges in $G(\mathcal{S})$) is at least $z - p$. We conclude that a setup time $t$ for vector $A$ implies a partition of $\Delta^A$ into at least $z - t$ disjoint PPZS vectors.

To complete the proof of Theorem 1 we show that a partition of $\Delta^A$ into $p$ disjoint PPZS vectors implies setup time at most $z - p$ for the vector $A$. Consider a PPZS vector $\Delta$ in the partition. Let $S^\Delta$ be the prefix sum vector of $\Delta$; that is $A^\Delta(0) = 0$, and for $i \in [1..n+1]$, $S^\Delta(i) = \sum_{j=1}^{i} \Delta(i)$. Since the sum of all the prefix sum vectors $S^\Delta$, for all PPZS vectors $\Delta$ in the partition, is exactly $A$. It is sufficient to prove the following lemma.

**Lemma 3** *Let $\Delta$ be a PPZS vector with $z$ nonzero entries and let $S^\Delta$ be its prefix sums vector. There are $z - 1$ interval vectors that sum to $S^\Delta$.*

**Proof**: We prove by induction on $z$. The base case is $z = 2$. In this case $\Delta$ consists of two nonzero entries at positions $\ell < r$. From the definition of a PPZS vector it follows that $\Delta(\ell) > 0$ and $\Delta(r) = -\Delta(\ell)$. Hence, the vector $S^\Delta$ is exactly the interval vector $(\ell, r, \Delta(\ell))$.

For the inductive step consider $z > 2$ assume that the lemma holds for $z' < z$. We simply show how to generate a single interval vector $V$ such that the difference vector $\Delta'$ of the vector $S^\Delta - V$ has no more than $z - 1$ nonzero entries. Consider a nonzero entry in $\Delta$ with the minimum absolute value. Let the index of this entry be $j$. If $\Delta(j) > 0$, then generate the interval vector $(j = \ell, r, \Delta(j))$, where $r > j$ is the minimum index such that $\Delta(r) < 0$. Similarly, if $\Delta(j) < 0$, then generate the interval vector $(\ell, j = r, -\Delta(j))$, where $\ell < j$ is the maximum index such that $\Delta(\ell) > 0$. Note that $S^\Delta(i) \geq |\Delta(j)|$, for $i \in [\ell, r-1]$. Thus $V$ is a valid interval vector. Also the difference vector of $S^\Delta - V$ has zero in position $j$ and also has zeroes in all positions $\Delta$ has zeroes. $\square$

## 3 APX Hardness of the setup minimization problem

**Theorem 4** *The setup minimization problem is APX-Hard even for vectors with entries polynomially bounded in $n$.*

**Proof**: The proof is by showing a gap preserving reduction from the 3-partition problem. The 3-partition problem is defined as follows. Given a threshold $B$ and $3m$ integers $p_1, \ldots, p_{3m}$ such that $\sum_{i=1}^{3m} p_i = mB$ and $B/4 < p_i < B/2$ for each $i$, is there a partition of the $3m$ integers into $m$ triples each of which sums exactly to $B$.

Petrank [14] showed that unless P=NP, there exists an $\epsilon > 0$ such that it is impossible to distinguish in polynomial time whether there are exactly $m$ or whether no more than $(1-\epsilon)m$ disjoint triples that sum to exactly $B$. This is true even for instances where $B$ is polynomially bounded in $m$.

Given an instance of the 3-partition problem we define an instance of the setup minimization problem. The input vector $A$ consists of $4m-1$ entries $A = (s_1, s_2, \ldots, s_{3m}, (m-1)B, (m-2)B, \ldots, B)$, where $s_i = \sum_{j=1}^{i} p_j$. Note that this instance is unimodal and each uptick has value $p_i$ for some $i$ and each downtick has value $B$. The corresponding difference vector with $4m$ entries is $\Delta^A = (p_1, \ldots, p_{3m}, -B, \ldots, -B)$.

Suppose that for any $\epsilon' > 0$ it is possible to distinguish in polynomial time whether the setup time for vector $A$ is $3m$ or at least $3m(1+\epsilon')$. This implies by Theorem 1 that it is possible to distinguish in polynomial time whether the disjoint partition of $\Delta^A$ into PPZS vectors is of size $4m - 3m = m$ or at most $4m - 3m(1+\epsilon') = m(1-3\epsilon')$. Note that each triple $p_{i_1}$, $p_{i_2}$ and $p_{i_3}$ that sums to $B$ corresponds to a PPZS vector with four nonzero entries in positions $i_1$, $i_2$ and $i_3$ and any one of the positions $3m+1, \ldots, 4m$. As each $p_i$ has size strictly between $B/4$ and $B/2$, each PPZS vector in the partition must have at least four nonzero entries: at least three positive and at least one negative (of value $-B$). Moreover, each PPZS vector that has more than four nonzero entries must have at least seven nonzero entries: at least five positive and at least two negative (of value $-B$).

Set $\epsilon' = \frac{1}{7}\epsilon$. If the partition of $\Delta^A$ is of size $m$ then clearly all the PPZS vectors have exactly four nonzero entries and thus there are $m$ disjoint triples that each sums to $B$, and vice versa. On the other hand, suppose there is a disjoint partition of $\Delta^A$ into $m(1-3\epsilon')$ or more PPZS vectors. Let $x$ denote the number of PPZS vectors with four nonzero entries and let $y$ denote the number of vectors of size at least seven. Thus we have than $x + y \geq m(1-3\epsilon')$ and that $4x + 7y \leq 4m$ which together imply that $x$ is at least $m(1-7\epsilon') = m(1-\epsilon)$, Thus, there are least $m(1-\epsilon)$ triples in the partition problem that sum to exactly $B$. □

## 4 Approximating the minimum setup time

In this section we describe several approximation algorithms for the minimum setup time for an input vector $A$. First, note that Lemma 3 implies a simple algorithm that finds $z-1$ interval vectors that sum to $A$, where $z$ is the number of nonzero entries in $\Delta^A$. This is since $\Delta^A$ is a PPZS vector. On the other hand, the maximum size of any partition of $\Delta^A$ into disjoint PPZS vectors is at most $\lfloor \frac{z}{2} \rfloor$, since each such vector must have at least two nonzero entries, implying that the minimum setup time is at least $z - \lfloor \frac{z}{2} \rfloor = \lceil \frac{z}{2} \rceil$. Thus a factor two approximation is trivial. Below, we show approximation algorithms with better ratios.

The basic idea for our algorithms will be the following. Note that a PPZS vector may be in a partition of $\Delta^A$ if its nonzero entries are a subset of the nonzero entries of $\Delta^A$. We call such a vector a *part* of $\Delta^A$. We compute the collection of all possible PPZS vectors with at most $y$ nonzero entries that are parts of $\Delta^A$. This can be done in time $O(n^y)$ and hence is polynomial for constant $y$. We then find a large (close to optimum) cardinality subset $S$ of disjoint PPZS vectors from the collection such that either $S$ is a partition of $\Delta^A$ or in case it is not a partition of $\Delta^A$, the vector $V$ consisting of all the nonzero entries of $\Delta^A$ that are not entries of vectors in $S$ is a PPZS vector. In this case $S \cup \{V\}$ is a disjoint partition of $\Delta^A$. Our algorithms depends on the "shape" of the vector $A$. Roughly speaking, if the vector $A$ has too many local minima, this complicates the task of finding the right set of PPZS vectors $S$ such that the remaining vector $V$ is also PPZS. We begin by considering the (simplest) unimodal case in which the only local minima of $A$ are at its two ends. Then, we consider the general case and some of its variants.

## 4.1 Unimodal input vectors

Consider a unimodal vector $A$ and let $\Delta^A$ be its corresponding difference vector. Note that $\Delta^A$ consists of a block of nonnegative entries followed by a block of non-positive entries. A useful property of such difference vectors is the following.

**Lemma 5** *Consider a PPZS vector $\Delta$ that consists of a block of nonnegative entries followed by a block of non-positive entries. Let $S$ be any set of disjoint PPZS vectors, each of which is a part of $\Delta^A$. The vector $V$ consisting of all the nonzero entries of $\Delta^A$ that are not entries of vectors in $S$ is a PPZS vector.*

**Proof**: Note that the vector $V$ also consists of a block of nonnegative entries followed by a block of non-positive entries. Also, the sum of all these entries is zero since the sum of all entries of the vectors in $S$ is zero. Hence, each prefix sum must be nonnegative and $V$ is a PPZS vector. $\square$

It follows that in order to find a good approximation we need to find a large cardinality set of disjoint PPZS vectors, each of which is a part of $\Delta^A$. We will find such a set consisting only of PPZS vectors with at most $y$ nonzero entries. Identify each such vector with the set of indices of its nonzero entries. Then the problem is reduced to the following set packing problem: Given a collection $C = \{S_1, S_2, \ldots\}$ of sets where each set $S_i$ has size at most $y$, find a maximum cardinality sub-collection $C' \subseteq C$ of disjoint sets.

The best known approximation algorithm for this set packing problem is an elegant local search based algorithm due to Hurkens and Schrijver [11] which achieves an approximation ratio of $y/2$.

Our algorithm for unimodal vectors is the following: Compute the set $\mathcal{S}_y$ of all PPZS vectors with at most $y$ nonzero entries that are part of $\Delta^A$, for $y = 2, 3$ and $4$. Apply the algorithm of [11] to find a subset $\mathcal{C}_y \subseteq \mathcal{S}_y$ of disjoint vectors, for each $y = 2, 3$ and $4$ and choose the subset $\mathcal{C}_y$ with the maximum cardinality.

**Theorem 6** *The algorithm described above is a $\frac{9}{7}$ approximation algorithm for the minimum setup time problem for unimodal input vectors.*

**Proof**: Let $Opt$ denote the optimum setup time for input vector $A$. By Theorem 1, $\Delta^A$ can be partitioned into $z - Opt$ PPZS vectors, where $z$ is the number of nonzero entries in $\Delta^A$. Let $n_k$ denote the number of PPZS vectors with $k$ nonzero entries in the partition defined by $Opt$. By definition $n_i$ satisfies

$$\sum_{i>1} i \cdot n_i = z \qquad \text{and} \qquad Opt = \sum_{i>1} (i-1) \cdot n_i \tag{1}$$

For a fixed choice of $y$, the algorithm of [11] guarantees that we can find at least $\sum_{i=2}^{y} \frac{2}{y} n_i$ disjoint vectors in $\mathcal{S}_y$, and hence a solution with setup cost at most $\sum_{i=2}^{y} (i - 2/y) \cdot n_i + \sum_{i>y} i \cdot n_i$. To show that our algorithm is a $\frac{9}{7}$ approximation, it suffices to show that for any choice of $n_2, n_3, \ldots$ that satisfies condition (1), the inequality

$$\sum_{i=2}^{y}(i - \frac{2}{y}) \cdot n_i + \sum_{i>y} i \cdot n_i \leq \frac{9}{7}\sum_{i\geq 2}(i-1)n_i$$

holds for at least one of $y = 2$, $y = 3$ or $y = 4$.

Without loss of generality, we can assume that $n_i = 0$ for $i \geq 5$, because $Opt$ incurs a setup time of at least $i - 1$ for any PPZS vector with $i$ nonzero entries while our algorithm incurs a setup time of at most $i$. As $\frac{5}{4} \leq \frac{9}{7}$, an instance where $n_i > 0$ for $i \geq 5$ can only improve the approximation ratio. Thus it suffices to show that one of the following inequalities holds. (The first corresponds to $y = 2$, the second to $y = 3$ and the third to $y = 4$.)

$$\begin{aligned} n_2 + 3n_3 + 4n_4 &\leq \frac{9}{7}(n_2 + 2n_3 + 3n_4) \\ \frac{4}{3}n_2 + \frac{7}{3}n_3 + 4n_4 &\leq \frac{9}{7}(n_2 + 2n_3 + 3n_4) \\ \frac{3}{2}n_2 + \frac{5}{2}n_3 + \frac{7}{2}n_4 &\leq \frac{9}{7}(n_2 + 2n_3 + 3n_4) \end{aligned}$$

Multiplying the first inequality by 2, second by 3 and third by 2 and summing each side of the resulting inequalities gives an identity which implies that one of these inequalities always holds for any $n_2, n_3$ and $n_4$. □

It is easily seen that the analysis for the above algorithm is tight by considering an instance where the optimum solution has $n_2 = 2(n+1)/11$, $n_3 = (n+1)/11$ and $n_4 = (n+1)/11$. As $2n_2+3n_2+4n_4 = n+1$, this is a valid choice of $n_i$ and the optimum setup time is $n+1-n_2-n_3-n_4 = 7(n+1)/11$. Our algorithm on the other hand, finds $n_2 = 2(n+1)/11$ vectors for $y = 2$, $2/3 \cdot (n_2 + n_3) = 2(n+1)/11$ vectors for $y = 3$, and $(n_2 + n_3 + n_4)/2 = 2(n+1)/11$ vectors for $y = 4$. In either case, our solution has setup time equal to $(n+1) - 2(n+1)/11 = 9(n+1)/11$.

## 4.2 Arbitrary input vectors

For unimodal input vectors we used the key fact (Lemma 5) that any collection of disjoint PPZS vectors that are parts of $\Delta^A$ can be used in the solution. However, this is not true in general. Consider for example when $A = (10, 5, 10)$ and hence $\Delta^A = (10, -5, 5, -10)$. In this case, the PPZS vector $S = (10, 0, 0, -10)$ is a valid part of $\Delta^A$. However we cannot choose this in the solution, because $V = \Delta^A - S = (0, -5, 5, 0)$ which is not a valid PPZS vector (the second prefix sum is negative).

Given an input vector $A$ to the setup minimization problem, let $z$ denote the number of non-zero entries in $\Delta^A$. We say that position $i$ is a *local minima* if $\Delta^A(i) < 0$ and $\Delta^A(j) > 0$ where $j$ is the smallest index greater than $i$ such that $\Delta^A(j)$ is non-zero. Note that the number of local minima can be no more than $z/2$. We will show the following three results:

1. We give a $3/(2-\epsilon)$ approximation when the number of local minima is no more than $\epsilon z$.

2. We improve this guarantee slightly when $\epsilon$ is $o(1/\log z)$. In particular, we give a $(11 + 9e^{-2} - 2e^{-3})/(8 + 6e^{-2} - 2e^{-3}) \approx 1.391$ approximation when the number of local minima is $o(z/\log z)$.

3. Finally, we give a $24/13 \approx 1.846$ approximation in the general case in which the number of local minima could be arbitrary.

Recall that our goal is to find a large cardinality set of PPZS vectors that are part of $\Delta^A$, such that (1) no two vectors share a nonzero position, and (2) the vector given by subtracting all the prefix sum vectors of these vectors from $A$ does not have any negative entries. We call property (1) *the independence property* and property (2) *the packing property*. Note that such a set of cardinality $p$ induces a solution of size $z - p - 1$, where $z$ is the number of nonzero entries in $\Delta^A$.

Our first and the second algorithms above are based on rounding the solution of a certain linear program (LP) that models the properties above. We also show certain structural properties of these linear programs which are essential to our rounding scheme. For lack of space we defer the description of these algorithms to Section B in the Appendix.

We describe here the algorithm for the general case, where the number of local minima could be arbitrary.

### 4.2.1 The algorithm for the general case:

Our main idea is the following. We will only be interested in PPZS vectors of size 2. Note that if some fixed optimum solution $Opt$ does not use size 2 vectors, then the setup time is at least 2z/3 and hence we trivially have a 3/2 approximation. Thus we will focus on the case when $Opt$ uses at lot of size 2 vectors (in particular it chooses close to $z/2$ such vectors). Our goal then will be to obtain a large subset of such vectors that simultaneously satisfies the independence and the packing properties.

We do not know how to find such a set of size 2 PPZS vectors directly and hence adopt an indirect approach. Recall that each such vector $\Delta$ with non-zeroes in positions $\ell$ and $r$ (for $\ell < r$) corresponds to an interval vector $(\ell, r, \Delta(\ell))$ in the solution. Call such interval vectors *candidate* interval vectors. We show that there exists a certain “minimal” set of independent interval vectors $\mathcal{R}_2$, such that any feasible collection of candidate interval vectors can be transformed (without any loss) into one that only uses vectors from $\mathcal{R}_2$. Since $\mathcal{R}_2$ only contains independent vectors, $|\mathcal{R}_2| \leq z/2$. Now, if $Opt$ uses close to $z/2$ vectors in its solution, then it must have discarded very few vectors from $\mathcal{R}_2$. This allows us to use the known results for the generalized caching problem considered by [2, 3], and hence find a solution where number of discarded intervals is no more than a constant times that under $Opt$.

We now describe the details. The following lemma describes the properties of the set $\mathcal{R}_2$. Due to space constraints its proof appears in Section C of the Appendix.

**Lemma 7** *Given a vector $A$, there exists a set $\mathcal{R}_2$ of interval vectors that satisfies the following properties:*

1. *No two interval vectors in $\mathcal{R}_2$ begin or end at the same position (hence they are independent).*

2. *For any set $S$ of candidate interval vectors that satisfies the independence and packing properties, there exists another set $S' \subseteq \mathcal{R}_2$, such that $|S'| \geq |S|$ and $S'$ satisfies the packing property. (Since $S' \subseteq \mathcal{R}_2$ it also satisfies the independence property.)*

*The set $\mathcal{R}_2$ is of linear size and can be obtained in linear time.*

To find a large subset of $\mathcal{R}_2^h$ that satisfies the independence and packing properties we use known results for the general caching problem defined below.

In the general caching problem (with unit profit) we are given a vector $A = (a_1, \ldots, a_n)$ where $a_i$ denotes the amount of cache available at time $i$. There is collection of tasks $\mathcal{T} = \{T_1, \ldots, T_m\}$, where each task $T_i$, specified by $(\ell_i, r_i, h_i)$, requires $h_i$ units of cache during the interval $[\ell_i, r_i - 1]$. A set of tasks is feasible if the required cache size for these tasks does not exceed the available cache size at any time. The goal is to find a feasible collection of tasks such that the total number of tasks not included in the collection is minimized.

This problem was first considered by [2] who gave a logarithmic approximation for the problem. Later [3] gave an algorithm with an approximation ratio of 4. Note that the approximation ratio is for the number of tasks excluded rather than included in the collection.

Our algorithm is as follows. We construct an instance of the general caching problem where $\mathcal{T} = \mathcal{R}_2$. We apply the algorithm of [3] to this instance and obtain a collection $S$ of interval vectors that satisfy the independence and packing properties. We use this to construct a solution with setup time at most $z - |S| - 1$, where $z$ is the number of nonzero entries in $\Delta^A$.

**Theorem 8** *The algorithm stated above is a* $\frac{24}{13} \approx 1.846$ *approximation algorithm for the setup minimization problem.*

**Proof**: Consider an optimal canonical solution $\mathcal{S}$ for the setup problem. Let $\mathcal{S}_2$ be the set of interval vectors in the solution that begin at an uptick and end at a downtick of the same value. By the definition of $n_2$, we have $n_2 = |\mathcal{S}_2|$. By Lemma 7 the set $\mathcal{R}_2$ contains a subset of size $n_2$ that satisfies the independence and packing properties. Thus, the solution returned by the algorithm of [3] for the general caching problem contains at least $\max\{0, |\mathcal{R}_2| - 4(|\mathcal{R}_2| - n_2)\}$ interval vectors. Moreover, since $\sum_{i \geq 2} i \cdot n_i = z$, we have that $\sum_{i \geq 3} n_i \leq (z - 2n_2)/3$. Since $Opt = \sum_{i \geq 2} (i-1) \cdot n_i$, we have $opt = \sum_{i \geq 2} i \cdot n_i - n_2 - \sum_{i \geq 3} n_i \geq z - n_2 - (z - 2n_2)/3 = (2z - n_2)/3$.

We consider two cases based on the magnitude of $\alpha = 4((|\mathcal{R}_2| - n_2)$. If $\alpha \geq |\mathcal{R}_2|$, or equivalently, $n_2 \leq 3|\mathcal{R}_2|/4 \leq 3z/8$, the approximation ratio is most

$$\frac{z}{Opt} \leq \frac{z}{(2z - n_2)/3} \leq \frac{3z}{2z - 3z/8} = \frac{24}{13}.$$

If $\alpha < |\mathcal{R}_2|$, let $\beta$ denote $|\mathcal{R}_2| - \alpha$. Then, the approximation ratio is at most

$$\frac{z - \beta}{(2z - n_2)/3} = \frac{3(z - \beta)}{2z - (\mathcal{R}_2 - \alpha/4)} = \frac{3(z - \beta)}{2z - \beta/4 - 3\mathcal{R}_2/4}$$

As $|\mathcal{R}_2| \leq z/2$ this is at most

$$\frac{24(z - \beta)}{13z - 2\beta}$$

which is clearly maximized when $\beta = 0$ and hence is at most $\frac{24}{13}$. □

## 5 Conclusions and open problems

In this paper we considered the beam-on time and setup time minimization problems in radiation therapy. We presented an efficient linear time algorithm for the beam-on time minimization problem. We proved that the setup time minimization problem is APX Hard, and gave approximation algorithms for the problems that are better than the naive 2 approximation for the problem.

The area still has a lot of open problems, such as maximizing the combination of beam-on and setup times, considering multileaf rather than leaf setup time, considering constrained shape matrices and more.

## References

[1] R.K. Ahuja and H.W. Hamacher. A network flow algorithm to minimize beam-on time for unconstrained multileaf collimator problems in cancer radiation therapy. *Networks,* 44 (2005), 36–41.

[2] S. Albers, S. Arora and S. Khanna. Page replacement for general caching problems. *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, 31–40, 1999.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM,* 48 (2001), 1069–1090.

[4] S. Bernstein. Theory of Probability. Moscow, 1927.

[5] A.L. Boyer. Use of MLC for intensity modulation. *Medical Physics*, 21 (1994), 1007.

[6] T.R. Bortfeld, D.L. Kahler, T.J. Waldron and A.L. Boyer. X-ray field compensation with multileaf collimators. *Int. Journal of radiation Oncology, Biology, Physics,* 28 (1994), 723–730.

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

[8] D.Z. Chen, X.S. Hu, S. Luan, C. Wang and X. Wu. Mountain reduction, block matching, and applications in intensity modulation radiation therapy. *Proc. 21st ACM Symp. on Computational Geometry,* 35–44, 2005.

[9] J. Dai and Y. Zhu. Minimizing the number of segments in a delivery sequence for intensity modulated radiation therapy with multileaf collimator. *Medical Physics,* 28 (2001), 2113–2120.

[10] H.N. Gabow, J.L. Bentley and R.E. Tarjan. Scaling and related techniques for geometry problems. *Proc. 16th ACM Symp. on Theory of Computing,* 135–143, 1984.

[11] C.A.J. Hurkens and A. Schrijver. On the size of systems of sets every $t$ of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics,* 2 (1989), 68–72.

[12] S. Kamath, S. Sahni, J. Palta and S. Ranka. Algorithms for optimal sequencing of dynamic multileaf collimators. *Physics in Medicine and Biology,* 49 (2004), 33–54.

[13] S. Kamath, S. Sahni, J. Palta, S. Ranka and J. Li. Optimal leaf sequencing with elimination of tongue-and-groove underdosage. *Physics in Medicine and Biology,* 49 (2004), N7–N19.

[14] E. Petrank. The hardness of approximation: gap location. *Computational Complexity,* 4 (1994), 133–157.

[15] J. Vuillemin. A Unifying Look at Data Structures. *Comm ACM,* 23(1980), 229–239.

# Appendix

## A Beam-on time minimization

The beam-on minimization problem is defined as follows. Given an intensity matrix $\mathcal{I}$ find a minimum set of 0-1 shape matrices $S_1, S_2 \ldots$ such that for every $i \in [1..m]$ and $j \in [1..n]$, we have $\sum_{x=1} S_x(i,j) = \mathcal{I}(i,j)$. Assuming unconstrained multileaf collimator the problem can be solved for each channel separately. The single channel problem is: given a vector of non-negative integers $A = (a_1, \ldots, a_n)$ find a minimum set of 0-1 vectors with the consecutive 1's property that sum to the input vector. We note that without loss of generality we may assume that all entries are positive since the problem can be solved separately for each consecutive sequence of positive entries.

To define our algorithm it is enough to specify for every input vector $A = (a_1, \ldots, a_n)$ a 0-1 vector, denoted $V_A$, that is in an optimal solution. In order to find a complete solution we can continue in the same manner; that is, find next the 0-1 vector that is in a solution for the input vector $A' = A - V_A = (a_1 - V_A(1), \ldots, a_n - V_A(n))$, then the vector $V_{A'-V_{A'}}$, and so forth until we reach the all zeroes vector.

All left to be done is to specify how to find $V_A$ given $A$. For convenience assume that $A$ has $n+1$ entries and that $a_{n+1} = 0$. Note that $V_A$ is defined by the half open interval $[L(V_A), R(V_A))$ of its consecutive "1"s, i.e., $L(V_A)$ and $R(V_A) - 1$ are the indices of the leftmost and rightmost "1"s in $V_A$. The following "greedy" algorithm finds $V_A$: $L(V_A)$ is the minimum index $i$ such that $a_i > 0$ and $R(V_A)$ is the minimum index $j \geq L(V_A)$ such that $a_j = 0$. The correctness follows from the following claim.

**Claim 9** *There is an optimal solution for $A$ that includes $V_A$.*

**Proof**: Note that any optimal solution for $A$ must include at least one 0-1 vector whose leftmost "1" is in position $L(V_A)$. Consider a solution that includes such a vector $U$ with the maximum number of 1's. We prove by contradiction that $U = V_A$. To obtain a contradiction assume that $R(U) \neq R(V_A)$. Clearly $R(U) < R(V_A)$. In this case there must be another vector $U'$ in the solution that has a "1" in position $R(U)$. Clearly, $L(U') > L(U)$ and $R(U') > R(U)$. However, in this case we can construct another optimal solution by replacing the pair of vectors $U$ and $U'$ by the pair $U \vee U'$ and $U \wedge U'$, where the $\vee$ and $\wedge$ operations are done entry-wise. However, $L(U \vee U') = L(U)$ and $R(U \vee U') = R(U') > R(U)$, contradicting our choice of the solution. $\square$

To compute the solution in linear time we use the Cartesian tree data structure. The Cartesian tree is a binary search tree proposed by Vuillemin [15] that is defined recursively for a vector $A = (a_1, \ldots, a_n)$ as follows. Each node $t$ of the tree corresponds to an entry $t(A)$ in $A$. The root corresponds to the minimum entry in $A$ (ties are broken arbitrarily). Suppose that $a_i$ is the minimum entry. The left subtree of the root is the Cartesian tree defined for $(a_1, \ldots, a_{i-1})$, and the right subtree is the Cartesian tree defined for $(a_{i+1}, \ldots, a_n)$. It is well known [10] that a Cartesian tree can be constructed in linear time. Associate an pair $(L(v), R(v))$ with each node of the tree where $L(v)$ is the index of the leftmost entry in the subtree rooted at $v$, and $R(v)$ is one plus the index of rightmost entry in the subtree. It is not difficult to see that the solution computed by our algorithm for the vector $A$ consists of the following 0-1 vectors defined by the nodes of the Cartesian tree. For the root node $t$ the solution contains $t(A)$ vectors $(L(t), R(t))$, for each non-root node $t$ the solution contains $t(A) - t'(A)$ vectors $(L(t), R(t))$, where $t'$ is the parent of $t$ in the tree.

We note that given a solution to the beam-on minimization problem, any feasible matching of the left endpoints in the solution to the right endpoints (i.e., a matching that always matches a left endpoint to a right endpoint with a larger of equal index) yields also a solution. This observation is useful in case there are side constraints that need to be satisfied by the solution.

## B LP based approximations

Let $\mathcal{S}_y$ denote the set of all PPZS vectors that are parts of $\Delta^A$ and have size at most $y$. Clearly, the cardinality of $\mathcal{S}_y$ is $O(n^y)$. We consider the following Integer Programming formulation to find a maximum cardinality subset of $\mathcal{S}_y$ that satisfies the independence and packing properties. We have a binary variable $x_i$ for each PPZS vector $\Delta^{V_i}$ in $\mathcal{S}_y$. Let $V_i(j)$ denote $j^{th}$ prefix sum of $\Delta^{V_i}$. Finally, let $c_i(j)$ be a constant which is 1 if $\Delta^{V_i}$ has a non-zero entry at $j$, and 0 otherwise. Consider the following formulation.

$$\max \sum_i x_i \qquad \text{subject to}$$

$$\sum_{\Delta^{V_i} \in \mathcal{S}_y} V_i(j) x_i \quad \leq \quad A(j) \qquad \forall 1 \leq j \leq n \tag{2}$$

$$\sum_{\Delta^{V_i} \in \mathcal{S}_y} c_i(j) x_i \quad \leq \quad 1 \qquad \forall 1 \leq j \leq n \tag{3}$$

$$x_i \quad \in \quad \{0,1\} \, \forall i \in [1, \ldots, |\mathcal{S}_y|] \tag{4}$$

Note that constraints 2 in the IP ensure that the subset of $\mathcal{S}_y$ satisfies the packing property, thus we call these constraints the packing constraints, and the constraints 3 ensure that the PPZS vectors are disjoint, thus we call them independence constraints. In the LP formulation we replace constraint 4 by $0 \leq x_i \leq 1$, for each $i \in [1, \ldots, |\mathcal{S}_y|]$.

### B.1 Simplifying the LP further

We now show how to reduce the number of packing constraints without affecting the quality of the solution.

**Lemma 10** *It suffices to consider packing constraints (2) only at positions $j$ that are a local minima in $A$.*

**Proof**: Consider an (infeasible) solution $x_i$, for $i \in [1, \ldots, |\mathcal{S}_y|]$. We show that if there is a position where the packing constraint is violated for this solution, then there is some local minima where the packing constraint is also violated.

Let $A^*$ denote the vector $\sum_i x_i V_i$, Let $j$ be some position that is not a local minima where $A^*(j) > A(j)$. Let $j_1 < j$ and $j_2 > j$ be the indices of the closest local minima to $j$. Suppose that $\Delta^A(j) > 0$, we show that the packing constraint is also violated at $j_1$. (If $\Delta^A(j) < 0$, a symmetric argument shows that the packing constraint is also violated at $j_2$.) As $\Delta^A(j) > 0$, and by definition of $j_1$, we have $\Delta^A(y) > 0$ for $y \in [j_1 + 1..j]$. Note that $A(j) = A(j_1) + \sum_{y=j_1+1}^{j} \Delta^A(y)$. Also the total height of the interval vectors in a canonical solution that begin in a position in $[j_1 + 1..j]$ is bounded by $\sum_{y=j_1+1}^{j} \Delta^A(y)$. Thus $A^*(j) \leq A^*(j_1) + \sum_{y=j_1+1}^{j} \Delta^A(y)$. Hence, $A^*(j) > A(j)$ implies $A^*(j_1) \geq A^*(j) - \sum_{y=j_1+1}^{j} \Delta^A(y) > A(j) - \sum_{y=j_1+1}^{j} \Delta^A(y) = A(j_1)$. □

We now describe our algorithms.

## B.2 At most $\epsilon z$ local minima

We will consider the LP described in Section B for PPZS vectors of size 2 (i.e. $\mathcal{S}_y = \mathcal{S}_2$). In this case, we can eliminate the independence constraints completely by using Lemma 7 and restricting the set of candidate PPZS vectors to $\mathcal{R}_2$. Observe that by Lemma 7, this leaves the solution unaffected.

Let $\mathcal{M}$ denote the positions of all local minima of $A$. By Lemma 10 it suffices to consider the following linear programming relaxation that we call LP2:

$$\max \sum_i x_i \qquad \text{subject to} \tag{5}$$

$$\sum_{\Delta^{V_i} \in \mathcal{R}_2} V_i(j) x_i \quad \leq \quad A(j) \qquad \forall j \in \mathcal{M} \tag{6}$$

$$0 \leq \quad x_i \quad \leq 1 \qquad \forall i \in [1, \ldots, |\mathcal{R}_2|] \tag{7}$$

As LP2 has at most $\mathcal{M}$ non-trivial constraints (6), any basic feasible solution to LP2 has at at most $|\mathcal{M}|$ variables that are set fractionally, i.e. their value is strictly between 0 and 1. Let $n_2$ denote the maximum number of vectors of size 2 that can be packed feasibly in any integral solution. This implies the following,

**Lemma 11** *A basic feasible solution to LP2 has at least $n_2 - |\mathcal{M}|$ variables integrally set to 1.*

**Theorem 12** *If $|\mathcal{M}| \leq \epsilon z$, the algorithm that just solves LP2 and chooses the interval vectors for which $x_i = 1$ is a $3/(2-\epsilon)$ approximation.*

**Proof**: Let $Opt$ be the value of an optimum solution for input vector $A$, and let $z$ be the number of non-zeros in $\Delta^A$. Then, $\sum_{i\geq 2} i \cdot n_i = z$ and $Opt = z - (\sum_{i\geq 2} n_i)$. Now the algorithm described above chooses $x = \max(0, n_2 - |\mathcal{M}|)$ interval vectors from $\mathcal{R}_2$, and thus has a setup time of $z - x$. Thus the approximation ratio is

$$\frac{z-x}{z - (\sum_{i\geq 2} n_i)}$$

As $\sum_{i\geq 3} n_i \leq (z - 2n_2)/3)$, the optimum can be lower bounded by $z - n_2 - (z - 2n_2)/3 = (2z - n_2)/3$. Finally, as $n_2 \leq x + |\mathcal{M}| \leq x + \epsilon z$, optimum is at least $((2-\epsilon)z - x)/3$. Thus the approximation ratio is at most

$$\frac{z-x}{((2-\epsilon)z - x)/3}$$

The above term is maximized when $x = 0$, and hence is at most $3/(2-\epsilon)$. $\square$

## B.3 $o(z/\log z)$ local minima

Consider $\mathcal{S}_3$, the set of all PPZS vectors $\Delta$ that have at most 3 non-zero entries and whose entries are a subset of $\Delta^A$. Let $\epsilon$ be an arbitrarily small constant. We call a PPZS vector $\Delta^{V_i} \in \mathcal{S}_3$ *bad* if there is some local minima $j \in \mathcal{M}$ such that $V_i(j)$, is greater than $\epsilon^2 A(j)/(10\log(z))$. Let $\mathcal{R}$ be the set obtained by removing all the bad PPZS vectors from $\mathcal{S}_3$.

Our algorithm is as follows:

1. We begin by solving the linear programming relaxation of the integer program defined by 2-4, with the set of PPZS vectors restricted to $\mathcal{R}$. Let $x^*$ denote the LP solution.

2. Let $\mathcal{R}^+$ denote the set of PPZS vectors $V_i$ for which $x_i^* > 0$. For each $i \in \mathcal{R}^+$, let $t_i$ denote the quantity $\frac{1}{x_i^*}\ln(\frac{1}{1-x_i^*(1-\epsilon)})$. We say that two vectors $i$ and $j$ are *neighbors* there is some position $k$ where both $c_i(k) = c_j(k) = 1$, i.e. if they share some uptick or downtick.

3. We round the solution as follows: We initialize each $x_i = 0$. For each $i \in \mathcal{R}^+$, we begin a Poisson process at time 0, that chooses $i$ (i.e. sets $x_i = 1$) with with rate $x_i^*$. That is, if $x_i = 0$ at time $t$, then the probability that $i$ is chosen during the infinitesimal time interval $(t, t+dt)$ is $x_i^* dt$.

   The process for $i$ continues until one the following event happens:

   (a) The PPZS vector $i$ gets chosen.

   (b) Some neighbor of $i$ gets chosen.

   (c) Time reaches $t_i$.

Note that though the above algorithm is stated in terms of time being continuous, it can be implemented in time polynomial in $z$ in a straightforward way. We now analyze the above algorithm. Let $S$ denote the solution obtained after rounding. Let $Opt$ denote the cost of some fixed optimum solution and let $n_2$ and $n_3$ be the number of PPZS vectors with 2 and 3 non-zero entries used by $Opt$. We can assume that $n_2 + n_3 = \Omega(z)$, since otherwise $Opt$ will have a setup time of at least $3z/4 - o(z)$ and hence even if $S = \emptyset$, we have a $4/3$ approximation (which is better than what we are aiming for). As at most $10 \log z/\epsilon^2$ PPZS vectors are bad for any local minima, at most $|\mathcal{M}| \cdot 10 \log z/\epsilon^2 = o(z)$ vectors are removed by restricting our attention to $\mathcal{R}$ instead of $\mathcal{S}_3$. Thus the value of the LP solution is at least $n_2 + n_3 - o(z)$ which is at least $(1-\epsilon)(n_2+n_3)$ as $n_2 + n_3 = \Omega(z)$.

Clearly, the rounding procedure ensures that if $i$ is chosen then, no neighbor of $i$ is chosen, and thus all PPZS vectors in $S$ will be disjoint. We first show that $S$ satisfies the packing constraints at the local minima (and hence at all $1 \leq j \leq n$, by Lemma 10) with high probability. Let $p_i$ denote the probability that $i$ is chosen in $S$. Since the process for $i$ definitely stops by time $t_i$, clearly $p_i \leq 1 - e^{-t_i x_i^*} = (1-\epsilon)x_i^*$.

Consider a particular local minima $k$. Let $Y(i)$ be the random variable that denotes the height contributed by $V_i$ to position $k$ in the solution $S$. Thus $Y(i) = V_i(k)$ with probability $p_i$ and 0 otherwise. Since $p_i$ is at most $(1-\epsilon)x_i^*$ it follows that $E[\sum_i Y(i)] \leq (1-\epsilon)A(k)$. We are interested in the upper bound the probability of the event that $\sum_i Y(i) > A(k)$. We use the following tail inequality due to Bernstein [4].

**Theorem 13 (Bernstein's Inequality)** *Let $X_1, \ldots, X_n$ be independent random variables with $X_i - E[X_i] \leq d$ for all $i \in \{1, \ldots, n\}$. Let $S = X_1 + \ldots + X_n$, and $t > 0$. Then, with $\sigma_i^2 = E[X_i^2] - E[X_i]^2$ we have that*

$$Pr[S - E[S] \geq t] \leq \exp\left(\frac{-t^2}{2\sum_{i=1}^n \sigma_i^2 + 2td/3}\right)$$

We apply Theorem 13 with $X_i = Y(i)$. As none of the vectors in $\mathcal{R}$ is bad, $V_i(k) \leq \epsilon^2 A(k)/(10 \log z)$, thus $X_i - E[X_i] \leq X_i \leq \epsilon^2 A(k)/(10 \log z)$ for all $i$. The variance

$$\sigma_i^2 \leq E[X_i]^2 \leq p_i V_i(k)^2 \leq (1-\epsilon)x_i^* V_i(k)^2 \leq x_i^* V_i(k)^2$$

Thus,

$$\sum_i \sigma_i^2 \leq \sum_i x_i^* V_i(k)^2 \leq \sum_i x_i^* V_i(k) \cdot \epsilon^2 A(k)/(10 \log z)$$

which is at most $\epsilon^2 A^2(k)/(10\log z)$ as $x_r^*$ satisfy the packing the constraints. As $E[\sum_i Y(i)] \leq (1-\epsilon)A(k)$, by Theorem 13 it follows that

$$\begin{aligned} Pr\left[\sum_i Y(i) > A(k)\right] &\leq Pr\left[\sum_i Y(i) - E[\sum_i Y(i)] > \epsilon A(k)\right] \\ &\leq \exp\left(\frac{-\epsilon^2 A(k)^2}{2\epsilon^2 A^2(k)/(10\log z) + 2\epsilon A(k)\cdot \epsilon^2 A^2(k)/(30\log z)}\right) \\ &\leq z^{-3} \end{aligned}$$

As there are at most $z/2$ local minima, by the union bound it follows that the rounded solution satisfies the packing constraints with probability at least $1 - 1/2z^2$.

We now analyze the number of vectors present in $S$ after the rounding.

**Lemma 14** *Let $n_2^*$ and be $n_3^*$ the contribution of vectors with size 2 and 3 respectively to the objective function in the optimum LP solution. Then, $S$ contains at least $(1-\epsilon)(n_2^*(1-e^{-2})/2 + n_3^*(1-e^{-3})/3)$ intervals in expectation.*

**Proof**: Consider a vector $\Delta^{V_i}$ with two non-zero entries and let $N(i)$ denote its neighbors. Due to the independence constraints 3, the quantity $\sum_{i' \in N(i)} x_{i'}^*$ is at most $2 - 2x_i^*$. Thus at any time the rate at which either $\Delta^{V_i}$ or some vector in $N(i)$ is chosen is at most $2 - x_i^*$ and hence at most 2. Hence the probability that none of these vectors is chosen by time $t$ is at least $e^{-2t}$. Thus the probability that $i$ is chosen during the infinitesimal interval $[t, t+dt]$ is equal to $x_i dt$ times the probability that none of $i$ or $N(i)$ has been chosen during $[0,t]$ which is at least $x_i e^{-2t} dt$. Thus the probability that $i$ is chosen in $S$ is at least

$$\begin{aligned} \int_0^{t_i} x_i e^{-2t} dt = \frac{x_i}{2}(1 - e^{-2t_i}) &= \frac{x_i}{2}(1 - (1-(1-\epsilon)x_i)^{\frac{2}{x_i}}) \\ &\geq \frac{x_i}{2}(1 - e^{-2(1-\epsilon)}) \\ &> (1-\epsilon)\frac{x_i}{2}(1 - e^{-2}) \end{aligned}$$

Thus $S$ contains at least $(1-\epsilon)n_2^*((1-e^{-2})/2)$ vectors of with two non-zero entries. A similar argument shows that $S$ contains at least $(1-\epsilon)n_3^*((1-e^{-3})/3)$ vectors with three non-zero entries. $\square$

Let $\alpha_2$ denote $(1-e^{-2})/2$ and let $\alpha_3$ denote $(1-e^{-3})/3$. As $n_2^* + n_3^* \geq n_2 + n_3 - o(z)$, and conditional on the fact that $S$ satisfies the packing constraints, $S$ contains at least $(1-2\epsilon)(\alpha_2 n_2 + \alpha_3 n_3)$ PPZS vectors. As $\epsilon$ is arbitrarily, this quantity approaches $\alpha_2 n_2 + \alpha_3 n_3$. By Theorem 12, we can also find $n_2 - |\mathcal{M}| = n_2 - o(z)$ disjoint PPZS vectors with two non-zero entries

Consider the algorithm that chooses the best of the two guarantees. Thus our algorithm has a setup time which is the minimum of $n_2 + 3n_3 + o(z)$ and $(2-\alpha_2)n_2 + (3-\alpha_3)n_3 + o(z)$. Since $n_2 + n_3 = \Omega(z)$, we will henceforth ignore the $o(z)$ term above.

Let $\alpha$ denote the desired approximation ratio $1 + \frac{1-\alpha_2}{2-2\alpha_2+\alpha_3} = \frac{11+9e^{-2}-2e^{-3}}{8+6e^{-2}-2e^{-3}} \approx 1.391$. It suffices to show that for any value of $n_2$ and $n_3$ at least one of the following inequality holds:

$$\begin{aligned} n_2 + 3n_3 &\leq \alpha(n_2 + 2n_3) \\ (2-\alpha_2)n_2 + (3-\alpha_3)n_3 &\leq \alpha(n_2 + 2n_3) \end{aligned}$$

Multiplying the first equation by $(1 - 2\alpha_2 + \alpha_3)$ and adding to the second, and doing some algebraic manipulation gives the identity $0 \leq 0$, which implies our result.

## C The algorithm for the general case

**Proof of Lemma 7**: We first show how to construct $\mathcal{R}_2$, and then show how to map a set $S$ of candidate interval vectors to a set $S' \subseteq \mathcal{R}_2$ with the required properties.

By the definition of candidate interval vectors any two such vectors with different heights do not begin and end at the same position. Thus, it suffices to show how to construct $\mathcal{R}_2^h$, the set $\mathcal{R}_2$ restricted to interval vectors of height $h$. Let $S_h$ denote $S$ restricted to interval vectors of height $h$. To find $S' \subseteq \mathcal{R}_2$, it suffices to show that we can find a set $S'_h \subseteq \mathcal{R}_2^h$ such that $|S'_h| \geq |S_h|$ and for each entry $i \in [1..n]$, $\sum_{V \in S'_h} V(i) \leq \sum_{V \in S_h} V(i)$.

To construct $\mathcal{R}_2^h$, we start with $\mathcal{R}_2^h = \emptyset$ and an empty stack $T$. For $i = 1, 2, \ldots, n$, we repeat the following: If $\Delta^A(i) = h$, then we push $i$ on to the stack, If $\Delta^A(i) = -h$ and the stack if not empty, we pop the top entry $t$ on the stack and add the interval vector $(t, i, h)$ to $\mathcal{R}_2^h$. If $\Delta^A(i) = -h$, and the stack is empty, we discard $i$ and move to $i + 1$.

It is easy to see that the sets $\mathcal{R}_2^h$, for all possible heights, can be constructed simultaneously in $O(n)$ time. By construction no two interval vectors in $\mathcal{R}_2^h$ begin or end at the same position. Moreover, any two intervals corresponding to two interval vectors in $\mathcal{R}_2^h$ are either non-overlapping or nested. Thus, these intervals form a *laminar* family of intervals and hence $\mathcal{R}_2^h$ can be viewed naturally as a collection of trees. The intervals along any path in this tree are nested. Finally, since $\mathcal{R}_2^h$ is constructed greedily, it is maximal in the sense that no interval vector $V \in \mathcal{R}_2^h$ can be replaced by another interval vector $V' \neq V$ such that the interval corresponding to $V'$ is contained in that corresponding to $V$, without violating the independence property of $\mathcal{R}_2^h$. Similarly, no interval vector can be added to $\mathcal{R}_2^h$ without violating the independence property.

Given $S_h$, we now show how to obtain $S'_h \subseteq \mathcal{R}_2^h$ with the required properties. We construct $S'_h$ recursively. Start with $S'_h = S_h \cap \mathcal{R}_2^h$ and remove these vectors from $S_h$ and $\mathcal{R}_2^h$. From now on assume that $S_h \cap \mathcal{R}_2^h = \emptyset$.

Consider an interval vector $U \in S_h$. Due to maximality of $\mathcal{R}_2^h$, there is some interval vector $V \in \mathcal{R}_2^h$ that either begins at the same position as $U$ or ends at the same position as $U$. Hence, there is some interval vector $(\ell, r, h) \in R_h$ such that $(\ell, r)$ is contained in the interval corresponding to $U$. Note that in the tree representation of the laminar family $\mathcal{R}_2^h$, the interval $L = (\ell, r, h)$ is a leaf in the subtree rooted at $V$.

If no interval vector in $S_h$ either begins at position $\ell$ or ends at position $r$, then we add $L$ to $S'_h$ and remove $U$ from $S_h$. Note that $L \leq U$ entry-wise and thus this maintains the properties. None of the remaining vectors in $\mathcal{R}_2^h$ and $S_h$ begin at $\ell$ or end at $r$, and thus we can proceed recursively.

Suppose that there is a vector $U' = (\ell, r', h) \in S_h$ that begins at position $\ell$. (The case where $U'$ ends at $r$ is symmetric.) By our assumption $U' \notin \mathcal{R}_2^h$. Since $(\ell, r, h)$ is a leaf it must be that $r' > r$. We consider two cases: If no interval vector in $S_h$ ends at position $r$, then we add $L$ to $S'_h$ and remove $U'$ from $S_h$. Note that $L \leq U'$ entry-wise and thus this maintains the properties. None of the remaining vectors in $\mathcal{R}_2^h$ and $S_h$ begin at $\ell$ or end at $r$, and thus we can proceed recursively.

If there is an interval vector $(\ell', r, h) \in S_h$ that ends at position $r$, then we must have $\ell' < \ell$. Replace the interval vectors $(\ell', r, h)$ and $(\ell, r', h)$ in $S_h$ by the interval vectors $(\ell, r, h)$ and $(\ell', r', h)$. Clearly this maintains the properties of $S_h$. Now, since $(\ell, r, h) \in S_h \cap \mathcal{R}_2^h$ remove it from both sets and add it to $S'_h$. None of the remaining vectors in $\mathcal{R}_2^h$ and $S_h$ begin at $\ell$ or end at $r$, and thus we can proceed recursively. $\square$