New algorithms for target delineation and radiation delivery in intensity-modulated radiation therapy

Xin Dou
University of Iowa

NEW ALGORITHMS FOR TARGET DELINEATION AND RADIATION DELIVERY IN INTENSITY-MODULATED RADIATION THERAPY

by

Xin Dou

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

December 2009

Thesis Supervisor: Assistant Professor Xiaodong Wu

ABSTRACT

Intensity modulated radiation therapy (IMRT) is a modern cancer therapy technique that aims to deliver a highly conformal radiation dose to a target tumor while sparing the surrounding normal tissues. The prescribed dose is specified by an intensity map (IM) matrix and often delivered by a multileaf collimator (MLC).

In this thesis, we study a set of combinatorial optimization problems arising in the field of IMRT: 1) the auto-contouring problems using region properties, which aim to optimize the intraclass variance of the target objects; 2) the field decomposition problems, whose goal is to decompose a “complex” IM to the sum of two “simpler” sub-IMs such that the two sub-IMs are delivered in orthogonal directions to improve the delivery efficiency; 3) the field splitting problems, which seek to split a large IM that can not be directly delivered by MLC into several separate sub-IMs of size no larger than the given MLC size and the delivery effectiveness is optimized.

Our algorithms are based on combinatorial techniques – mostly graph-based algorithms. We strive to find the globally optimal solution efficiently – in a linear or low polynomial time. In the case that the exact algorithm is not efficient enough, an approximation algorithm is also developed for solving the problem.

We have implemented all the proposed algorithms and experimented on computer-generated phantoms and clinical data. Comparing with results supervised by experts, the auto-contouring algorithms yield highly accurate results for all tested datasets. The field decomposition and field splitting methods produce treatment plans of much better quality while comparing with the state-of-the-art commercial treatment planning system.

NEW ALGORITHMS FOR TARGET DELINEATION AND RADIATION DELIVERY IN INTENSITY-MODULATED RADIATION THERAPY

by

Xin Dou

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

December 2009

Thesis Supervisor: Assistant Professor Xiaodong Wu

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Xin Dou

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Electrical and Computer Engineering at the December 2009 graduation.

Thesis Committee: ______________________________
Xiaodong Wu, Thesis Supervisor

____
Milan Sonka

____
Yusung Kim

____
Steve Collins

____
Kasturi Varadarajan

ACKNOWLEDGEMENTS

I would first of all like to thank Dr. Xiaodong Wu for his patient support and guidance for this work throughout my study at the University of Iowa. I would also like to thank Prof. Milan Sonka for imparting to me his broad knowledge in image processing.

During my study in this department, I have been benefited from interactions with many faculty members here, including Andreas Wahle in the Department of Electrical and Computer Engineering, and Yusung Kim, John E. Bayouth, and John M. Buatti in the Department of Radiation Oncology. Special thanks go to Steve M. Collins, and Kasturi R. Varadarajan, who served on my defense committee. I would also like to sincerely thank my workmates in my lab - Qi Song, Yunlong Liu, and William Monroe for their help in my research, and other students and staff in the ECE department, including Kunlin Cao, Yin Yin, Yinxiao Liu, Honghai Zhang, Shanhui Sun, Yao Wang, Xun Tang, and Fan Yang.

At last I want to thank my parents. Without their continuous love and support, this work would not be possible to finish.

This research was supported in part, by NIH grants R01-EB004640, K25-CA123112, R01-HL64368, R01-HL071809, and US National Science Foundation grant CCF- 0830402 and CCF-0844765.

ABSTRACT

Intensity modulated radiation therapy (IMRT) is a modern cancer therapy technique that aims to deliver a highly conformal radiation dose to a target tumor while sparing the surrounding normal tissues. The prescribed dose is specified by an intensity map (IM) matrix and often delivered by a multileaf collimator (MLC).

In this thesis, we study a set of combinatorial optimization problems arising in the field of IMRT: 1) the auto-contouring problems using region properties, which aim to optimize the intraclass variance of the target objects; 2) the field decomposition problems, whose goal is to decompose a "complex" IM to the sum of two "simpler" sub-IMs such that the two sub-IMs are delivered in orthogonal directions to improve the delivery efficiency; 3) the field splitting problems, which seek to split a large IM that can not be directly delivered by MLC into several separate sub-IMs of size no larger than the given MLC size and the delivery effectiveness is optimized.

Our algorithms are based on combinatorial techniques – mostly graph-based algorithms. We strive to find the globally optimal solution efficiently – in a linear or low polynomial time. In the case that the exact algorithm is not efficient enough, an approximation algorithm is also developed for solving the problem.

We have implemented all the proposed algorithms and experimented on computer-generated phantoms and clinical data. Comparing with results supervised by experts, the auto-contouring algorithms yield highly accurate results for all tested datasets. The field decomposition and field splitting methods produce treatment plans of much better quality while comparing with the state-of-the-art commercial treatment planning system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

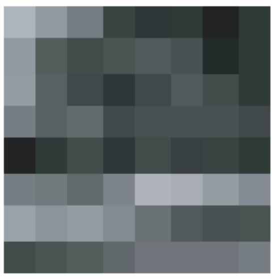
The study of mathematical and geometric optimization problems is becoming an important research area. Many application problems arise in this field and extensive studies have been done to solve these problems. In this thesis, we propose to solve a set of combinatorial optimization problems in the field of intensity-modulated radiation therapy (IMRT).

1.1 Basic Concepts in IMRT

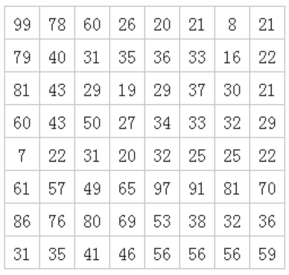
IMRT is a modern cancer therapy technique [23, 10] that aims to deliver a highly conformal radiation dose to a target tumor while sparing the surrounding normal tissues.

The prescribed dose distribution of radiation is commonly described by an *intensity map* (IM), which is specified by a set of nonnegative integers on a 2-D grid (see Figure 1.1(a,b)). The number in a grid cell indicates the amount (in unit) of radiation to be delivered. The delivery is done by a set of radiation beams orthogonal to the IM grid.

Most commonly the external radiation beams are delivered by a linear accelerator (LINAC) (see Figure 1.2). The linear accelerator uses microwave technology to accelerate electrons in a “wave guide”, then allows these electrons to collide with a high atom number metal (usually tunsten) target. As a result of the collisions, high-energy x-rays are produced from the target and become the radiation source.

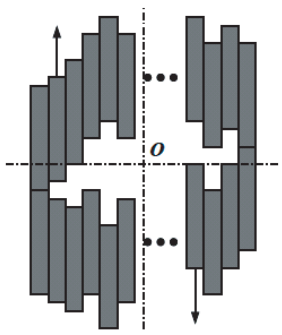


(a)



99	78	60	26	20	21	8	21
79	40	31	35	36	33	16	22
81	43	29	19	29	37	30	21
60	43	50	27	34	33	32	29
7	22	31	20	32	25	25	22
61	57	49	65	97	91	81	70
86	76	80	69	53	38	32	36
31	35	41	46	56	56	56	59

(b)



(c)

Figure 1.1: Illustration of intensity map(IM) and multileaf collimator (MLC). (a) An example of dose distribution. (b) The intensity map (IM) corresponding to dose distribution in (a). (c) An example of multileaf collimator (MLC).

Figure 1.2: A linear accelerator (LINAC).

An advanced tool today for IM delivery is the multileaf collimator (MLC) [74]. An MLC consists of many pairs of tungsten alloy leaves of the same rectangular shape and size (see Figure 1.1(c)). The leaves can move left and right to form a rectilinear region, called an *MLC-aperture*. Each MLC-aperture is associated with an integer representing the radiation units delivered by its radiation beam.

Comparing with 3D conformal radiation therapy (3DCRT), which uses 3D anatomic information and delivers highly "conformed" radiation to the target volume as closely as possible [49], IMRT has several advantages:

1. IMRT has the ability to conform the treatment volume to concave tumor shapes, which yields better tumor targeting and less normal tissue complication.
2. IMRT can deliver higher radiation to tumor and thus we can achieve better tumor control.

1.2 The Basic Workflow of IMRT

The workfolw of an MLC-based radiation treatment usually consists of the following steps:

1. Determine the tumor contour and the contours of nearby organs-at-risk. Table 1.1 lists some contour related terms used in treatment planning.

   Radiation therapy seeks to effectively irradiate CTVs while avoiding surrounding normal tissues and critical structures (OARs). To minimize radiation to normal tissue, CTV and surrounding OARs must be precisely defined. Currently this is done on CT or MRI scans of the patient (Figure 1.3) and this

Table 1.1: Contour related terms [35]

gross tumor volume	GTV	Gross palpable, visible, or clinically demonstrable disease
clinical target volume	CTV	GTV plus an extension for subclinical (microscopic) malignant disease
planning target volume	PTV	CTV plus a setup margin (SM) for uncertainties in patient positioning and alignment of the therapeutic beams
organ at risk	OAR	any organ or compartment of normal tissue which might be significantly impacted by the radiation dose delivered

procedure is called contouring. Currently, contouring is carried out manually by physician based upon clinical disease information by marking the perimeter of CTV on each image slice [50]. This may not be effective, and it may be time-consuming and subjective – there are considerable variations of contouring. Many interactive semi-auto contouring methods were then developed: these methods start with specialists' manual contours and then an intelligent algorithm searches around the contours and shapes the contours to the structures so they match the structures [56, 50]. This greatly improves the efficiency of the contouring process and reduces the subjectivity of GTV. These computer-aided contour modules are usually integrated into the treatment planning software and are the currently desirable approaches in clinical practice. With the development of image segmentation techniques, less and less interaction needs to be involved in the contouring process (e.g., only a 3-D box is needed as the bound of the contour, which is not necessarily tight to the tumor/structure.) There are

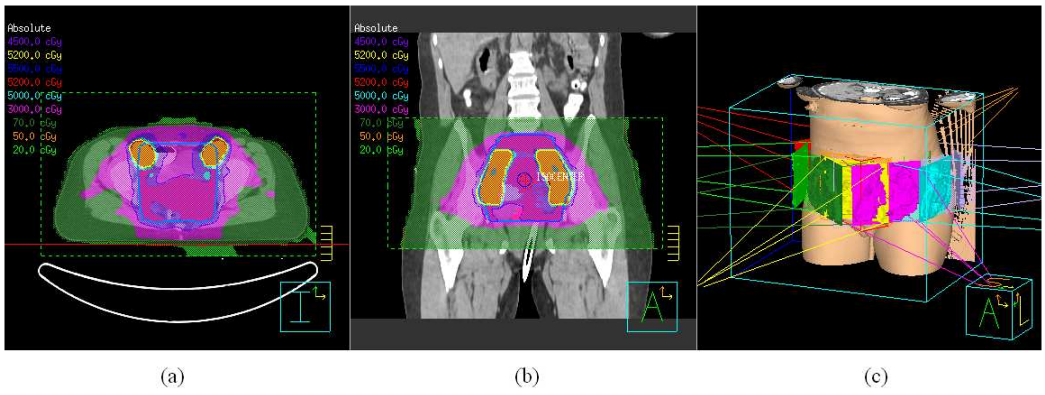


Figure 1.3: CT scans of a cervix tumor superimposed with dose distribution. (a) Transverse view (b) coronal view and (c) beam angle selection.

also fully automatic contour segmentation algorithms being developed [19, 54].

Besides determining the contours, the prescription dose (i.e., energy we want to impart to the irradiated PTV, e.g., $72Gy$), fractionation plan (e.g., $1.8Gy$ per day) and constraints, such as priority for overlapped structures and penalties, are also specified in this step.

2. Beams are then selected and optimized [12, 17]. We want to simplify the use of beams while maintaining a good quality of the plan. Less sensitive tissues or organs should be chosen as passage of radiation. Usually 4 to 9 such passages are chosen to apply beams (see Figure 1.3(c)).

   Beams are then optimized. The quantity optimized is an intensity map for each beam direction, comprised of beam elements (beamlets) that are typically square and the side length of the beamlet is equal to the width of an MLC leaf. Optimization is accomplished via a deterministic method that assesses the dose distribution [75, 4]. After each iteration of optimization, the dose distribution along each beam is calculated and then compared with the goal. It determines how much the dose goal is violated and how much change should be applied to the beamlets. At the end of each iteration, an objective function is evaluated to see how well the current dose distribution conforms the dose goal. The optimization terminates when an objective function goal is achieved or the maximum number of iterations is reached.

3. Radiation delivery. The intensity maps computed in step 2 is the dose to be delivered in intensity-modulated radiation therapy[24, 74, 82]. The system needs to know how to used MLC to deliver the dose. This process is known as leaf sequencing or conversion. There are two basic types of radiation delivery scheme: static and dynamic leaf sequencing.

    - Static leaf sequencing (SLS): SLS is also known as “step-and-shoot” delivery. In SLS, MLC does not move when the beam is on - the beam is turned off and wait the MLC-aperture to form, and then the beam is turned on to deliver a specific dose [7, 6]. Figure 1.4 shows an example of leaf sequencing for a given intensity map. The reader is referred to [82, 3, 30, 15] for more details on the step-and-shoot IMRT technique.

    - Dynamic leaf sequencing: In dynamic leaf sequencing [22, 51], the MLC leaves keep moving across the IM field with the radiation beam remains on.

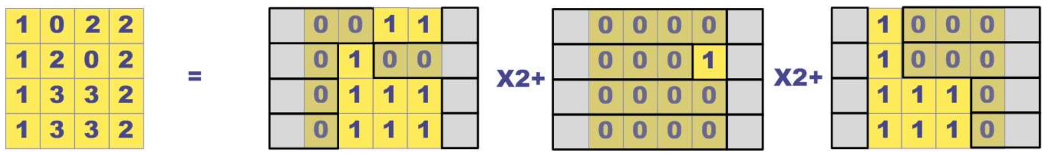


Figure 1.4: An example of static leaf sequencing for an IM.

Following the leaf sequencing, the prescribed dose is delivered to the patient.

4. Verification. The is the final step of IMRT. Patient-specific quality assurance (QA) procedures such as the comparison of the percent difference between measurements and planned results for IMRT treatments are carried out. Film and ion-chamber measurements are often used as verification methods. The goal of this procedure is to ensure accurate and consistent treatments for each patient.

1.3 Efficiency and Quality Measure of Static Leaf Sequencing

Mathematically, the static leaf sequencing planning can be viewed as the following matrix decomposition problem: Given an intensity map A (i.e., a matrix), decompose A into the form of $A = \sum_{i=1}^{\kappa} \alpha_i S_i$, where S_i is a special 0-1 matrix specifying an MLC-aperture, α_i is the amount of radiation delivered through S_i, and κ is the number of MLC-apertures used to deliver M (see Figure 1.4).

There are two obvious measures for the efficiency of the step-and-shoot delivery: (1) the *number of MUs (beam-on time)* which is given by $\sum_{i=1}^{\kappa} \alpha_i$, and (2) *the number κ of MLC-apertures* used. The number of MUs is the actual amount of radiation delivered to the patient and is proportional to the time that the patient is exposed to the radiation (beam-on time). Minimizing number of MUs is crucial to reduce the patient's risk under irradiation and to reduce the delivery error caused by the tumor motion [3]. On the other hand, minimizing the number of MLC-apertures used for each IM (hence, minimizing the treatment time of each IM) is also important because it not only lowers the treatment cost for each patient but also enables hospitals to treat more patients [15].

The delivery error (accuracy) can also be used as a quality measure of SLS planing. Due to the special geometric shapes of the MLC leaves [14, 57, 69, 74, 82, 83] (i.e., the tongue-and-groove interlock feature), an MLC-aperture cannot be delivered perfectly. Instead, there is a delivery error between the planned dose and actual delivered dose [14, 57] (called the tongue-and-groove error in medical literature [74, 82, 83]). Chen *et al.* [16] showed that for an IM $A = (a_{i,j})_{m\times n}$ of size $m \times n$, the minimum amount of error for delivering A is captured by the following formula (note that A contains only nonnegative integers):

$$Err(A) = \sum_{j=1}^{n} \left\{ a_{1,j} + \sum_{i=1}^{m-1} |a_{i,j} - a_{i+1,j}| + a_{m,j} \right\}. \tag{1.1}$$

Minimizing the delivery error is important because according to a recent study [26], the maximum delivery error can be up to 10%, creating underdose/overdose spots in the target region.

1.4 Summary of Target Problems and Our Main Results

In this thesis, we study various combinatorial optimization problems that arise in the treatment planning and delivery process to improve the effectiveness of IMRT.

1.4.1 Auto-Contouring Using Regional Properties

As mentioned in Section 1.2, tumor and organ contouring is the first step of the workflow of IMRT. An accurate contouring is the basis of a good IMRT planning. Many image segmentation methods have been developed to facilitate automatic contouring.

While edges defined by image gradients are commonly used for image segmentation, many object boundaries in medical image data may lack strong edges, e.g., when multiple adjacent objects with similar intensity profiles that may be locally noisy and not exhibit distinct edge properties, are present in an image. Image segmentation having the capability of handling weak edges is crucially important in medical image analysis. Intraclass variance has been successfully used in the well-known Chan-Vese active contour model *without* using image gradient [11], which is based on a piecewise constant minimal variance criterion of the Mumford-Shah functional [60]. The following formula captures the intraclass variance, which is the data-driven term of the energy function used by Chan and Vese:

$$\begin{aligned}\mathcal{E}(\mathcal{S}) = &\int_{inside(\mathcal{S})} |u_0(x,y,z) - c_1|^2 dxdydz \\ &+ \int_{outside(\mathcal{S})} |u_0(x,y,z) - c_2|^2 dxdydz \end{aligned} \tag{1.2}$$

where u_0 is the image, $\mathcal{S}$ is a variable boundary surface, and the constants c_1, c_2, depending on $\mathcal{S}$, are the averages of u_0 inside and outside $\mathcal{S}$, respectively. This energy function (intraclass variance) was proved capable of producing promising results [11]. However, Chan and Vese's method lacks the ability of finding the global optimality. Chan and Vese also considered two regularization terms in their energy function, which regularize the length of the boundary and the area of the region. These regularization terms tend to smooth the boundary of the target object.

Our algorithm is based on the shape probing technique used in computational geometry and computes a sequence of minimum-cost closed sets in a derived parametric graph.

Furthermore, we develop an approximation method which runs much faster than the exact algorithm while yielding highly comparable segmentation accuracy. A more efficient algorithm for a simplified version of the problem - single surface detection (SSD) problem - is also provided.

Methods using graph-searching principles [59, 58, 63, 71, 32] have become ones of the most frequently utilized medical image segmentation tools in 2-D. But it was not until recently that graph-search methods were extended to d-D ($d \geq 3$). Wu *et* *al.* [79] first developed a graph-search based method for single surface detection and then Li *et al.* [53] extended the work for detection of multiple interacting surfaces. The basic idea is to formulate the tasks as a minimum-cost closed set problem in graph theory, which can be solved by a minimum s-t cut algorithm.

Recently, the graph-cut based segmentation methods of Boykov *et al.* (e.g., [8]), which represent an option for optimally performing segmentation tasks in 3-D, have attracted a lot of attention. The cost function employed in their work follows the "Gibbs model", which is general enough to include both the region and boundary properties of the target objects. Their approach, which is topologically flexible and shares some elegance with the level set methods, has been quite successful. Though desirable for segmenting objects of unknown structures in many applications, topology flexibility in medical image segmentation may sometimes be considered a liability rather than an advantage since many medical structures have known topologies. Furthermore, Boykov *et al.*'s methods are at least non-trivial to be extended to simultaneous detection of coupled surfaces. Grady [37] recently developed a method for computing discrete minimal surfaces using linear programming.

The method we propose has been validated on computer-synthetic volumetric images and in X-ray CT-scanned datasets of plexiglas tubes of known sizes. Its applicability to clinical data sets is also demonstrated. In all cases, the approach yields highly accurate results.

1.4.2 Field Decomposition

In current SLS method, MLC leaves move along one direction (say, horizontally or vertically) to deliver the IMs during the entire delivery process. This uni-direction delivery may not fully utilize the capacity of the advanced MLC, which is rotatable. In fact, in order to improve the effectiveness of the IMRT delivery, it was proposed recently to rotate the MLC between the delivery of different segments of an IM [20, 29, 40, 44]. Several authors have reported variations on standard MLC-based techniques that attempt to improve the spatial resolution of fluence maps by rotating MLC [70, 31]. Later people considered delivering IMs by rotating the MLC to reduce interleaf leakage, tongue-and-groove effects and increase maximum deliverable field size [62]. The rotating capacity of MLC was also used to increase the maximum deliverable dose [66].

We propose to use two orthogonal directions to deliver an IM (i.e., horizontal and vertical). The intensity map is first decomposed to the sum of two "simpler" sub-intensity maps (sub-IM), and then, the sub-IMs are delivered in two orthogonal directions. This is a decomposition problem optimizing the total complexities of the two resulting sub-IMs. The complexity of an IM is not well defined, we use two different measures.

For each complexity measure, we develop an efficient algorithm for solving the IM matrix orthogonal decomposition problem.

Our algorithm is based on a non-trivial graph construction scheme, which enables us to formulate the decomposition problem as computing a minimum *s*-*t* cut in a 3-D geometric multi-pillar graph. Experiments on clinical intensity maps on Pinnacle show that our algorithm can improve the efficiency of the treatment plan. Using the first complexity measure, our algorithm produces as much as 27.3% less MLC-apertures with an average of 13.1% comparing with the SLS method using a single direction for delivery. Our second complexity measure performs better in terms of number of MUs. Using our decomposition algorithm, the average improvement percentage of number of MUs is 45.1%.

1.4.3 Field Splitting

One common constraint of the MLC is called the maximum leaf spread: each MLC leaf can only travel away from the vertical center line of the MLC within a certain threshold distance. Note that during the delivery of an IM, the vertical center line of the MLC is always aligned with the center of the IM. Geometrically, the maximum leaf spread means the rectilinear y-monotone polygon corresponding to each MLC-aperture has a maximum horizontal "width" ϖ (e.g., $\varpi = 14.5cm$ for the Varian MLCs). But we need to point out here that not all MLC systems require field splitting, e.g., Siemens MLCs and Tomotherapy, in which the radiation is delivered slice-by-slice, do not require field splitting in clinical practice.

In current clinical radiation therapy, large intensity maps occur [28, 42, 76]. Hong *al.* [42] did experiments on whole abdomen irradiation for 10 patients in which the GTV (gross target volume, which is the palpable, visible, or clinically demonstrable disease) included the entire peritoneal cavity. The PTV (planning target volume) extended $1cm$ beyond the GTV in both the superior and the inferior direction and has a margin of $5mm$ around GTV. Among the 10 patients, the ranges of PTV dimensions were: length 35 - $46cm$ (median $44cm$), width 27 - $35cm$ (median $30cm$), and depth 17 - $23cm$ (median $19cm$). Due to the maximum leaf spread constraint, a large IM needs to be split into several sub-IMs each being delivered separately using the step-and-shoot delivery technique. However, such splitting may result in lowered efficiency and increased delivery error, and thus compromise the treatment quality. The field splitting problem, roughly speaking, is to split an IM of a large size into multiple sub-IMs whose sizes are no larger than a threshold size, such that the treatment quality is optimized.

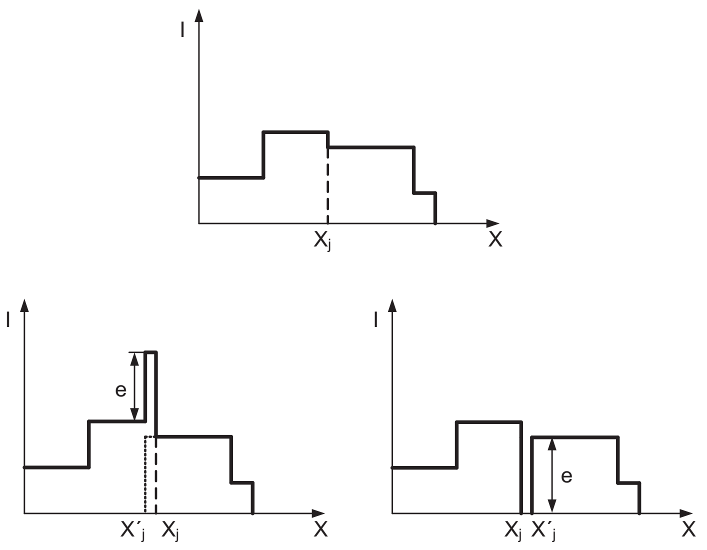


Figure 1.5: Illustration of hotspot and coldspot caused by field splitting, the top profile shows the desired profile split at x_j, due to field mismaching, the left end of right field is positioned at x'_j and the fields may overlap as in bottom left to cause hotspot or be separated as in bottom right to cause coldspot.

One simple way to split a large IM is to use straight lines, yielding abutting sub-IMs. One of the problems associated with this field splitting method is the field mismatching problem that occurs in the field junction region due to the uncertainties in setup and organ motion [46, 76]. If the borders of two abutting sub-IMs do not precisely align each other, it may result in hotspots or coldspots (see Figure 1.5). To alleviate the field mismatching problem, a commonly used medical practice is to apply a so-called field feathering technique [28, 46, 76]. Using this technique, a large IM A is split into a set of sub-IMs, $A_1, A_2, \ldots, A_K$, such that each sub-IM A_k is subject to the maximum field size constraint, and any two adjacent sub-IMs overlap over a central feathering region. Note that in the former splitting method, each IM cell belongs to exactly one sub-IM; but in the latter method, each cell of the feathering region can belong to two adjacent sub-IMs, with non-negative intensity value in both sub-IMs as in Figure 1.6. While splitting an IM into multiple sub-IMs to minimize the total complexity, it is also desirable to minimize the maximum number of MUs of the resulting sub-IMs. The motivation for this optimization is that, during the delivery of each sub-IM, the patient may move, and the larger the number of MUs of a sub-IM, the higher chance of body motion is. Thus it is good not to have a sub-IM after splitting with a large number of MUs.

A few field splitting algorithms have been recently reported in the literature to address the issue of treatment delivery efficiency and accuracy for large IMs. Early studies focused more on measures other than efficiency, e.g., the magnitude of hot/cold spots. Wu et al. [76] proposed a dynamic ”feathering” technique where the subfields

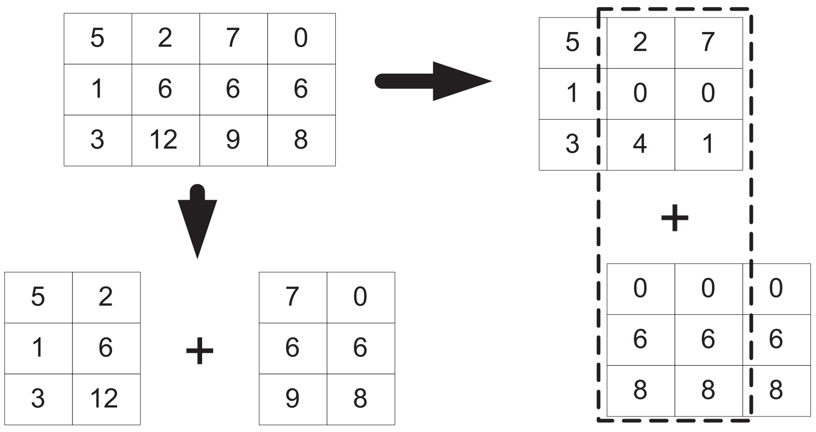


Figure 1.6: Illustrating field splitting without (the lower left panel) and with (the right panel) feathering. The upper left panel shows the original IM. The feathering region consists of two columns.

overlap each other by a small amount, and the intensity in the overlapping region gradually decreases in one subfield and increases in the other. Dogan *et al.* [28] employed the methods of shifting the isocenter position along the target width and introduced a "pseudo-target" to modify the split line positions, thereby reducing the magnitude of hot/cold spots. Later researchers put more efforts on preserving delivery efficiency when splitting the intensity map. To our best knowledge, Kamath *et al.* [47] first gave an $O(mn^2)$ time algorithm to split an $m \times n$ IM with or without feathering using vertical lines into at most three sub-IMs (thus restricting the maximum width of a large IM) while minimizing the total number of MUs. They further extended their algorithm to a more general case in which the field width was the only constraint [45]. However, their algorithm again works only when the width of the input IM is $\leq 3\varpi$, where ϖ is the maximum allowed field width. However, the current use of the high resolution motorized micro multi-leaf collimator (micro-MLC), which is usually designed for the treatment of lesions smaller than $8.0cm$ (e.g., a micro-MLC system manufactured by MRC systems GmbH Heidelberg has a maximum field width of $7.2cm$ [61]), may require the field splitting method to have the capability of splitting an IM into over three sub-IMs in order to treat large tumor sites. Wu [77] formulated the field splitting without feathering problem for an arbitrary field width ϖ using vertical lines as a k-link shortest path problem and developed an $O(mn\varpi)$ time algorithm. Chen and Wang [18] further developed an algorithm for optimally splitting an IM of size $m \times n$ with feathering while minimizing the total beam-on time of the resulting sub-IMs. Their algorithm runs in $O(mn + m\xi^{d-2})$ time, where d is the number of resulting sub-IMs and ξ is the remainder of n divided by ϖ . Very recently, field splitting while addressing delivery accuracy was studied in [81, 13]. Chen *et al.* [13] also considered field splitting problems based on other clinical objectives.

We study the problem of splitting a large IM into arbitrary number of sub-IMs if necessary, with the maximum leaf spread and the minimum and maximum feathering width as the only constraints, such that the total complexity of the resulting sub-IMs is minimized. Our algorithm is based on the shortest-path approach. Meanwhile, our algorithm strives to minimize the maximum number of MUs of those sub-IMs considering an interesting min-max slope path problem in a monotone polygon which is solvable in linear time. Our method has no size limit of tumor and is applicable to treating large tumor with micro-MLCs. A more general case is also studied and solved using dynamic programming approach.

A comprehensive comparison study is conducted against a commercial treatment planning system in radiation therapy. Our field splitting method outperforms Pinnacle in terms of both the total MU efficiency and the total MLC-aperture efficiency. To the best of our knowledge, this is the first study of the field splitting method considering both total MU and total MLC-aperture efficiencies.

1.5 Significance of Our Work and Their Impact on Radiation Therapy

In current treatment planning software, good success is achieved for high-contrast objects such as the external skin surface and outlines of the lungs, and for bone, all of which have high contrast interfaces, but for features with less contrast, only little success is achieved [35]. Using region information, the auto-contouring algorithm we proposed can help in this situation. By using the auto-contouring algorithm, subjectivity of the result is greatly reduced and efficiency is improved.

One of the known disadvantages of IMRT is its long delivery time. The treatment delivery time of the MLC-based technique is one to five times as long as that of conventional treatment [33]. In step-and-shoot radiation delivery, the total treatment delivery time consists of beam-on time, leaf-travel time and verification-and-recording (V&R) overhead time, and is equal to the summation of the first component and the larger of the other two components. For a system with a longer V&R overhead time, minimizing the number of MLC-apertures may significantly reduce the treatment delivery time. Thus reducing number of MUs (beam-on time) and/or number of MLC-apertures can help improve the delivery time. Our field decomposition method, which can be used as a preprocessing step of static leaf sequencing, has been proved to help in reducing number of MUs and number of MLC-apertures and thus improve the treatment efficiency.

When field splitting is necessary (although not all systems require field splitting for large fields), our field splitting algorithm produce feathering near the splitting region, which helps reduce the sensitivity to displacement at the field junction region due to uncertainties in setup and organ motion. Our algorithm also achieved better MU and MLC-aperture efficiency compared with commercial treatment planning software. As described above, treatment delivery time is closely related to number of MUs and number of MLC-apertures of a treatment plan, and thus our field splitting algorithm can achieve better treatment efficiency.

By applying our auto-contouring algorithms to the current planning software, we expect a higher treatment plan design efficiency (for shorter contouring time) and a better quality of the GTV/OAR contour. With this improvement, we expect a better tumor control and a lower normal tissue complication probability from the plan and thus reduce tumor recurrence and improve the patient survival rate.

By applying our field decomposition and field splitting algorithms to radiation therapy, we can expect a shorter delivery time. Thus uncertainties due to organ motion and systematic errors will be greatly reduced so less margin is needed and we may achieve a reduced normal tissue complication. When field splitting is required, the fluence map we generate may have a reduced sensitivity to displacement at the field junction and thus a better quality because its lowered intensities around splitting regions.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2,3, and 4 present our main results on region-based contouring algorithm, the field decomposition problems, and the field splitting problems, respectively. Chapter 5 concludes the thesis and proposes some interesting problems for future research.

CHAPTER 2 GLOBALLY OPTIMAL SEGMENTATION USING REGIONAL PROPERTIES

2.1 Introduction

In this chapter, we develop a novel algorithm that can find a globally optimal solution to segmentation by minimizing the *intraclass variance*. Our approach detects an optimal region between two coupled terrain-like surfaces in a volumetric image in a low-order polynomial time. Instead of adding the smoothness regularization term to the objective function as in the Chan-Vese model, we explicitly enforce the smoothness of the target surfaces with geometric constraints between neighboring voxels on the surfaces (see details in Section 2.2). The proposed method is limited to handling those target objects that can be "unfolded" into two coupled terrain-like surfaces, which may seem to highly limit the application scope of the method. However, as we will demonstrate, the guarantee of global optimality and the freedom to design problem-specific cost functions allow the method to be applied to various medical image segmentation problems, for instance, the delineation of inner and outer airway wall surfaces in pulmonary CT images and the detection of endocardial and epicardial boundaries of the left ventricle from cardiac MR, both of which are difficult to solve by existing techniques. We show that the optimal solution can be obtained via the construction of the convex hull for a set of $O(n)$ unknown 2-D points using the shape probing technique [21, 27] in computational geometry, where n is the size of the input image. The probing oracles are implemented by computing a minimum s-t cut in a weighted directed graph. The intraclass variance can then be minimized by a sequence of calls to the minimum s-t cut algorithm. The shape probing technique has been used for image segmentation in the past [2, 78]. To the best of our knowledge, our method is the *first* algorithm for *globally* minimizing the intraclass variance to detect a region bounded by two coupled terrain-like surfaces in a volumetric image. We believe that the developed technique is of interest on its own. We expect that it can help solving other important optimization problems existing in computer vision. We further develop an approximation method which runs much faster than the exact algorithm while yielding highly comparable segmentation accuracy. A more efficient algorithm for a simplified version of the problem - single surface detection (SSD) problem - is also provided.

The remaining of the chapter is organized as follows: Section 2.2 shows the problem modeling and some basic notations, Section 2.3 demonstrates the exact algorithm of the main problem, Section 2.4 shows the approximation method, and the SSD method is developed in Section 2.5.

2.2 Problem Modeling

Let I be a given 3-D volumetric image of $n = X \times Y \times Z$ voxels, where X, Y, and Z denote the image sizes in **x**, **y**, and **z** directions, respectively. The intensity level of every voxel (x, y, z) ($1 \leq x \leq X, 1 \leq y \leq Y$, and $1 \leq z \leq Z$) is denoted by $I(x, y, z)$. We consider the desired region (target object) R that is bounded by two coupled terrain-like surfaces, S_l and S_u, and oriented as shown in Figure 2.1. Each of the bounding surfaces intersects with exactly one voxel of every *column* parallel to the **z**-axis. We look for an *optimal region* by minimizing the intraclass variance among all feasible regions that can be defined in the 3-D volumetric image I. Let μ_0 (resp., μ_1) be the average intensity of the desired region R (resp., the background $\overline{R} = I - R$), that is $\mu_0 = \frac{1}{|R|}\sum_{(x,y,z)\in R} I(x,y,z)$, and $\mu_1 = \frac{1}{|\overline{R}|}\sum_{(x,y,z)\in \overline{R}} I(x,y,z)$. The intraclass variance is

$$\mathcal{E}_{CV}(R) = \sum_{(x,y,z)\in R} (I(x,y,z) - \mu_0)^2 + \sum_{(x,y,z)\in \overline{R}} (I(x,y,z) - \mu_1)^2. \tag{2.1}$$

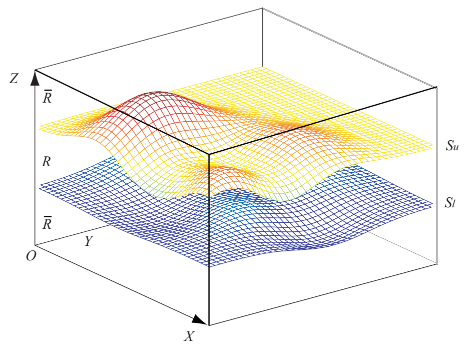


Figure 2.1: A region R enclosed by two coupled terrain-like surfaces S_l and S_u.

The feasibility of a region in I is constrained by two sets of application-specific parameters: (1) *surface smoothness parameters*, $\Delta_{\mathbf{x}}$ and $\Delta_{\mathbf{y}}$, and (2) *surface separation parameters*, δ^l and δ^u. The surface smoothness parameters guarantee the continuity of the bounding surfaces of R. More precisely, if (x,y,z) and $(x+1,y,z')$ (resp.,

$(x, y+1, z'))$ are two voxels on a feasible bounding surface, then $|z - z'| \leq \Delta_{\mathbf{x}}$ (resp., $|z - z'| \leq \Delta_{\mathbf{y}}$). The surface separation parameters ensure that the two bounding surfaces, S_l and S_u, of the desired region R are at a certain distance range apart, that is, for every pair (x, y), $\delta^l \leq S_u(x, y) - S_l(x, y) \leq \delta^u$, where $S(x, y)$ denotes the z-coordinate of the intersection voxel of the surface S and the column (x, y) of I. Comparing to the regularizing terms used in Chan and Vese's method [11], our geometric constraints not only regulate the smoothness of the bounding surfaces, they also incorporate essential shape information: the guarantee of monotonicity and topological constraints.

2.3 The Algorithm

Although minimizing the intraclass variance for general object shapes is computationally intractable, we are able to *optimally* detect the region bounded by two coupled terrain-like surfaces (or those regions that can be "unfolded" into two coupled terrain-like surfaces) in low-order polynomial time using the techniques of parametric search [48], hand probing [21, 27] in computational geometry, and 3-D graph-search [79, 53, 80].

Let $\mu = \frac{1}{n} \sum_{(x,y,z) \in I} I(x, y, z)$ be the average intensity of the entire image I. It is known that minimizing the intraclass variance $\mathcal{E}_{CV}(R)$ is equivalent to the maximization of the following objective function [39],

$$\mathcal{E}_{Inter}(R) = |R|(\mu - \mu_0)^2 + |\overline{R}|(\mu - \mu_1)^2,$$

which is called the *interclass variance* of R and $\overline{R}$.

The equivalency of the two objective functions can be shown as follows,

$$\begin{aligned}
\mathcal{E}_{CV}(R) &= \sum_{(x,y,z)\in R} (I(x,y,z)^2 - 2\mu_0 I(x,y,z) + \mu_0^2) \\
&\quad + \sum_{(x,y,z)\in \overline{R}} (I(x,y,z)^2 - 2\mu_1 I(x,y,z) + \mu_1^2) \\
&= \sum_{(x,y,z)\in I} I(x,y,z)^2 - 2|R|\mu_0^2 + |R|\mu_0^2 - 2|\overline{R}|\mu_1^2 + |\overline{R}|\mu_1^2 \\
&= \sum_{(x,y,z)\in I} I(x,y,z)^2 - |R|\mu_0^2 - |\overline{R}|\mu_1^2 \qquad (2.2)
\end{aligned}$$

and

$$\begin{aligned}
-\mathcal{E}_{Inter}(R) &= -n\mu^2 + 2\mu(|R|\mu_0 + |\overline{R}|\mu_1) - |R|\mu_0^2 - |\overline{R}|\mu_1^2 \\
&= \left(\sum_{(x,y,z)\in I} I(x,y,z)\right)^2 /n - |R|\mu_0^2 - |\overline{R}|\mu_1^2 \qquad (2.3)
\end{aligned}$$

Noticing that both $\sum_{(x,y,z)\in I} I(x,y,z)^2$ and $(\sum_{(x,y,z)\in I} I(x,y,z))^2/n$ are constants for a given image, the two objective functions differ by a constant, and thus minimizing $\mathcal{E}_{CV}(R)$ is equivalent to maximizing $\mathcal{E}_{Inter}(R)$.

Note that the objective function $\mathcal{E}_{Inter}(R)$ is invariant if we replace $I(x,y,z)$ by $\tilde{I}(x,y,z) = I(x,y,z) - \mu$ for every voxel (x,y,z) in I. We thus, without loss of generality (WLOG), assume that $\mu = 0$ and, accordingly,

$$\begin{aligned}
\mathcal{E}_{Inter}(R) &= |R|\left(\frac{U(R)}{|R|}\right)^2 + |\overline{R}|\left(\frac{-U(R)}{|\overline{R}|}\right)^2 \\
&= \frac{n}{|R|\cdot|\overline{R}|}(U(R))^2, \qquad (2.4)
\end{aligned}$$

where $U(R) = \sum_{(x,y,z)\in R} I(x,y,z)$. Note that $U(R)$ could be negative. If $U(R) < 0$ for an optimal region R, then we can define a new image such that the intensity of each of its voxel (x,y,z) is $-I(x,y,z)$. It is not difficult to see that an optimal region in this new image is also an optimal region in the original image. Hence, WLOG, we can assume $U(R) \geq 0$, and thus minimizing $\mathcal{E}_{CV}(R)$ is equivalent to maximizing

$$\Psi(R) \equiv \frac{U(R)}{\sqrt{|R|(n-|R|)}} = \frac{\sum_{(x,y,z)\in R} I(x,y,z)}{\sqrt{|R|(n-|R|)}}. \tag{2.5}$$

Let us further demonstrate how to find an optimal region R while maximizing $\Psi(R)$, where R is bounded by two coupled terrain-like surfaces.

2.3.1 Overview of the Algorithm

To maximize $\Psi(R)$, the following straightforward observation holds: for each $n_0 = 0, 1, \ldots, n$, if an optimal region $R^*_{n_0}$ of size n_0 can be computed so that it maximizes the total sum of intensity of all voxels in the region (denoted by $U(R^*_{n_0})$), the problem is solved. Unfortunately, that is not an easy task at all. However, viewing the problem in this way provides a basis for further exploitation of the intrinsic geometric structure of the problem.

For each $n_0 = 0, 1, \ldots, n$, the pair $(n_0, U(R^*_{n_0}))$ defines a point on the 2-D plane, on which the x-axis represents the number of voxels of a desired region R and the y-axis represents $U(R)$, thus forming a set $\mathcal{P}$ of $n+1$ points. A key observation here is that it may not be necessary to compute all points in $\mathcal{P}$. A classical concept in computational geometry [25], called *convex hull*, plays an important role. The convex hull $CH(\mathcal{P})$ of a point set $\mathcal{P}$ is the unique convex polygon which contains $\mathcal{P}$ and all vertices of which are points from $\mathcal{P}$.

Lemma 1. *The point* $(|R^*|, U(R^*))$ *defined by an optimal region* R^* *in* I *(i.e.,* $\Psi(R^*) = \max_R \Psi(R)$*), must be a vertex of the convex hull* $CH(\mathcal{P})$*.*

Proof. Let $\psi^* = \Psi(R^*) = U(R^*)/\sqrt{|R^*|(n - |R^*|)}$. Consider the curve $\xi : y = \psi^* \sqrt{x(n-x)}$ in the 2-D plane (see Figure 2.2). Since $U(R^*) = \psi^* \sqrt{|R^*|(n - |R^*|)}$, the point $(|R^*|, U(R^*))$ is on the curve ξ. Notice that $\psi^* = \max_R \{U(R)/\sqrt{|R|(n - |R|)}\}$. Thus, for any region R bounded by two coupled terrain-like surfaces, we have $U(R) \leq \psi^* \sqrt{|R|(n - |R|)}$, i.e., every point $(n_0, U(R^*_{n_0})) \in \mathcal{P}$ $(n_0 = 0, 1, \ldots, n)$ lies below or on the curve ξ. Furthermore, due to the concavity of the curve $\xi : y = \psi^* \sqrt{x(n-x)}$, all points in $\mathcal{P}$ lie below or on the tangent line l to ξ at the point $(|R^*|, U(R^*))$. Hence, $(|R^*|, U(R^*))$ is a vertex of the upper chain of the convex hull $CH(\mathcal{P})$ of $\mathcal{P}$. $\square$

Thus, finding the optimum can be simplified to examining all convex hull vertices. However, directly computing the hull vertices of $CH(\mathcal{P})$ appears to be quite involved. Inspired by the *shape probing* method [21, 27] which can be viewed as recognizing a convex polygon by "touching with lines", we use the following *probing oracle* to construct $CH(\mathcal{P})$ when the coordinates of the points in $\mathcal{P}$ are unknown. The probing oracle is:

Given a slope θ*, report the tangent line with slope* θ *to* $CH(\mathcal{P})$ *and the tangent point.*

Using this probing oracle, the convex hull $CH(\mathcal{P})$ can be constructed as follows.

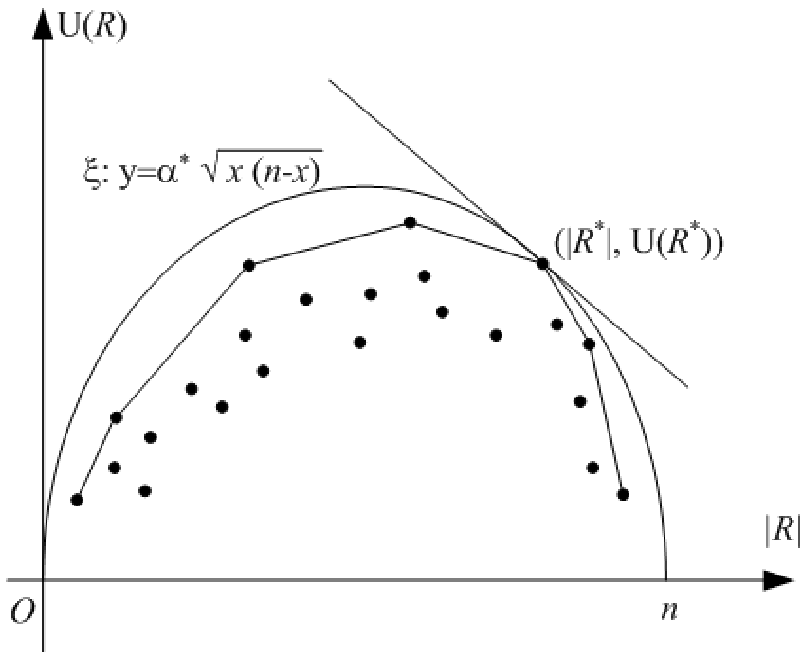


Figure 2.2: Illustrating the proof of Lemma 1.

Start with slopes $+\infty$ and $-\infty$ to find the two endpoints of $\mathcal{P}$ (leftmost and rightmost points, which are always $(0, 0)$ and $(n, 0)$ in this problem). Note that the convex hull $CH(\mathcal{P})$ is always an upper convex chain. Now suppose that two vertices u and v have been computed on the hull and that so far there is no vertex of $CH(\mathcal{P})$ between u and v. Let θ be the slope of the line through u and v. Then, employ a probing oracle with respect to θ (see Figure 2.3). Consequently, we either find a new vertex on $CH(\mathcal{P})$ between u and v or conclude that $\overline{uv}$ is an edge of $CH(\mathcal{P})$. Thus, employing a probing oracle results in either a new vertex or a new edge of $CH(\mathcal{P})$. Hence, the convex hull $CH(\mathcal{P})$ with k vertices can be computed in $O(k)$ probing oracle application steps.

A major challenge is to implement this oracle for a given slope θ. The para-

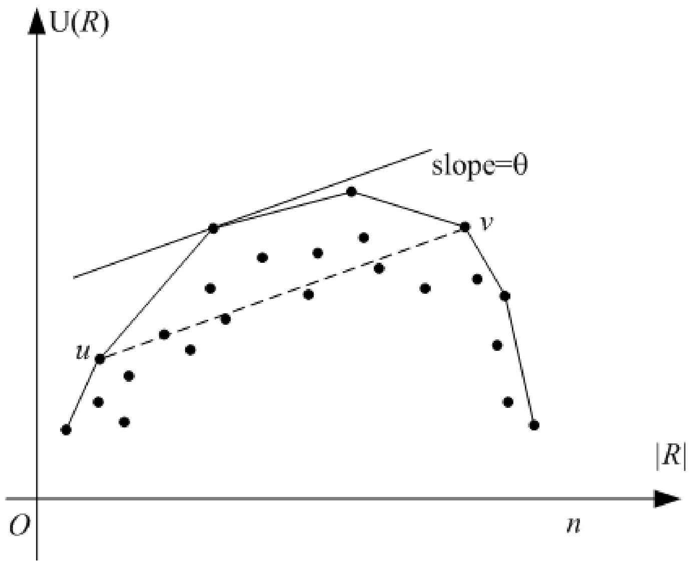


Figure 2.3: Illustrating the construction of a convex hull using the shape probing technique.

metric approach [48] in computational geometry is utilized. For a given real-valued parameter θ, we define the *parametric intensity sum* of a region R as the sum of intensities of all voxels in R minus $\theta|R|$ (i.e., $U(R) - \theta|R|$), denoted by $U_\theta(R)$. We show in Lemma 2 below, that the tangent point of the tangent line with slope θ to $CH(\mathcal{P})$ is defined by the optimal feasible region with a maximized intensity sum in the parametric image I_θ, where $I_\theta(x, y, z) = I(x, y, z) - \theta$ for every (x, y, z) tuple. This last step of the optimal-region-finding process can be solved using the existing graph-based segmentation method [53].

The key shape probing procedure can therefore be summarized as follows.

```
ShapeProbe($I$,$n_{left}$,$U_{left}$,$n_{right}$,$U_{right}$)
1  $\theta \leftarrow (U_{right} - U_{left})/(n_{right} - n_{left})$
```

```
2 Find the region $R^*(\theta)$ such that $U_\theta(R^*(\theta)) = \max_R U_\theta(R)$.
3 if $|R^*(\theta)| \neq n_{right}$ then do
4         ShapeProbe($I$,$n_{left}$,$U_{left}$,$|R^*(\theta)|$,$U(R^*(\theta))$)
5         ShapeProbe($I$,$|R^*(\theta)|$,$U(R^*(\theta))$,$n_{right}$,$U_{right}$)
```

The input to the subroutine ShapeProbe includes two known hull vertices $u(n_{left}, U_{left})$ and $v(n_{right}, U_{right})$, with no known hull vertices in between so far. It finds all hull vertices of $CH(\mathcal{P})$ between u and v. Line 1 calculates the slope θ of the probing lines. Line 2 computes an optimal region $R^*(\theta)$ whose parametric intensity sum is maximized to implement the probing oracle. Then, if a new hull vertex $(|R^*(\theta)|, U(R^*(\theta)))$ between u and v is found, as shown in Line 3, the procedure recursively computes all hull vertices on both left and right convex hull chains divided by $(|R^*(\theta)|, U(R^*(\theta)))$. Thus, the convex hull $CH(\mathcal{P})$ is computed. Based on Lemma 1, we can examine every vertex of $CH(\mathcal{P})$ to find the optimum. We next show the efficient implementation of the probing oracle.

2.3.2 Implementation of the Probing Oracle

Given a real-valued parameter θ, we show in this section that the probing oracle can be implemented by computing in I an optimal region $R^*(\theta)$ whose parametric intensity sum $U_\theta(R^*(\theta))$ is maximized. We call $R^*(\theta)$ an *optimal parametric region* associated with the parameter θ. Recall that R^*_k denotes an optimal region in I whose total intensity sum $U(R^*_k)$ is maximized.

Lemma 2. *There exists a tangent line to $CH(\mathcal{P})$ at the point $(n_0, U(R^*_{n_0}))$ with a slope θ if and only if $|R^*(\theta)| = n_0$ and $U(R^*(\theta)) = U(R^*_{n_0})$.*

Proof. "$\Rightarrow$" Suppose that $l : y = \theta x + b$ is a tangent line to $CH(\mathcal{P})$ at the point $(n_0, U(R^*_{n_0}))$. This implies that $b = U(R^*_{n_0}) - n_0\theta = U(R^*_{n_0}) - |R^*_{n_0}|\theta$. Note that $CH(\mathcal{P})$ is actually the upper chain of the convex hull. Thus for any $k \neq n_0$, the point $(k, k\theta + b)$ on l is on or above $CH(\mathcal{P})$. Alternatively, $k\theta + b \geq U(R^*_k)$, that is $U(R^*_{n_0}) - n_0\theta \geq U(R^*_k) - k\theta$ for any $k \neq n_0$ (see Figure 2.4). Hence, the region $R^*_{n_0}$ achieves $\max_k\{U(R^*_k) - k\theta\}$. Note that $U_\theta(R^*(\theta)) = \max_{R(\theta)}\{U(R(\theta)) - |R(\theta)| \cdot \theta\} = \max_k\{U(R^*_k) - k\theta\}$. Thus, $U(R^*(\theta)) = U(R^*_{n_0})$.

"$\Leftarrow$" The fact that $|R^*(\theta)| = n_0$ indicates that, for any feasible region $R(\theta)$ bounded by two coupled terrain-like surfaces, if $|R(\theta)| \neq n_0$, then $U(R(\theta)) - |R(\theta)| \cdot \theta \leq U_\theta(R^*(\theta))$. Thus, for any $k \neq n_0$, $U(R^*_k) - k\theta \leq U_\theta(R^*(\theta))$. Based on the assumption that $U(R^*(\theta)) = U(R^*_{n_0})$ and $|R^*(\theta)| = n_0$, we have $U(R^*_k) - k\theta \leq U(R^*_{n_0}) - n_0\theta$ for any $k \neq n_0$. Consider the line $l : y - \theta x = b$ with $b = U(R^*_{n_0}) - n_0\theta$. Obviously, the point $(n_0, U(R^*_{n_0}))$ is on Line l and any other point $(k, U(R^*_k))$ with $k \neq n_0$ is on or below Line l (see Figure 2.4). Hence, Line l is a tangent line to $CH(\mathcal{P})$ at the point $(n_0, U(R^*_{n_0}))$ with a slope θ.

This proves Lemma 2. $\square$

Consequently, for a given slope θ, we need to compute an optimal parametric region $R^*(\theta)$ bounded by two coupled terrain-like surfaces in I. If the size of $R^*(\theta)$ is n_0, based on Lemma 2, the line l: $y = \theta x + (U(R^*(\theta)) - n_0 \cdot \theta)$ is a tangent line to $CH(\mathcal{P})$ at the point $(n_0, U(R^*(\theta)))$ with slope θ. Let $R^*_{n_0} = R^*(\theta)$. We thus recognize a hull vertex on $CH(\mathcal{P})$. Next, we develop an efficient algorithm for computing such an optimal parametric region $R^*(\theta)$ in I.

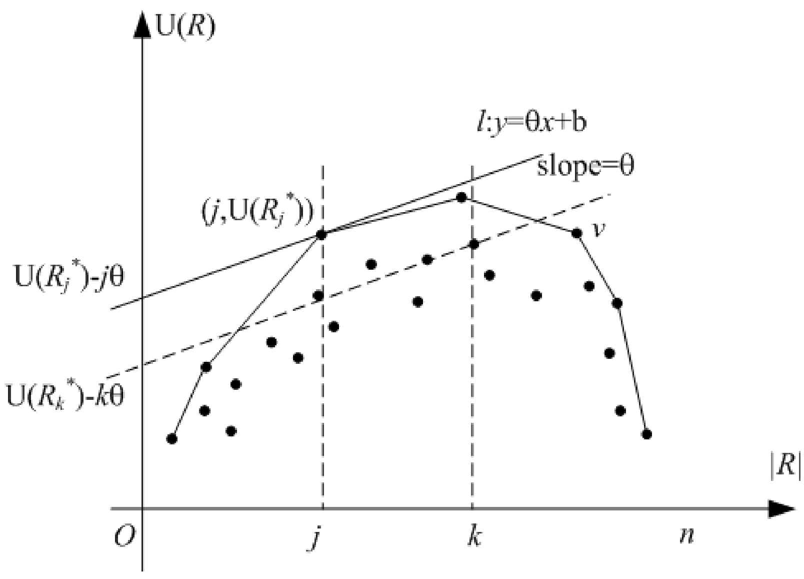


Figure 2.4: Illustrating the proof of Lemma 2.

2.3.3 Computing an Optimal Parametric Region

Given a parameter θ, we reduce the problem of computing an optimal parametric region $R^*(\theta)$ in I to the problem of finding two coupled terrain-like 3-D surfaces on the transformed images while minimizing the total sum of the cost on both surfaces. The detection of two coupled terrain-like surfaces can be formulated as a surface segmentation problem proposed by Li *et al.* [53].

First, we perform the following transformations on the image I:

$$I'_\theta(x,y,z) = \begin{cases} 0 & \text{if } z = 0; \\ \sum_{0 \le z' < z} \{I(x,y,z') - \theta\} & \text{otherwise.} \end{cases} \tag{2.6}$$

and

$$I''_\theta(x, y, z) = \sum_{0 \le z' \le z} -\{I(x, y, z') - \theta\}. \tag{2.7}$$

Hence, for any feasible region $R(\theta)$ bounded by two coupled terrain-like surfaces, S_l and S_u, with S_u above S_l, we have

$$\sum_{(x,y,z)\in S_l} I'_\theta(x, y, z) + \sum_{(x,y,z)\in S_u} I''_\theta(x, y, z) = -U_\theta(R(\theta)). \tag{2.8}$$

Note that both bounding surfaces S_l and S_u satisfy the smoothness constraint and the surface separation constraint.

In this way, we convert the optimal parametric region problem to a surface segmentation problem, which can be optimally solved using Li *et al.*'s method [53].

The basic idea of Li *et al.*'s surface segmentation method is to transform the problem into computing a minimum *s*-*t* cut in a derived arc-weighted directed graph. Denote by $T(n', m')$ the time for finding a minimum *s*-*t* cut in an arc-weighted directed graph with $O(n')$ vertices and $O(m')$ edges. For example, by Goldberg and Tarjan's algorithm [36], $T(n', m') = O(m'n' \log \frac{n'^2}{m'})$.

Lemma 3. *For a given θ, an optimal parametric region $R^*(\theta)$ in I can be computed in $O(T(n, n))$ time.*

In summary, it suffices to compute the convex hull $CH(\mathcal{P})$ to detect in I an optimal region while minimizing the intraclass variance by Lemma 1. Based on Lemma 2, we can perform $O(n)$ probing oracles to obtain all vertices on $CH(\mathcal{P})$. Each probing oracle can be implemented in $O(T(n, n))$ time by Lemma 3. Thus, the total running time of our algorithm for minimizing the intraclass variance is $O(nT(n,n))$. However, in our experimentation, the number of the employed probing oracles is much less than n.

2.4 Approximating Minimum Intraclass Variance

Although our exact algorithm for minimizing the intraclass variance in Section 2.3 is efficient for the moderate size of input images, the large image size may prevent the method from being computationally practical. In this section, we develop an approximation method that improves the running time while still producing close-to-optimal solutions.

Our method is based on the property of interclass variance explored by Asano *et al.* [2]. They utilized the property to obtain a $(1-\epsilon)$-approximation algorithm for computing an optimal connected rectilinear region with maximum interclass variance in 2-D. We characterize a similar property for a region bounded by two coupled terrain-like surfaces in 3-D, as stated in Lemma 4. The proof of the lemma is similar to that in [2].

Lemma 4. *Let $\hat{\theta}$ denote the optimal parameter value with which the optimal parametric region $R^*(\hat{\theta})$ maximizes $\Psi(R)$. If $\hat{\theta} \neq 0$, then an optimal parametric region $R^*((1+\epsilon)\hat{\theta})$ gives an $(1-\epsilon)$-approximate solution to the problem of maximizing $\Psi(R)$, that is, for any $0 < \epsilon < 1$, $\Psi(R^*((1+\epsilon)\hat{\theta})) \geq (1-\epsilon)\Psi(R^*(\hat{\theta}))$.*

We assume that the cost $I(x,y,z)$ of each voxel in the input image I is an integer, and $\mathcal{L}$ is the total sum of the absolute values of the voxel costs in I. Due to the integrality of $I(x,y,z)$, it is not difficult to see that $\frac{1}{n} \leq |\hat{\theta}| \leq \mathcal{L}$. Our idea is to partition the θ-space $[-\mathcal{L} \cdots -\frac{1}{n}] \cup [\frac{1}{n} \cdots \mathcal{L}]$ into intervals $[\theta_i \cdots \theta_{i+1}]$ such that either $\frac{\theta_i}{\theta_{i+1}} = 1+\epsilon$ or $\frac{\theta_{i+1}}{\theta_i} = 1+\epsilon$. Note that Lemma 4 actually indicates that for any θ in between $(1+\epsilon)\hat{\theta}$ and $\hat{\theta}$, an optimal parametric region $R^*(\theta)$ gives an $(1-\epsilon)$-approximate solution to the problem of maximizing $\Psi(R)$. Thus, if $\hat{\theta} \in [\theta_i \cdots \theta_{i+1}]$, then either $R^*(\theta_i)$ or $R^*(\theta_{i+1})$ is an $(1-\epsilon)$-approximate solution. Hence, we partition the θ-space into the following intervals:

$$\left(\bigcup_{i=0}^{\lfloor \log_{1+\epsilon} n\mathcal{L} \rfloor} [-\frac{1}{n}(1+\epsilon)^{i+1} \cdots -\frac{1}{n}(1+\epsilon)^{i}] \right) \bigcup \left(\bigcup_{i=0}^{\lfloor \log_{1+\epsilon} n\mathcal{L} \rfloor} [\frac{1}{n}(1+\epsilon)^{i} \cdots \frac{1}{n}(1+\epsilon)^{i+1}] \right).$$

Considering the sequence $\{-\theta_i, \theta_i\}$ of parameters for θ, with $\theta_i = \frac{1}{n}(1+\epsilon)^i$ ($i = 0, 1, \ldots, \lfloor \log_{1+\epsilon} n\mathcal{L} \rfloor$), we compute an optimal parametric region $R^*(\theta_i)$ for each θ_i. Among those optimal parametric regions, the one that maximizes $\Psi(R)$ is chosen as the approximation solution. It is clear that such a solution is an $(1-\epsilon)$-approximation solution. The number of parameters that we search is $O(\log_{1+\epsilon} n\mathcal{L}) = O(\epsilon^{-1} \log n\mathcal{L})$. Note that an optimal parametric region $R^*(\theta)$ for a given parameter θ can be computed in $T(n,n)$ time using minimum s-t cut algorithm. Thus, the running time of this approximation algorithm is $O(\frac{\log n\mathcal{L}}{\epsilon} T(n,n))$.

Lemma 5. *A $(1-\epsilon)$-approximation solution to the problem of maximizing $\Psi(R)$ can be computed in $O(\frac{\log n\mathcal{L}}{\epsilon} T(n,n))$ time.*

We then output the $(1-\epsilon)$-approximation solution to the problem of maximizing $\Psi(R)$ as our approximate solution to the problem of minimizing the intraclass variance $\mathcal{E}_{CV}(R)$. Although we have not yet proven a tight bound of the approximation ratio of this approximation algorithm for minimizing the intraclass variance, our implementation demonstrated its high segmentation accuracy with much less execution times for all data we tested for our exact algorithm.

2.5 Fast Single Surface Detection Algorithm

Since the exact algorithm is computationally expensive, we consider a simplified version of the problem - the single surface detection (SSD) problem, whose complexity can be greatly reduced. In this section, we develop a fast algorithm for the SSD problem.

SSD problem seeks to find an optimal terrain-like surface S^* such that (1) S^* satisfies the smoothness constraints, (2) S^* divides the input image into two regions: the target object R consisting of all voxels on and below S^* and background $\overline{R}$ consisting of all voxels above S^* (with respect to z-axis), and (3) the total sum of intraclass variance of R and $\overline{R}$ is minimized. Instead of considering a target region bounded by two coupled terrain-like surfaces as in Section 2.3, the SSD problem looks for an optimal region bounded by a single terrain-like surface.

Recall that we can still use the algorithmic framework in Section 2.3 and the only difference is the implementation of the probing oracle. For a given parameter θ, we need to compute an optimal parametric region bounded by a single terrain-like surface, as in Section 2.3.3. By applying Li *et al.*'s method [53], this problem can be formulated as computing a maximum-cost closed set in a constructed graph $G(\theta)$

associated with the parameter θ.

Let V^+ and V^- denote the sets of nodes in G with non-negative and negative costs, respectively. Define a new directed graph $G_{st} = (V \cup \{s,t\}, E \cup E^{st})$. The node set of G_{st} consists of the node set V of G plus a source node s and a sink node t. The arc set of G_{st} is the arc set E of G plus a new arc set E^{st}. We assign an infinity cost to each arc in E. E_{st} consists of the following arcs: The source node s is connected to each node $V(x,y,z) \in V^-$ by a directed arc of cost $-I(x,y,z)+\theta$; every node $V(x,y,z) \in V^+$ is connected to the sink node t by a directed arc of cost $I(x,y,z)-\theta$. It can be proved that the minimum-cut corresponding to the maximum-flow in G_{st} is the same as the cut separating the minimum closed-set in G.

Observing that when θ increases, the cost of an arc from the source node s to a node $V(x,y,z) \in V$ increases at a slope of 1; the cost of an arc from a node $V(x,y,z) \in V$ to the sink node t decreases at a slope of 1 until it reaches 0, when this link is cut and a new arc is created from the source node s.

We then model a parametric maximum-flow problem. The graph $G'_{st}(\theta) = (V \cup \{s,t\}, E \cup E'^{st})$. The node set of $G'_{st}(\theta)$ is the same as that of G_{st}. E'^{st} consists of the following arcs: there is an arc from the source node s to each node $V(x,y,z)$ with a cost of $\max(0, -I(x,y,z)+\theta)$; an arc is connected from $V(x,y,z)$ to the sink node t with a cost of $\max(0, I(x,y,z)-\theta)$.

Noticing that the arc weights of the graph G_{st} have the following good properties:

1. for each arc from the source s to a vertex, the arc weight is a nondecreasing

function of θ;

2. for each arc from a vertex to the sink t, the arc weight is a nonincreasing function of θ; and
3. for each of the other arcs, the weight is a constant $(+\infty)$,

we proposed to solve this problem using Gusfield and Martel's parametric minimum s-t cut algorithm [38].

$G'_{st}(\theta)$ is a monotone parametric flow network [34, 38] with respect to the parameter θ. Following the algorithmic framework in Section 2.3, we need to compute a minimum s-t cut for each of a sequence of parameters $\{\theta_1, \theta_2, \ldots, \theta_\tau\}$ generated by the shape probing process, where $\tau = O(n)$. Due to the monotonicity of $G'_{st}(\theta)$, the problem can be used to compute all those τ minimum s-t cuts in the complexity of solving a single maximum flow problem.

The running time of this parametric maximum-flow algorithm is $O(n'^3 + kn')$ if the number of nodes in the graph is n' and the number of different θ's is k. In our problem, it can be shown the number of nodes in the graph is $n+2$ and the number of different θ's is bounded by n. So the time complexity of our algorithm is $O(n^3)$. The running time can be further reduced to $O(n^2 \log n)$ if dynamic tree data structure is used.

Lemma 6. *The globally optimal solution to our single surface detection using region properties problem can be computed in $O(n^2 \log n)$ time.*

2.6 Experimental Methods

The experiments were performed on computer phantoms, physical phantoms and 3-D CT and MR medical images. Both regions bounded by terrain-like surfaces and by tubular surfaces were used.

2.6.1 Data

Computer Phantoms: To validate the correctness of the modeling techniques, we tested our method on a set of computer-generated phantoms, with sizes of $256 \times 256 \times 3$ voxels, containing differently textured regions or shapes (Figure 2.7). For the evaluation of the execution times, a second set of computer phantoms was produced that contained a region bounded by two coupled terrain-like surfaces with various shapes and mutual positions. Two sets of different patterns were used to differentiated the target region from the background. The sizes of the phantom images ranged from $30 \times 30 \times 40$ to $100 \times 100 \times 50$. To test the performance of our algorithm in noisy images, we added this set of computer phantoms with Gaussian noise of $\sigma = 0.5$, 2.0, and 3.0, respectively. In addition, we generated a third set of 160 computer phantoms, each of size $100 \times 100 \times 10$, to evaluate the approximation ratio of the approximation algorithm. Each phantom image included a region bounded by two coupled terrain-like surfaces with specific position, distance, and smoothness randomly generated. Please refer to Figure 2.7(a) and (b) in Section 2.7 for samples of computer phantoms.

Physical Phantoms: To quantitatively assess the performance of our method, a physical phantom was imaged by multi-detector CT and analyzed. The phantom contained six plexiglas tubes, numbered 1 through 6, with nominal inner diameters of 1.98, 3.25, 6.40, 6.50, 9.50 and 19.25mm, respectively. The corresponding outer diameters were 4.45, 6.30, 9.70, 12.60, 15.60 and 25.50mm, respectively. The phantom was scanned using Philips Mx8000 4-slice CT scanner with 3 different scan settings (low dose, regular dose, and high dose). Under each setting, the scans were taken at 4 distinct angles of 0°, 5°, 30°, and 90°, rotated in the coronal plane, resulting in a total of 12 datasets for use in the validation. The regular dose scanning was intentionally repeated, yielding additional 4 datasets used for initial calibration of the cost functions. In all cases, a resolution of $0.39 \times 0.39 \times 0.6mm^3$ was used. Images consisted of 200-250 slices, 512×512 pixels each. Figure 2.5 shows a sample slice of the data.

CT Images of Pulmonary Airway Walls: To demonstrate the utility of our method in quantitative analysis of medical images, the method was applied to human pulmonary CT images to concurrently segment the inner and outer wall surfaces of intrathoracic airways imaged by multi-detector CT. 20 airway wall segments extracted from 12 *in vivo* CT scans of 6 human objects were used for the experiments. The airway wall segments had a resolution of $0.7 \times 0.7 \times 0.6mm^3$ and consisted of 30-50 slices, each having an average size of 50×50 pixels. Figure 2.6(a) shows some sample slices of the pulmonary airway wall data.

The inner and outer surfaces needed to be unfolded to apply our method. The centerlines of the airways were manually chosen for the unfolding process. B-spline interpolation was used in the unfolding process. Segmentation was performed only on the trunks of the airway tree between bifurcations since our method can not

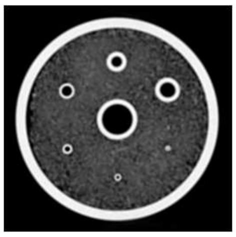


Figure 2.5: A sample slice of the physical phantom data.

handle topological changes at this point. The inner and outer walls were segmented simultaneously.

MR Images of Cardiac Ventricular Walls: To study the applicability of the method to a broader range of medical image segmentation applications, we also performed our method on MRI images of left human cardiac ventricle. 20 MRI scans of normal human hearts and 10 MRI scans of patient hearts were segmented. The images had a resolution of $2.08 \times 2.08 \times 2.08mm^3$. Each dataset consisted of 30-40 mid-papillary slices and each slice had an average size of 40×40 pixels. To facilitate the unfolding process of the tubular shape of ventricular walls, the centerlines (long axes) of the cardiac ventricle were manually determined. Figure 2.6(b) shows some sample slices of the cardiac ventricular wall data.

CT Images of Liver Lesion for Single Surface Detection: To study the applicability of the single surface detection method to medical image segmentation applications, we performed our method on CT images of liver lesion. 15 CT scans of liver lesions were segmented. The images had a resolution of $0.97 \times 0.97 \times 0.97mm^3$. Each dataset consisted of 10-30 slices and each slice had an average size of 50×50 pixels.

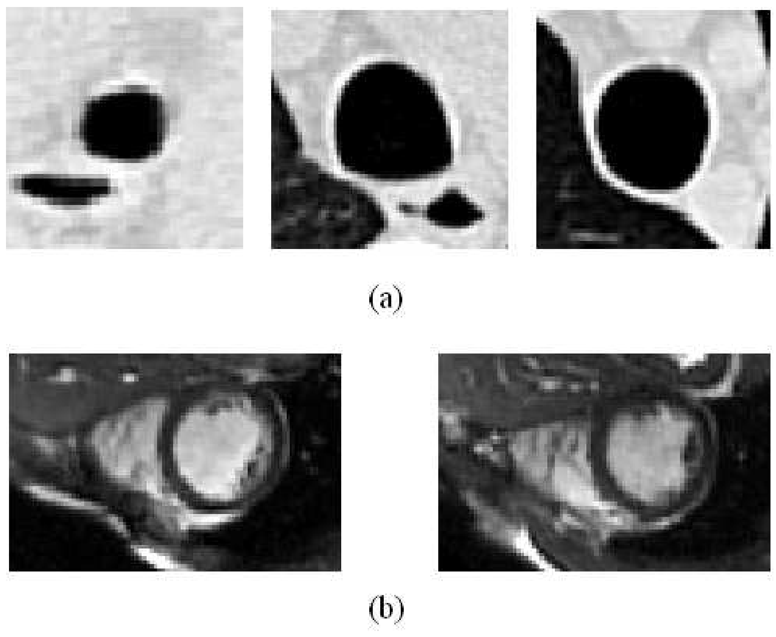


Figure 2.6: Sample slices of pulmonary airway wall data and cardiac ventricular wall data. (a) Pulmonary airway wall data. (b) Cardiac ventricular wall data.

The user initiated the segmentation by drawing a single line over the target object in a cross-sectional slice, which roughly indicated the extent of the object. Then an ellipsoidal coordinate transformation was performed on the region of interest to unfold the bounding surface and the surface detection algorithm was used to segment the tumor. Please refer to Figure 2.19 in Section 2.7.8 for sample slices of the data.

CT Images of Lung Tumor for Single Surface Detection: Lung tumor data was also used for the verification of our single surface detection method. 12 CT scans of lung tumors were segmented. The images had a resolution of $0.97 \times 0.97 \times 0.97 mm^3$. Each dataset consisted of 12-40 slices and each slice had an average size of 60×60 pixels. A ellipsoidal transformation similar to the one used for liver lesion segmentation was performed. Please refer to Figure 2.20 in Section 2.7.8 for sample slices of the data.

2.6.2 The Cost Functions

Cost function design is very important in graph-based segmentation. Since our method minimizes the intraclass variance, a cost function reflecting the homogeneity was used. For the experiments, the intensity or its linear transformation was utilized as the cost of a voxel. For the texture related phantom images, the cost function also included orientation and/or curvature information [11]. For the clinical data, our algorithm was run on the original image to get the estimated positions of the bounding surfaces and then this estimated position information was combined with the voxel intensities to form the cost function. For liver lesion cases, because basically the intensities in both the parenchyma and lesion are homogeneous, the voxel intensity and transformation of intensity were used in cost function. For lung tumor cases, generally intensity was used for cost function. In case a tumor is close to the boundary of the lung, lung boundary was segmented first and the information was combined into the cost function.

2.6.3 Execution Time

All experiments were conducted on a Pentium-D 2.4GHz PC with 3.5 GB of memory. The execution time included the graph initialization time and the actual computation time. The execution times varied across the test cases even with the same image size. We thus averaged the running times over 20 test cases for each image size. The execution time for each test case was measured three times and the results were averaged. In our implementation, Boykov and Kolmogorov 's algorithm [9] was used to compute the minimum *s*-*t* cut.

2.6.4 Performance Indices

Besides visual inspection, which was carried out on computer phantom images, we used several quantitative indices to measure the performance of our method.

2.6.4.1 Diameter Calculation for Physical Phantom

For the physical phantoms, the average diameter of the segmentation result of each slice was calculated using the circle fitting method to minimize the mean-squared error, and then the diameter was compared with the known tube diameter. The diameter errors are reported as mean $\pm$ standard deviation in absolute measurements and as percentages of the diameter.

2.6.4.2 Positioning Error Measure for Clinical Cases

Surface detection accuracy was determined in clinical test cases in comparison with the independent standard. The mean surface positioning errors were computed and expressed in pixels. Corresponding points were defined as pairs of points, the first point being from a computer detected border and the second point from the reference standard border that was closest to each other using the Euclidean distance metric. The positioning errors were calculated both signed and unsigned, where signed errors were calculated by adding a sign to each pair-wise unsigned errors, and the sign was positive while the point from a computer detected border was outside the reference standard contour, and negative otherwise. These errors were reported as mean $\pm$ standard deviation.

2.6.4.3 Optimizing Intraclass Variance with Iterations

Since our reported approach and the Chan-Vese original method [11] both optimize the intraclass variance, it is of interest to compare their performance. Considering that the regularization terms and constraints are different, a fair comparison is very difficult. We thus compared our approach with a modified Chan-Vese method, which keeps the iteration-based principle of the Chan-Vese method to optimize the intraclass variance while it incorporates the geometric constraints of our method. We briefly call the method an *iterative intraclass variance or iterative ICV* method.

The basic idea of the Chan-Vese method [11] is to first find an initial region and calculate the average intensities of the initial region and the background as μ_0^e and μ_1^e. These two parameters are used as constants in the next iteration to find an optimal region optimizing the objective function,

$$\begin{aligned} \mathcal{E}_{Chan-Vese}(R) &= \sum_{(x,y,z)\in R}(I(x,y,z)-\mu_0^e)^2+\sum_{(x,y,z)\in\overline{R}}(I(x,y,z)-\mu_1^e)^2 \\ &\quad +T_{regulariztion}, \end{aligned} \tag{2.9}$$

where $T_{regularization}$ is a regularization term. Then, the new average intensities, μ_0^e and μ_1^e, of the newly computed region and the background are calculated. This process iterates until both μ_0^e and μ_1^e converge.

The iterative ICV method "simulates" the Chan-Vese method to find an optimal region bounded by two coupled terrain-like surfaces while minimizing the intraclass variance of the region. With given average intensities, μ_0^e and μ_1^e, of the initial region and the background, the Li *et al.*'s graph searching method [53] was employed to find the desired region with the same geometric constraints as used in the presented method.

We performed the following transformations on image I:

$$I'(x,y,z) = \sum_{0\le z'<z} (I(x,y,z') - \mu_1^e)^2 + \sum_{z\le z'<Z} (I(x,y,z') - \mu_0^e)^2. \tag{2.10}$$

and

$$I''(x,y,z) = -\sum_{z<z'<Z} (I(x,y,z') - \mu_0^e)^2 + \sum_{z<z'<Z} (I(x,y,z') - \mu_1^e)^2. \tag{2.11}$$

Hence, for any feasible region R bounded by two coupled terrain-like surfaces, S_l and S_u with S_u above S_l, we have

$$\begin{aligned} &\sum_{(x,y,z)\in S_l} I'(x,y,z) + \sum_{(x,y,z)\in S_u} I''(x,y,z) \\ = &\sum_{(x,y,z)\in R} (I(x,y,z) - \mu_0^e)^2 + \sum_{(x,y,z)\in \overline{R}} (I(x,y,z) - \mu_1^e)^2. \end{aligned} \tag{2.12}$$

Note that both bounding surfaces S_l and S_u satisfy the surface smoothness and separation constraints. Using Li *et al.*'s method [53], the optimal surfaces S_l^* and S_u^* are simultaneously detected in I' and I'', respectively. From Equation (2.12), the region R^* bounded by S_l^* and S_u^* is the desired region in I. This process iterates until both

μ_0^e and μ_1^e converge, as in the Chan-Vese method.

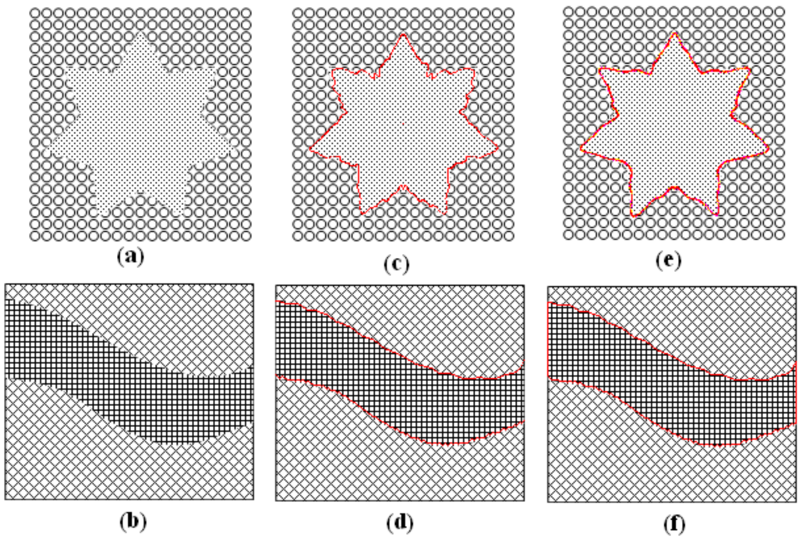


Figure 2.7: Computer phantoms with textures and segmentation results. (a),(b) Original images. (c),(d) The segmentation results from our method. (e),(f) The segmentation results from the Chan-Vese method.

2.7 Results

2.7.1 Computer Phantoms

Figure 2.7 presents segmentation examples obtained by our algorithm. The objects and background were differentiated by their different textures. The curvature and edge orientation in each slice were used (as descriptions of the patterns) in the cost functions. Using the Chan-Vese method with the same cost images yielded similar results.

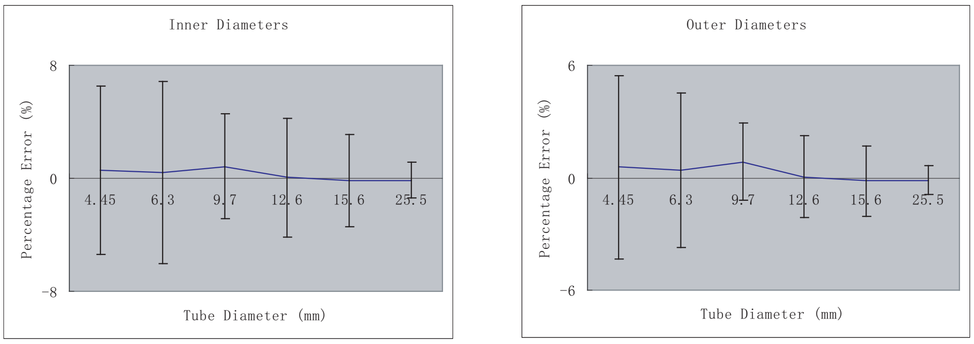


Figure 2.8: The signed percentage errors of inner- and outer-diameter measurements in the CT-imaged physical phantom tubes.

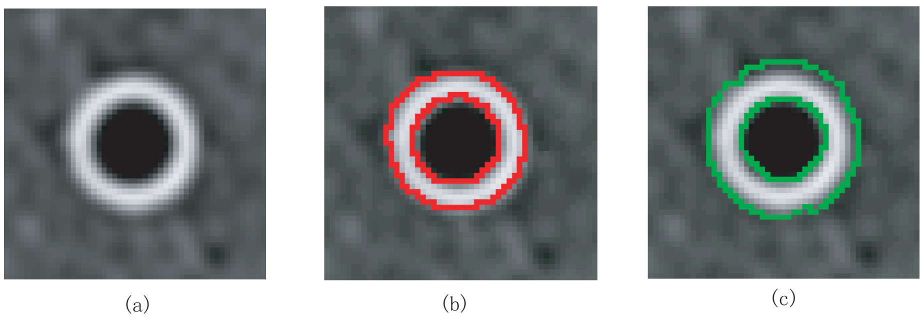


Figure 2.9: Segmentation result on CT-imaged physical phantom tubes. a) An example 2-D slice of a 3-D CT-imaged physical phantom. b) Segmentation result from Chan-Vese method. The errors of the measured diameters of the inner and outer walls were -0.139mm and -0.314mm, respectively. c) Segmentation result from our method. The errors of the measured diameters of the inner and outer walls were 0.042mm and 0.071mm, respectively.

2.7.2 Accuracy Assessment in CT-imaged Physical Phantoms

The signed errors of the measured diameters of the inner and outer walls were 0.026 ± 0.072mm and 0.045 ± 0.094mm, respectively, given a voxel size of $0.39 \times 0.39 \times 0.6$mm^3. The corresponding unsigned errors were 0.057 ± 0.059mm and 0.079 ± 0.071mm. The signed percentage errors of the computer segmented and the measured diameters are presented in Figure 2.8, where mean errors $\pm$ standard deviations are shown as a function of the phantom tube diameters.

Using the Chan-Vese method (the input was the cost image used in our method), we obtained the segmentation results with signed errors of 0.311 ± 0.094mm and 0.097 ± 0.064mm, and unsigned errors of 0.411 ± 0.090mm and 0.359 ± 0.063mm for the inner- and outer-diameters, respectively. Comparing with the Chan-Vese method, our approach is more accurate. Figure 2.9 shows the example segmentation results using the Chan-Vese method and our approach on a 2-D slice of a 3-D CT-imaged physical phantom.

2.7.3 Airway Wall Segmentation

While the inner airway wall surfaces are well visible in CT images, the outer airway wall surfaces are very difficult to segment due to their blurred and discontinuous appearance. The results showed a high accuracy and 3-D consistency of our method (see Figures 2.10 and 2.11). Compared with the manual tracings, our method yielded signed border positioning errors of 0.422 ± 0.381 and -0.127 ± 0.460 voxel for inner and outer wall surfaces, respectively. The corresponding unsigned errors were

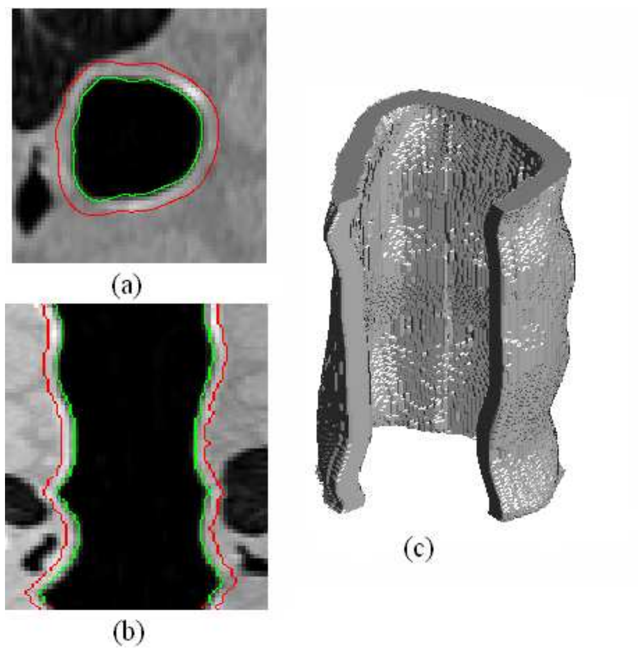
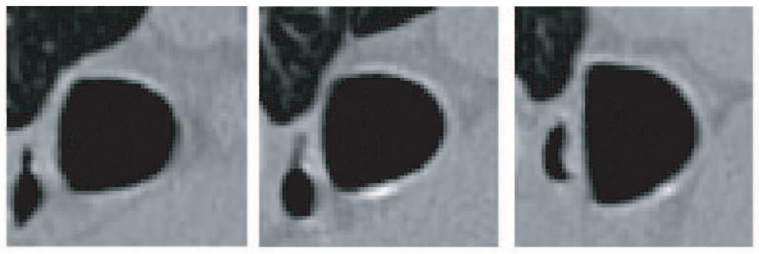


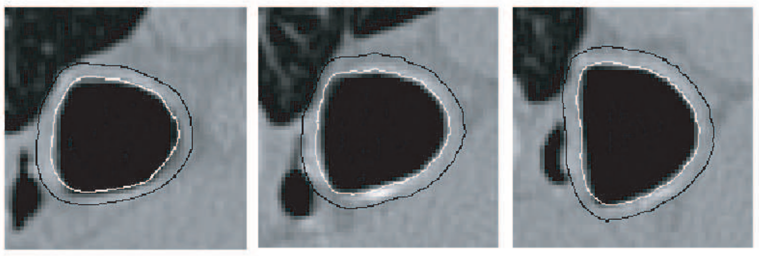
Figure 2.10: Segmented inner and outer walls of human pulmonary airways imaged with multi-detector CT. (a) The transverse, (b) sagittal cross-sections, and (c) 3-D view of an airway segment.

0.657 ± 0.149 and 0.572 ± 0.032 voxel, respectively. We attempted to perform the Chan-Vese method on these data, but without enforcing the separation constraints, their method failed to produce meaningful segmentations.

2.7.4 Cardiac Ventricular Wall Segmentation

Our 3-D method for cardiac ventricular wall segmentation demonstrated low surface positioning errors as well as robust performance when compared with the expert-traced results, although the simultaneous inner- and outer-wall segmentation is challenging due to blurriness and discontinuity of the boundaries in the cardiac ventricular images. Detailed results are given in Table 2.1. Examples of our segmentation


(a)


(b)

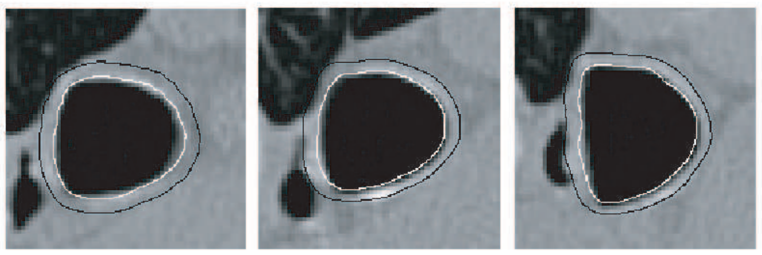

(c)

Figure 2.11: Comparison of computer-segmented and expert-traced inner and outer airway wall borders. (a) Original image, (b) expert-traced borders, and (c) 3-D surface obtained using our method.

results and the corresponding manual tracings are shown in Figure 2.12.

2.7.5 Comparison With the Iterative ICV Method

We demonstrated that our method outperformed the the iterative ICV method (Section 2.6.4.3) for optimally detecting a region bounded by two coupled terrain-like surfaces. The experiments were conducted on both phantoms and 3-D MR cardiac ventricular datasets. Although it always converged, the iterative ICV method

Table 2.1: Surface positioning errors of 3-D cardiac ventricular wall segmentation

	Inner		Outer	
	Our Method	ICV	Our Method	ICV
Normal	0.77 ± 0.43	0.81 ± 0.57	0.79 ± 0.44	0.86 ± 0.56
Patient	0.78 ± 0.48	0.79 ± 0.51	0.95 ± 0.59	1.11 ± 0.73

(a) Unsigned errors in voxels

	Inner		Outer	
	Our Method	ICV	Our Method	ICV
Normal	-0.71 ± 0.54	-0.72 ± 0.59	-0.58 ± 0.43	-0.62 ± 0.45
Patient	-0.49 ± 0.77	-0.54 ± 0.77	0.53 ± 0.59	-0.94 ± 0.72

(b) Signed errors in voxels

frequently did not find the globally optimal solution with respect to the interclass variance objective function.

For the physical phantom images tested, our method outperformed the iterative ICV method by a factor of 0.23% on average in terms of the interclass variance (i.e., the value of the objective function) obtained. Figure 2.13 shows an example of the segmentation results produced by the iterative ICV method and by our method. For this example, our method gave a segmentation with an interclass variance of 9.6071×10^6, compared with 9.5909×10^6 achieved by the iterative ICV method.

For the 3-D MR left ventricular datasets, our method outperformed the iterative ICV method by a factor of about 9% on average in terms of the surface positioning errors. For the fairness of the comparison, the cost functions and geometric constraints used for the iterative ICV method were the same as those used for our method. Table 2.1 shows the signed and unsigned positioning errors of results

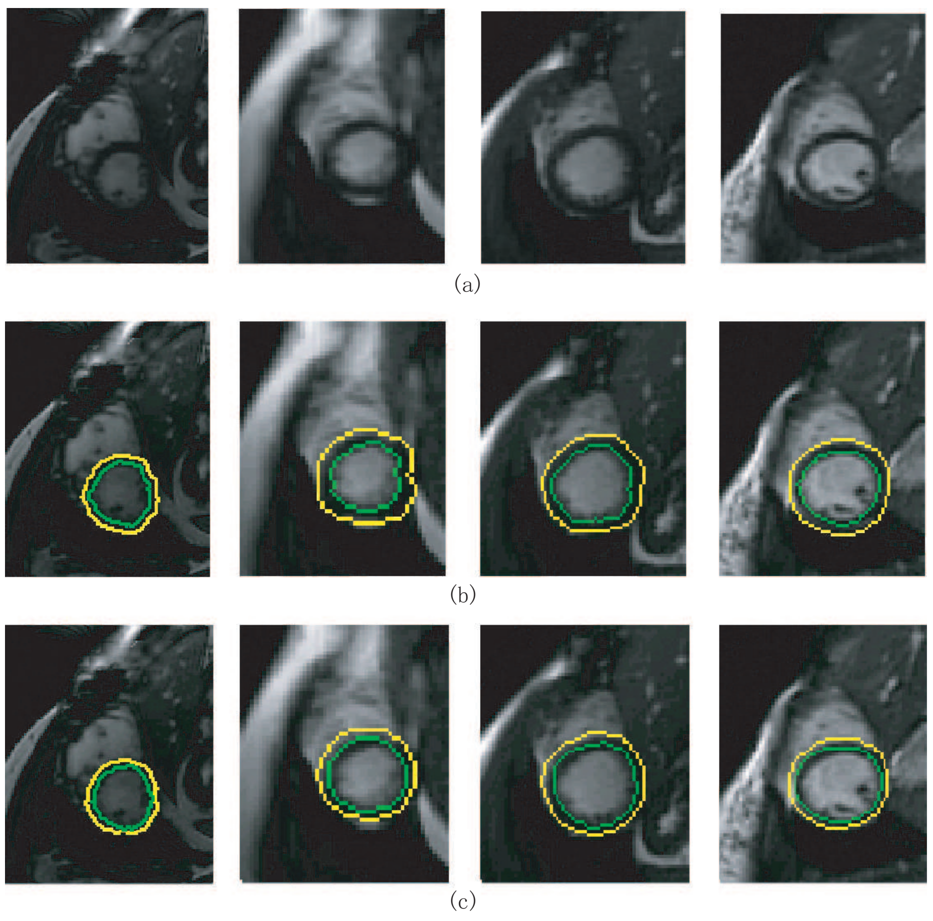


Figure 2.12: Segmentation of MR ventricular walls. (a) Original images. (b) Computer-segmented inner- and outer-wall borders using our method. (c) Manually identified inner- and outer-walls.

obtained by the iterative ICV method.

2.7.6 Execution Time

The execution time of our method mainly depends on the number of shape probings performed (i.e., the number of hull vertices of the constructed convex hull) and the time for performing each probing oracle.

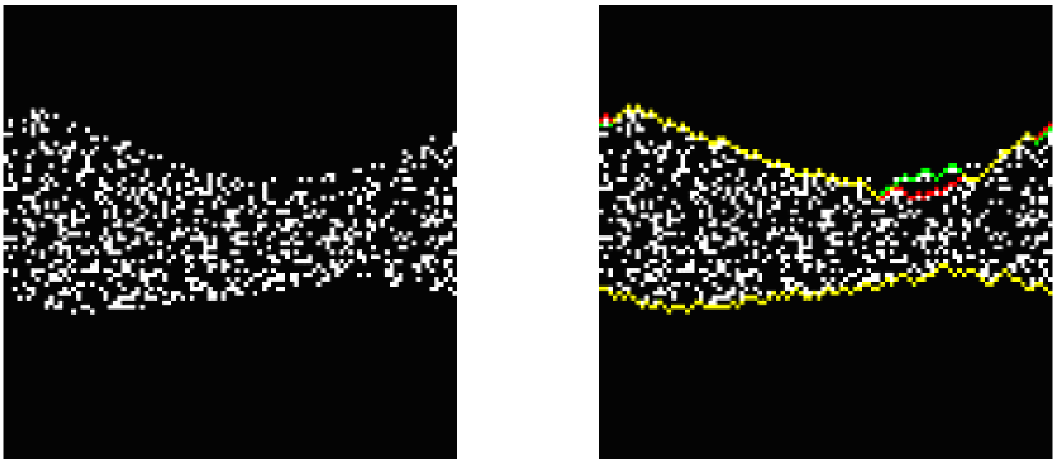


Figure 2.13: Comparison of segmentation results for an example computer phantom image. (a) The phantom image. (c) The overlay of the results by our method and by the iterative ICV method. The green contour is the result from our method, the red one shows the result from the iterative method, and the yellow one is the overlapping part.

The average execution times for employing each probing oracle are shown in Table 2.2. The minimum *s*-*t* cut was computed using Boykov and Kolmogorov's maximum flow algorithm [9] with a "forward-star" graph representation.

For the airway wall segmentation, the number of the hull vertices of $CH(\mathcal{P})$ ranged from 1.50% to 2.86% of the image size; while for the 3-D ventricular datasets,

Table 2.2: Average execution times of each shape probing process

Image Size	Execution Time (s)	Image Size	Execution Time (s)
$30^2 \times 40$	0.5	$80^2 \times 40$	5.9
$50^2 \times 40$	2.1	$100^2 \times 40$	15.7

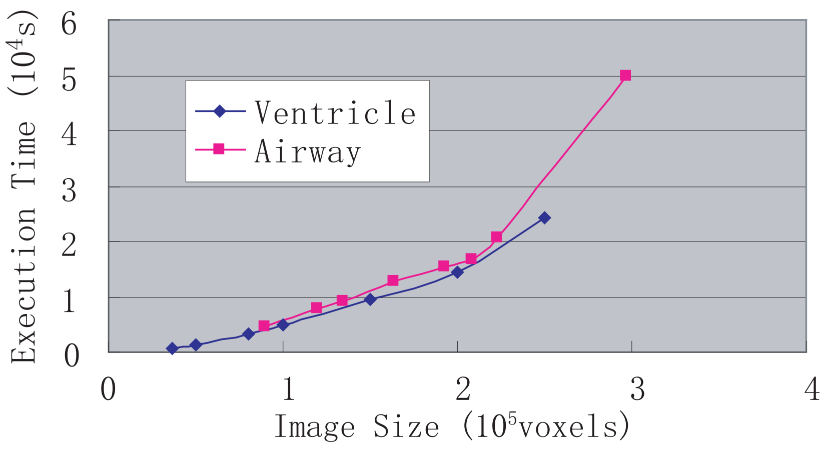


Figure 2.14: Execution times for 3-D airway CT images and 3-D ventricular MR images.

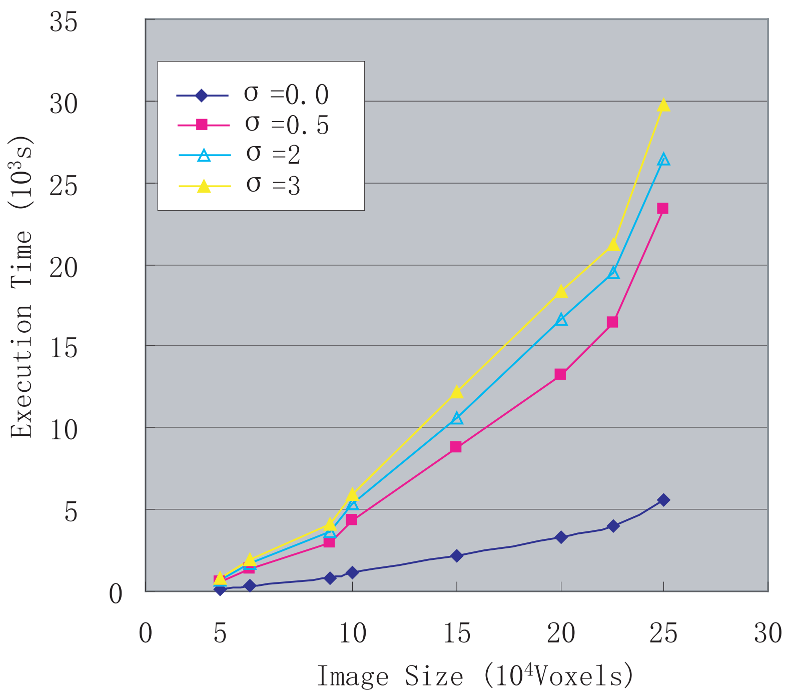


Figure 2.15: Execution times for computer phantoms with different levels of Gaussian noise ($\sigma = 0.0, 0.5, 2.0, 3.0$).

the percentage ranged between 1.64% and 2.89%. The average total execution times of our algorithm are shown in Figure 2.14.

For the computer phantoms with no Gaussian noise added ($\sigma = 0.0$), the number of hull vertices of $CH(\mathcal{P})$ was ranging from 1.32% to 1.61% of the image size. Our experiments showed that the added Gaussian noise affected the performance of our algorithm significantly. The average execution times on the computer phantoms with different levels of Gaussian noise are shown in Figure 2.15.

2.7.7 Performance of the Approximation Method

The performance of our developed approximation method was tested on computer phantoms, physical phantoms, 3-D CT images of pulmonary airway walls, and 3-D MR images of cardiac ventricular walls, and was compared to that of our exact algorithm. Our experiments demonstrated that the approximation method ran much faster than the exact algorithm while the segmentation accuracy was highly comparable to that of the exact algorithm.

We chose $\epsilon = 0.01, 0.02, 0.05, 0.1, 0.2$, and 0.5 to test the approximation method on segmentation accuracy. In addition to the surface positioning errors, the so-called *approx-opt ratio*, which is the ratio of the objective function value produced by the approximation algorithm over that yielded by the exact algorithm, was also used to measure the segmentation performance. Interestingly, while $\epsilon \leq 0.05$, the approximation method always computed the optimal solutions to the problem of minimizing intraclass variance $\mathcal{E}_{CV}(R)$ for all the datasets we tested. For the six physical phantoms, even while we took $\epsilon = 0.1, 0.2, 0.5$, we were still able to obtain the optimal solutions by using the approximation method. Table 2.3 presents, for the different ϵ's, the average approx-opt ratios, the maximum approx-opt ratios, and the percentage of the test cases that the approximation algorithm did not produce the optimal solution. Table 2.4 and Table 2.5 show the average approx-opt ratio and the average surface positioning errors with different ϵ's for the airway wall images and the ventricular wall images, respectively. Note that for $\epsilon = 0.05$ in both tables, the objective function values were optimal and the average surface positioning errors were the same as those obtained by the exact algorithm. Thus, our experiments showed that the approximation method is able to produce highly comparable segmentation results as those produced by the exact algorithm even for relatively large ϵ's.

Table 2.3: Segmentation accuracy of the approximation method on computer phantoms

ϵ	0.05	0.1	0.2	0.5
Average Approx-opt Ratio	100.00%	100.03%	100.22%	101.02%
Max Approx-opt Ratio	100.00	100.15%	100.80%	104.35%
% of Non-Optimal Results	1.25%	7.50%	25.00%	51.25%

The running times of the approximation method were evaluated on computer phantoms and clinical data. Our experiments demonstrated great improvement of running time over the exact algorithm. Figure 2.16 shows the comparison of the execution times of the approximation algorithm with different ϵ's on the computer phantom images against the exact algorithm. We did not explicitly add Gaussian noise to those phantoms. Our further experiments on those phantom images with added

Table 2.4: Segmentation accuracy of the approximation method on 3-D CT airway wall images

	Average	Unsigned Error(Pixels)	
ϵ	Approx-opt Ratio	Inner Wall	Outer Wall
0.05^*	100.00%	0.4220 ± 0.3810	-0.1270 ± 0.4600
0.1	100.10%	0.4222 ± 0.3810	-0.1270 ± 0.4601
0.2	100.52%	0.4220 ± 0.3810	-0.1270 ± 0.4600
0.5	100.97%	0.4220 ± 0.3810	-0.1270 ± 0.4600
	Average	Signed Error(Pixels)	
ϵ	Approx-opt Ratio	Inner Wall	Outer Wall
0.05^*	100.00%	0.6570 ± 0.1490	0.5720 ± 0.0320
0.1	100.10%	0.6562 ± 0.1490	0.5726 ± 0.0320
0.2	100.52%	0.6570 ± 0.1490	0.5724 ± 0.0320
0.5	100.97%	0.6570 ± 0.1490	0.5724 ± 0.0320

Table 2.5: Segmentation accuracy of the approximation method on 3-D MR ventricular images

	Average	Unsigned Error(Pixels)	
ϵ	Approx-opt Ratio	Inner Wall	Outer Wall
0.05^*	100.00%	-0.7103 ± 0.5384	-0.5841 ± 0.4367
0.1	100.14%	-0.7090 ± 0.5375	-0.5859 ± 0.4430
0.2	100.86%	-0.7076 ± 0.5354	-0.5860 ± 0.4440
0.5	101.43%	-0.7201 ± 0.5470	-0.5869 ± 0.4275
	Average	Signed Error(Pixels)	
ϵ	Approx-opt Ratio	Inner Wall	Outer Wall
0.05^*	100.00%	0.7776 ± 0.4356	0.7929 ± 0.4367
0.1	100.14%	0.7742 ± 0.4366	0.8068 ± 0.4430
0.2	100.86%	0.7674 ± 0.4398	0.8207 ± 0.4440
0.5	101.43%	0.8033 ± 0.4288	0.7516 ± 0.4275

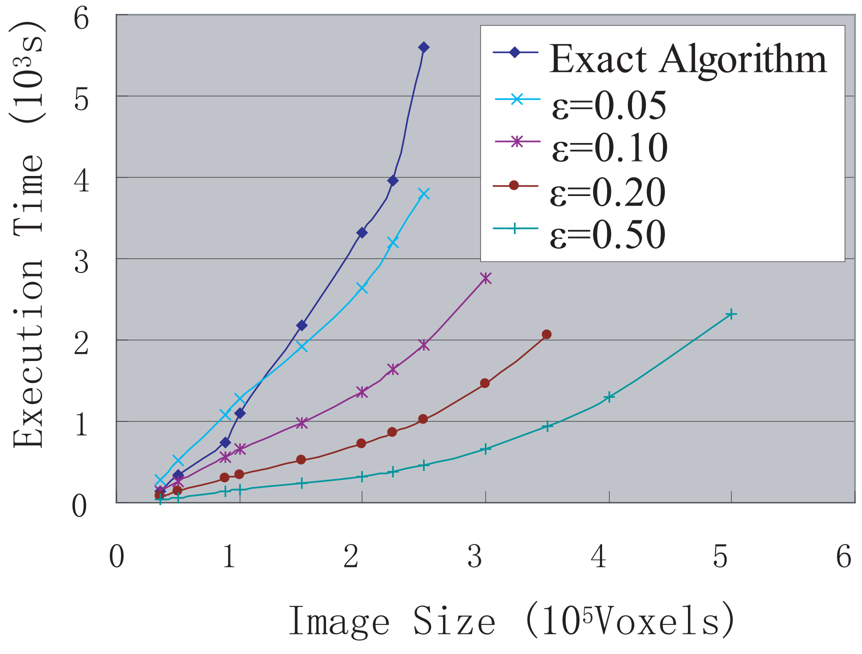


Figure 2.16: Comparison of execution times on computer phantoms by the approximation algorithm against the exact algorithm.

Gaussian noise ($\sigma = 0.5, 2.0, 3.0$) revealed that the efficiency of the approximation method did not decrease with the presence of noise. In contrast, the performance of the exact algorithm on running time deteriorated significantly for noisy images. The dramatic improvement on the execution times of the approximation algorithm over the exact algorithm are shown in Figures 2.17 and 2.18, respectively, for 3-D airway CT images and 3-D ventricular MR images, which show less intensity homogeneity.

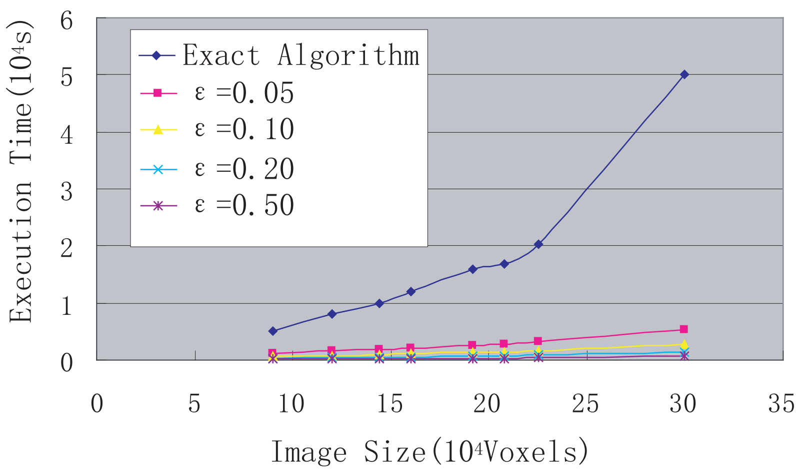


Figure 2.17: Comparison of execution times on 3-D airway CT image data by the approximation algorithm against the exact algorithm.

2.7.8 Performance of the SSD Method

Our SSD method for liver lesion segmentation demonstrated low surface positioning errors as well as robust performance when compared with the expert-traced

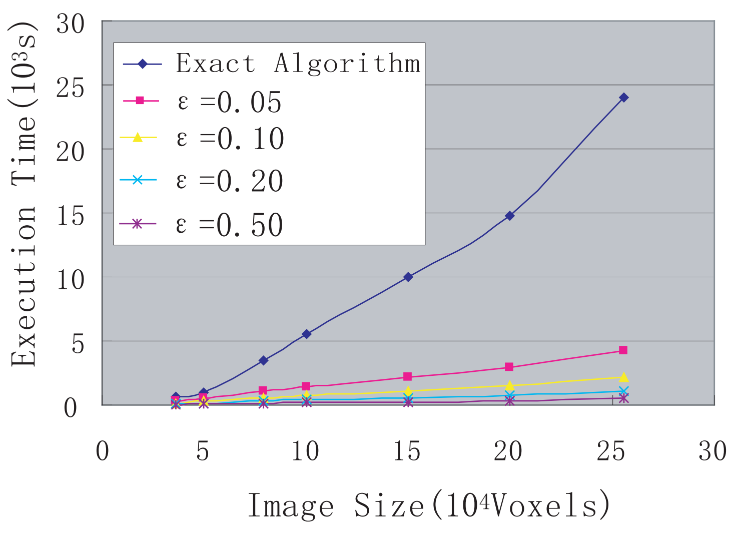


Figure 2.18: Comparison of execution times on 3-D ventricular MR images by the approximation algorithm against the exact algorithm.

results. The average signed and unsigned positioning errors were -0.07 ± 0.31 and 0.77 ± 0.16 voxels. Examples of our segmentation results and the corresponding manual tracings are shown in Figure 2.19.

For lung tumor data, the average signed and unsigned positioning errors were -0.03 ± 0.10 and 0.78 ± 0.13 voxels compared with the expert-traced results. Figure 2.20 shows examples of our segmentation results.

The average execution times for the initial preflow and the total running time of our algorithm are shown in Table 2.6. It shows that the total execution time is just a few times higher than the single non-parametric maximum-flow algorithm while our exact double surface algorithm takes $O(n)$ runs of the non-parametric maximum-flow algorithm.

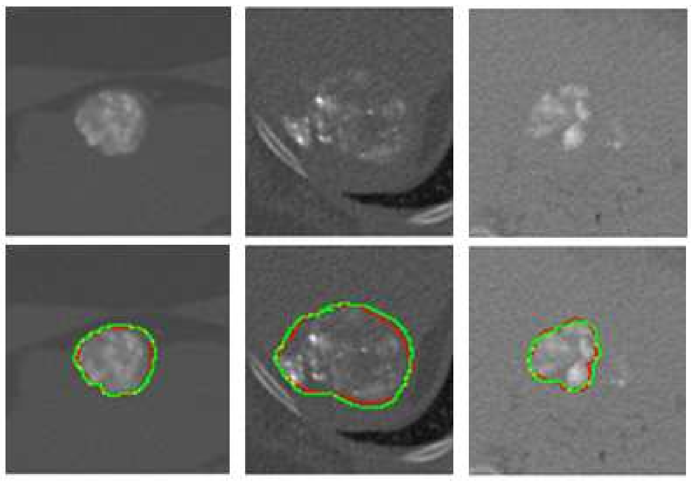


Figure 2.19: Example slices showing our segmentation result (red) and expert-traced result (green) on liver lesion data. Top row shows the original images.

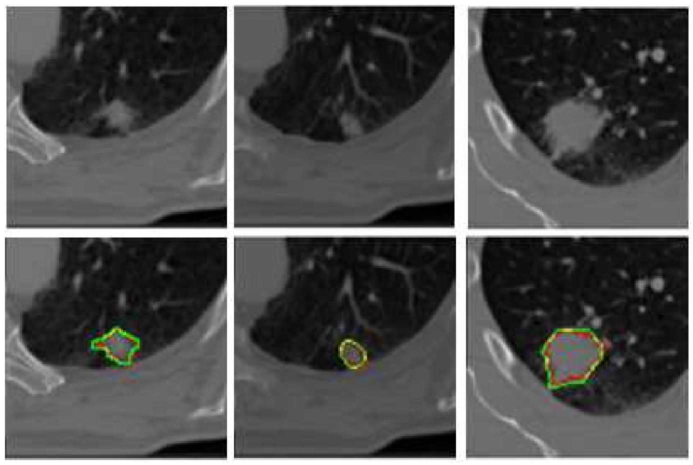


Figure 2.20: Example slices showing our segmentation result (red) and expert-traced result (green) on lung tumor data. Top row shows the original images.

Table 2.6: Average execution times

Image Size	Single Max-flow (s)	Total Time (s)
$40 \times 40 \times 40$	5.0	18.8
$60 \times 60 \times 40$	12.0	62.5
$80 \times 80 \times 40$	20.0	145.5
$100 \times 100 \times 40$	33.1	272.5
$140 \times 140 \times 40$	66.1	733.9

2.8 Discussion

The algorithm efficiently detects the globally optimal region in the entire region-of-interest (ROI), enabling highly accurate image segmentation that is a prerequisite for reliable quantitative image analyses. One obvious limitation is that that our method only allows optimal detection of a single object (region). This may prevent the method from being used for a wider range of medical applications, in which multiple interacting objects are present. The extension of our method for dealing with simultaneous multiple object segmentation is an interesting and challenging future work. The second limitation is that the method can only detect those surfaces that can be unfolded to be terrain-like, including cylindrical or tubular surfaces. Unfolding techniques have been developed for objects of relatively complex shapes, such as liver [41], knee bone and cartilage [52, 43], heart [67], and pulmonary airway and vascular trees [55]. However, one may experience difficulty to unfold more complex objects into terrain-like surfaces. Thus, another challenge is to make the method work with complex topology.

The approximation algorithm was demonstrated as having much faster running times while still producing highly comparable segmentation results as the exact algorithm. Unfortunately, we have not yet proven a tight bound of the approximation ratio, although the experiments indicated that it was pretty small. We plan to further study the approximation ratio of our approximation algorithm.

CHAPTER 3
FIELD DECOMPOSITION PROBLEM

3.1 Introduction and Problem Modeling

In this chapter, we propose to use two orthogonal directions to deliver an IM (i.e., horizontal and vertical) and formulate the following **IM matrix orthogonal decomposition (IMOD)** problem: Given an $m \times n$ non-negative integer matrix $A = (a_{i,j}) \in \mathbb{Z}^{+m \times n}$ (i.e., an IM) and an integer $\lambda \geq 1$, find two matrices (i.e., sub-IMs) $Q = (q_{i,j}), R = (r_{i,j}) \in \mathbb{Z}^{+m \times n}$ such that:

(1) $A = \lambda Q + R$,

(2) the sum of the *horizontal complexity* $C_H(Q)$ of Q and the *vertical complexity* $C_V(R)$ of R is minimized.

Since the complexity of an IM is not well defined, we consider the following two sets of complexity definitions:

1. *Positive Gradient Sum Cost*:

$$\begin{aligned} C_H(Q) &= \sum_{i=1}^{m} \left(q_{i,1} + \sum_{j=2}^{n} \max(0, q_{i,j} - q_{i,j-1}) \right) \\ C_V(R) &= \sum_{j=1}^{n} \left(r_{1,j} + \sum_{i=2}^{m} \max(0, r_{i,j} - r_{i-1,j}) \right) \end{aligned} \tag{3.1}$$

2. Total Variation Cost:

$$\begin{aligned} C_H(Q) &= \sum_{i=1}^{m}\sum_{j=2}^{n}(q_{i,j} - q_{i,j-1})^2 \\ C_V(R) &= \sum_{j=1}^{n}\sum_{i=2}^{m}(r_{i,j} - r_{i-1,j})^2 \end{aligned} \tag{3.2}$$

The rationale behind the positive gradient sum cost definition is based on the following observations. The minimum number of MUs $T_{MU}(B[i])$ for delivering each row $B[i]$ of B equals to $\Big(b_{i,1} + \sum_{j=2}^{n} \max(0, b_{i,j} - b_{i,j-1})\Big)$ [30]. The complexity $C(A)$ measures the total minimum number of MUs of all rows of the IM A when it is delivered horizontally. Hence, the complexity of an IM that we use is closely related to the number of MUs of the IM.

For the total variation cost, Süss and Küfer found that if the total variation of an intensity map increases, the minimum number of MUs used to deliver it is expected to increase as well [72]. This indicates that the total variation is a good measure of the complexity of an IM, especially number of MUs of the IM.

Then, the sub-IM Q and R are delivered in two orthogonal directions. It is helpful to note that two IMs A and B with $A = \lambda \cdot B$ for some integer $\lambda > 1$, can be delivered by the same set of MLC-apertures. By adding the factor λ, it is very likely to reduce the total number of MLC-apertures since this can reduce the elements in R and thus the number of MLC-apertures used to deliver R. Most of current approaches for the SLS problem are based on a method for reducing the intensity level of IM matrices, then compute a set of MLC-apertures for the IM matrices with a smaller maximum intensity level [82, 15, 64, 65, 73, 16]. Our decomposition results in two “simpler” sub-IMs with smaller maximum intensity level, which, in turn, yields a more efficient delivery plan using fewer MLC-apertures and/or less total number of MUs.

We develop an efficient algorithm for the IM matrix orthogonal decomposition problem. Our algorithm is based on a non-trivial graph construction scheme, which enables us to formulate the decomposition problem as computing a minimum s-t cut in a 3-D geometric multi-pillar graph. The algorithm was implemented and tested on clinical datasets.

3.2 Algorithm for the Positive Gradient Sum Cost

This section presents our efficient IM matrix orthogonal decomposition algorithm. We start from the first cost function, and we model the problem as a minimum s-t cut problem on a 3-D geometric multi-pillar graph G by a complicated graph transformation scheme. The s-t cuts in the graph G characterize the solution space of the IMOD problem.

3.2.1 Modeling the IMOD Problem

We define a 3-D geometric multi-pillar graph $G = (V, E)$ on a 2-D $m \times n$ grid Γ from the given IM matrix $A = (a_{i,j})_{m\times n}$ and the integer $\lambda > 0$, as follows.

Note that each element $a_{i,j}$ of A has $\lfloor \frac{a_{i,j}}{\lambda} \rfloor + 1$ possible decompositions, that is, $a_{i,j} = \lambda \cdot q_{i,j} + r_{i,j}$ for $q_{i,j} = 0, 1, \ldots, \lfloor \frac{a_{i,j}}{\lambda} \rfloor$. We then introduce exact one vertex

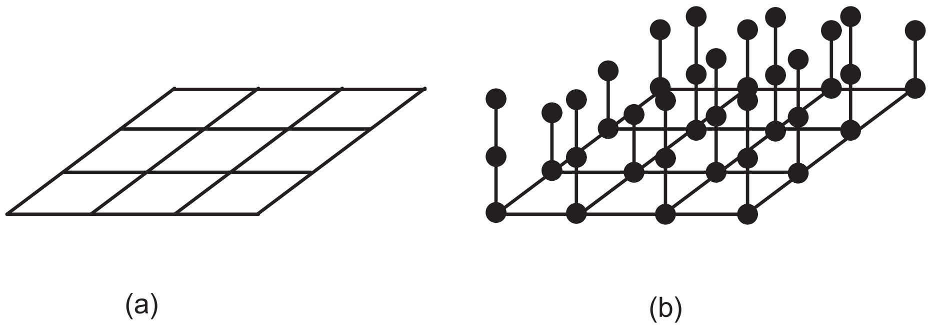


Figure 3.1: Illustration of 2-D grid and multi-pillar vertices. (a) A 2-D grid. (b) Multi-pillar vertices.

$v(i,j,k) \in V$ for each possible decomposition $(q_{i,j}, r_{i,j})$ of $a_{i,j}$, where $k = q_{i,j} + 1$. For the ease of our graph construction, a special vertex $v(i,j,\lfloor\frac{a_{i,j}}{\lambda}\rfloor + 2)$ is put in G for each $a_{i,j}$. Thus, there is a set $Col(i,j)$ of $\lfloor\frac{a_{i,j}}{\lambda}\rfloor + 2$ vertices in G corresponding to $a_{i,j}$ of the IM matrix A; $Col(i,j) = \{v(i,j,k) | k = 1, 2, \ldots, \lfloor\frac{a_{i,j}}{\lambda}\rfloor + 1, \lfloor\frac{a_{i,j}}{\lambda}\rfloor + 2\}$, called the (i,j)-pillar of G (see Figure 3.1(a) and (b) for an example). In addition, we add two dummy vertices, a source s and a sink t, in G to formulate our IMOD problem as computing a minimum s-t cut in G.

Before adding edges into G, we first introduce some notations. The *height* $h(i,j)$ of the pillar $Col(i,j)$ is $\lfloor\frac{a_{i,j}}{\lambda}\rfloor + 2$. We say that two pillars $Col(i,j)$ and $Col(i',j')$ are *adjacent* to each other if $|i - i'| + |j - j'| = 1$. For each pillar $Col(i,j)$, $v(i,j,1)$ is called the *base* vertex and $v(i,j,\lfloor\frac{a_{i,j}}{\lambda}\rfloor + 2)$ is called the *top* vertex of the pillar.

We are now ready to put directed edges in G. Our basic idea is that, for a feasible decomposition $(q_{i,j}, r_{i,j})$ of each $a_{i,j}$ of A, we expect that

- $q_{i,j}$ partitions the (i,j)-pillar into two subsets, $S_{i,j} = \{v(i,j,k) | k = 1, 2, \ldots, q_{i,j}+$

1} and $\overline{S}_{i,j} = \{v(i,j,k) | k = q_{i,j}+1, \ldots, \lfloor \frac{a_{i,j}}{\lambda} \rfloor + 2\}$, and $\mathcal{C} = \left(\bigcup_{i,j} S_{i,j}, \bigcup_{i,j} \overline{S}_{i,j}\right)$ is a valid s-t cut in G (a cut $\mathcal{C}$ in G is valid if the total edge weight $w(\mathcal{C})$ of $\mathcal{C}$ is finite); and

- the total weight of the cut $\mathcal{C}$ equals to the sum of the horizontal complexity $C_H(Q)$ of $Q = (q_{i,j})_{m\times n}$, with

$$C_H(Q) = \sum_{i=1}^{m} \left(q_{i,1} + \sum_{j=2}^{n} \max(0, q_{i,j} - q_{i,j-1}) \right),$$

  and the vertical complexity $C_V(R)$ of $R = (r_{i,j})_{m\times n}$, with

$$C_V(R) = \sum_{j=1}^{n} \left(r_{1,j} + \sum_{i=2}^{m} \max(0, r_{i,j} - r_{i-1,j}) \right).$$

We thus introduce two types of directed edges: one is used to enforce the feasibility of a solution, which includes two subsets of edges E_{mo} and E_{ad}; the other is used to realize total complexity of a feasible decomposition of A, which consists of four subsets of edges, E_{vt}, E_{hz}, E_r, and E_q. The four subsets are used to realize different parts of the complexity measurement, $\sum_{j=1}^{n}\sum_{i=1}^{m-1} \max\{0, r_{i+1,j} - r_{i,j}\}$, $\sum_{i=1}^{m}\sum_{j=1}^{n-1} \max\{0, q_{i,j+1} - q_{i,j}\}$, $\sum_{j=1}^{n} r_{1,j}$, and $\sum_{i=1}^{m} q_{i,1}$, of a feasible decomposition (Q, R) of A, respectively.

- *The edges in E_{mo}*: On each pillar $Col(i,j)$, an edge of weight $+\infty$ is added from every vertex $v(i,j,k)$ to vertex $v(i,j,k-1)$ for $k = 2, 3, \ldots, h(i,j)$. The set of edges in E_{mo} is used to guarantee the monotonicity property of the solution, since exactly one vertex on each pillar of G can be used to define a feasible solution to the IMOD problem.

- *The edges in* E_{ad}: An edge of weight $+\infty$ is put in E_{ad} from the source s to the base vertex of each pillar. Meanwhile, an edge of weight $+\infty$ is added from the top vertex of each pillar to the sink t.

- *The edges in* E_{vt}: Consider any two adjacent pillars $Col(i,j)$ and $Col(i+1,j)$ on the same column of Γ ($0 < i < m$ and $0 < j \leq n$). Recall that edges in E_{vt} are used to realize the part $\sum_{j=1}^{n}\sum_{i=1}^{m-1}\max\{0, r_{i+1,j} - r_{i,j}\}$ of the vertical complexity $C_V(R)$. We thus expect that the edges between $Col(i,j)$ and $Col(i+1,j)$ are put in such a way that the total weight of the edges that contribute to the corresponding cut of the decomposition (Q,R) equals to $\max\{0, r_{i+1,j} - r_{i,j}\}$. Let us take a close look at the term $r_{i+1,j} - r_{i,j}$.

$$\begin{aligned} r_{i+1,j} - r_{i,j} &= (a_{i+1,j} - \lambda \cdot q_{i+1,j}) - (a_{i,j} - \lambda \cdot q_{i,j}) \\ &= (a_{i+1} - a_{i,j}) + \lambda \cdot (q_{i,j} - q_{i+1,j}) \\ &= \lambda \cdot \left(\lfloor \frac{a_{i+1,j} - a_{i,j}}{\lambda} \rfloor + (q_{i,j} - q_{i+1,j}) \right) + (a_{i+1,j} - a_{i,j}) \% \lambda \end{aligned}$$

To give an intuitive idea on how to introduce edges in E_{vt}, we assume that $a_{i+1,j} \geq a_{i,j}$. Based on the equation above, we could "align" vertex $v(i,j,k) \in Col(i,j)$ with vertex $v(i+1,j,k') \in Col(i+1,j)$ such that $k' = k + \lfloor \frac{a_{i+1,j} - a_{i,j}}{\lambda} \rfloor$

and add an edge in E_{vt} from $v(i,j,k)$ to $v(i+1,j,k'+1)$ with a weight of $(a_{i+1,j}-a_{i,j})\%\lambda$. Notice that when the difference between $q_{i,j}$ and $q_{i+1,j}$ increases by 1, the complexity increases by λ. To accommodate this, we might put an edge from $v(i,j,k)$ to $v(i+1,j,k')$ with a weight of $\lambda-(a_{i+1,j}-a_{i,j})\%\lambda$.

Formally, for every vertex $v(i,j,k)$ with $i<m$ and $0<k<h(i,j)$ in G, we define its *lower neighbor* and its *upper neighbor* on the pillar $Col(i+1,j)$ adjacent to $Col(i,j)$:

1. if $1 \leq (\lfloor \frac{a_{i+1,j}-a_{i,j}}{\lambda} \rfloor + k) \leq h(i+1,j)-1$, the lower neighbor of $v(i,j,k)$ is $v(i+1,j,\lfloor \frac{a_{i+1,j}-a_{i,j}}{\lambda} \rfloor + k)$;
2. if $v(i+1,j,z_l)$ is the lower neighbor of $v(i,j,k)$ and $z_l < h(i+1,j)$, the upper neighbor of $v(i,j,k)$ is $v(i+1,j,z_l+1)$.

Note that we do not define neighbors for the top vertices.

Then, for each non-base vertex $v(i,j,k)$ on $Col(i,j)$ (i.e., $k>1$), two directed edges are put in E_{vt}:

1. a *lower edge* from $v(i,j,k)$ to its lower neighbor, and
2. an *upper edge* from $v(i,j,k)$ to its upper neighbor (see Figure 3.2(a)).

The weight of the lower edge of $v(i,j,k)$ is assigned as $(\lambda-[a_{i+1,j}-a_{i,j}]\%\lambda)$, and the weight of the upper edge of $v(i,j,k)$ is set to $([a_{i+1,j}-a_{i,j}]\%\lambda)$. Meanwhile, for the base vertex $v(i,j,1)$, we put an *upper-base edge* with a weight of $([a_{i+1,j}-a_{i,j}]\%\lambda)$ from $v(i,j,1)$ to its upper neighbor on $Col(i+1,j)$. If the lower neighbor

$v(i+1, j, z_l)$ of $v(i, j, 1)$ is not the base vertex of the pillar $Col(i+1, j)$, then a set of directed edges (called the *lower-base edges*) from $v(i, j, 1)$ to $v(i+1, j, k')$ for every $2 \leq k' \leq z_l$ is introduced into E_{vt}; the weight of each of these edge is λ. Note that all the above edges are added only when the corresponding neighbor exists or the neighbor is not the base vertex. Figure 3.2(a) illustrates the construction with an example.

- *The edges in E_{hz}*: The edges in E_{hz} are used to realize the part $\sum_{i=1}^{m} \sum_{j=1}^{n-1} \max\{0, q_{i,j+1} - q_{i,j}\}$ of the horizontal complexity $C_H(Q)$. Consider any two adjacent pillar $Col(i, j)$ and $Col(i, j+1)$ on the same row of Γ ($0 < i \leq m$ and $0 < j < n$). For each non-base vertex $v(i, j+1, k)$ on $Col(i, j+1)$ (i.e., $k \neq 1$), if $k < \min\{h(i, j), h(i, j+1)\}$, we put an edge from $v(i, j+1, k)$ to $v(i, j, k) \in Col(i, j)$ with a weight of 1 (see Figure 3.2(b)). If the height $h(i, j+1)$ of $Col(i, j+1)$ is larger than the height $h(i, j)$ of $Col(i, j)$, a directed edge of weight 1 is also introduced from each vertex $v(i, j+1, k)$ on the pillar $Col(i, j+1)$ to the top vertex $v(i, j, h(i, j))$ of $Col(i, j)$, for every $k = h(i, j), \ldots, h(i, j+1) - 1$.

- *The edges in E_r*: The part $\sum_{j=1}^{n} r_{1,j}$ of the vertical complexity $C_V(R)$ is intended to be realized using edges in E_r. The top vertex of each pillar $Col(1, j)$ on the first row of Γ (i.e., for every $j = 1, 2, \ldots, n$), has a directed edge with a weight of $a_{1,j}\%\lambda$ from the source s. Additionally, for each vertex $v(1, j, k)$ of every pillar $Col(1, j)$, if $v(1, j, k)$ is not the base vertex or the top vertex, then we add a directed edge of weight λ from the source s to $v(1, j, k)$. Figure 3.2(a) shows

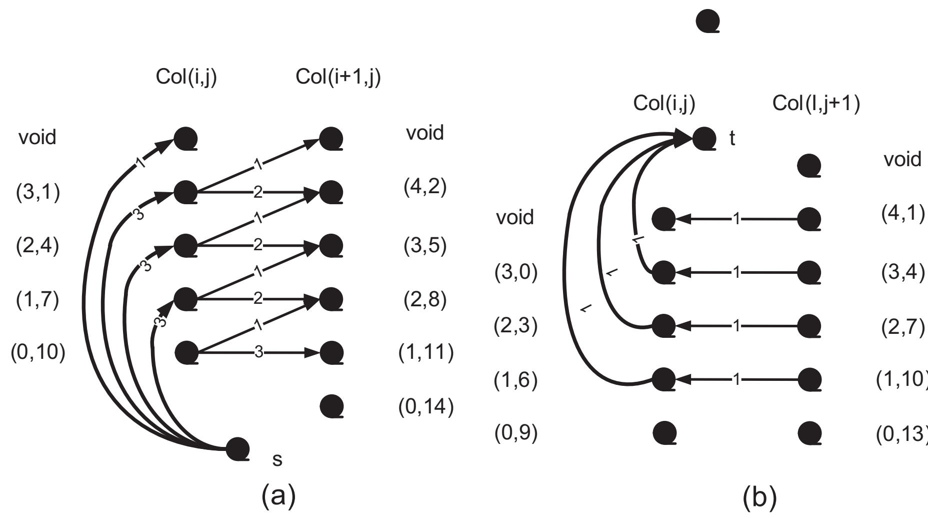


Figure 3.2: Illustrating the construction of the multi-pillar graph G from a given IM matrix A. Each element $a_{i,j}$ corresponds to a pillar $Col(i,j)$ of $\lfloor a_{i,j} \rfloor + 1$ vertices, each specifying a possible decomposition $(q_{i,j}, r_{i,j})$ of $a_{i,j}$, and a top vertex of the pillar, which is denoted by "void". (a) The construction of E_{vt} (thin edges) and E_r (thick edges) of the case $a_{i,j} = 10$, $a_{i+1,j} = 14$, and $\lambda = 3$. Each vertex of $Col(i,j)$, except the base and top ones, has in E_{vt} a lower edge with a weight of 2 $(= (\lambda - [a_{i+1,j} - a_{i,j}]\%\lambda))$ and an upper edge with a weight of 1 $(= ([a_{i+1,j} - a_{i,j}]\%\lambda))$ to $Col(i+1,j)$. The base vertex of $Col(i,j)$ has in E_{vt} an upper-base edge of weight 1 $(= ([a_{i+1,j} - a_{i,j}]\%\lambda))$ and *only* one lower-base edge of weight λ to $Col(i+1,j)$ in this case. Assume that $i = 1$. The source s has in E_r a directed edge with a weight of 1 $(= a_{1,j}\%\lambda)$ to the top vertex of $Col(1,j)$ and a directed edge of weight λ to each of other non-base vertices of $Col(1,j)$. (b) The construction of E_{hz} (thin edges) and E_q (thick edges) of the case $a_{i,j} = 9$, $a_{i,j+1} = 13$, and $\lambda = 3$. Each non-base vertex $v(i,j+1,k) \in Col(i,j+1)$ has in E_{hz} a directed edge of weight 1 to the corresponding vertex $v(i,j,k) \in Col(i,j)$ if $k < 5$ (the smaller height of $Col(i,j)$ and $Col(i,j+1)$). For each non-top vertex $v(i,j+1,k)$ with $k \geq 5$, a directed edge of weight 1 is introduced to the top vertex of $Col(i,j)$. Assuming $j = 1$, each vertex of $Col(i,j)$, except the base and top ones, has a directed edge with a weight of 1 to the sink t.

an example for this construction. Note that there is no edge in E_r from source s to the base vertex of each pillar $Col(1, j)$.

- *The edges in* E_q: The edges in E_q are used to realize the part $\sum_{i=1}^{m} q_{i,1}$ of the horizontal complexity $C_H(Q)$. For each vertex $v(i, 1, k)$ of every pillar Col(i, 1) on the first column of Γ (i.e., $i = 1, 2, \ldots, m$), if $v(i, 1, k)$ is not the base vertex or the top vertex, then we put a directed edge of weight 1 from $v(i, 1, k)$ to the sink t (see Figure 3.2(b) when $j = 1$).

Hence, the edge set E of G is $E_{vt} \cup E_{hz} \cup E_q \cup E_r \cup E_{mo} \cup E_{ad}$. We thus complete the construction of the multi-pillar graph G.

3.2.2 Computing an Optimal Matrix Orthogonal Decomposition

The graph G constructed in Section 3.2.1 allows us to find the optimal matrix orthogonal decomposition for the given IM matrix A, by computing a minimum-weight s-t cut in G. In order to do that, below we prove the following facts: (1) Any valid s-t cut $\mathcal{C}$ defines a feasible decomposition (Q, R) of A (i.e., $A = \lambda \cdot Q + R$), such that $C_H(Q) + C_V(R) = w(\mathcal{C})$; (2) any feasible decomposition (Q, R) of A specifies a valid s-t cut $\mathcal{C}$ in G, such that $w(\mathcal{C}) = C_H(Q) + C_V(R)$. Consequently, a valid s-t cut in G with the minimum total edge weight can specify an optimal matrix orthogonal decomposition of A.

We first argue that any valid s-t cut in G corresponds to a feasible decomposition of A and any feasible decomposition corresponds to a valid s-t cut in G.

Observation 1. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, the base vertices of all pillars are included in the source set S and all top vertices of pillars are included in the sink set $\overline{S}$.*

Proof. The observation can be proved by contradiction. If a base vertex is included in the sink set $\overline{S}$, since we add edges of weight $+\infty$ from the source s to every base vertex during the construction of $E_{ad} \subset E$, the total edge weight $w(\mathcal{C})$ is thus equal to $+\infty$, which is a contradiction. Similarly, all top vertices have to be in the sink set $\overline{S}$. □

Observation 2. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, if a vertex $v(i,j,k) \in Col(i,j)$ is in the source set S, each vertex $v(i,j,k')$ with $k' < k$ is also in S; if a vertex $v(i,j,k) \in Col(i,j)$ is in the sink set $\overline{S}$, every vertex $v(i,j,k')$ with $k' > k$ is also in the sink set $\overline{S}$.*

Proof. We prove this observation also by contradiction. Without loss of generality (WLOG), assume that $v(i,j,k)$ is in the source set S and $v(i,j,k-1)$ is in the sink set $\overline{S}$. Due to the construction of edges in E_{mo}, there is an edge of weight $+\infty$ from vertex $v(i,j,k)$ to vertex $v(i,j,k-1)$ for any $k = 2, 3, \ldots, h(i,j)$. Thus, $w(\mathcal{C}) = +\infty$, which is a contradiction. Hence, if a vertex $v(i,j,k) \in Col(i,j)$ is in the source set S, each vertex $v(i,j,k')$ with $k' < k$ is also in S. A similar argument can show that the second part of the observation holds. □

Thus, we can define a matrix $D = d_{i,j} \in \mathbb{Z}^{+^{m \times n}}$ $(1 \leq d_{i,j} \leq h(i,j) - 1)$ to describe a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, such that for each pillar $Col(i,j)$, $S \cap$

$Col(i,j) = \{v(i,j,k) | k = 1, 2, \ldots, d_{i,j}\}$ and $\overline{S} \cap Col(i,j) = \{v(i,j,k) | k = d_{i,j}+1, d_{i,j}+2, \ldots, h(i,j)\}$. Then, a feasible decomposition (Q, R) of A, with $A = \lambda \cdot Q + R$, can be defined, as follows. For every pair (i,j) $(1 \leq i \leq m$ and $1 \leq j \leq n)$, $q_{i,j} = d_{i,j} - 1$ (Note that $r_{i,j}$ is uniquely determined by $q_{i,j}$).

On the other hand, given a feasible decomposition (Q, R) of A, a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G can be specified: $S = \{(v(i,j,k) | (i,j) \in \Gamma$ and $k = 1, 2, \ldots, d_{i,j}\}$ and $\overline{S} = \{(v(i,j,k) | (i,j) \in \Gamma$ and $k = d_{i,j}+1, d_{i,j}+2, \ldots, h(i,j)\}$. It is easy to see the weight $w(\mathcal{C})$ of the cut $\mathcal{C}$ is finite. Hence, the following lemma holds.

Lemma 7. *Any valid s-t cut in G has a one-to-one correspondence to a feasible decomposition of the IM matrix A.*

For a given valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, based on Lemma 7, it specifies a feasible decomposition (Q, R) of A. Next, we show that the total edge weight $w(\mathcal{C})$ of $\mathcal{C}$ equals to the sum of the horizontal complexity $C_H(Q)$ and the vertical complexity $C_V(R)$.

From Observations 1 and 2, edges in E_{mo} or in E_{ad} cannot be in $\mathcal{C}$. We thus only need to consider edges in E_{vt}, E_{hz}, E_q, and E_r. Actually, we are able to show that the total edge weight of the intersection of $\mathcal{C}$ with E_{vt}, E_{hz}, E_q, and E_r, equals to $\sum_{j=1}^{n} \sum_{i=1}^{m-1} \max\{0, r_{i+1,j} - r_{i,j}\}$, $\sum_{i=1}^{m} \sum_{j=1}^{n-1} \max\{0, q_{i,j+1} - q_{i,j}\}$, $\sum_{i=1}^{m} q_{i,1}$, and $\sum_{j=1}^{n} r_{1,j}$, respectively.

Lemma 8. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, the total edge weight of $\mathcal{C} \cap E_{vt}$ equals to $\sum_{j=1}^{n} \sum_{i=1}^{m-1} \max\{0, r_{i+1,j} - r_{i,j}\}$.*

Proof. In the construction of the edge set E_{vt}, all edges are added between two adjacent pillars on the same column of Γ, we thus first consider the subset $\mathcal{C}_{i,j}$ of edges that are between two adjacent pillars $Col(i,j)$ and $Col(i+1,j)$, and belong to the *s*-*t* cut $\mathcal{C}$. Recall that we distinguish four types of edges in G: lower edges, upper edges, lower-base edges, and upper-base edges. The weights of the edges of the same type are uniform. The basic idea to prove this lemma is to calculate in $\mathcal{C}_{i,j}$ the number of edges of each of those four types. We then compute the total edge weight of $\mathcal{C}_{i,j}$.

From the decomposition defined by the cut $\mathcal{C}$, we know that the set of vertices $\{v(i,j,k)|k=1,\ldots,q_{i,j}+1\} \cup \{v(i+1,j,k)|k=1,\ldots,q_{i+1,j}+1\}$ are in the source set S; while the set of vertices $\{v(i,j,k)|k=q_{i,j}+2,\ldots,h(i,j)\} \cup \{v(i+1,j,k)|k=q_{i+1,j}+2,\ldots,h(i+1,j)\}$ are in the sink set $\overline{S}$. Recall that for the lower edge le incident to a non-base vertex $v(i,j,k) \in Col(i,j)$, we have

$$1 \leq \lfloor \frac{a_{i+1,j}-a_{i,j}}{\lambda} \rfloor + k \leq h(i+1,j)-1.$$

Further, the edge le can be in the cut $\mathcal{C}$ only if the lower neighbor of $v(i,j,k)$ is in the sink set $\overline{S}$. Hence,

$$q_{i+1,j}+2 \leq \lfloor \frac{a_{i+1,j}-a_{i,j}}{\lambda} \rfloor + k \leq h(i+1,j)-1.$$

Another condition needed for the edge le to be in $\mathcal{C}$ is that $v(i,j,k) \in S$, which means

$$2 \le k \le q_{i,j} + 1.$$

Combining both conditions above we have

$$\max\left\{2, q_{i+1,j} + 2 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor\right\} \le k$$
$$\le \min\left\{h(i+1,j) - 1 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor, q_{i,j} + 1\right\}.$$

Hence, the number of lower edges in the cut $\mathcal{C}$ is,

$$\max\left\{0, \min\left\{h(i+1,j) - 1 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor, q_{i,j} + 1\right\}\right.$$
$$\left. - \max\left\{2, q_{i+1,j} + 2 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor\right\} + 1\right\}$$

For the upper edges between $Col(i,j)$ and $Col(i+1,j)$, in a similar way, we can calculate that the number of such edges in the cut $\mathcal{C}$ is

$$\max\left\{0, \min\left\{h(i+1,j) - 1 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor, q_{i,j} + 1\right\}\right.$$
$$\left. - \max\left\{1, q_{i+1,j} + 1 - \lfloor\frac{a_{i+1,j} - a_{i,j}}{\lambda}\rfloor\right\} + 1\right\}$$

The number of the upper-base and lower-base edges between $Col(i,j)$ and

$Col(i+1,j)$ that are in the cut $\mathcal{C}$ is

$$\max\{\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor - q_{i+1,j}, 0\}.$$

Thus, the total weight of the edges in the intersection of the s-t cut and the edges between $Col(i,j)$ and $Col(i+1,j)$ is calculated, as follows.

$$\begin{aligned}
& (\lambda - [a_{i+1,j}-a_{i,j}]\%\lambda) \times \max\left\{0, \min\left\{h(i+1,j) - 1 - \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor,\right.\right. \\
& \left.\left. q_{i,j}+1\right\} - \max\left\{2, q_{i+1,j}+2-\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor\right\} + 1\right\} \\
+ \quad & ([a_{i+1,j}-a_{i,j}]\%\lambda) \times \max\left\{0, \min\left\{h(i+1,j) - 1 - \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor, q_{i,j}+1\right\}\right. \\
& \left. - \max\left\{1, q_{i+1,j}+1-\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor\right\} + 1\right\} \\
+ \quad & \lambda \times \max\left\{\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor - q_{i+1,j}, 0\right\} \\
= \quad & \left\{\begin{array}{cc} 0 & f(\cdot) < 0 \\ \lambda\left(f(\cdot) + \max\left\{\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor - q_{i+1,j}, 0\right\}\right) + [a_{i+1,j}-a_{i,j}]\%\lambda & f(\cdot) \geq 0 \end{array}\right.,
\end{aligned} \tag{3.3}$$

where

$$f(\cdot) = \min\left\{\lfloor\frac{a_{i+1,j}}{\lambda}\rfloor - \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor, q_{i,j}\right\} - \max\left\{0, q_{i+1,j} - \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor\right\}$$

When $0 \leq a_{i+1,j} - a_{i,j} < \lambda$ or $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor = 0$, function $f(\cdot)$ can be reduced to

$$f(\cdot) = \min\left\{\lfloor\frac{a_{i+1,j}}{\lambda}\rfloor, q_{i,j}\right\} - \max\{0, q_{i+1,j}\} = q_{i,j} - q_{i+1,j}$$

And thus the expression in Equation 3.3 can be rewritten as

$$\begin{aligned} & \begin{cases} 0 & q_{i,j} - q_{i+1,j} < 0 \\ (q_{i,j} - q_{i+1,j}) \times \lambda + [a_{i+1,j} - a_{i,j}]\%\lambda & q_{i,j} - q_{i+1,j} \geq 0 \end{cases} \\ = & \lambda \cdot \max\left\{0, \frac{a_{i+1,j} - a_{i,j}}{\lambda} + q_{i,j} - q_{i+1,j}\right\} \\ = & \max\{r_{i+1,j} - r_{i,j}, 0\} \end{aligned}$$

When $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor < 0$, we can increase $a_{i+1,j}$ by $\lambda\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor$ and $q_{i+1,j}$ by $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor$, and denote the results by $a'_{i+1,j}$ and $q'_{i+1,j}$, respectively. This operation does not change $r_{i+1,j}$ or any edges in the cut $\mathcal{C}$. However, after the operation, $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor = 0$ holds, and thus we still have the total weight of the edges in the intersection of the s-t cut C and the edges between $Col(i,j)$ and $Col(i+1,j)$, is $\max\{r_{i+1,j} - r_{i,j}, 0\}$. Figure 3.3 (a) and (b) show an example to illustrate the key idea.

When $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor > 0$, we can decrease $a_{i+1,j}$ by $\lambda\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor$ and $q_{i+1,j}$ by $\min\{q_{i+1,j}, \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor\}$ to $a'_{i+1,j}$ and $q'_{i+1,j}$, respectively, to make sure that $q'_{i+1,j} \geq 0$. Observe that the case for $q_{i+1,j} \geq \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor$ is the same as the case for $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor < 0$. However, if $q_{i+1,j} < \lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor$, the new $r'_{i+1,j} = a'_{i+1,j} - \lambda q'_{i+1,j}$ will be $(\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor - q_{i+1,j}) \times \lambda$ less than the actual $r_{i+1,j}$. The term

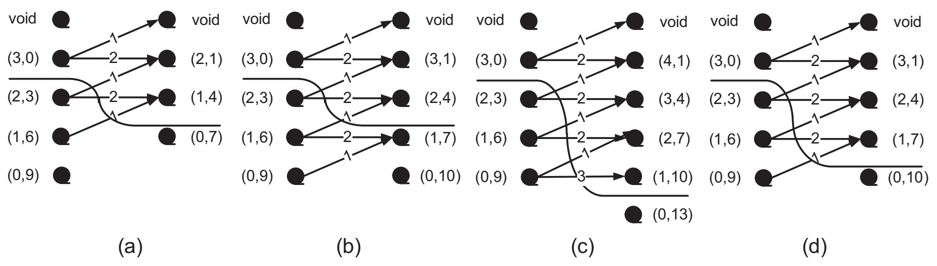


Figure 3.3: Illustrating the proof of Lemma 8. (a) An example with $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor < 0$, wherein $a_{i,j} = 9$, $a_{i+1,j} = 7$, $q_{i,j} = 2$, $q_{i+1,j} = 0$, and $\lambda = 3$. (b) Increasing $a_{i+1,j}$ to 10 and $q_{i+1,j}$ to 1, $r_{i,j}$ will not be changed, neither are the edges across the cut. (c) An example with $\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor > 0$, wherein $a_{i,j} = 9$, $a_{i+1,j} = 13$, $q_{i,j} = 2$, $q_{i+1,j} = 0$, and $\lambda = 3$. (d) Decreasing $a_{i+1,j}$ to 10 and keeping $q_{i+1,j}$ unchanged, $r_{i+1,j}$ is decreased by 3, but an edge of weight 3 is introduced to counteract this change.

$$\max\left\{\lfloor\frac{a_{i+1,j}-a_{i,j}}{\lambda}\rfloor - q_{i+1,j}, 0\right\} \times \lambda$$

can then counteract the change. Hence, in this case, we again have the total weight of the edges in the intersection of the s-t cut $\mathcal{C}$ and the edges between $Col(i,j)$ and $Col(i+1,j)$, is $\max\{r_{i+1,j} - r_{i,j}, 0\}$. Figure 3.3 (c) and (d) illustrate the essential idea using an example.

Taking all the above possibilities into account, we conclude that the total weight of the edges in the intersection of the s-t cut $\mathcal{C}$ and the edges between $Col(i,j)$ and $Col(i+1,j)$, is $\max\{r_{i+1,j} - r_{i,j}, 0\}$. By considering all pairs of adjacent pillars on the same columns of Γ, we have $w(\mathcal{C} \cap E_{vt}) = \sum_{j=1}^{n}\sum_{i=1}^{m-1}\max\{0, r_{i+1,j} - r_{i,j}\}$. Thus, Lemma 8 follows.

$\square$

Using a similar argument as for Lemma 8, we have the following lemma.

Lemma 9. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, the total edge weight of $\mathcal{C} \cap E_{hz}$ equals to $\sum_{i=1}^{m} \sum_{j=1}^{n-1} \max\{0, q_{i,j+1} - q_{i,j}\}$.*

We next investigate the total edge weight of the intersections of a valid s-t cut $\mathcal{C}$ with E_r and E_q, respectively.

Lemma 10. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, the total edge weight of $\mathcal{C} \cap E_r$ equals to $\sum_{j=1}^{n} r_{1,j}$.*

Proof. For each pillar $Col(1, j)$ on the first row of Γ, all vertices in $\{v(1, j, k)|k = 1, \ldots, d(1, j)$ or $k = 1, \ldots, q_{1,j}+1\}$ are in the source set S, while vertices in $\{v(1, j, k)|k = q_{1,j} + 2, \ldots, h(1, j)\}$ are in the sink set $\overline{S}$. Recall that each edge in $\{(s, v(1, j, k))|k = 2, \ldots, h(1, j)-1\}$ has a weight of λ, and the edge $(s, v(1, j, h(1, j)))$ is of weight $a_{1,j}\%\lambda$. Here only edges in $E'' = \{(s, v(1, j, k))|k = q_{1,j}+2, \ldots, h(1, j)-1\} \cup \{s, v(1, j, h(1, j))\}$ are in the cut $\mathcal{C}$. Thus, we have

$$\begin{aligned}
w(E'') &= (h(1, j) - 1 - (q_{1,j} + 2) + 1) \times \lambda + a_{1,j}\%\lambda \\
&= \left(\lfloor \frac{a_{1,j}}{\lambda} \rfloor + 2 - 1 - q_{1,j} - 2 + 1\right) \times \lambda + a_{1,j}\%\lambda \\
&= \lfloor \frac{a_{1,j}}{\lambda} \rfloor \times \lambda + a_{1,j}\%\lambda - \lambda q_{1,j} \\
&= a_{1,j} - \lambda \times q_{1,j} = r_{1,j}.
\end{aligned}$$

Hence, $w(\mathcal{C} \cap E_r) = \sum_{j=1}^{n} r_{1,j}$ □

Lemma 11. *For a valid s-t cut $\mathcal{C} = (S, \overline{S})$ in G, the total edge weight of $\mathcal{C} \cap E_q$ equals to $\sum_{i=1}^{m} q_{i,1}$.*

Proof. Consider each pillar $Col(i,1)$ on the first column of the grid Γ. All vertices in the set $\{v(i,1,k) | k = 1, \dots, d(i,1) \text{ or } k = 1, \dots, q_{i,1}+1\}$ are in the source set S, while vertices in $\{v(i,1,k) | k = q_{i,1}+2, \dots, h(i,1)\}$ are in the sink set $\overline{S}$. Thus, only edges in $E' = \{(v(i,1,k), t) | k = 2, \dots, q_{i,1}+1\}$ (recall that no edge in E_q is incident to a base vertex) is in the cut $\mathcal{C}$. Note that the weight of each of those edges is 1. Thus, the total edge weight of E' is $q_{i,1} + 1 - 2 + 1 = q_{i,1}$. Hence, $w(\mathcal{C} \cap E_q) = \sum_{i=1}^{m} q_{i,1}$. □

Putting Lemmas 8 - 11 all together, we have the following fact.

Lemma 12. *For any valid s-t cut $\mathcal{C}$ in G and its specified decomposition (Q,R) of A, with $A = \lambda \cdot Q + R$, we have $w(\mathcal{C}) = C_H(Q) + C_V(R)$.*

From Lemmas 7 and 12, an minimum-weight *s-t* $\mathcal{C}$ in G can be used to define an optimal matrix orthogonal decomposition (Q^*, R^*) of A, with $A = \lambda \cdot Q^* + R^*$, such that $C_H(Q^*) + C_V(R^*)$ is minimized. Note that $|V| = O(mn\lfloor \frac{H}{\lambda} \rfloor)$ and $|E| = O(mn\lfloor \frac{H}{\lambda} \rfloor)$, where H is the largest intensity level in the IM matrix A. Denote by $T(n', m')$ the time for finding a minimum *s-t* cut in an edge-weighted directed graph with $O(n')$ vertices and $O(m')$ edge. We have our first main result.

Theorem 1. *The IMOD problem with positive gradient sum cost can be solved in $T\left(mn\lfloor \frac{H}{\lambda} \rfloor, mn\lfloor \frac{H}{\lambda} \rfloor\right)$ time.*

3.3 Our Algorithm for the Total Variation Cost

For the second cost function, we first formulate the problem as searching an optimal surface in a 3-D geometric multi-pillar graph. The graph is constructed from the given intensity map A , such that each feasible surface in the graph defines a corresponding orthogonal decomposition of A (i.e., $A = \lambda Q + R$) and the cost of the surface equals to the horizontal complexity of Q plus the vertical complexity of R . Thus, the minimum-cost surface in the graph defines an optimal orthogonal decomposition of A . Then the minimum-cost surface is detected by computing a minimum s-t cut in a derived graph [79].

3.3.1 The Graph Transformation Scheme for the Total Variation Cost

In this section, we formulate the IMOD problem as the minimum-cost surface detection problem. Given an IM matrix $A = (a_{i,j})_{m\times n}$ and the integer $\lambda > 1$, a 3-D geometric multi-pillar graph $G = (V, E)$ is constructed on a 2-D $m \times n$ grid Γ (see Figure 3.4(a)), as follows.

Let $g(i, j)$ ($0 < i \leq m$ and $0 < j \leq n$) denote a grid point in Γ and each grid point represents an element $a_{i,j}$ of the given intensity map. Similar to the graph construction for the positive gradient sum cost, for each grid point $g(i, j)$, we create a set $Col(i, j)$ of $\lfloor \frac{a_{i,j}}{\lambda} \rfloor + 1$ (defined as *height* $h_{i,j}$) vertices in G, each representing a decomposition of $a_{i,j}$; $Col(i, j) = \{v(i, j, k) | k = 0, 1, \ldots, h_{i,j} - 1\}$, is called the (i, j)-*pillar* of G (see Figure 3.4(b)). Two pillars $Col(i, j)$ and $Col(i', j')$ are *adjacent* to each other if $|i - i'| + |j - j'| = 1$. We add edges between any two vertices if their corresponding pillars are adjacent. The edge weights are assigned as follows,

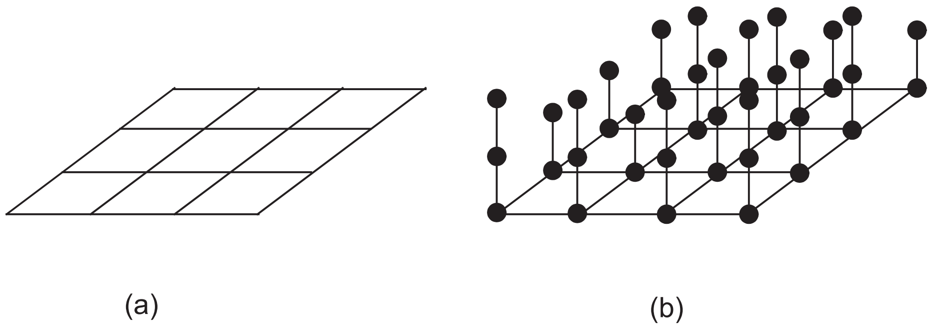


Figure 3.4: Illustration of a 2-D grid and multi-pillar vertices. (a) A 2-D grid. (b) Multi-pillar vertices.

1. For two vertices $v_1(i,j,k) \in Col(i,j)$ and $v_2(i,j{+}1,k') \in Col(i,j{+}1)$, the weight of the edge $e = (v_1, v_2)$ is assigned as $c(e) = f_{(i,j)(i,j+1)}(|k - k'|) = (k - k')^2$.

2. For two vertices $v_1(i,j,k) \in Col(i,j)$ and $v_2(i+1,j,k') \in Col(i+1,j)$, the weight of the edge $e = (v_1, v_2)$ is assigned as $c(e) = f_{(i,j)(i+1,j)}(k - k') = (a_{i,j} - a_{i+1,j} - \lambda k + \lambda k')^2$.

A *surface* in G is defined by a function $\mathcal{N} : \Gamma \rightarrow \{0, 1, \ldots, h_{i,j} - 1\}$, that is, each $(i,j) \in \Gamma$ is mapped to $k = \mathcal{N}(i,j)$ and $v(i,j,k)$ is a vertex of G. Intuitively the net surface is a surface that "intersects" exactly once with each pillar at a vertex. For simplicity, we also denote the surface by $\mathcal{N}$. Note that for any two adjacent vertices on the surface $\mathcal{N}$, $v_1(i,j,k)$ and $v_2(i',j',k')$ with $|i - i'| + |j - j'| = 1$, (v_1, v_2) is an edge of G. The cost $c(\mathcal{N})$ of the surface $\mathcal{N}$ is the total sum of the weights of the edges on $\mathcal{N}$. Apparently a feasible surface in G has a one-to-one relation with the decomposition of the IM A by setting $q_{i,j} = \mathcal{N}(i,j)$ and $r_{i,j} = a_{i,j} - q_{i,j}$. The following equation shows that the cost $c(N)$ of the surface $\mathcal{N}$ equals to the total complexity of the resulting decomposition:

$$\begin{aligned}
c(\mathcal{N}) &= \sum_{i=1}^{m}\sum_{j=2}^{n}(\mathcal{N}(i,j)-\mathcal{N}(i,j-1))^2 \\
&+ \sum_{j=1}^{n}\sum_{i=2}^{m}(a_{i+1,j}-a_{i,j}-\lambda\cdot\mathcal{N}(i,j)+\lambda\cdot\mathcal{N}(i,j-1))^2 \\
&= \sum_{i=1}^{m}\sum_{j=2}^{n}(q_{i,j}-q_{i,j-1})^2+\sum_{j=1}^{n}\sum_{i=2}^{m}((a_{i,j}-\lambda\cdot q_{i,j})-(a_{i-1,j}-\lambda\cdot q_{i-1,j}))^2 \\
&= C_H(Q)+C_V(R) \qquad (3.4)
\end{aligned}$$

Thus, an optimal surface $\mathcal{N}^*$ in G defines an optimal decomposition of IM A.

3.3.2 Optimal Surface Detection

In this section, we present an efficient algorithm for detecting an optimal surface in G. The basic idea is to transform the optimal surface detection problem as a minimum s-t cut problem by adapting Wu and Chen's technique for the optimal net surface problem [79].

To solve the problem by the minimum s-t cut algorithm, a weighted directed graph $\tilde{G}=(\tilde{V},\tilde{E})$ is constructed from G, as follows. The vertex set $\tilde{V}$ consists of the vertex set V of G, a set V_{top} of dummy vertices, a source vertex s and a sink vertex t. To differentiate the description of from G, we call the set of nodes $Col(i,j)=\{v(i,j,k)|k=0,1,\ldots,h_{i,j}-1\}$ a *column* (i.e., the (i,j)-column) in $\tilde{G}$. For each column $Col(i,j)$, we add a dummy vertex $v_{top}(i,j)\in V_{top}$ on the top of the column, called the *top vertex* of $Col(i,j)$ (see Figure 3.4(b) for example). Note that $v_{top}(i,j) \notin Col(i,j)$. The arcs (directed edges) are put in $\tilde{G}$ such that,

- for any feasible surface $\mathcal{N}$ in G, $\mathcal{N}(i,j)$ partitions each (i,j)-pillar into two subsets, $S_{i,j} = \{v(i,j,k) | k = 0, 1, \ldots, \mathcal{N}(i,j)\}$ and $\bar{S}_{i,j} = \{v(i,j,k) | k = \mathcal{N}(i,j) + 1, \ldots, h_{i,j} - 1\}$, and the induced cut $\kappa = (\bigcup_{i,j} S_{i,j}, \bigcup_{i,j} \bar{S}_{i,j} \cup V_{top})$ is a feasible s-t cut in (a cut κ is *feasible* if the total edge weight $c(\kappa)$ is finite);

- any feasible cut $\kappa = (S, \bar{S} \cup V_{top})$ in $\tilde{G}$ can be used to define a feasible surface $\mathcal{N}$ in G, as follows. Let $v(i,j,k_{top})$ be the "topmost" vertex in $S \cap Col(i,j)$ (i.e., the vertex with the largest k-coordinate in $S \cap Col(i,j)$). Then, $\mathcal{N}(i,j) = k_{top}$; and

- the total weight of the cut equals to the cost of the surface $\mathcal{N}$.

We thus introduce two types of arcs: one is used to enforce the feasibility of a solution, which includes two subsets of arcs E_{mo} and E_{ad}; the other is used to realize the total cost of a feasible surface $\mathcal{N}$, which consists of E_{hz} and E_{vt}.

- *The arcs in E_{mo}*: On each column $Col(i,j)$, an arc of weight $+\infty$ is added from every vertex $v(i,j,k)$ to vertex $v(i,j,k-1)$ for $k = 1, 2, \ldots, h_{i,j} - 1$ and from every top vertex $v_{top}(i,j)$ to $v(i,j,h_{i,j} - 1)$. The set of arcs in E_{mo} is used to guarantee that exactly one vertex on each pillar of G can be on a feasible surface $\mathcal{N}$.

- *The arcs in E_{ad}*: An arc of weight $+\infty$ is put in E_{ad} from the source s to the vertex $v(i,j,0)$ of each column $Col(i,j)$. Meanwhile, an arc of weight $+\infty$ is added from the top vertex of each column to the sink t.

- *The arcs in E_{hz}*: These arcs are added between two adjacent columns $Col(i,j)$ and $Col(i,j+1)$. As in Ref. [79], we first define the functions $\Delta_{(i,j)(i,j+1)}(x)$ from $f_{(i,j)(i,j+1)}(x)$, as follows,

$$\Delta_{(i,j)(i,j+1)}(0) = f_{(i,j)(i,j+1)}(1) - f_{(i,j)(i,j+1)}(0) = 1^2 - 0^2 = 1$$

$$\begin{aligned}\Delta_{(i,j)(i,j+1)}(x+1) &= f_{(i,j)(i,j+1)}(x) + f_{(i,j)(i,j+1)}(x+2) - 2f_{(i,j)(i,j+1)}(x+1) \\ &= x^2 + (x+2)^2 - 2(x+1)^2 = 2,\ x = 0,1,2,\ldots\end{aligned}$$

In fact, $\Delta_{(i,j)(i,j+1)}(x)$ is a discrete equivalent of the second derivative of $f_{(i,j)(i,j+1)}(x)$ at x.

Now we can put arcs in E_{hz}. Consider each vertex $v(i,j,k) \in Col(i,j)$, we differentiate two cases.

Case 1. If $k < h_{i,j+1}$, an arc is added from $v(i,j,k)$ to each vertex $v(i,j+1,k')$ of $Col(i,j+1)$ with $k' = 0,1,\ldots,k$. The weight of the arc is set to $\Delta_{(i,j)(i,j+1)}(|k-k'|)$.

Case 2. If $k \geq h_{i,j+1}$, we put in E_{hz} an arc from $v(i,j,k)$ to each vertex $v(i,j+1,k')$ of $Col(i,j+1)$. The weight of the arc is set to $\Delta_{(i,j)(i,j+1)}(|k-$

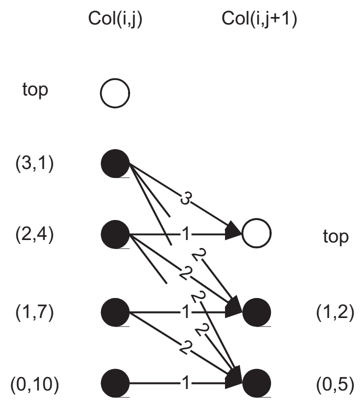


Figure 3.5: Illustrating E_{hz} of the case $a_{i,j} = 10$, $a_{i,j+1} = 5$, and $\lambda = 3$. Solid circles represent vertices in the columns and hollow circles represent the top vertices.

$k'|)$. Since there is no cost-penalty arc for $\Delta_{(i,j)(i,j+1)}(0)$, $\Delta_{(i,j)(i,j+1)}(1)$,..., $\Delta_{(i,j)(i,j+1)}(k-h_{i,j+1})$, the weight of the arc to vertex $v_{top}(i, j+1)$ is updated to

$$\sum_{k'=0}^{k-h_{i,j+1}} \Delta_{(i,j)(i,j+1)}(k') = 2(k - h_{i,j+1}) + 1.$$

Arcs are symmetrically added from $v(i, j + 1, k) \in Col(i, j + 1)$ to vertices of $Col(i, j)$. An example of the construction of E_{hz} is shown in Figure 3.5.

- *The arcs in E_{vt}*: These arcs are added between two adjacent pillars $Col(i, j)$ and $Col(i + 1, j)$. The purpose is to ensure that the total weight of the arcs between $Col(i, j)$ and $Col(i+1, j)$ that are in a feasible s-t cut κ of $\tilde{G}$, equals to a constant (possible 0) plus the weight of the corresponding edge on the surface in G defined by the cut κ.

  Consider each vertex $v(i, j, k)$ of $Col(i, j)$. Recall that the weight of an edge between $v(i,j,k)$ and $v(i+1,j,k')$ of G is $f_{(i,j)(i+1,j)}(k-k') = (a_{i,j} - a_{i+1,j} - \lambda k + \lambda k')^2$. To "distribute" the weight of $f_{(i,j)(i+1,j)}(k-k')$ to appropriate arcs between $Col(i,j)$ and $Col(i+1,j)$, we develop the following novel arc weight penalty embedding scheme.

For each vertex $v(i,j,k)$ of $Col(i,j)$, we first find a vertex $v(i+1,j,\overrightarrow{\eta}_k)$ such that $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k)$ is minimized, where $\overrightarrow{\eta}_k$ is an integer. In this way, we essentially want to create a correspondence between vertices of $Col(i,j)$ and $Col(i+1,j)$. Two possible cases need to be differentiated.

Case 1) : $a_{i,j} \geq a_{i+1,j}$. It is not difficulty to see that when $k - \overrightarrow{\eta}_k = \frac{a_{i,j}-a_{i+1,j}}{\lambda}$, $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k)$ is minimized. Note that although k and $\overrightarrow{\eta}_k$ are integers, $\frac{a_{i,j}-a_{i+1,j}}{\lambda}$ may not be an integer. With a close eye on the function $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k)$, we know that for any k, $k-\overrightarrow{\eta}_k$ is a fixed constant while $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k)$ is minimized. Thus, we only need to consider $k = h_{i,j}-1$, where $h_{i,j}$ is the height of $Col(i,j)$. Then, $\overrightarrow{\eta}_k$ only has three possible values: $h_{i+1,j}-2$, $h_{i+1,j}-1$, or $h_{i+1,j}$. Let us investigate the function value of $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k)$ for each of those three possible $\overrightarrow{\eta}_k$ values.

a) If $k = h_{i,j}-1$ and $\overrightarrow{\eta}_k = h_{i+1,j}-2$, $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k) = [\lambda - (a_{i,j}\%\lambda - a_{i+1,j}\%\lambda)]^2$.

b) If $k = h_{i,j}-1$ and $\overrightarrow{\eta}_k = h_{i+1,j}-1$, $f_{(i,j)(i+1,j)}(k-\overrightarrow{\eta}_k) = (a_{i,j}\%\lambda - a_{i+1,j}\%\lambda)^2$.

c) If $k = h_{i,j} - 1$ and $\overrightarrow{\eta}_k = h_{i+1,j}$, $f_{(i,j)(i+1,j)}(k - \overrightarrow{\eta}_k) = [\lambda + (a_{i,j}\%\lambda - a_{i+1,j}\%\lambda)]^2$.

Hence, we can compute $\overrightarrow{\eta}_{h_{i,j}-1}$, as follows.

i) If $a_{i,j}\%\lambda - a_{i+1,j}\%\lambda \geq 0$ and $2(a_{i,j}\%\lambda - a_{i+1,j}\%\lambda) < \lambda$, then $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j} - 2$.

ii) If $a_{i,j}\%\lambda - a_{i+1,j}\%\lambda \geq 0$ and $2(a_{i,j}\%\lambda - a_{i+1,j}\%\lambda) \geq \lambda$, then $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j} - 1$.

iii) If $a_{i,j}\%\lambda - a_{i+1,j}\%\lambda < 0$ and $2(a_{i,j}\%\lambda - a_{i+1,j}\%\lambda) \leq \lambda$, then $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j} - 1$.

iv) If $a_{i,j}\%\lambda - a_{i+1,j}\%\lambda < 0$ and $2(a_{i,j}\%\lambda - a_{i+1,j}\%\lambda) > \lambda$, then $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j}$.

Then, for each $k = 0, 1, \ldots, h_{i,j} - 1$, $\overrightarrow{\eta}_k = k - h_{i,j} + \overrightarrow{\eta}_{h_{i,j}-1} + 1$.

To establish one-to-one correspondences between vertices of $Col(i,j)$ and $Col(i+1,j)$, we add the following vertices to $Col(i,j)$ or $Col(i+1,j)$.

For each $k = 0, 1, \ldots, h_{i,j} - 2$, if $v(i+1, j, \overrightarrow{\eta}_k)$ is not a vertex of $Col(i+1, j)$ (i.e., $\overrightarrow{\eta}_k < 0$), we add this new vertex to $Col(i+1,j)$ (Figure 3.6 (a)). In this way, we say that *vertex*$v(i,j,k)$ of $Col(i,j)$ corresponds to $v(i+1, j, \overrightarrow{\eta}_k)$ of $Col(i+1,j)$. For $k = h_{i,j} - 1$, we need the following special treatment.

a) $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j} - 2$ (Figure 3.6(b)). We add a new vertex $v(i, j, h_{i,j})$ to $Col(i,j)$. The vertex $v(i+1, j, h_{i+1,j} - 1)$ of $Col(i+1,j)$ corresponds

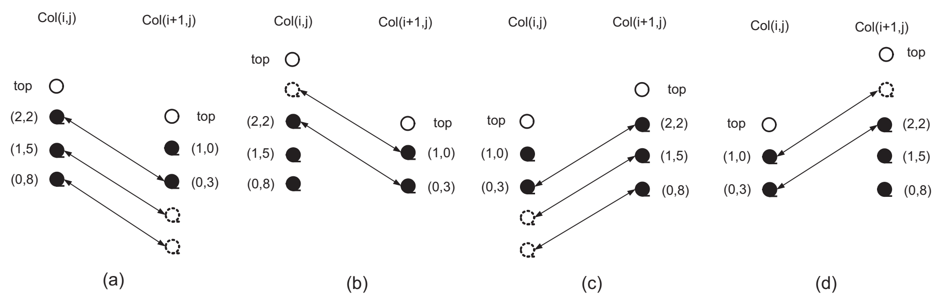


Figure 3.6: Illustrating the establishment of the correspondence of vertices between $Col(i, j)$ and $Col(i + 1, j)$. Solid circles represent vertices in the columns and hollow circles represent the top vertices. Dashed circles are vertices added and arrows mark the correspondence between two vertices. (a) $a_{i,j} = 8$, $a_{i+1,j} = 3$, and $\lambda = 3$, vertices are created below $Col(i+1, j)$. (b) $a_{i,j} = 8$, $a_{i+1,j} = 3$, and $\lambda = 3$, a vertex is created on top of $Col(i, j)$. (c) $a_{i,j} = 3$, $a_{i+1,j} = 8$, and $\lambda = 3$, vertices are created below $Col(i, j)$. (d) $a_{i,j} = 3$, $a_{i+1,j} = 8$, and $\lambda = 3$, a vertex is created on top of $Col(i+1, j)$.

to this new vertex $v(i, j, h_{i,j})$, and vertex $v(i, j, h_{i,j} - 1)$ corresponds to $v(i + 1, j, h_{i+1,j} - 2)$.

b) $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j} - 1$. The vertex $v(i, j, h_{i,j} - 1)$ corresponds to $v(i + 1, j, h_{i+1,j} - 1)$ of $Col(i + 1, j)$.

c) $\overrightarrow{\eta}_{h_{i,j}-1} = h_{i+1,j}$. We add a new vertex $v(i+1, j, h_{i+1,j})$ to $Col(i+1, j)$. The vertex $v(i, j, h_{i,j} - 1)$ of $Col(i, j)$ corresponds to this new vertex $v(i + 1, j, h_{i+1,j})$.

Case 2) : $a_{i,j} < a_{i+1,j}$. Similar to the case $a_{i,j} \geq a_{i+1,j}$, we establish the vertex correspondence between $Col(i, j)$ and $Col(i+1, j)$. In this case, for each $k' = 0, 1, \ldots, h_{i+1,j} - 1$ of $Col(i + 1, j)$, we compute $\overleftarrow{\eta}_{k'}$ such that $f_{(i,j)(i+1,j)}(\overleftarrow{\eta}_{k'} - k')$ is minimized. The computation of $\overleftarrow{\eta}_{k'}$'s is the same as that for Case 1). Then, we establish the correspondence between vertex

$v(i, j, \overleftarrow{\eta}_{k'})$ and $v(i+1, j, k')$ for $k' = 0, 1, \ldots, h_{i+1,j} - 1$ (Figure 3.6 (c)). Note that if $v(i, j, \overleftarrow{\eta}_{k'})$ does not exist in $Col(i, j)$, we add it in. As in Case 1), if $\overleftarrow{\eta}_{h_{i+1,j}-1} = h_{i,j} - 2$ (Figure 3.6 (d)), we need to add a new vertex $v(i+1, j, h_{i+1,j})$ such that the vertex $v(i, j, h_{i,j} - 1)$ corresponds to this new vertex.

Up to this point, we set the one-to-one correspondences between vertices between the new columns $Col(i, j)$ and $Col(i + 1, j)$, which include the newly added vertices. Intuitively, these correspondences indicate the pairs of vertices of two adjacent columns who have the closest r-values. To facilitate the addition of weighted arcs, we define the following functions $\overrightarrow{\Delta}_{(i,j)(i+1,j)}(x)$ and $\overleftarrow{\Delta}_{(i,j)(i+1,j)}(x)$ from $f_{(i,j)(i,j+1)}(x)$. Recall that $f_{(i,j)(i+1,j)}(k-k') = (a_{i,j} - a_{i+1,j} - \lambda k + \lambda k')^2$ and $k - \overrightarrow{\eta}_k = (h_{i,j} - 1) - \overrightarrow{\eta}_{h_{i,j}-1}$. Let $\chi = a_{i,j} - a_{i+1,j} - \lambda(h_{i,j} - 1) + \lambda \overrightarrow{\eta}_{h_{i,j}-1}$.

For $k' \leq \overrightarrow{\eta}_k$, $k - k' = x + (k - \overrightarrow{\eta}_k) = x + [(h_{i,j} - 1) - \overrightarrow{\eta}_{h_{i,j}-1}], x = 0, 1, 2, \ldots$. We have $f_{(i,j)(i+1,j)}(x) = f_{(i,j)(i+1,j)}(k - k') = (a_{i,j} - a_{i+1,j} - \lambda k + \lambda k')^2 = (\lambda x - \chi)^2$.

$$\begin{aligned}
\overrightarrow{\Delta}_{(i,j)(i+1,j)}(0) &= f_{(i,j)(i+1,j)}(1) - f_{(i,j)(i+1,j)}(0) \\
&= (\lambda - \chi)^2 - \chi^2 = \lambda^2 - 2\lambda\chi; \\
\overrightarrow{\Delta}_{(i,j)(i+1,j)}(x+1) &= [f_{(i,j)(i+1,j)}(x+2) - f_{(i,j)(i+1,j)}(x+1)] \\
&- [f_{(i,j)(i+1,j)}(x+1) - f_{(i,j)(i+1,j)}(x)] \\
&= [(\lambda(x+2) - \chi)^2 - (\lambda(x+1) - \chi)^2] \\
&- [(\lambda(x+1) - \chi)^2 - (\lambda x - \chi)^2] \\
&= 2\lambda^2, \text{for} x = 0, 1, 2, \ldots
\end{aligned}$$

Then, we consider $k' > \overrightarrow{\eta}_k$. In this case, $k - k' = -x + (k - \overrightarrow{\eta}_k) = -x + [(h_{i,j} - 1) - \overrightarrow{\eta}_{h_{i,j}-1}], x = 0, 1, 2, \ldots$. We have $f_{(i,j)(i+1,j)}(x) = f_{(i,j)(i+1,j)}(k - k') = (a_{i,j} - a_{i+1,j} - \lambda k + \lambda k')^2 = (\lambda x + \chi)^2$.

$$\begin{aligned}
\overleftarrow{\Delta}_{(i,j)(i+1,j)}(0) &= f_{(i,j)(i+1,j)}(1) - f_{(i,j)(i+1,j)}(0) \\
&= (\lambda+\chi)^2 - \chi^2 = \lambda^2 + 2\lambda\chi; \\
\overleftarrow{\Delta}_{(i,j)(i+1,j)}(x+1) &= [f_{(i,j)(i+1,j)}(x+2) \\
&- f_{(i,j)(i+1,j)}(x+1)] - [f_{(i,j)(i+1,j)}(x+1) - f_{(i,j)(i+1,j)}(x)] \\
&= [(\lambda(x+2)+\chi)^2 - (\lambda(x+1)+\chi)^2] \\
&- [(\lambda(x+1)+\chi)^2 - (\lambda x+\chi)^2] \\
&= 2\lambda^2, \mathrm{for} x = 0, 1, 2, \ldots
\end{aligned}$$

We are now ready to put in E_{vt} arcs. For every vertex $v(i,j,k)$ of $Col(i,j)$ including the newly added vertices, we add an arc to each vertex $v(i+1,j,k')$ with $k' \leq \overrightarrow{\eta}_k$ (note that $v(i+1,j,\overrightarrow{\eta}_k)$ is the corresponding vertex of $v(i,j,k)$ in $Col(i+1,j)$, and the weight of the arc is $\overrightarrow{\Delta}_{(i,j)(i+1,j)}(\overrightarrow{\eta}_k - k')$. In the meanwhile, for every vertex $v(i+1,j,k')$ of $Col(i+1,j)$ including the newly added vertices, denote by $v(i,j,\overleftarrow{\eta}_{k'})$ the corresponding vertex of $v(i+1,j,k')$ in $Col(i,j)$. An arc from every vertex $v(i+1,j,k')$ to each vertex $v(i,j,k)$ with $k \leq \overleftarrow{\eta}_{k'}$ is added, and the weight of the arc is $\overleftarrow{\Delta}_{(i,j)(i+1,j)}(\overleftarrow{\eta}_{k'} - k)$.

So far, we finish adding new vertices and arcs into the graph $\tilde{G}$. Applying a similar argument in Ref. [79], it is not difficult to show that the weight of the edge between $Col(i,j)$ and $Col(i+1,j)$ on a feasible surface $\mathcal{N}$ in G equals to a constant (possible 0) plus the total weight of the arcs between $Col(i,j)$ and

$Col(i+1, j)$ in $\tilde{G}$, which are in the corresponding s-t cut κ of defined by $\mathcal{N}$. However, a feasible s-t cut κ of $\tilde{G}$ may not define a feasible surface in G due to the addition of new vertices in $\tilde{G}$. We thus use the following scheme to remove the newly added vertices from $\tilde{G}$.

For each column $Col(i, j)$ in $\tilde{G}$, note that a vertex $v(i, j, k)$ is a newly added on if $k < 0$ or $k = h_{i,j}$. Consider each newly added vertex $v(i, j, k)$ with $k < 0$.

– For each incoming arc of $v(i, j, k)$, that is, an arc from $v(i', j, k')$ to $v(i, j, k)$ ($i' = i + 1$, or $i - 1$), it cannot be in a valid s-t cut (a feasible s-t cut is valid if it can be used to define a feasible surface in G). We thus simply remove the arc $(v(i', j, k'), v(i, j, k))$.

– For each outgoing arc of $v(i, j, k)$, that is, an arc from $v(i, j, k)$ to $v(i', j, k')$ ($i' = i + 1$, or $i - 1$), if the arc $(v(i, j, k), v(i', j, k'))$ is in a valid s-t cut, then the arc $(v(i, j, 0), v(i', j, k'))$ must be in the cut. Thus, we remove the arc $(v(i, j, k), v(i', j, k'))$ and add the weight of that arc to the arc $(v(i, j, 0), v(i', j, k'))$.

After removing all the associated arcs of $v(i, j, k)$, we then remove that vertex.

Next, we consider the newly added vertex $v(i, j, k)$ with $k = h_{i,j}$. In this case, the outgoing arc of $v(i, j, k)$ cannot be in a valid s-t cut, we thus simply remove each outgoing arc. For each incoming edge of $v(i, j, k)$, that is, an arc from $v(i', j, k')$ to $v(i, j, k)$ ($i' = i + 1$, or $i - 1$), if it is in a valid s-t cut, then $v(i', j, k')$ must be in the source set of the cut. We thus introduce an arc, having the weight of the arc $(v(i', j, k'), v(i, j, k))$, from $v(i', j, k')$ to the sink vertex t. We then remove the vertex $v(i, j, k)$ after no arc is associated with it. Figure 3.7 shows the construction of arcs in E_{vt} for the example in Figure 3.6(a).

After performing the above scheme for the removal of the newly added vertices, any feasible s-t cut κ in $\tilde{G}$ can be used to define a feasible surface $\mathcal{N}$ in G. Furthermore, using a similar argument in Ref. [79], we can prove that the total weight of the arcs between $Col(i, j)$ and $Col(i+1, j)$ that are in differs from the weight of the edge on the surface $\mathcal{N}$ by a constant.

We thus finish the construction of graph $\tilde{G}$. Any feasible s-t cut in can be used to define a feasible surface $\mathcal{N}$ in G, and vice verse; furthermore, the total weight of differs by a constant from the total cost of the corresponding surface $\mathcal{N}$. Hence, an optimal surface $\mathcal{N}^*$ in G, which defines an optimal decomposition of the input IM A, can be obtained by computing a minimum s-t cut κ^* in $\tilde{G}$. Note that a minimum s-t cut κ^* in $\tilde{G}$ can be computed by Goldberg and Tarjan's efficient maximum flow algorithm [36]. We thus solve the IMOD with total variation cost problem efficiently.

3.4 Material and Experimental Method

To evaluate performance of the method with the positive gradient sum cost, we performed some statistical studies using 1000 randomly generated 15×15 IM matrices each with intensity levels ranging from 4 to 64 in powers of 2. The number of MLC-apertures are computed using Xia and Verhey's algorithm [82] without considering interleaf motion constraint.

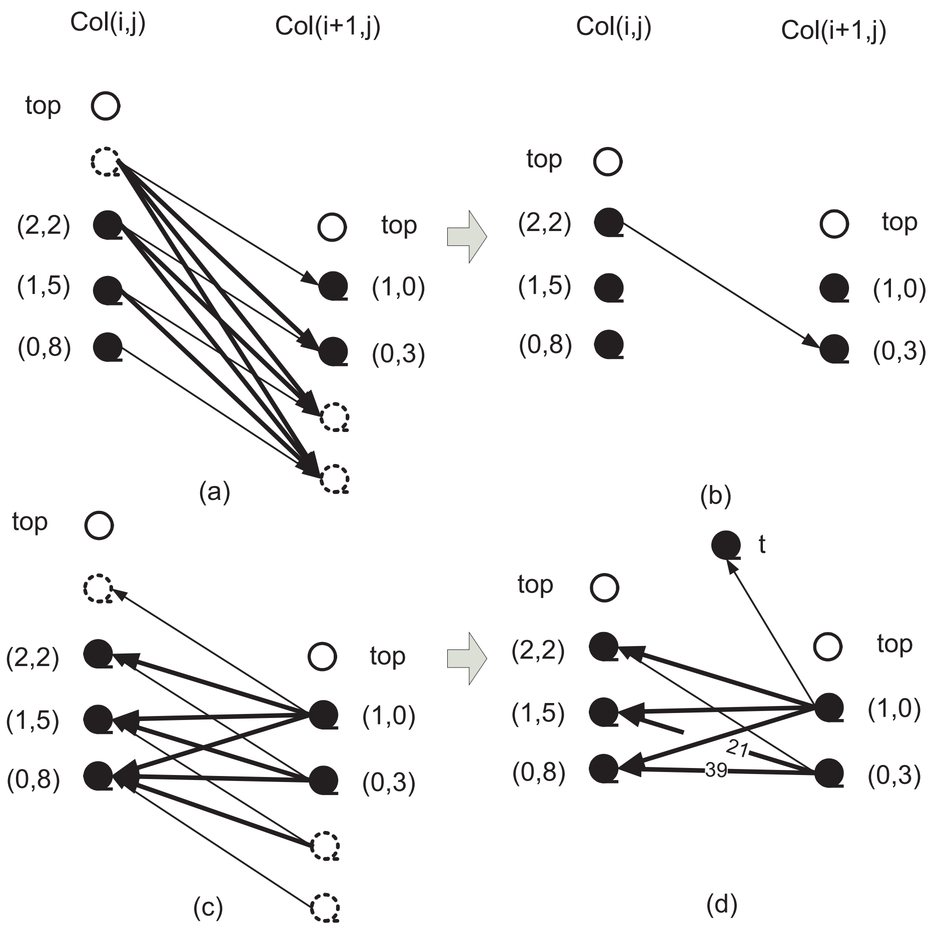


Figure 3.7: Illustrating the construction of arcs in E_{vt}. In this example, $a_{i,j} = 8$, $a_{i+1,j} = 3$, and $\lambda = 3$. (a) Arcs added from $Col(i,j)$ to $Col(i+1,j)$. The thin line arrows represent arcs with a weight of $\overrightarrow{\Delta}_{(i,j)(i+1,j)}(0) = 15$ and the thick arcs have a weight of $\overrightarrow{\Delta}_{(i,j)(i+1,j)}(1) = 18$. (b) Arc(s) after removing the newly added vertices (the dashed circular ones). (c) Arcs added from $Col(i+1,j)$ to $Col(i,j)$, the thin arcs have a weight of $\overleftarrow{\Delta}_{(i,j)(i+1,j)}(0) = 3$ and the thick arc have a weight of $\overleftarrow{\Delta}_{(i,j)(i+1,j)}(1) = 18$. (d) Arcs after removing the newly added vertices. The numbers on the two arcs denote the weight of each individual arc after the removal operations. The weights of other arcs are not changed.

We have also experimented with some real medical data sets available to us.

The performance of the method with the total variation cost was tested using 73 intensity maps that were generated with a commercial inverse treatment planning system (Pinnacle v8.0m). 32 IMs were from woman pelvic data and 41 IMs were from head & neck data to discover how complexity of IMs affects the performance of our algorithm.

We assume to do the planning for the Varian LINAC System. Our decomposition method was conducted and then leaf sequencing was performed in Pinnacle. The results were compared between with that without field decomposition conducted. The total number of MUs required to deliver the IMs by Pinnacle was recorded when doing the leaf sequencing algorithm in Pinnacle.

3.5 Experimental Results

3.5.1 Results for the Positive Gradient Sum Cost

Table 3.1 shows percentage of IMs getting improved and the average results (both number of MUs and number of MLC-apertures) before and after performing our decomposition method (the average is calculated based only on those IMs getting improved). We observed that our IMOD algorithm generated as much as 38.1% less MLC-apertures and 33.3% less number of MUs than single direction delivery.

For the medical data. 77.0% IMs that we tested on got improved number of MLC-apertures. Our IMOD algorithm produced as much as 27.3% less MLC-apertures with an average of 13.1% comparing with the SLS method using single direction delivery.

Table 3.1: The average number of MUs and the number of MLC-apertures

IM size	Number of MLC-apertures		
	%Improved	Avg before	Avg after
4	9%	6.0±0.6	5.8±0.4
8	24%	9.1±0.6	8.5±0.7
16	25%	12.0±0.7	11.4±0.6
32	45%	14.9±0.9	14.2±0.7
64	54%	18.2±0.9	17.1±0.7
IM size	Number of MUs		
	%Improved	Avg before	Avg after
4	25%	7.8±1.0	7.3±0.8
8	66%	17.1±2.0	15.5±1.6
16	73%	35.3±4.3	32.1±4.0
32	81%	69.7±8.9	63.4±6.5
64	94%	144.2±18.2	129.0±13.1

3.5.2 Results for the Total Variation Cost

For all the 73 datasets, the average decrease in MUs obtained using our algorithm over Pinnacle v8.0m is 45.1%. The maximum decrease was 60.9% for the number of MUs. 66 (90.6%) intensity maps got decreased number of MUs and the average decrease for those intensity maps was 50.8%.

For woman pelvic data (as shown in Table 3.2), the average improvement for number of MUs was 50.7%. The maximum decrease was 80.8% for number of MUs. For number of MUs, 31 out of the 32 IMs (96.9%) got reduced number of MUs with an average of 52.5%.

Table 3.2: Results for woman pelvic data

Patient	IM	# of MUs			% Decrease
		Pinnacle	Our Method	%Decrease	per patient
1	1	160	81	49.4%	37.4%
	2	149	96	35.6%	
	3	101	107	-5.9%	
	4	144	63	56.3%	
	5	130	54	58.5%	
	6	170	166	2.4%	
	7	172	86	50.0%	
	8	136	75	44.9%	
2	1	113	44	61.1%	51.3%
	2	163	60	63.2%	
	3	163	56	65.7%	
	4	129	80	38.0%	
	5	151	55	63.6%	
	6	164	129	21.3%	
	7	167	32	80.8%	
	8	165	136	17.6%	
3	1	191	46	75.9%	66.9%

Table 3.2 Continued

Patient	IM	# of MUs			% Decrease
		Pinnacle	Our Method	%Decrease	per patient
3	2	151	43	71.5%	66.9%
	3	110	53	51.8%	
	4	186	55	70.4%	
	5	139	51	63.3%	
	6	179	60	66.5%	
	7	115	45	60.9%	
	8	169	58	65.7%	
4	1	141	61	56.7%	49.0%
	2	127	83	34.7%	
	3	198	133	32.8%	
	4	160	68	57.5%	
	5	186	71	61.8%	
	6	131	68	48.1%	
	7	167	92	44.9%	
	8	130	56	56.9%	

For head & neck data (as shown in Table 3.3), the average improvement for number of MUs was 40.7%. 35 (85.4%) of the 41 IMs got improvement in number of MUs and the average improvement is 48.9%.

Table 3.3: Results for head & neck data

Patient	IM	# of MUs			% Decrease
		Pinnacle	Our Method	%Decrease	per patient
1	1	120	87	27.5%	48.9%
	2	75	55	26.8%	
	3	66	12	81.8%	
	4	87	42	51.7%	
	5	76	44	42.1%	
	6	79	35	55.7%	
	7	84	74	11.9%	
	8	84	25	70.2%	
	9	96	18	81.3%	
2	1	148	91	38.5%	22.9%
	2	112	56	50.0%	
	3	67	68	-1.5%	
	4	117	58	50.4%	
	5	98	112	-14.3%	
	6	121	101	16.5%	
	7	70	79	-12.9%	
3	1	111	59	46.9%	42.1%

Table 3.3 – Continued

Patient	IM	# of MUs			% Decrease
		Pinnacle	Our Method	%Decrease	per patient
3	2	49	40	18.4%	42.1%
	3	145	58	60.0%	
	4	71	81	-14.1%	
	5	123	36	70.7%	
	6	121	67	44.6%	
	7	125	59	52.8%	
	8	93	56	39.8%	
	9	134	107	20.2%	
4	1	224	89	60.3%	63.7%
	2	121	66	45.5%	
	3	168	58	65.5%	
	4	144	43	70.1%	
	5	155	70	54.8%	
	6	130	41	68.5%	
	7	212	53	75.0%	
	8	115	20	82.6%	
	9	85	51	40.0%	
5	1	64	68	-6.3%	31.4%

Table 3.3 – Continued

Patient	IM	# of MUs			% Decrease
		Pinnacle	Our Method	%Decrease	per patient
5	2	76	38	50.0%	31.4%
	3	100	79	21.0%	
	4	75	80	-6.7%	
	5	78	50	35.9%	
	6	101	67	33.7%	
	7	142	54	62.0%	

3.6 Discussion

From the results in Table 3.2, our algorithm constantly got good results for all the women pelvic data if we consider the per patient performance. It is noticeable that the improvement of the first patient, whose diagnosis category is uterus, is not as good as the other three whose diagnosis categories are cervix. This may be an implication that the performance of our algorithm does depend on the diagnosis case. As for each IM, there is no clear relation between the complexity of the IM (number of MUs without decomposition) and the improvement – although the only IM without any improvement has the lowest number of MUs without decomposition, but other IMs with lower improvement have relatively large complexities (e.g., the 6th IM of the first patient and the 8th IM of the second patient).

For head & neck data in Table 3.3, the per patient performance is also constantly good with the lowest per patient improvement of 22.9%. For head & neck case, there is a relation between the complexity of the IM (number of MUs without decomposition) and the improvement - all IMs without any improvement have a number of MUs without decomposition of 98 or less. If we only consider IMs having a number of MUs without decomposition of 100 or more, all IMs got improvement and the average improvement in number of MUs is 50.8%. The average improvement for the remaining IMs is 28.9%.

About the two cost function used, the first cost function demonstrates fair improvement in both number of MUs and number of MLC-apertures while the second cost function generates much better improvement in number of MUs but a little compromised number of MLC-apertures. This is compliant with Süss and Küfer's statement [72].

CHAPTER 4
FIELD SPLITTING PROBLEM

4.1 Problem Modeling

In this chapter, we study the following **optimal field splitting with balanced beam-on times (OFSB)** problem. Given an IM $A = (a_{i,j})_{m \times n}$ of size $m \times n$, an integral maximum leaf spread $\varpi > 0$, and the width range $[\delta \mathinner{..} \Delta]$ of each feathering region ($0 < \delta < \Delta < \varpi$), split A using vertical lines into a sequence of $d = \lceil \frac{n-\delta}{\varpi-\delta} \rceil$ (≥ 2) sub-IMs, such that: (1) the width of each sub-IM is ϖ, except the width of the last sub-IM is larger than 0 and no larger than ϖ; (2) any two neighboring sub-IMs in the sequence overlap each other and the width of the overlapping (feathering) region ranges from δ to Δ; (3) no sub-IM overlaps completely with its neighboring sub-IM(s); and (4) the total complexity of all these d sub-IMs is minimized. In our algorithm, we use the *sum of positive gradients* to measure the complexity $C(A)$ of an IM A [5, 70, 13], more precisely, $C(A) = \sum_{i=1}^{m} \Big(a_{i,1} + \sum_{j=2}^{n} \max(0, a_{i,j} - a_{i,j-1}) \Big)$.

The resulting d sub-IMs, however, may have a large minimum beam-on time, which is undesirable. We thus seek to further decompose the induced (d-1) feathering regions of those d sub-IMs, yielding a *split* $\mathcal{S}$ of d sub-IMs $\{S_1, S_2, \ldots, S_d\}$ (from left to right), such that the maximum $M_{bot}(\mathcal{S})$ of all the minimum beam-on times $T_{bot}(\cdot)$ of these sub-IMs in $\mathcal{S}$ (i.e., $M_{bot}(\mathcal{S}) = \max_{S_k \in \mathcal{S}} T_{bot}(S_k)$) is minimized, while imposing the lower bound on the total complexity $TC(\mathcal{S})$ of the split $\mathcal{S}$ with $TC(\mathcal{S}) = \sum_{S_k \in \mathcal{S}} C(S_k)$. Note that d is the minimum number of sub-IMs needed. We may use more sub-IMs, which however, could undesirably increase the total treatment time. Thus, we assume that each sub-IM has a maximum width ϖ since we can introduce columns filled with 0's to the sub-IM without increasing its complexity.

Notice that the complexity definition we use is the sum of positive gradients (see Figure 4.1(a)) and recall the equation we used to capture delivery error in Chapter 1:

$$Err(A) = \sum_{j=1}^{n} \left\{ a_{1,j} + \sum_{i=1}^{m-1} |a_{i,j} - a_{i+1,j}| + a_{m,j} \right\}.$$

This definition is actually the sum of all gradients, both positive and negative(see Figure 4.1(b)) and thus the delivery error of an IM is twice the value of our complexity definition. So our method helps not only in improving the efficiency but also in reducing the delivery error.

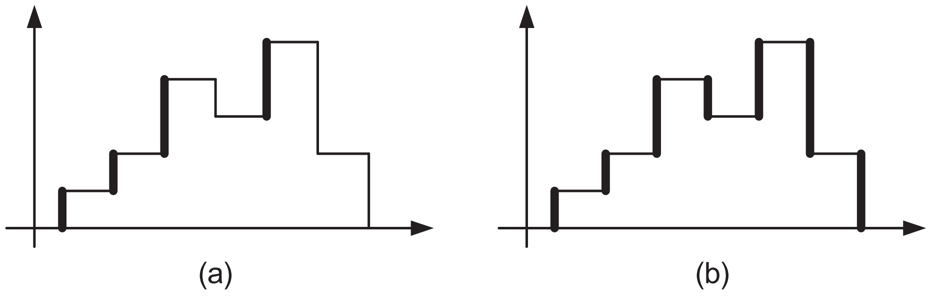


Figure 4.1: Comparison between our complexity definition and delivery error on vector $(1,2,4,3,5,2)$. (a) Our complexity definition: sum of positive gradients (in thick edges). (b) Delivery error: sum of all gradients (in thick edges).

We present the first ever near linear time algorithm for solving the above field splitting problem. In our algorithm, we first model the computation of an "optimal" set of (d-1) feathering regions (i.e., with minimum total increase of the complexity) as a shortest path problem in a directed acyclic graph (DAG) with $O(n)$ vertices and $O(n(\Delta - \delta))$ edges. This DAG has a special "layered" structure, which consists of d layers of vertices with any two adjacent layers inducing a bipartite graph. We are able to calculate each edge weight in constant time after a certain preprocessing. Moreover, the edge weights of the DAG satisfy the Monge property [1]. Thus, we can solve this shortest path problem by examining only a small portion of the graph, and our algorithm runs in a near linear time (Section 4.2). Then, the decomposition of the resulting feathering regions is modeled as computing a min-max slope path problem in a special monotone polygon, which is of its own interest. We develop an interesting geometric algorithm, which runs in $O(mn\alpha(\varpi))$ time, where $\alpha(\cdot)$ is the inverse Ackermann function., for solving the min-max slope path problem (Section 4.3). A less effective algorithm for a more general case is provided in Section 4.4.

4.2 Computing an Optimal Set of Feathering Regions

In this section, we compute a set $\mathcal{F}$ of (d-1) feathering regions for a given instance of the OFSB problem, such that the decomposition of the feathering regions in $\mathcal{F}$ yields a split of the input IM A, whose total complexity is minimized. The problem is modeled as computing a shortest path in a directed acyclic graph, for which we can exploit the Monge property to speed up the computation. The running time of our algorithm is $O(mn\alpha(\varpi))$, where $\alpha(\cdot)$ is the inverse Ackermann function.

4.2.1 The Shortest Path Model

Denote by $A[j]$ the j-th column of IM A, i.e., $A[j] = \{a_{1,j}, a_{2,j}, \ldots, a_{m,j}\}$, and $A[j \,..\, k]$ consists of all elements of A from Column j to Column k. Since the width of each sub-IM is fixed as ϖ, d vertical lines $\{j_1, j_2, \ldots, j_d\}$ are needed to determine the starting column of each sub-IM in the split (including the first vertical line which is always corresponding to the first column of A, i.e., $j_1 = 1$). The k-th feathering region F_k consists of multiple columns of A starting from Column j_{k+1} to Column $j_k + \varpi - 1$. F_k is somehow decomposed into $F_k^{(0)}$ and $F_k^{(1)}$ such that $F_k = F_k^{(0)} + F_k^{(1)}$ (i.e., the value of every element in F_k is decomposed into two non-negative integers, one in $F_k^{(0)}$ and the other in $F_k^{(1)}$). Then, a feasible split $\mathcal{S} = \{S_1, S_2, \ldots, S_d\}$ of A is defined, as follows. For each $k = 1, 2, \ldots, d$, $S_k = F_{k-1}^{(1)} \,||\, A[j_{k-1} + \varpi \,..\, j_{k+1} - 1] \,||\, F_k^{(0)}$, where $||$ is a concatenation operator, $F_0^{(1)} = F_d^{(0)} = \varnothing$, $j_0 = -\varpi + 1$ and $j_{d+1} = n + 1$. The decomposition of each feathering region F_k may increase the total complexity. Our goal is to find a set $\mathcal{F}$ of (d-1) feathering regions such that the total increase of the complexity resulting from the decomposition of all the feathering regions in $\mathcal{F}$ is minimized. We next model this problem as computing a shortest path in a weighted directed acyclic graph $G = (V, E)$.

1) The graph G has d layers of vertices, where each vertex in the k-th layer (denoted by L_k) defines a possible starting column of the k-th sub-IM. The first layer L_1 consists of only one vertex u_1, i.e., the first sub-IM S_1 has to start at the first column of A. For the k-th sub-IM S_k, there are $k - 1$ (resp., $d - k$) sub-IMs to the left (resp., right) of it. Thus, the rightmost (resp., leftmost) possible

starting column of sub-IM S_k is $(k-1)(\varpi-\delta)+1$ (resp., $(k-1)(\varpi-\delta)+1-\mu$), where $\mu=(n-\delta)\%(\varpi-\delta)$. Thus, each layer L_k $(k=2,\ldots,d)$ contains $\mu+1$ vertices $\{u_j \mid (k-1)(\varpi-\delta)+1-\mu \leq j \leq (k-1)(\varpi-\delta)+1\}$. Note that no sub-IM overlaps *completely* with its neighboring sub-IMs. Hence, the maximum width Δ of a feathering region is less than $(\varpi-\delta)$. We thus can prove that the layers are mutually exclusive (i.e., $L_k \cap L_{k+1}=\varnothing$ for $k=1,2,\ldots,d-1$).

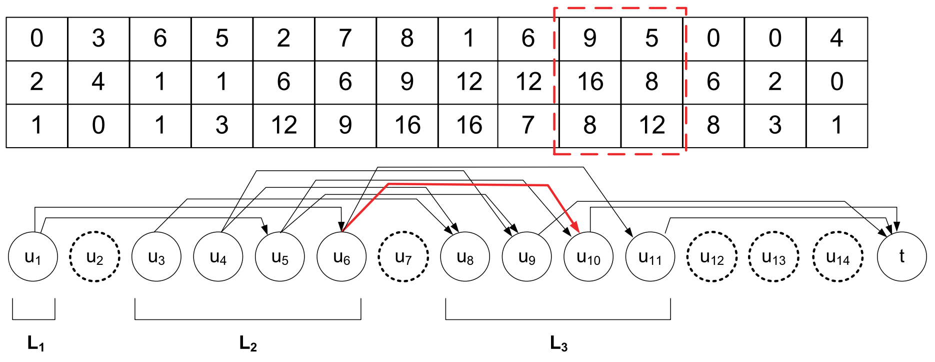
2) For each vertex $u_j \in L_k$ in G, there is a directed edge from u_j to every $u_{j'} \in L_{k+1}$ as long as the two corresponding sub-IMs overlap each other with an overlapping region of width between δ and Δ. Thus, each edge $(u_j, u_{j'})$ defines a feathering region $A[j' \mathinner{..} j+\varpi-1]$.

3) For each edge $e=(u_j,u_{j'})$ in G, we compute the minimum increased complexity Δ_{cpl} of the corresponding feathering region and assign it as the weight of the edge, denoted by $c(u_j,u_{j'})$.

4) For the ease of our algorithm description, we introduce a dummy vertex t. Each vertex in the d-th layer L_d has a directed edge to t with a weight of 0.

An example of the constructed graph is shown in Figure 4.2. Our algorithm then computes a shortest u_1-to-t path p: $u_1 \rightarrow u_{j_2} \rightarrow \cdots \rightarrow u_{j_{d-1}} \rightarrow u_{j_d} \rightarrow t$ in G, where $u_{j_k} \in L_k$. Obviously, the d sub-IMs defined by the starting columns in $\{1, j_2, \ldots, j_{d-1}, j_d\}$ specify a feasible split $\mathcal{S}^*$ of A. The total increase of the complexity due to the split $\mathcal{S}^*$ equals the total path weight $c(p)$, which is minimized. Thus, $\mathcal{S}^*$ is a split of A minimizing the total increase of the complexity.

We next in Section 4.2.2 show how to efficiently compute Δ_{cpl} for each pos-



0	3	6	5	2	7	8	1	6	9	5	0	0	4
2	4	1	1	6	6	9	12	12	16	8	6	2	0
1	0	1	3	12	9	16	16	7	8	12	8	3	1

Figure 4.2: Illustrating the constructed graph for an example intensity map. In the example $n = 14$, $\varpi = 6$, $\delta = 1$ and $\Delta = 2$. L_1, L_2, and L_3 shows layered structure of the nodes. Edges are added to guarantee the minimum and maximum feathering width. The red thick edge is corresponding to the increased complexity by splitting the feathering region surrounded by the red dashed box.

sible feathering regions. Actually, the computation can be done in $O(m)$ time after an $O(mn)$ time preprocessing. The Monge property of the graph G is explored in Section 4.2.3 to speed up the computation of the shortest u_1-to-t path in G.

4.2.2 Computing the Minimum Increase of the Complexity for a Feathering Region

In this section, we characterize the complexity increase due to the feathering region decomposition. Then, a linear time algorithm is developed to compute the minimum increase of the complexity Δ_{cpl} for a feathering region.

4.2.2.1 Characterizing the Increase of the Complexity

Consider a feathering region $F_k = A[l\,..\,r]$ with $\delta \leq r - l = \omega \leq \Delta$ $(1 < l < r < n)$, which is the overlapping region of sub-IMs S_k and S_{k+1}. Assume that the decomposition of F_k is $F_k^{(0)} = (x_{i,j})_{m\times\omega}$ plus $F_k^{(1)} = (y_{i,j})_{m\times\omega}$. Then, S_k ends with

$F_k^{(0)}$ and S_{k+1} starts with $F_k^{(1)}$. The contribution $R(F_k)$ of the feathering region F_k ($A[l\,..\,r]$) to the complexity of IM A is $\sum_{i=1}^{m}\sum_{j=l}^{r+1}\max\{0, a_{i,j} - a_{i,j-1}\}$. While the contribution $R(F_k^{(0)})$ to the complexity of S_k is

$$\sum\nolimits_{i=1}^{m}(\max\{0, x_{i,l} - a_{i,l-1}\} + \sum\nolimits_{j=l+1}^{r}\max\{0, x_{i,j} - x_{i,j-1}\} + \max\{0, 0 - x_{i,r}\}),$$

and the contribution $R(F_k^{(1)})$ to the complexity of S_{k+1} is

$$\sum\nolimits_{i=1}^{m}(\max\{0, y_{i,l} - 0\} + \sum\nolimits_{j=l+1}^{r}\max\{0, y_{i,j} - y_{i,j-1}\} + \max\{0, a_{i,r+1} - y_{i,r}\}).$$

Thus, the *increase* of the complexity due to the decomposition (i.e., $F_k^{(0)}$ and $F_k^{(1)}$) of the feathering region F_k is $R(F_k^{(0)}) + R(F_k^{(1)}) - R(F_k)$. We next develop a linear time algorithm for an optimal decomposition of an feathering region to minimizing the increase of the complexity.

Observing that the decomposition of F_k can be performed row by row, we define the following **optimal vector decomposition (OVD)** problem. Define the *weight* $W_{ovd}(\mathbf{z})$ of a vector $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ as $\sum_{j=2}^{N}\max\{0, z_j - z_{j-1}\}$. Given a non-negative integer vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$, decompose $\mathbf{b}$ into two non-negative vectors $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, such that (1) $x_1 = b_1$ and $y_N = b_N$; (2) for each $j = 1, 2, \ldots, N$, $b_j = x_j + y_j$, and (3) the total weight of $\mathbf{x}$ and $\mathbf{y}$ (i.e., $W_{ovd}(\mathbf{x}) + W_{ovd}(\mathbf{y})$) is minimized.

Then, for a given feathering region $F_k = A[l \mathinner{..} r]$, each *extended* row $(a_{i,l-1}, a_{i,l}, \ldots, a_{i,r}, a_{i,r+1})$ is viewed as a vector $\mathbf{a}_i$. Applying the OVD algorithm, $\mathbf{a}_i$ is decomposed into two vectors, $\mathbf{x}_i = (a_{i,l-1}, x_{i,l}, \ldots, x_{i,r}, 0)$ and $\mathbf{y}_i = (0, y_{i,l}, \ldots, y_{i,r}, a_{i,r+1})$, with $W_{ovd}(\mathbf{x}_i) + W_{ovd}(\mathbf{y}_i)$ being minimized. Clearly, $(\mathbf{x}_i)_{i=1}^m$ and $(\mathbf{y}_i)_{i=1}^m$ can be used to specify a decomposition of F_k (i.e., $F_k^{(0)}$ and $F_k^{(1)}$). Thus, $F_k^{(0)}$ and $F_k^{(1)}$ is an optimal decomposition of F_k minimizing the increase of the complexity.

We next develop a linear time algorithm for solving the OVD problem in Section 4.2.2.2. Chen and Wang [18] also independently studied this OVD problem for a different purpose and a linear time algorithm was obtained. Our algorithm is essentially the same as the algorithm of Chen and Wang; however, in light of earlier result, our formulation is arguably more natural and general.

4.2.2.2 Linear Time Algorithm for Optimal Vector Decomposition (OVD) Problem

This section presents our optimal $O(N)$ time algorithm for computing an optimal decomposition of a given vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$. The OVD problem is modeled as computing a shortest path in a directed acyclic graph (DAG) of pseudo-polynomial size. By exploiting the convexity of the edge weight functions of the graph, we show that the OVD problem can be optimally solved in linear time without explicitly constructing and searching the whole graph.

1. The Graph Model of the OVD Problem

Given a non-negative integer vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$, we define an edge-weighted DAG $H = (V_H, E_H)$ for the OVD problem, as follows.

The element b_1 (resp., b_N) of $\mathbf{b}$ corresponds to exactly one vertex $v_1(b_1)$ (resp., $v_N(b_N)$) in V_H, briefly called the *source* (resp., *sink*) vertex s (resp., t) of H. For every other element b_j ($j = 2, 3, \ldots, N-1$), there is a set $Col(j)$ of $b_j + 1$ vertices in H corresponding to b_j, with $Col(j) = \{v_j(h) \mid h = 0, 1, \ldots, b_j\}$, namely the *$b_j$-column* of H. Intuitively, the vertices in $Col(j)$ give all possible distinct ways to decompose b_j into two non-negative integers (i.e., each vertex $v_j(h)$ corresponds to decomposing b_j into h and $b_j - h$). Note that Column $Col(1)$ (resp., $Col(N)$) consists of only one vertex $v_1(b_1)$ (resp., $v_N(b_N)$). For any two adjacent columns $Col(j)$ and $Col(j+1)$ ($j = 1, 2, \ldots, N-1$), each vertex $v_j(h) \in Col(j)$ has a directed edge e to every vertex $v_{j+1}(h')$, with an edge *weight* $w(e) = \max\{0, h' - h\} + \max\{0, (b_{j+1} - h') - (b_j - h)\}$. For the convenience of our discussion, we let $w_{\mathbf{x}}(e) = \max\{0, h' - h\}$ and $w_{\mathbf{y}}(e) = \max\{0, (b_{j+1} - h') - (b_j - h)\}$.

Consider any s-t path p in H, with $p = v_1(h_1) \to v_2(h_2) \to \ldots \to v_{N-1}(h_{N-1}) \to v_N(h_N)$, where $h_1 = b_1$ and $h_N = b_N$ (i.e., $v_1(h_1)$ is the source s and $v_N(h_N)$ is the sink t). Let $\mathbf{x}(p) = (x_1, x_2, \ldots, x_N)$ and $\mathbf{y}(p) = (y_1, y_2, \ldots, y_N)$ be two non-negative integer vectors defined from the path p, in the following way: for each $j = 1, 2, \ldots, N$, $x_j = h_j$ and $y_j = b_j - h_j$. Note that each s-t path p in H actually define a feasible decomposition of $\mathbf{b}$, i.e., $\mathbf{b} = \mathbf{x}(p) + \mathbf{y}(p)$. The total weight of $\mathbf{x}(p)$ and $\mathbf{y}(p)$, $W_{ovd}(\mathbf{x}(p)) + W_{ovd}(\mathbf{y}(p))$, equals to the total sum of the weights of the edges on p, i.e., $w(p) = W_{ovd}(\mathbf{x}(p)) + W_{ovd}(\mathbf{y}(p))$. We further introduce the following notation: $w_{\mathbf{x}}(p) = W_{ovd}(\mathbf{x}(p))$ and $w_{\mathbf{y}}(p) = W_{ovd}(\mathbf{y}(p))$. Hence, a shortest s-t path in H, which can be computed in $O(|V| + |E|)$ time, specifies an optimal decomposition of $\mathbf{b}$.

This is a pseudo-polynomial time algorithm for the OVD problem, which may not be efficient enough, especially when the elements of **b** are large. However, this DAG model lays down a base for further exploring the intrinsic structures of the OVD problem.

2. Our Optimal OVD Algorithm

Our OVD algorithm hinges on the piecewise linearity and convexity of the edge weight functions of H. For each $j = 1, 2, \ldots, N-1$, based on b_j and b_{j+1}, we define a function $f_j(\mathit{\Delta}h)$: $\mathbb{Z} \rightarrow \mathbb{Z}^+$, as follows.

(1) If $b_j \leq b_{j+1}$,

$$f_j(\mathit{\Delta}h) = \begin{cases} \mathit{\Delta}h + (b_{j+1} - b_j), & \text{if } \mathit{\Delta}h > 0 \\ b_{j+1} - b_j, & \text{if } b_j - b_{j+1} \leq \mathit{\Delta}h \leq 0 \\ -\mathit{\Delta}h, & \text{if } \mathit{\Delta}h < b_j - b_{j+1} \end{cases} \qquad (*)$$

(2) If $b_j > b_{j+1}$,

$$f_j(\mathit{\Delta}h) = \begin{cases} \mathit{\Delta}h + (b_{j+1} - b_j), & \text{if } \mathit{\Delta}h > b_j - b_{j+1} \\ 0, & \text{if } 0 \leq \mathit{\Delta}h \leq b_j - b_{j+1} \\ -\mathit{\Delta}h, & \text{if } \mathit{\Delta}h < 0 \end{cases} \qquad (**)$$

Note that $f_j(\mathit{\Delta}h)$ is piecewise linear and convex with respect to $\mathit{\Delta}h$. For any edge $(v_j(h), v_{j+1}(h'))$ between two adjacent columns $Col(j)$ and $Col(j+1)$, Lemma 13 reveals the relation between the edge weight $w(v_j(h), v_{j+1}(h'))$ and the function

$f_j(\Delta h)$.

Lemma 13. *$w(v_j(h), v_{j+1}(h')) = f_j(h - h')$.*

We next explore the structures of any shortest s-t path p passing a specific vertex $v_{\hat{j}}(\hat{h}) \in Col(\hat{j})$ in H. The path p consists of two shortest paths: one is from s to $v_{\hat{j}}(\hat{h})$, denoted by $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$, and the other is from $v_{\hat{j}}(\hat{h})$ to t, denoted by $v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t$. Both shortest paths $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$ and $v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t$ can defined using the following series $\{h_j^-\}_{j=1}^{\hat{j}}$ and $\{h_j^+\}_{j=\hat{j}}^{N}$, respectively.

$$h_j^- = \begin{cases} \hat{h}, & j = \hat{j} \\ \min\left\{b_j, h_{j+1}^- + \max\{0, b_j - b_{j+1}\}\right\}, & 1 < j < \hat{j} \\ b_1, & j = 1 \end{cases}$$

$$h_j^+ = \begin{cases} \hat{h}, & j = \hat{j} \\ \max\left\{0, h_{j-1}^+ - \max\{0, b_{j-1} - b_j\}\right\}, & \hat{j} < j < N \\ 0, & j = N \end{cases}$$

Intuitively, for any $j < \hat{j}$, $v_j(h_j^-)$ is the "top-most" vertex on $Col(j)$ such that the edge $(v_j(h_j^-), v_{j+1}(h_{j+1}^-))$ has the minimum edge weight among all edges connecting a vertex in $Col(j)$ to $v_{j+1}(h_{j+1}^-)$; while for any $j > \hat{j}$, $v_j(h_j^+)$ is the "lowest" vertex on $Col(j)$ such that the edge $(v_{j-1}(h_{j-1}^+), v_j(h_j^+))$ has the minimum edge weight among all edges connecting $v_{j-1}(h_{j-1}^+)$ to every vertex of $Col(j)$. Figure 4.3 demonstrates some example of these basic ideas.

Lemma 14. *(14a) The path $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$ defined by the series $\{h_j^-\}_{j=1}^{\hat{j}}$, with $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h}) =$*

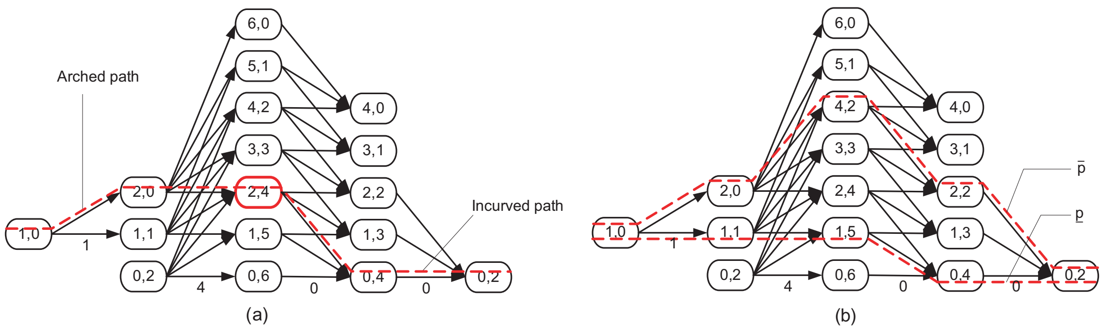


Figure 4.3: Illustrating the graph model for decomposing the vector $(1, 2, 6, 4, 2)$. Only edges with minimum weights between two columns are shown and the minimum edge weight is labeled below each set of edges. (a) The shortest s-t path through $v_3(2)$, which is the concatenation the arched path $s \overset{p^-}{\rightsquigarrow} v_3(2)$ and the incurved path $v_3(2) \overset{p^+}{\rightsquigarrow} t$. (b) $\overline{p} = s \overset{p^-}{\rightsquigarrow} t$ (upper) and $\underline{p} = s \overset{p^+}{\rightsquigarrow} t$ (lower).

$v_1(h_1^-) \to v_2(h_2^-) \to \ldots \to v_{\hat{j}}(h_{\hat{j}}^-)$ $(h_{\hat{j}}^- = \hat{h})$, is a shortest path from s to $v_{\hat{j}}(\hat{h})$, and $w_{\mathbf{y}}(s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})) = b_{\hat{j}} - \hat{h}$.

(14b) The path $v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t$ defined by the series $\{h_j^+\}_{j=\hat{j}}^N$, with $v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t = v_{\hat{j}}(h_{\hat{j}}^+) \to v_{\hat{j}+1}(h_{\hat{j}+1}^+) \to \ldots \to v_N(h_N^+)$ $(h_{\hat{j}}^+ = \hat{h})$, is a shortest path from $v_{\hat{j}}(\hat{h})$ to t, and $w_{\mathbf{x}}(v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t) = 0$.

(14c) The weight $w(s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h}))$ of the path $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$ is $\max\{\hat{h}, \sum_{j=2}^{\hat{j}} \max\{0, b_j - b_{j-1}\}\}$.

Proof. Let $sw_j^-(h)$ denote the shortest path weight from s to $v_j(h)$ and $sw_j^+(h)$ denote the shortest path weight from $v_j(h)$ to t. We first prove the following claim.

Claim 1. *(i) $sw_j^-(h+1) \leq sw_j^-(h) \leq sw_j^-(h+1) + 1$, for $j = 2, 3, \ldots, N$ and $h = 0, \ldots, b_j - 1$.*

(ii) $sw_j^+(h-1) \leq sw_j^+(h) \leq sw_j^+(h-1)+1$, for $j = 1, 2, \ldots, N-1$ and $h = 1, 2, \ldots, b_j$.

Proof. We only show by induction on j that the first part of the claim holds. The second part of the claim can be proved in a similar way.

For $j = 2$, based on Lemma 13, $sw_2^-(h) = w(s, v_2(h)) = f_1(b_1 - h)$. It follows from the definition of $f_1(\Delta h)$ that $sw_2^-(h+1) \leq sw_2^-(h) \leq sw_2^-(h+1)+1$ for $h = 0, 1, \ldots, b_2 - 1$. We next consider the shortest path weight $sw_j^-(h)$ from s to $v_j(h)$ for $j > 2$.

Let $s \rightsquigarrow v_j(h+1) = s \rightarrow v_2(h_2) \rightarrow \ldots \rightarrow v_{j-1}(h') \rightarrow v_j(h+1)$ be a shortest path. From Lemma 13, $w(v_{j-1}(h'), v_j(h)) = f_{j-1}(h' - h)$ and $w(v_{j-1}(h'), v_j(h+1)) = f_{j-1}(h' - h - 1)$. It is not hard to see that $f_{j-1}(\Delta h) \leq f_{j-1}(\Delta h - 1) + 1$ holds from the definition of $f_{j-1}(\cdot)$. Thus, $w(v_{j-1}(h'), v_j(h)) \leq w(v_{j-1}(h'), v_j(h+1)) + 1$. By replacing edge $(v_{j-1}(h'), v_j(h+1))$ by $(v_{j-1}(h'), v_j(h))$ on the path $s \rightsquigarrow v_j(h+1)$, we obtain a path p' from s to $v_j(h)$. Hence, $sw_j^-(h) \leq w(s \overset{p'}{\rightsquigarrow} v_j(h)) = sw_j^-(h+1) - w(v_{j-1}(h'), v_j(h+1)) + w(v_{j-1}(h'), v_j(h)) \leq sw_j^-(h+1) + 1$.

Next, we show that $sw_j^-(h) \geq sw_j^-(h+1)$. Let $s \rightsquigarrow v_j(h) = s \rightarrow v_2(h_2) \rightarrow \ldots \rightarrow v_{j-1}(h') \rightarrow v_j(h)$ be a shortest path. Two cases are distinguished.

Case (1): $h' = b_{j-1}$. Note that $f_{j-1}(b_{j-1} - h)$ is decreasing with respect to h. While $w(v_{j-1}(h'), v_j(h)) = f_{j-1}(h' - h)$ and $w(v_{j-1}(h'), v_j(h+1)) = f_{j-1}(h' - h - 1)$. Thus, $w(v_{j-1}(h'), v_j(h+1)) \leq w(v_{j-1}(h'), v_j(h))$. If we substitute edge $(v_{j-1}(h'), v_j(h))$ on the path $s \rightsquigarrow v_j(h)$ by edge $(v_{j-1}(h'), v_j(h+1))$, we have a path p' from s to $v_j(h+1)$. Consequently, $sw_j^-(h+1) \leq w(s \overset{p'}{\rightsquigarrow} v_j(h+1)) = sw_j^-(h) - w(v_{j-1}(h'), v_j(h)) + w(v_{j-1}(h'), v_j(h+1)) \leq sw_j^-(h)$.

Case (2): $h' \neq b_{j-1}$. We have $w(v_{j-1}(h'), v_j(h)) = f_{j-1}(h' - h) = w(v_{j-1}(h' +

$1), v_j(h+1))$. By the induction hypothesis, $sw^-_{j-1}(h'+1) \leq sw^-_{j-1}(h')$. As a result, $sw^-_j(h+1) \leq sw^-_{j-1}(h'+1) + w(v_{j-1}(h'+1), v_j(h+1)) \leq sw^-_{j-1}(h') + w(v_{j-1}(h'), v_j(h)) = sw^-_j(h)$.

This proves the first part of the claim. ¶

We are now ready to prove Lemma 14. Again, we only prove by induction Lemma 14a and 14c. Lemma 14b follows by a similar argument for Lemma 14a.

(14a) For $\hat{j} = 2$, it trivially holds. Assuming that the statement holds for any $j < \hat{j}$, we show that it also holds for $\hat{j}$. Note that $sw^-_{\hat{j}}(\hat{h}) = \min_{h=0}^{b_{\hat{j}-1}}(sw^-_{\hat{j}-1}(h) + w(v_{\hat{j}-1}(h), v_{\hat{j}}(\hat{h})))$. Let $h^-_{\hat{j}-1} = \min\left\{b_{\hat{j}-1}, \hat{h} + \max\{0, b_{\hat{j}-1} - b_{\hat{j}}\}\right\}$ $(\hat{h} = h^-_{\hat{j}})$. By the definition of $f_{\hat{j}-1}(\cdot)$ and Lemma 13, for any h with $h^-_{\hat{j}-1} \leq h < b_{\hat{j}-1}$, we have $w(v_{\hat{j}-1}(h+1), v_{\hat{j}}(\hat{h})) = w(v_{\hat{j}-1}(h), v_{\hat{j}}(\hat{h})) + 1$; while for each $h < h^-_{\hat{j}-1}$, $w(v_{\hat{j}-1}(h+1), v_{\hat{j}}(\hat{h})) \leq w(v_{\hat{j}-1}(h), v_{\hat{j}}(\hat{h}))$. On the other hand, from Claim 1, $sw^-_{\hat{j}-1}(h)$ is monotonically decreasing with respect to h and $sw^-_{\hat{j}-1}(h) - sw^-_{\hat{j}-1}(h+1) \leq 1$. Thus, when $h = h^-_{\hat{j}-1}$, $sw^-_{\hat{j}}(\hat{h})$ achieves its minimum value. Together by the induction hypothesis, the statement holds for $\hat{j}$.

We then show that $w_{\mathbf{y}}(s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})) = b_{\hat{j}} - \hat{h}$. Consider the corresponding decomposition $\mathbf{x}(p^-)$ and $\mathbf{y}(p^-) = (y_1, y_2, \ldots, y_{\hat{j}})$ of the path $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$. We have $y_1 = 0$, and $y_j = b_j - h^-_j$ $(j = 2, 3, \ldots, \hat{j})$. Since $h^-_j = \min\left\{b_j, h^-_{j+1} + \max\{0, b_j - b_{j+1}\}\right\}$, we consider two cases: (1) $h^-_j = b_j$, and (2) $h^-_j = h^-_{j+1} + \max\{0, b_j - b_{j+1}\}$. A careful analysis reveals that in both cases $y_j \leq y_{j+1}$. Hence, $w_{\mathbf{y}}(p^-) = W_{ovd}(\mathbf{y}) = \sum_{j=1}^{\hat{j}-1} \max\{0, y_{j+1} - y_j\} = \sum_{j=1}^{\hat{j}-1}(y_{j+1} - y_j) = y_{\hat{j}} = b_{\hat{j}} - \hat{h}$.

Thus, Lemma 14a follows.

(14c) To demonstrate $w(s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})) = \max\{\hat{h}, \sum_{j=2}^{\hat{j}} \max\{0, b_j - b_{j-1}\}\}$, it suffices to show that for any $j \in \{2, 3, \ldots, \hat{j}\}$, $sw_j(h_j^-) = \max\{h_j^-, \sum_{k=2}^{j} \max\{0, b_k - b_{k-1}\}\}$. We prove it by induction on j. For $j = 2$, it is trivial. Assuming that the statement holds for all $j \leq r$, we show that it also holds for $j = r + 1$.

Since $h_r^- = \min\{b_r, h_{r+1}^- + \max\{0, b_r - b_{r+1}\}\}$, it is easy to show $0 \leq h_r^- - h_{r+1}^- \leq \max\{0, b_r - b_{r+1}\}$. By induction hypothesis, $sw_r(h_r^-) = \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\}$. Hence

$$\begin{aligned}
sw_{r+1}(h_{r+1}^-) &= sw_r(h_r^-) + w(v_r(h_r^-), v_{r+1}(h_{r+1}^-)) \\
&= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} + \max\{0, h_{r+1}^- - h_r^-\} \\
&\quad + \max\{0, (b_{r+1} - h_{r+1}^-) - (b_r - h_r^-)\} \\
&= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} \\
&+ h_{r+1}^- - h_r^- + \max\{0, b_{r+1} - b_r - (h_{r+1}^- - h_r^-)\} \\
&= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} + \max\{h_{r+1}^- - h_r^-, b_{r+1} - b_r\}.
\end{aligned}$$

Consider two possible cases.

Case (I) $b_{r+1} < b_r$. In this case, we have $h_r^- = h_{r+1}^-$ and $\max\{0, b_{r+1} - b_r\} = 0$. Thus, it follows

$$\begin{aligned}
sw_{r+1}(h_{r+1}^-) &= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} + 0 \\
&= \max\{h_{r+1}^-, \sum_{k=2}^{r+1} \max\{0, b_k - b_{k-1}\}\}.
\end{aligned}$$

Case (II) $b_{r+1} \geq b_r$. In this case, $h_r^- = \max\{0, h_{r+1}^- - b_{r+1} + b_r\}$, $h_r^- + b_{r+1} - b_r = \max\{h_{r+1}^-, b_{r+1} - b_r\}$, and $b_{r+1} - b_r \geq h_{r+1}^- - h_r^-$. Hence,

$$\begin{aligned}
sw_{r+1}(h_{r+1}^-) &= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} + \max\{h_{r+1}^- - h_r^-, b_{r+1} - b_r\} \\
&= \max\{h_r^-, \sum_{k=2}^{r} \max\{0, b_k - b_{k-1}\}\} + b_{r+1} - b_r \\
&= \max\{h_r^- + b_{r+1} - b_r, \sum_{k=2}^{r+1} \max\{0, b_k - b_{k-1}\}\} \\
&= \max\{\max\{h_{r+1}^-, b_{r+1} - b_r\}, \sum_{k=2}^{r+1} \max\{0, b_k - b_{k-1}\}\} \\
&= \max\{h_{r+1}^-, b_{r+1} - b_r, \sum_{k=2}^{r+1} \max\{0, b_k - b_{k-1}\}\} \\
&= \max\{h_{r+1}^-, \sum_{k=2}^{r+1} \max\{0, b_k - b_{k-1}\}\}.
\end{aligned}$$

□

We call the path $s \overset{p^-}{\rightsquigarrow} v_{\hat{j}}(\hat{h})$ an *arched path* from s to $v_{\hat{j}}(\hat{h})$, while $v_{\hat{j}}(\hat{h}) \overset{p^+}{\rightsquigarrow} t$ an *incurved path* from $v_{\hat{j}}(\hat{h})$ to t. In the rest of the paper, we use the notation $u \overset{p^-}{\rightsquigarrow} v$ (resp., $u \overset{p^+}{\rightsquigarrow} v$) to denote an arched (resp., incurved) path from u to v. It immediately follows from Lemma 14 that when $v_{\hat{j}}(\hat{h}) = v_N(0)$, the series $\{h_j^-\}_{j=1}^N$ defines an s-t shortest path in H, denoted by $\overline{p}$ (i.e., $\overline{p} = s \overset{p^-}{\rightsquigarrow} t$), and when $v_{\hat{j}}(\hat{h}) = v_1(b_1)$, the series $\{h_j^+\}_{j=1}^N$ specifies another s-t shortest path in H, denoted by $\underline{p}$ (i.e., $\underline{p} = s \overset{p^+}{\rightsquigarrow} t$).

From the graph model of the OVD problem and Lemmas 13 and 14, the following lemma immediately follows.

Lemma 15. *(1) Each of the s-t shortest paths $\overline{p}$ and $\underline{p}$ in H can be computed in*

$O(N)$ time, and $w_{\mathbf{y}}(\overline{p}) = b_N$ and $w_{\mathbf{x}}(\underline{p}) = 0$.

(2) $w(\overline{p}) = \max\{b_N, \sum_{j=2}^{N} \max\{0, b_j - b_{j-1}\}\}$. (3) The OVD problem can be solved in an optimal linear time.

For a given non-negative vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$, define two vectors $\mathbf{x}^* = (x_1, x_2, \ldots, x_N)$ and $\mathbf{y}^* = (y_1, y_2, \ldots, y_N)$, as follows.

$$x_j = \begin{cases} b_1, & j = 1 \\ \max\{0, x_{j-1} - \max\{0, b_{j-1} - b_j\}\}, & 1 < j < N \\ 0, & j = N \end{cases} \tag{4.1}$$

For every $j = 1, 2, \ldots, N$, $y_j = b_j - x_j$. The following lemma then immediately follows from Lemma 15.

Lemma 16. *Given a non-negative vector $\mathbf{b} = (b_1, b_2, \ldots, b_N)$, an optimal decomposition $\mathbf{x}^*$ and $\mathbf{y}^*$ of $\mathbf{b}$ can be computed in $O(N)$ time; furthermore, $W_{ovd}(\mathbf{x}^*) = 0$ and the total weight of $\mathbf{x}^*$ and $\mathbf{y}^*$, $W_{ovd}(\mathbf{x}^*) + W_{ovd}(\mathbf{y}^*) = \max\{b_N, \sum_{j=2}^{N} \max\{0, b_j - b_{j-1}\}\}$*

4.2.2.3 Computing the minimum increase of the complexity

As analyzed in Section 4.2.2.1, for a given feathering region $F_k = A[l\,..\,r]$, we can view each extended row $(a_{i,l-1}, a_{i,l}, \ldots, a_{i,r}, a_{i,r+1})$ as a vector $\mathbf{a}_i$. Applying the OVD algorithm to decompose each $\mathbf{a}_i$ into two vectors, $\mathbf{x}_i$ and $\mathbf{y}_i$, we have the minimum increase $\Delta_{cpl}(F_k)$ of the complexity for F_k is $\sum_{i=1}^{m}[W_{ovd}(\mathbf{x}_i) + W_{ovd}(\mathbf{y}_i) - W_{ovd}(\mathbf{a}_i)]$, which equals $\sum_{i=1}^{m} \max\{0, a_{i,r+1} - \sum_{j=l}^{r+1} \max\{0, a_{i,j} - a_{i,j-1}\}\}$ based on Lemma 16. We thus can introduce an additional matrix $B = (b_{i,j})_{m\times n}$ such that

$b_{i,j} = \sum_{k=1}^{j+1} \max\{0, a_{i,k} - a_{i,k-1}\}\}$ $(a_{i,0} = a_{i,n+1} = 0)$. The matrix B can be computed in $O(mn)$ time. Then, $\Delta_{cpl}(F_k)$ for any feathering region F_k can be obtained in $O(m)$ time.

Lemma 17. *After an $O(mn)$ time preprocessing, the minimum increase $\Delta_{cpl}(F_k)$ of the complexity for any feathering region $F_k = A[l\,..\,r]$ can be computed in $O(m)$ time, with $\Delta_{cpl}(F_k) = \sum_{i=1}^{m} \max\{0, a_{i,r+1} - \sum_{j=l}^{r+1} \max\{0, a_{i,j} - a_{i,j-1}\}\}$.*

4.2.3 Speeding Up the Computation of the Feathering Regions

In this section, we exploit the Monge property [1] of the graph $G = (V, E)$ defined in Section 4.2.1. This property enables us to compute in a near linear time a shortest u_1-to-t path in G, which defines an optimal set of feathering regions for the field splitting problem.

Lemma 18. *Given four vertices $u_{j'}, u_{j'+1} \in L_k$ and $u_{j''}, u_{j''+1} \in L_{k+1}$ in G with $2 \leq k < d$, $c(u_{j'}, u_{j''}) + c(u_{j'+1}, u_{j''+1}) \leq c(u_{j'}, u_{j''+1}) + c(u_{j'+1}, u_{j''})$.*

Proof. First, if either $(u_{j'}, u_{j''+1})$ or $(u_{j'+1}, u_{j''})$ is not an edge in G, then we can assume that the weight $c(\cdot,\cdot)$ of an unexisting edge is $+\infty$ and the lemma holds. We thus consider the case that both $(u_{j'}, u_{j''+1})$ and $(u_{j'+1}, u_{j''})$ are edges in G, which implies $\varpi - \Delta + 1 \leq j'' - j' \leq \varpi - \delta - 1$ due to the construction of the graph G in Section 4.2.1, and thus both $(u_{j'}, u_{j''})$ and $(u_{j'+1}, u_{j''+1})$ are edges. Hence, we only need to prove the lemma assuming all $(u_{j'}, u_{j''})$, $(u_{j'+1}, u_{j''+1})$, $(u_{j'}, u_{j''+1})$, and $(u_{j'+1}, u_{j''})$ are edges in G.

Note that an edge (u_j, u_k) in G defines a feathering region $F = A[k \mathinner{..} j+\varpi-1]$ and $c(u_j, u_k)$ is equal to $\Delta_{cpl}(F)$. Based on Lemma 17, it suffices to prove that for each row i of A,

$$\begin{aligned}
&\max\{0, a_{i,j'+\varpi} - \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
+\ &\max\{0, a_{i,j'+\varpi+1} - \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
&\leq \max\{0, a_{i,j'+\varpi} - \sum_{j=j''+1}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
+\ &\max\{0, a_{i,j'+\varpi+1} - \sum_{j=j''}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\}
\end{aligned}$$

Further, it suffices to show that

$$\begin{aligned}
&\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
+\ &\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
&\leq \max\{a_{i,j'+\varpi}, \sum_{j=j''+1}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} \\
+\ &\max\{a_{i,j'+\varpi+1}, \sum_{j=j''}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\}
\end{aligned}$$

Based on the possible values of $\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\}$ and $\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\}$, we distinguish four cases:

I) $\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} = a_{i,j'+\varpi}$,

$\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} = a_{i,j'+\varpi+1}$;

II) $\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} = a_{i,j'+\varpi}$,

$\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} = \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}$;

III) $\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} = \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}$,

$\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} = a_{i,j'+\varpi+1}$;

IV) $\max\{a_{i,j'+\varpi}, \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}\} = \sum_{j=j''}^{j'+\varpi} \max\{0, a_{i,j} - a_{i,j-1}\}$,

$\max\{a_{i,j'+\varpi+1}, \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}\} = \sum_{j=j''+1}^{j'+\varpi+1} \max\{0, a_{i,j} - a_{i,j-1}\}$.

The proof of case III) is shown below. The other cases can be proved in a similar way.

$$\begin{aligned}
& \max\{a_j, \sum_{k=i+1}^{j} \max\{0, a_k - a_{k-1}\}\} + \max\{a_{j+1}, \sum_{k=i+2}^{j+1} \max\{0, a_k - a_{k-1}\}\} \\
= \quad & \sum_{k=i+1}^{j} \max\{0, a_k - a_{k-1}\} + a_{j+1} \\
\leq \quad & \sum_{k=i+1}^{j} \max\{0, a_k - a_{k-1}\} + \max\{0, a_{j+1} - a_j\} + a_j \\
\leq \quad & \max\{a_{j+1}, \sum_{k=i+1}^{j+1} \max\{0, a_k - a_{k-1}\}\} + \max\{a_j, \sum_{k=i+2}^{j} \max\{0, a_k - a_{k-1}\}\}.
\end{aligned}$$

Thus, the lemma holds. □

The Monge property as shown in Lemma 18 can be used to speed up the computation of the shortest u_1-to-t path in G. For every vertex u_j in the k-th layer

L_k, let $sw_k(j)$ denote the shortest path length from u_1 to $u_j \in L_k$ in G. Clearly, $sw_k(j) = \min\{sw_{k-1}(j') + c(u_{j'}, u_j) \mid u_{j'} \in L_{k-1} \text{ and } \varpi - \Delta \leq j - j' \leq \varpi - \delta\}$. Recall that an edge $(u_{j'}, u_j) \in E$ if and only if $u_{j'} \in L_{k-1}$, $u_j \in L_k$, and $\varpi - \Delta \leq j - j' \leq \varpi - \delta$. Hence, the set of all outgoing edges of each vertex $u_{j'}$ and the set of all incoming edges of each u_j can be represented implicitly (such that we can access any edge of G in $O(1)$ time and compute its weight in $O(m)$ time as shown in Section 4.2.2). Note that the Monge property is normally defined on a matrix [1]. We consider the matrix M_k containing the path weight $sw_{k-1}(j') + c(u_{j'}, u_j)$ for every edge $(u_{j'}, u_j)$ between the vertices on two consecutive layers L_k and L_{k+1} of G, with $1 < k < d$. Lemma 18 actually shows that M_k is a staircase matrix with the concave Monge property [1]. Thus, by using the staircase Monge matrix searching technique, it takes $O(m\varpi\alpha(\varpi))$ time to compute all shortest paths from u_1 to all vertices on the k-th layer when knowing all $sw_{k-1}(j')$'s of Layer L_{k-1}, where $\alpha(\cdot)$ is the inverse Ackermann function. Hence, a shortest u_1-to-t path in G can be obtained in $O(dm\varpi\alpha(\varpi)) = O(mn\alpha(\varpi))$ time.

Lemma 19. *Given an IM A of size $m \times n$, and an maximum allowable field width ϖ, an optimal set of feathering regions for the OFSB problem can be computed in $O(mn\alpha(\varpi))$ time.*

4.3 Balancing the Minimum Beam-On Times in an Optimal Splitting

After obtaining an optimal set of feathering regions $\{F_k \mid k = 1, 2, \ldots, d-1\}$, we can decompose each extended row of every feathering region F_k by Lemma 16, yielding a split of the IM A with d sub-IMs, whose total sum of the complexity is minimized. However, some of the resulting sub-IMs may have an undesirably large minimum beam-on time.

In this section, we thus consider the following **min-max beam-on time (MBoT)** problem: Given an optimal set of feathering regions $\{F_k \mid k = 1, 2, \ldots, d - 1\}$, decompose the feathering regions to achieve a set $\mathcal{S}$ of d sub-IMs $\{S_1, S_2, \ldots, S_d\}$, such that the maximum $M_{bot}(\mathcal{S})$ of all the minimum beam-on times $T_{bot}(\cdot)$ of these sub-IMs in $\mathcal{S}$ (i.e., $M_{bot}(\mathcal{S}) = \max_{S_k \in \mathcal{S}} T_{bot}(S_k)$) is minimized, while imposing the lower bound on total complexity of the split. We formulate the MBoT problem as computing a polygonal path in a polygon, such that the maximum slope of the segments on the path is minimized, which can be solved in linear time.

4.3.1 The Min-Max Slope Path Model

Assume that each feathering region $F_k = A[j_{k+1} \mathinner{..} j_k + \varpi - 1]$ is decomposed into $F_k^{(0)}$ and $F_k^{(1)}$, and denote by $\alpha_k(i)$ and $\beta_k(i)$ the weights of each row i of $F_k^{(0)}$ and $F_k^{(1)}$, respectively, as defined in Section 4.2.2.1. Since we impose the lower bound on the total complexity of the split while performing the decomposition of F_k, by Lemma 16, $\alpha_k(i) + \beta_k(i)$ is a fixed constant, denoted by $\rho_k(i)$, and $0 \leq \alpha_k(i), \beta_k(i) \leq \rho_k(i)$. Then, for each $k = 1, 2, \ldots, d$, as shown in Section 4.2.1, the k-th sub-IM in the split $\mathcal{S}$ is $S_k = F_{k-1}^{(1)} \,||\, A[j_{k-1} + \varpi \mathinner{..} j_{k+1} - 1] \,||\, F_k^{(0)}$. For a given set of $(d - 1)$ feathering regions, $A[j_{k-1} + \varpi \mathinner{..} j_{k+1} - 1]$ in each S_k is fixed, and thus we let $c_k(i) = \sum_{l=j_{k-1}+\varpi}^{j_{k+1}-1} \max\{0, a_{i,l} - a_{i,l-1}\}$, which is a constant. Note that the minimum beam-on time $T_{bot}(B)$ of an IM $B = (b_{i,j})_{m' \times n'}$ equals to $\max_{i=1}^{m'}\{b_{i,1} + \sum_{j=2}^{n'} \max\{0, b_{i,j} -$

$b_{i,j-1}\}\}$ [30]. Hence, $T_{bot}(S_k) = \max_{i=1}^{m}(\beta_{k-1}(i) + c_k(i) + \alpha_k(i))$, where $\alpha_k(i)$ and $\beta_k(i)$ are varying depending on the decompositions of F_k's. Then, the maximum $M_{bot}(\mathcal{S})$ of all the minimum beam-on times of the sub-IMs in $\mathcal{S}$ is $\max_{S_k \in \mathcal{S}} \max_{i=1}^{m}(\beta_{k-1}(i) + c_k(i) + \alpha_k(i))$, that is, $M_{bot}(\mathcal{S}) = \max_{i=1}^{m} \max_{k=1}^{d}(\beta_{k-1}(i) + c_k(i) + \alpha_k(i))$, where $\beta_0(i) = \alpha_d(i) = 0$.

Thus, to minimize $M_{bot}(\mathcal{S})$, we need to solve the following problem: Given a vector $\rho = \{\rho_0, \rho_1, \ldots, \rho_{d-1}, \rho_d\}$ with $\rho_0 = \rho_d = 0$ and $\rho_k \geq 0$ ($k = 1, 2, \ldots, d-1$), and a vector $c = (c_1, c_2, \ldots, c_d)$ with $c_k \geq 0$ for every $k = 1, 2, \ldots, d$, decompose ρ into two non-negative integral vectors $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_d)$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$, such that:

(1) $\alpha_k + \beta_k = \rho_k$ for each $k = 0, 1, \ldots, d$;

(2) $0 \leq \alpha_k, \beta_k \leq \rho_k$; and

(3) $\max_{k=1}^{d}(\beta_{k-1} + c_k + \alpha_k)$ is minimized.

Without loss of generality, we assume that $\rho_k > 0$ for $k = 1, 2, \ldots, d-1$. Interestingly, we are able to model this problem as a **min-max slope path (MSP)** problem in a polygon, as follows.

Define a monotone polygon $\mathcal{P}$ in the 2-D **x-y** plane, whose boundary consists of two **x**-monotone polygonal chains, the upper chain $\mathcal{C}_u$ and the lower one $\mathcal{C}_l$. Both $\mathcal{C}_u$ and $\mathcal{C}_l$, each consisting of $d+1$ chain vertices, start at the point $s(0,0)$ and end at the point $t(d, \sum_{i=1}^{d} c_i + \sum_{i=0}^{d} \rho_i)$. The k-th vertex on the lower (resp., upper) chain $\mathcal{C}_l$ (resp., $\mathcal{C}_u$) is at $\underline{B}_k(k, \sum_{i=1}^{k} c_i + \sum_{i=0}^{k-1} \rho_i)$ (resp., $\overline{B}_k(k, \sum_{i=1}^{k} c_i + \sum_{i=0}^{k} \rho_i)$) for $k = 1, 2, \ldots, d-1$, i.e., $\underline{B}_k$ and $\overline{B}_k$ are on the vertical line $x = k$ and $\overline{B}_k$ is ρ_k

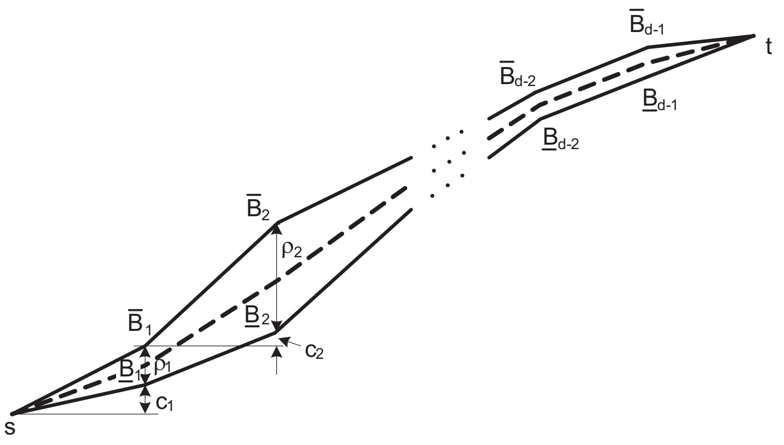


Figure 4.4: Illustrating the construction of polygon $\mathcal{P}$.

"higher" than $\underline{B}_k$ (see Figure 4.4). Let $x(P)$ (resp, $y(P)$) denote the x-coordinate (resp., y-coordinate) of a point P. The **min-max slope path (MSP)** problem seeks a polygonal path $L = P_0P_1 \dots P_{q-1}P_q$ in $\mathcal{P}$ such that: (1) $P_0 = s$ and $P_q = t$ and (2) the maximum slope of the line segments on L, denoted by $MS(L)$ (i.e., $MS(L) = \max_{i=1}^{q} \frac{y(P_i)-y(P_{i-1})}{x(P_i)-x(P_{i-1})}$), is minimized. Such a path L is called a min-maxslope path from s to t.

For any feasible decomposition of ρ, $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_{d-1}, \alpha_d)$ and $\beta = (\beta_0, \beta_1, \dots, \beta_{d-1}, \beta_d)$, we can define a polygonal path $L = P_0P_1, \dots, P_{d-1}P_d$ with $P_0 = s$, $P_d = t$, and $P_k = (k, y(\underline{B}_k) + \alpha_k)$ for $k = 1, 2, \dots, d-1$ (see Figure 4.4). It is not difficult to see that L is in the polygon $\mathcal{P}$, and $MS(L) = \max_{k=1}^{d}(\beta_{k-1} + c_k + \alpha_k)$. We next show how to compute a min-max slope s-to-t path which defines an optimal decomposition of ρ.

4.3.2 Computing a Min-Max Slope s-to-t Path

We show a min-max slope s-to-t path in $\mathcal{P}$ may only use the polygonal vertices of $\mathcal{P}$, which leads to a linear time algorithm for computing a min-max slope s-to-t path.

In the polygon $\mathcal{P}$, we call the line segments $\underline{B}_k\overline{B}_k$ (resp., $\overline{B}_k\underline{B}_{k+1}$) as *V-diagonals* (resp., *H-diagonals*) of $\mathcal{P}$ (see Figure 4.4). Note that we do not consider the line segments $\underline{B}_k\overline{B}_{k+1}$ in $\mathcal{P}$. A diagonal d_i of $\mathcal{P}$ is denoted by $\overline{v}_i\underline{v}_i$ with $\overline{v}_i$ on the upper chain $\mathcal{C}_u$ of $\mathcal{P}$ and $\underline{v}_i$ on the lower chain $\mathcal{C}_l$ of $\mathcal{P}$. Denote by $L(v, v')$ the min-max slope path from a polygonal vertex v to v' in the polygon $\mathcal{P}$. Thus, $L(s, \overline{v}_i)$ and $L(s, \underline{v}_i)$ be the min-max slope paths from s to $\overline{v}_i$ and $\underline{v}_i$, respectively. There exists a unique common vertex u of both $L(s, \overline{v}_i)$ and $L(s, \underline{v}_i)$, which is the farthest one from s on both paths. The region bounded by the diagonal d_i, the sub-path $L(u, \overline{v}_i)$ of $L(s, \overline{v}_i)$ from u to $\overline{v}_i$, and the one $L(u, \underline{v}_i)$ of $L(s, \underline{v}_i)$ from u to $\underline{v}_i$, is called a *funnel*, denoted by R_i, with u being the *apex* of the funnel (see Figure 4.5).

Lemma 20. *There exists a min-max slope s-to-t path $L^* = P_0P_1 \dots P_{d-1}P_d$ such that $P_0 = s$, $P_d = t$, and P_k is a polygonal vertex of $\mathcal{P}$ for every $k = 1, 2, \dots, d-1$.*

Proof. We first show that there exists a min-max slope s-to-t path such that the x-coordinate of each vertex on the path is an integer. The idea is that, for a given min-max slope s-to-t path, if the x-coordinates of the vertices on the path are not an integer, we convert it to another min-max slope s-to-t path such that the x-coordinate of each vertex on the new path is an integer.

For a given a min-max slope s-to-t path $L^* = P_0P_1 \dots P_{d-1}P_d$, assume that

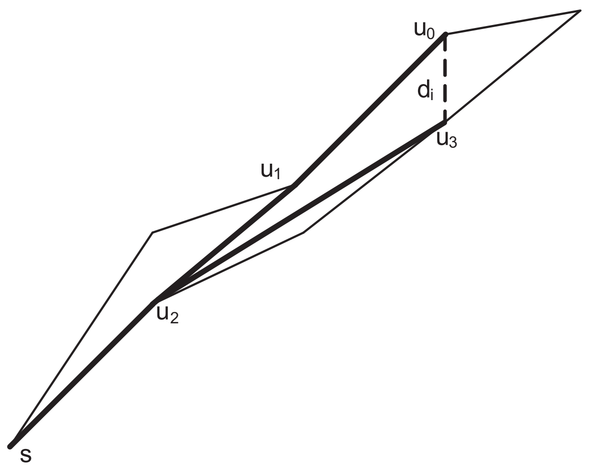


Figure 4.5: An example of R_i. For the diagonal d_i with u_0 and u_3 being the two endpoints, $\overline{su_2u_1u_0}$ and $\overline{su_2u_3}$ are the min-max slope paths from s to u_0 and u_3, respectively. The region bounded by $\overline{u_2u_1u_0}$, u_2u_3 and d_i is R_i and u_2 is the apex.

$P_{i-1}P_i$ is the first segment on L^* such that $x(P_{i-1})$ is an integer while $x(P_i)$ is not. Here, $i < d$; otherwise, L^* is the path that we want. Let P_jP_{j+1} be the segment of L^* that intersects with the line $x = \lceil x(P_i) \rceil$. Denote by P' the intersection of the segment $P_{i-1}P_i$ with Line $x = \lfloor x(P_i) \rfloor$, and let P'' be the intersection of the segment P_jP_{j+1} with Line $x = \lceil x(P_i) \rceil$. Three cases need to be considered.

Case (1) Both P_i and P_j are in the same half plane defined by the line segment $P'P''$. In this case, the slope of either $P_{i-1}P_i$ or P_jP_{j+1} is no less than that of $P'P''$. See Figure 4.6(a) for example.

Case (2) P_i is in the upper half plane and P_j is in the lower half plane defined by $P'P''$. Then, the slopes of both $P_{i-1}P_i$ and P_jP_{j+1} are no less than that of $P'P''$. See Figure 4.6(b) for example.

Case (3) P_i is in the lower half plane and P_j is in the upper half plane defined by $P'P''$. In this case, $P_i \neq P_j$. Let $L(P_i, P_j)$ denote the sub-path of L^* from P_i to P_j. Due to the monotonicity of $L(P_i, P_j)$, the maximum slope of the line segments on $L(P_i, P_j)$ ($MS(L(P_i, P_j))$) is no less than the slope of P_iP_j. The slope of P_iP_j is no less than that of $P'P''$ since $x(P_i) \leq x(P_j)$ and $y(P_i) \leq y(P_j)$. Hence, $MS(L(P_i, P_j))$ is no less than the slope of $P'P''$. See Figure 4.6(c) for example.

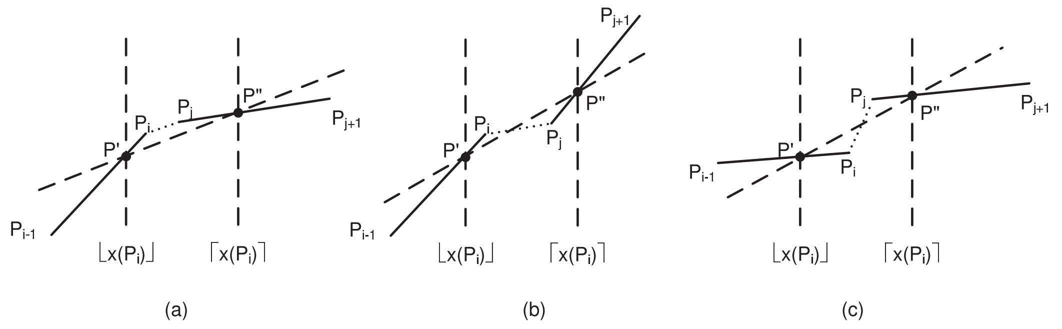


Figure 4.6: Illustrating the proof of Lemma 20. Illustrating the three possible cases for the proof that there exists a min-max slope s-to-t path such that the x-coordinate of each vertex on the path is an integer.

We thus consider the path $L' = P_0 \ldots P_{i-1}P'P''P_{j+1} \ldots P_d$. Obviously, L' is in $\mathcal{P}$ and $MS(L') \leq MS(L^*)$. Hence, L' is a min-max slope s-to-t path. Repeat this operation, we can generate a min-max slope s-to-t path in $\mathcal{P}$ such that the x-coordinate of each vertex on the path is an integer.

We next show that each vertex on L^* can be a polygonal vertex of $\mathcal{P}$. Otherwise, let $P_i = (x_i, y_i)$ be the first vertex of L^* that is not a polygonal vertex of $\mathcal{P}$. We

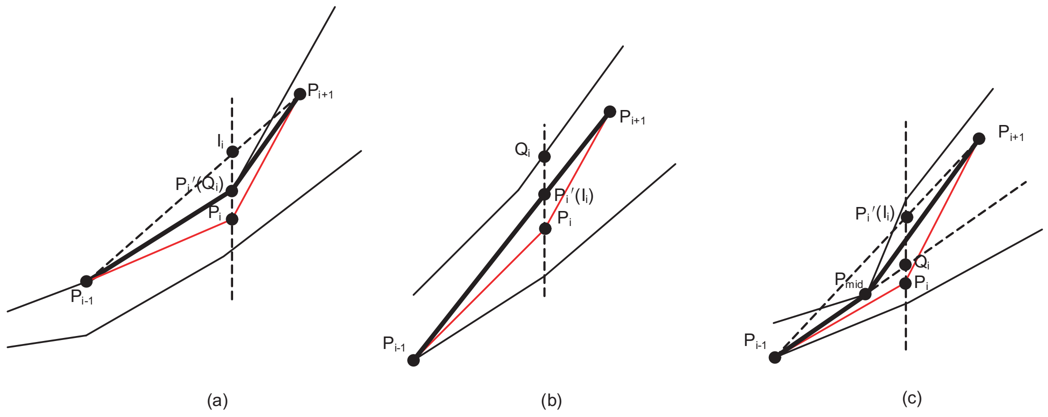


Figure 4.7: Illustrating the proof of Lemma 20. Thin solid lines show the polygon. Red solid lines show the original min-max slope path and the thick solid lines show the revised min-max slope path.

have two possible cases: (i) The slope of P_iP_{i+1} is no less than that of $P_{i-1}P_i$; and (ii) The slope of P_iP_{i+1} is smaller than that of $P_{i-1}P_i$.

We first consider the case that the slope of P_iP_{i+1} is no less than that of $P_{i-1}P_i$. In this case, there must exist a point $Q_i = (x_i, y_i + \Delta y^*)$ with $\Delta y^* > 0$, such that both $P_{i-1}Q_i$ and Q_iP_{i+1} are within the polygon $\mathcal{P}$, but for any point $Q_\epsilon(x_i, y_\epsilon)$ above Q_i on the line $x = x_i$ (i.e., $y_\epsilon > y_i + \Delta y^*$), either $P_{i-1}Q_\epsilon$ or $Q_\epsilon P_{i+1}$ is not within $\mathcal{P}$. That is, Q_i is the "highest" point on the line $x = x_i$ such that both $P_{i-1}Q_i$ and Q_iP_{i+1} are within $\mathcal{P}$. We then consider a point $P_i'(x_i, y_i + \Delta y)$ with $\Delta y = \min\{\sum_{k=1}^{x_i} c_k + \sum_{k=1}^{x_i} \rho_k, \frac{y_{i+1}-y_{i-1}}{x_{i+1}-x_{i-1}}(x_i - x_{i-1}) + y_{i-1}\} - y_i$. Note that the point $\bar{B}_i(x_i, \sum_{k=1}^{x_i} c_k + \sum_{k=1}^{x_i} \rho_k)$ is the polygonal vertex of $\mathcal{P}$ on the line $x = x_i$, and the point $I_i(x_i, \frac{y_{i+1}-y_{i-1}}{x_{i+1}-x_{i-1}}(x_i - x_{i-1}) + y_{i-1})$ is the intersection point of the line segment $P_{i-1}P_{i+1}$ with the line $x = x_i$. The point P_i' is no "higher" than the point I_i on the line $x = x_i$.

If $\Delta y \leq \Delta y^*$ (i.e., Q_i is no lower than P'_i), then both $P_{i-1}P'_i$ and P'_iP_{i+1} are still within $\mathcal{P}$. Since $y(P'_i) > y(P_i)$ and $y(I_i) > y(P_i)$, the slopes of P'_iP_{i+1} and I_iP_{i+1} are both smaller than that of P_iP_{i+1}. Note that the point P'_i is no "higher" than the point I_i on the line $x = x_i$ (i.e, $y(P'_i) \leq y(I_i)$). Thus, the slope of $P_{i-1}P'_i$ is no larger than that of $P_{i-1}I_i$. Since I_i is on the line segment $P_{i-1}P_{i+1}$, the slope of $P_{i-1}P'_i$ is smaller than that of P_iP_{i+1}. Hence, the slopes of both $P_{i-1}P'_i$ and P'_iP_{i+1} are smaller than that of P_iP_{i+1}. We thus can replace $P_{i-1}P_i$ by $P_{i-1}P'_i$ and P_iP_{i+1} by P'_iP_{i+1} to get a new s-to-t path L_{new} from L^*. Obviously, the maximum slope of L_{new} is no larger than that of L^*. Therefore, L_{new} is a min-max slope s-to-t path in $\mathcal{P}$. In addition, P'_i is either a polygonal vertex (see Figure 4.7(a)) of $\mathcal{P}$ if $P'_i = \bar{B}_i$, or P'_i is a point on the straight line segment $P_{i-1}P_{i+1}$ if $P'_i = I_i$, that is, P'_i is no longer a vertex on the path L_{new} (see Figure 4.7(b)). We thus obtain a new min-max slope s-to-t path with one less non-polygonal vertex.

If $\Delta y > \Delta y^*$, then either $P_{i-1}P'_i$ or P'_iP_{i+1} is not within $\mathcal{P}$. Without loss of generality, we assume that $P_{i-1}P'_i$ is not within $\mathcal{P}$. Let P_{mid} be the first polygonal vertex of $\mathcal{P}$ such that $x(P_{mid}) > x(P_{i-1})$ and the intersection point of $P_{i-1}P'_i$ and line $x = x(P_{mid})$ is outside of $\mathcal{P}$. Replacing $P_{i-1}P_i$ and P_iP_{i+1} by $P_{i-1}P_{mid}$ concatenated with $P_{mid}P_{i+1}$ yields a new s-to-t path L_{new}. We now exams the slopes of $P_{i-1}P_{mid}$ and $P_{mid}P_{i+1}$. Since $y(I_i) > y(P_i)$ and I_i is on $P_{i-1}P_{i+1}$, the slope of $P_{i-1}P_{i+1}$ is less than that of P_iP_{i+1}. Note that the slope of $P_{i-1}P_{i+1}$ is no less than that of $P_{i-1}P_{mid}$. Thus, the slope of $P_{i-1}P_{mid}$ is less than that of P_iP_{i+1}. It is not difficult to see that the slope of $P_{mid}P_{i+1}$ is less than that of P_iP_{i+1}. Hence, the maximum slope of L_{new}

does not increase. If $P_{mid}P_{i+1}$ is within the polygon $\mathcal{P}$, then L_{new} is a min-max slope s-to-t path with one less non-polygonal vertex on the path. Otherwise, we can repeat the above operations to replace $P_{mid}P_{i+1}$ by a sub-path $L(P_{mid}, P_{i+1})$ whose maximum segment slope is less than the slope of P_iP_{i+1}. We thus again can obtain a new min-max slope s-to-t path with one less non-polygonal vertex.

For the second case that the slope of P_iP_{i+1} is smaller than that of $P_{i-1}P_i$, we can perform the segment replacement operations in a similar way as for the first case to eliminate the non-polygonal vertices on an optimal min-max slope s-to-t path. Thus, the lemma is proved. □

From Lemma 20, we develop the following algorithm to compute a min-max slope s-to-t path L^*, such that all vertices of L^* are the polygonal vertices of $\mathcal{P}$.

Step 1: Initially, the min-max slope paths from s to $\overline{v}_1 = \overline{B}_1$ and $\underline{v}_1 = \underline{B}_1$ are obvious. Set $k = 1$ and $d_k = \overline{v}_k\underline{v}_k$. The region bounded by d_k, $s\overline{v}_1$ and $s\underline{v}_1$ forms the funnel R_1.

Step 2: Let u_a be the apex of the funnel R_k, at which the two subchains $\overline{u_a u_{a+1} \dots u_b}$ and $\overline{u_a u_{a-1} \dots u_0}$ diverge, where $\overline{v}_k = u_0$ and $\underline{v}_k = u_b$ (Figure 4.8). Consider the next diagonal $d_{k+1} = \overline{v}_{k+1}\underline{v}_{k+1}$.

To determine $\overline{v}_{k+1}$ and $\underline{v}_{k+1}$, and to compute the min-max slope paths from s to $\overline{v}_{k+1}$ and $\underline{v}_{k+1}$, we have the following possible cases:

(1) d_k is an H-diagonal. Then, d_{k+1} is a V-diagonal and $\underline{v}_{k+1} = \underline{v}_k$. Based on the definition of V-diagonals, $\overline{v}_{k+1}$ is determined. Start from u_0 to scan the sequence $u_0, u_1, \dots, u_b$ and let j be the smallest index for which $\overline{v}_{k+1}u_j$ is a supporting

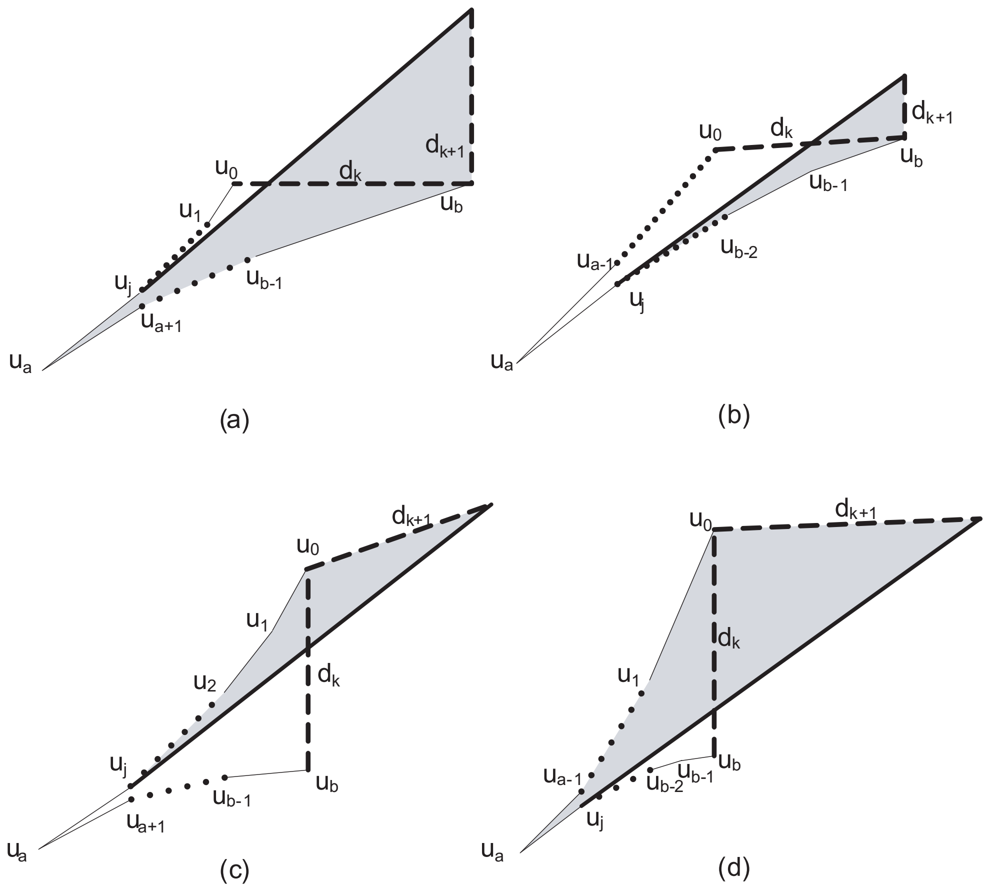


Figure 4.8: Illustrating the algorithm for computing the min-max slope path in $\mathcal{P}$. (a), (b), (c), and (d) are four possible cases in the algorithm.

segment of $\overline{u_a u_{a-1} \dots u_0}$ or $\overline{u_a u_{a+1} \dots u_b}$. A line segment is a *supporting segment* of an open convex (concave) curve if it has at least one point on the curve and it is a tangent line of the curve.

(i) If $j \leq a$ (Figure 4.8(a)), then remove all edges $u_i u_{i+1}$ for $0 \leq i \leq j-1$ and add edge $u_j \overline{v}_{k+1}$.

(ii) If $j > a$ (Figure 4.8(b)), then remove all edges $u_i u_{i+1}$ for $0 \leq i \leq j-1$ and add $\overline{u_j\overline{v}_{k+1}}$. u_j becomes the apex of the funnel R_{k+1}.

(2) d_k is a V-diagonal. Then, d_{k+1} is an H-diagonal and $\overline{v}_{k+1} = \overline{v}_k$. Based on the definition of V-diagonals, $\overline{v}_{k+1}$ is determined. Start from u_0 to scan the sequence $u_0, u_1, \ldots, u_b$ and let j be the smallest index for which $\underline{v}_{k+1}u_j$ is a supporting segment of $\overline{u_a u_{a-1} \ldots u_0}$ or $\overline{u_a u_{a+1} \ldots u_b}$.

(i) If $j \leq a$ (Figure 4.8(c)), the remove all edges u_iu_{i+1} for $j \leq i \leq b-1$ and add edge $u_j\overline{v}_{k+1}$. u_j becomes the apex of the funnel R_{k+1}.

(ii) If $j > a$ (Figure 4.8(d)), then remove all edges u_iu_{i+1} for $j \leq i \leq b-1$ and add edge $u_j\overline{v}_{k+1}$.

Step 3: Let $k = k+1$ and repeat Step 2 until t is reached. Then, we obtain a min-max slope s-t path L^*.

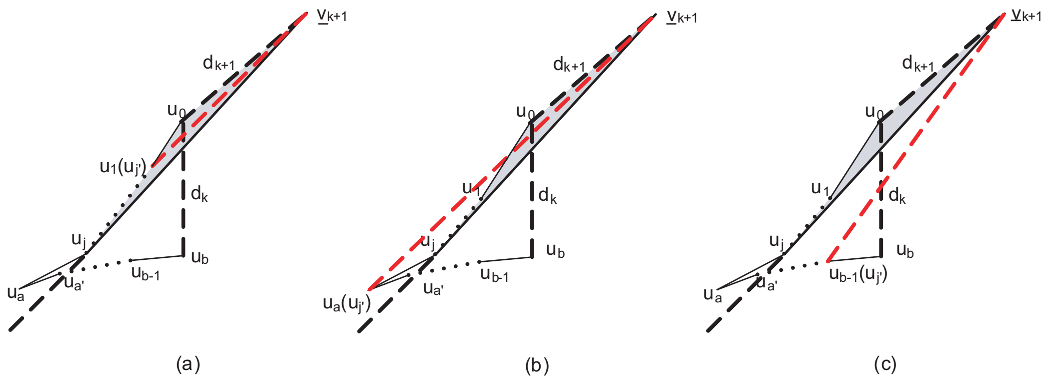


Figure 4.9: Illustrating the correctness of the MSP algorithm.

To prove the correctness of the algorithm, we first prove the path generated in Step 2 of the algorithm is a min-max slope path from u_a to $\overline{v}_{k+1}$. We prove it by contradiction. Let us take case (2)(i) as an example. Suppose the last segment of the min-max slope path from u_a to $\overline{v}_{k+1}$ is not $\overline{u_{j'}\overline{v}_{k+1}}$. We have the following three cases (see Figure 4.9):

(1) $j' < j$ (Figure 4.9(a)). We know the min-max slope path from u_a to $u_{j'}$ is $\overline{u_a u_{a-1} \ldots u_{j'}}$, so the min-max slope path from u_a to $\underline{v}_{k+1}$ is $\overline{u_a u_{a-1} \ldots u_{j'} \underline{v}_{k+1}}$. Note that $u_j \underline{v}_{k+1}$ is a supporting segment of $\overline{u_a u_{a-1} \ldots u_0}$, the slope of any segment on $\overline{u_j u_{j-1} \ldots u_{j'}}$ is larger than the slope of $u_j \underline{v}_{k+1}$. Thus, the min-max slope of the path $\overline{u_a u_{a-1} \ldots u_j \ldots u_{j'} \underline{v}_{k+1}}$ is larger than that of the path $\overline{u_a u_{a-1} \ldots u_j \underline{v}_{k+1}}$.

(2) $j < j' \leq a'$ (Figure 4.9(b)), where a' is the largest index such that $u_{a'}$ is on the upper half-plane defined by the line segment $\underline{v}_{k+1} u_j$. Since $\underline{v}_{k+1} u_j$ is a supporting segment of $\overline{u_a u_{a-1} \ldots u_0}$, the segment $\underline{v}_{k+1} u_{j'}$ is not entirely in the polygon. Thus, the min-max slope u_a-to-$\underline{v}_{k+1}$ path does not pass any vertex between $u_{a'}$ and u_{j+1}.

(3) $j > a'$ (Figure 4.9(c)). Again, note that $\underline{v}_{k+1} u_j$ is a supporting segment of $\overline{u_a u_{a-1} \ldots u_0}$. The min-max slope of the path $\overline{u_a u_{a-1} \ldots u_j \underline{v}_{k+1}}$ is the slope of $u_j \underline{v}_{k+1}$. However, the slope of $u_{j'} \underline{v}_{k+1}$ is larger than that of $u_j \underline{v}_{k+1}$. Thus, the min-max slope of the path $\overline{u_a u_{a+1} \ldots u_{j'} \underline{v}_{k+1}}$ is larger than that of the path $\overline{u_a u_{a-1} \ldots u_j \underline{v}_{k+1}}$.

Consider all the above three cases, we can conclude that the path computed in

Step 2 of the algorithm is a min-max slope path from u_a to $\underline{v}_{k+1}$. A similar argument is applicable for the other three cases.

Next, we show that for any point w on diagonal d_{k+1}, a min-max slope s-to-w path passes through the apex of the funnel R_{k+1}. We prove it by induction on k. Obviously, the claim is true for R_1. Assume that it is true for $1, 2, \ldots, k$, we want to prove it is also true for $k + 1$.

Suppose the min-max slope s-to-w path intersects with d_k at w'. The assumption immediately leads to the conclusion that the min-max slope s-to-w' path passes through the apex u_a of R_k, so the min-max s-to-w path is the min-max slope s-to-u_a path concatenated with the min-max slope u_a-to-w path. The min-max slope u_a-to-w path can be constructed using Step 2 of our algorithm (although we only show the computation of the min-max slope path for the end points of the diagonal). For case (1)(i) and (2)(ii), the apex of the funnel R_{k+1} is the same as that of R_k. Thus, the claim for $k + 1$ is held. We now consider Cases (1)(ii) and (2)(i). Since a similar argument is applicable to both cases, we only detail the proof for Case (2)(i) in the following.

It is not difficulty to see that the path $\overline{u_a u_{a-1} \ldots u_0}$ is inward convex, that is, the slopes of the line segments, $u_a u_{a-1}, u_{a-1} u_{a-2}, \ldots, u_1 u_0$ on the path, are monotonically increasing. Also note that to compute the supporting segment from a vertex v of the diagonal d_{k+1} to $\overline{u_a u_{a-1} \ldots u_0}$ is essentially finding the largest index i such that the slope of $u_i u_{i+1}$ is larger than that of $u_i v$. The fact that $u_j \underline{v}_{k+1}$ is the supporting line from $\underline{v}_{k+1}$ to the path $\overline{u_a u_{a-1} \ldots u_0}$ indicates that the slope of $u_j \underline{v}_{k+1}$ is less than that of $u_j u_{j+1}$. Suppose that $u_{j''}w$ is the supporting line from w to the path $\overline{u_a u_{a-1} \ldots u_0}$. We have the slope of $u_{j''}w$ is less than that of $u_{j''}u_{j''+1}$. Since $0 < j'' < a$, the slope of $u_{j''}\underline{v}_{k+1}$ is less than that of $u_{j''}w$. Thus, the slope of $u_{j''}\underline{v}_{k+1}$ is less than that of $u_{j''}u_{j''+1}$, which indicates that $j > j''$ (Figure 4.10). Hence, the computed min-max slope u_a-to-w path passes through the apex u_j of the funnel R_{k+1}. We can thus conclude that a min-max slope s-to-w path passes through the apex of the funnel R_{k+1}.

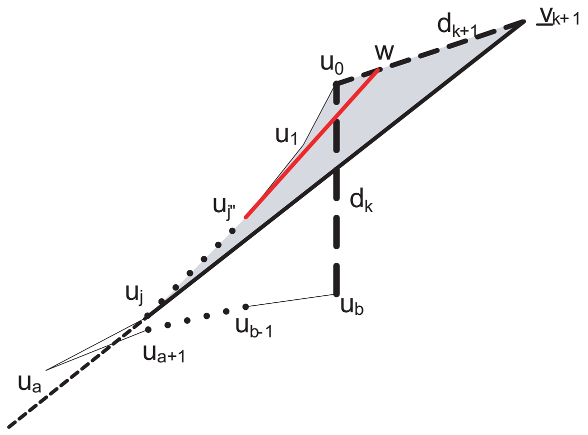


Figure 4.10: If both $u_j\underline{v}_{k+1}$ and $u_{j''}w$ are supporting segments of $\overline{u_a u_{a-1} \ldots u_0}$, then $j \geq j''$.

Lemma 21. *A min-max slope s-to-t path in $\mathcal{P}$ can be computed in $O(N)$ time, where N is the number of vertices on $\mathcal{P}$.*

4.3.3 Solution Integerization

The min-max slope s-to-t path L^* in $\mathcal{P}$ computed in Section 4.3.2 may not intersect with each line $x = k$ ($k = 1, 2, \ldots, d-1$) at a point whose y-coordinate is an integer. This prevents us from defining a valid decomposition of ρ for solving the MBoT problem.

However, if the maximum slope of a min-max slope s-to-t path L^* in $\mathcal{P}$ is S_{max}, an s-to-t path L'^* in $\mathcal{P}$, which intersects with each line $x = k$ ($k = 1, 2, \ldots, d-1$) at an integer point, can be found using the following integerization operation and the maximum slope of the line segments on L'^* is $\lceil S_{max} \rceil$. Obviously, this is an optimal integer solution to the MSP problem.

Suppose $P_k = (x, y)$ and $P_{k+1} = (x + \Delta x, y + \Delta y)$ are two adjacent polygonal vertices on the path L^*. We define the following integerization operation on this line segment: For each $i = 1, 2, \ldots, \Delta x - 1$, we insert a new point $(x+i, \lceil y + i\Delta y/\Delta x \rceil)$. It is easy to show that none of the slopes of all these segments is larger than $\lceil \Delta y/\Delta x \rceil$.

Performing this integerization operation on each line segment of L^*, we then obtain an s-to-t path L'^* whose vertices are integer points and maximum slope is $\lceil S_{max} \rceil$. We call L'^* an integer min-max slope path from s to t in $\mathcal{P}$. Consider each point $P(k, y(\underline{B}_k) + \alpha_k) \in L'^*$ for $k = 1, 2, \ldots, d-1$. Then, ρ_k can be decomposed into α_k and β_k with $\beta_k = \rho_k - \alpha_k$. Thus, ρ can be decomposed into integer vectors α and β such that $\max_{k+1}^{d}(\beta_{k-1} + c_k + \alpha_k)$ is minimized.

4.3.4 Constrained Feathering Region Decomposition

Up to this point, to balance the minimum beam-on times in an optimal split, we decompose the total complexity $\rho_k(i)$ of each row i of every feathering region F_k into $\alpha_k(i)$ and $\beta_k(i)$ (i.e., $\alpha_k(i) + \beta_k(i) = \rho_k(i)$). Next, we need to decompose each extended row i of the feathering region F_k into two vectors, $\mathbf{x}$ and $\mathbf{y}$, such that the weight $W_{ovd}(\mathbf{x})$ of $\mathbf{x}$ equals to $\alpha_k(i)$ and $W_{ovd}(\mathbf{x}) + W_{ovd}(\mathbf{y}) = \rho_k(i)$. In this section, we present a linear algorithm for such a decomposition by characterizing the optimal solution space of the OVD problem in Section 4.2.2.2.

Denote by $\rho(\mathbf{b})$ the total complexity of an optimal decomposition $\mathbf{x}$ and $\mathbf{y}$ of $\mathbf{b}$, i.e., $\rho(\mathbf{b}) = W_{ovd}(\mathbf{x}) + W_{ovd}(\mathbf{y})$. Clearly, by Lemma 15, we are able to compute an optimal decomposition of $\mathbf{b}$ such that $W_{ovd}(\mathbf{x}) = 0$ (i.e., $\mathbf{x} = \mathbf{x}(\underline{p})$) or $W_{ovd}(\mathbf{x}) = \rho(\mathbf{b}) - b_N$ (i.e., $\mathbf{x} = \mathbf{x}(\overline{p})$). Further, note that $x_1 = b_1$ and $x_N = 0$, and $y_1 = 0$ and $y_N = b_N$. Thus, $W_{ovd}(\mathbf{x}) \geq b_1 \geq 0$, and $W_{ovd}(\mathbf{y}) = \sum_{j=1}^{N-1} \max\{0, y_{j+1} - y_j\} \geq b_N$. Hence, for any optimal decomposition $\mathbf{x}$ and $\mathbf{y}$ of $\mathbf{b}$, we have $0 \leq W_{ovd}(\mathbf{x}) \leq \rho(\mathbf{b}) - b_N$. We next show that, in fact, for any integer τ with $0 \leq \tau \leq \rho(\mathbf{b}) - b_N$, we can compute an optimal decomposition $\mathbf{x}$ and $\mathbf{y}$ of $\mathbf{b}$ such that $W_{ovd}(\mathbf{x}) = \tau$.

Let the arched shortest path in G from s ($v_1(b_1)$) to t ($v_N(0)$) be $\overline{p} = v_1(h_1^-) \rightarrow v_2(h_2^-) \rightarrow \ldots \rightarrow v_{N-1}(h_{N-1}^-) \rightarrow v_N(h_N^-)$, where $h_1^- = b_1$ and $h_N^- = 0$. For each $j = 1, 2, \ldots, N$, we then define an s-t path p_j, which is the concatenation of an arched path $s \overset{p^-}{\rightsquigarrow} v_j(h_j^-)$ and an incurved path $v_j(h_j^-) \overset{p^+}{\rightsquigarrow} t$, denoted by $\left(s \overset{p^-}{\rightsquigarrow} v_j(h_j^-)\right) \bigvee \left(v_j(h_j^-) \overset{p^+}{\rightsquigarrow} t\right)$ (Figure 4.11(a)). It is not difficult to see that each path p_j is a shortest s-t path in G. Each path p_j induces a feasible decomposition $\mathbf{x}$ and $\mathbf{y}$ of vector $\mathbf{b}$. Considering the weight of the vector $\mathbf{x}$ induced by each path p_j, denoted by $w_x(p_j)$ (in fact, $w_x(p_j) = W_{ovd}(\mathbf{x})$), we have the following lemma.

Lemma 22. $0 = w_x(p_1) \leq w_x(p_2) \leq \ldots \leq w_x(p_{N-1}) \leq w_x(p_N) = \rho(\mathbf{b}) - b_N$, *and all these* $w_x(p_j)$*'s* ($j = 1, 2, \ldots, N$) *can be computed in* $O(N)$ *time.*

Proof. We first note that p_1 actually is the incurved shortest path $\underline{p}$ from s to t in

G and p_N is the arched shortest path $\overline{p}$. By Lemma 15, it immediately follows that $w_x(p_1) = 0$ and $w_x(p_N) = \rho(\mathbf{b}) - b_N$.

For any $j = 1, \ldots, N$, $p_j = \left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right) \bigvee \left(v_j(h_j^-) \overset{p^+}{\leadsto} t\right)$. Thus, $w_x(p_j) = w_x\left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right) + w_x\left(v_j(h_j^-) \overset{p^+}{\leadsto} t\right)$. It follows from Lemma 15 that $w_x\left(v_j(h_j^-) \overset{p^+}{\leadsto} t\right) = 0$. Hence, $w_x(p_j) = w_x\left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right)$. Considering two paths p_j and p_{j+1} ($j < N$),

$$\begin{aligned} w_x(p_{j+1}) - w_x(p_j) &= w_x\left(s \overset{p^-}{\leadsto} v_{j+1}(h_{j+1}^-)\right) - w_x\left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right) \\ &= w_x\left(s \overset{p^-}{\leadsto} v_j(h_j^-) \to v_{j+1}(h_{j+1}^-)\right) - w_x\left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right) \\ &= \max\{0, h_{j+1}^- - h_j^-\} \geq 0. \end{aligned}$$

Thus, we have $w_x(p_1) \leq w_x(p_2) \leq \ldots \leq w_x(p_{N-1}) \leq w_x(p_N)$.

Note that for each $j = 1, 2, \ldots, N$, $s \overset{p^-}{\leadsto} v_j(h_j^-)$ is the portion of $\overline{p}$ from s to $v_j(h_j^-)$. While $\overline{p}$ can be obtained in $O(N)$ time from Lemma 15. Hence, all $w_x(p_j)$'s can be computed in $O(N)$ time from $\overline{p}$. This proves the lemma. □

Lemma 23. *For any integer τ with $w_x(p_j) \leq \tau \leq w_x(p_{j+1})$, a shortest s-t path q in G with $w_x(q) = \tau$ can be computed in $O(N)$ time.*

Proof. Let $p_j = \left(s \overset{p^-}{\leadsto} v_j(h_j^-)\right) \bigvee \left(v_j(h_j^-) \overset{p^+}{\leadsto} t\right) = \left(s \overset{p^-}{\leadsto} v_j(h_j^-) \to v_{j+1}(h_{j+1}^+) \overset{p^+}{\leadsto} t\right)$ and

$p_{j+1} = \left(s \overset{p^-}{\leadsto} v_{j+1}(h_{j+1}^-)\right) \bigvee \left(v_{j+1}(h_{j+1}^-) \overset{p^+}{\leadsto} t\right) = \left(s \overset{p^-}{\leadsto} v_j(h_j^-) \to v_{j+1}(h_{j+1}^-) \overset{p^+}{\leadsto} t\right)$.

From Lemma 22, it is easy to see that $h_{j+1}^- \geq h_{j+1}^+$.

If $\tau = w_x(p_j)$ or $\tau = w_x(p_{j+1})$, obviously, the lemma holds (Figure 4.11(a)). We thus assume that $w_x(p_j) < \tau < w_x(p_{j+1})$ (Figure 4.11(b)). Two cases are needed

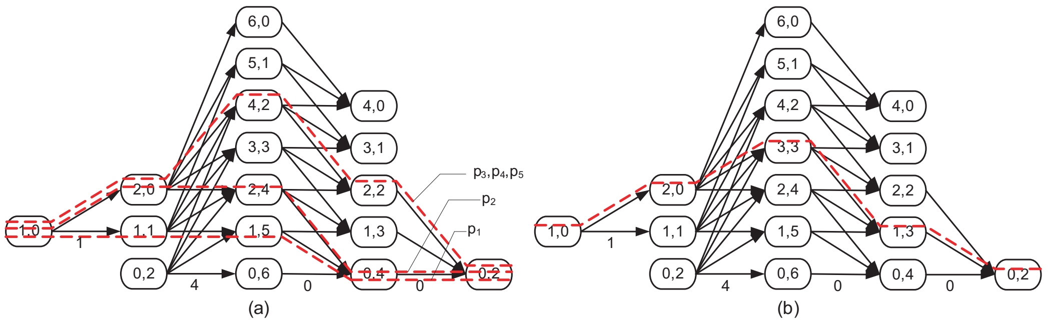


Figure 4.11: Illustrating the solution space of decomposing the vector $(1, 2, 6, 4, 2)$. Only edges with minimum weights between two columns are shown and the minimum edge weight is labeled below each set of the edges. (a) The shortest s-t paths p_j, $j = 1, 2, \ldots, 5$, with $w_x(p_1) = 0$, $w_x(p_2) = 1$ and $w_x(p_3) = w_x(p_4) = w_x(p_5) = 3$. (b) The shortest s-t path with $w_x = 2$.

to be considered.

Case (1): $b_j \leq b_{j+1}$. In this case, $h^+_{j+1} = h^-_j$. Let $\hbar = h^-_j + (\tau - w_x(p_j))$. Clearly, $\hbar > h^-_j$ and $h^-_{j+1} > \hbar > h^+_{j+1}$. We then define an s-t path q in G as $s \overset{p^-}{\rightsquigarrow} v_j(h^-_j) \rightarrow v_{j+1}(\hbar) \overset{p^+}{\rightsquigarrow} t$. Note that $w_x\left(v_j(h^-_j), v_{j+1}(\hbar)\right) = \tau - w_x(p_j)$, $w_x\left(s \overset{p^-}{\rightsquigarrow} v_j(h^-_j)\right) = w_x(p_j)$, and $w_x\left(v_{j+1}(\hbar) \overset{p^+}{\rightsquigarrow} t\right) = 0$. Hence, $w_x(q) = \tau$.

Case (2): $b_j > b_{j+1}$. In this case, $h^+_{j+1} = h^-_j + (b_{j+1} - b_j)$. For any integer h with $h^+_{j+1} \leq h \leq h^-_j$, $w_x\left(s \overset{p^-}{\rightsquigarrow} v_j(h^-_j) \rightarrow v_{j+1}(h)\right) = w_x(p_j)$, and for any $h^-_j < h \leq h^-_{j+1}$, $w_x\left(s \overset{p^-}{\rightsquigarrow} v_j(h^-_j) \rightarrow v_{j+1}(h)\right) = w_x(p_j) + (h - h^-_j)$. Let $\hbar = h^-_j + (\tau - w_x(p_j))$. Clearly, $h^-_{j+1} > \hbar > h^+_{j+1}$. As in Case (1), we can define an s-t path q in G as $s \overset{p^-}{\rightsquigarrow} v_j(h^-_j) \rightarrow v_{j+1}(\hbar) \overset{p^+}{\rightsquigarrow} t$, and $w_x(q) = \tau$.

We next need to prove that path q is a shortest s-t path in G. Actually, we go a step further to show that for any h with $h^+_{j+1} < h < h^-_{j+1}$, the path $q = s \overset{p^-}{\rightsquigarrow} v_j(h^-_j) \rightarrow v_{j+1}(h) \overset{p^+}{\rightsquigarrow} t$ is a shortest s-t path in G.

Consider the three shortest paths, $v_{j+1}(h^+_{j+1}) \overset{p^+}{\rightsquigarrow} t$, $v_{j+1}(h) \overset{p^+}{\rightsquigarrow} t$, and $v_{j+1}(h^-_{j+1}) \overset{p^+}{\rightsquigarrow} t$. By Claim 1, we have $w\left(v_{j+1}(h^+_{j+1}) \overset{p^+}{\rightsquigarrow} t\right) \leq w\left(v_{j+1}(h) \overset{p^+}{\rightsquigarrow} t\right) \leq w\left(v_{j+1}(h^-_{j+1}) \overset{p^+}{\rightsquigarrow} t\right)$. Further, it follows from Lemma 13 and the definition of $f(\cdot)$ that $w(v_j(h^-_j), v_{j+1}(h^+_{j+1}))$ $\leq w(v_j(h^-_j), v_{j+1}(h)) \leq w(v_j(h^-_j), v_{j+1}(h^-_{j+1}))$. Hence,

$$w\left(v_j(h^-_j) \to v_{j+1}(h^+_{j+1}) \overset{p^+}{\rightsquigarrow} t\right) \leq w\left(v_j(h^-_j) \to v_{j+1}(h) \overset{p^+}{\rightsquigarrow} t\right) \leq w\left(v_j(h^-_j) \to v_{j+1}(h^-_{j+1}) \overset{p^+}{\rightsquigarrow} t\right).$$

Note that all the three paths, p_j, q, and p_{j+1}, share the same sub-path $s \overset{p^-}{\rightsquigarrow} v_j(h^-_j)$. Thus, $w(p_j) \leq w(q) \leq w(p_{j+1})$. Since both p_j and p_{j+1} are shortest s-t paths, it immediately follows that q is a shortest s-t path in G.

This proves the lemma. $\square$

The following Lemma immediately follows.

Lemma 24. *Given a non-negative vector* $\mathbf{b} = (b_1, b_2, \ldots, b_N)$ *and a non-negative integer* τ*, there exists an optimal decomposition* $\mathbf{x}$ *and* $\mathbf{y}$ *of* $\mathbf{b}$ *with* $W_{ovd}(\mathbf{x}) = \tau$ *if and only if* $0 \leq \tau \leq \rho(\mathbf{b}) - b_N$*, and such an optimal decomposition* $\mathbf{x}$ *and* $\mathbf{y}$ *can be computed in* $O(N)$ *time.*

To compute an optimal split of IM A, we perform the following steps: (1) Compute an optimal set of $d-1$ feathering regions F_k's. (2) For each row i of these feathering regions, compute $\rho(i) = (\rho_1(i), \rho_2(i), \ldots, \rho_{d-1}(i))$ by Lemma 16. (3) Using our MSP algorithm to decompose each $\rho(i)$ into $\alpha(i)$ and $\beta(i)$. (4) Based on Lemma 24, each row i of every feathering region F_k can be decomposed into two vectors $\mathbf{x}_k(i)$ and $\mathbf{y}_k(i)$ such that $W_{ovd}(\mathbf{x}_k(i)) = \alpha_k(i)$. We thus obtain an optimal spit $\mathcal{S} = \{S_1, S_2, \ldots, S_d\}$ from the vectors $\mathbf{x}_k(i)$ and $\mathbf{y}_k(i)$ ($k = 1, 2, \ldots, d-1$ and $i = 1, 2, \ldots, m$). Hence, the following theorem holds.

Theorem 2. *Given an IM $A = (a_{i,j})_{m\times n}$,an integral maximum field width $\varpi > 0$, and the width range $[\delta \mathinner{..} \Delta]$ of each feathering region ($0 < \delta < \Delta < \varpi$), the OFSB problem can be solved in $O(mn\alpha(\varpi))$ time, where $\alpha(\cdot)$ is the inverse Ankermann function.*

4.4 An Alternative Method for Computing the Optimal Set of Feathering Regions

Our algorithm for computing the optimal set of feathering regions in Sections 4.2 and 4.3 require the width of each sub-IMs to be ϖ and a upper bound of the feathering width Δ. By removing these two constraints we can get a more general description of the field splitting problem but the shortest path approach may not work. We then turn to a less efficient approach – dynamic programming.

After removing the two constraints, the problem can be modeled as the following **general optimal field splitting with balanced beam-on times (GOFSB)** problem:

Given an IM $A = (a_{i,j})_{m\times n}$ of size $m \times n$, an integral maximum leaf spread $\varpi > 0$, and the minimum width δ of each feathering region ($0 < \delta < \Delta < \varpi$), split A using vertical lines into a sequence of $d = \lceil \frac{n-\delta}{\varpi-\delta} \rceil$ (≥ 2) sub-IMs, such that: (1) the width of each sub-IM is no larger than ϖ; (2) any two neighboring sub-IMs in the sequence overlap each other and the width of the overlapping (feathering) region ranges from δ to Δ; (3) no sub-IM overlaps completely with its neighboring sub-IM(s); and (4) the total complexity of all these d sub-IMs is minimized.

To apply the dynamic programming approach, we explore the optimal substructure of the problem, as follows. Let $C(x, N, L, k)$ be the minimum complexity increase of the following subproblem of the field splitting:

- The intensity map consists of the first x columns of the input IM A, which is to be split into k sub-IMs.
- The first $(k-1)$ sub-IMs cover the first $(x-N)$ columns of the input IM A.
- The width of the feathering region between the last two sub-IMs (i.e., the $(k-1)$-th and k-th sub-IMs) is L.
- The width of each sub-IM is no larger than ϖ.
- Any two adjacent sub-IMs overlap each other with a feathering width no less than δ.
- No sub-IM overlaps completely with its neighboring sub-IM(s).

Note that $N+L$ is the width of the k-th sub-IM of the split, thus $N+L \leq \varpi$. We then have the following recurrence formula.

$$\begin{aligned} C(x, N, L, k) &= \min_{\substack{N'+L' \leq \varpi \\ L' \geq \delta, N' \geq L}} \{C(x-N, N', L', k-1)\} \\ &+ \Delta_{cpl}(A[x-N-L+1 \,..\, x-N]), \end{aligned} \tag{4.2}$$

where $\Delta_{cpl}(A[x-N-L+1 \,..\, x-N])$ is the minimum complexity increase while decomposing the feathering region $A[x-N-L+1 \,..\, x-N]$, which can be solve by

Lemma 17. The condition $L' + N' \leq \varpi$ is to make sure that the width of each sub-IM is no larger than the maximum leaf spread ϖ. The minimum feathering width is guaranteed by $L' \geq \delta$. To make sure that no sub-IM overlaps completely with its neighboring sub-IM(s), we let $N' \geq L$.

The optimal solution to $\min_{\substack{N+L\leq\varpi \\ L\geq\delta}} C(n, N, L, d)$ gives an optimal split of A. Initially, if $n \leq \varpi$, we do not need to split A. Thus, for any $x \leq \varpi$, we have $C(x, N, L, 1) = 0$ for any $N + L \leq \varpi$ and $L \geq \delta$. To make sure that the number of sub-IMs in the split is d, let $C(x, N, L, 0) = +\infty$ for any $x > 0$, $N + L \leq \varpi$, and $L \geq \delta$. Intuitively, after using d sub-IMs to split IM , if the residual IM of A is not empty (i.e., $x > 0$), that means we have to use more than d sub-IMs in that split. We thus let $C(x, N, L, 0) = +\infty$ to prevent using more than d sub-IMs for splitting A.

After the optimal feather regions are determined, the actual decomposition of the feathering regions are done by our MSP method and then the splitting is achieved.

Note that it takes $O(n^2\varpi + mn)$ time to compute the optimal set of feathering regions by dynamic programming. The MSP algorithm takes a linear $O(mn)$ time. We thus have the following theorem.

Theorem 3. *Given an IM $A = (a_{i,j})_{m\times n}$, an integral maximum leaf spread $\varpi > 0$, and the minimum feathering width δ ($0 \leq \delta < w/2$), the GOFSB problem can be solved in $O(n^2\varpi + mn)$ time.*

4.5 Implementation and Experiments

4.5.1 Experiments for the OFSB Algorithm

To study the performance of our new OFSB algorithm for clinical applications, we implemented the algorithm using Matlab on Windows XP system. We experimented with the resulting software on 1000 randomly generated intensity maps for each problem configuration and also on 105 sets of clinical intensity maps obtained from the Department of Radiation Oncology, the University of Iowa. We conducted comparisons with Chen and Wang's FSMP algorithm [18] for field splitting with feathering, which devotes to minimize the total beam-on time of the resulting sub-IMs. With the FSMP algorithm, a given IM was split into d sub-IMs with each one being of the same width of ϖ, where ϖ was the maximum field width. Actually, Chen and Wang's method did not guarantee overlapping between two adjacent sub-IMs, especially, when $n\%\varpi < d$, that was impossible. To ensure a fair comparison, the minimum feathering width δ in our algorithm was set to 0. In addition, one may also note that in our algorithm, we did not require that the last sub-IM in a split be of width of ϖ although the first d-1 sub-IMs had the same width of ϖ. Both the total minimum beam-on time and the number of MLC apertures were used as measures of the results. Furthermore, we compared the total minimum beam-on time and the the number of MLC apertures of the resulting sub-IMs by our OFSB algorithm to those obtained without splitting. In these experiments, the minimum feathering width δ was set to 3 in our OFSB algorithm.

For the randomly generated intensity maps, the widths of the tested intensity

Table 4.1: Comparison results on randomly generated intensity maps of various sizes and on different maximum field widths.

	ϖ	$n = 25$	$n = 35$	$n = 45$	$n = 60$	$n = 75$
beam-on time	14	-2.03%[‡]/2	4.81%/3	14.41%/4	11.73%/5	9.03%/6
	16	0.33%/2	12.56%/3	1.57%/3	2.87%/4	3.73%/5
	18	1.89%/2	-0.23%[‡]/2	5.87%/3	9.96%/4	14.88%/5
# of MLC	14	0.20%/2	-0.43%/3	-1.30%/4	-0.54%/5	-0.03%/6
	16	-3.29%/2	-3.92%/3	0.24%/3	0.18%/4	0.13%/5
	18	-8.18%/2	0.01%/2	-0.96%/3	-0.92%/4	1.08%/5

(a)

	ϖ	$n = 25$	$n = 35$	$n = 45$	$n = 60$	$n = 75$
beam-on time	14	6.84%/2	11.50%/3	14.49%/4	19.03%/6	19.47%/7
	16	7.19%/2	11.35%/3	14.60%/4	15.66%/5	16.74%/6
	18	9.86%/2	13.50%/3	10.23%/3	12.68%/4	14.16%/5
# of MLC	14	12.64%/2	19.65%/3	24.06%/4	31.07%/6	32.25%/7
	16	12.52%/2	19.40%/3	23.97%/4	26.47%/5	27.81%/6
	18	12.19%/2	18.51%/3	17.21%/3	20.73%/4	23.30%/5

(b)

(a) The average increment of the total minimum beam-on time and the number of MLC apertures of the resulting sub-IMs by using our OFSB algorithm over Chen and Wang's FSMP algorithm. The values after "/" show the numbers of the resulting sub-IMs after splitting. (b) The average increment of the total minimum beam-on time and the number of MLC apertures of the resulting sub-IMs after splitting using our OFSB algorithm over those obtained without splitting.

[‡]: Our OFSB algorithm does not require that the last sub-IM of the split have a maximum field width, while the FSMP algorithm does. Thus, our algorithm allows some more flexibility for splitting, which indicates that it may outperform the FSMP algorithm in term of the total minimum beam-on time.

Table 4.2: Comparison results on 105 clinical intensity maps.

ϖ	d	beam-on time			# of MLC apertures		
		FSMP	OFSB	increment%	FSMP	OFSB	increment%
14	2	6081	6232	2.48	837	840	0.36
	3	15502	16490	6.37	2324	2385	2.62
	4	2061	2267	10.00	346	352	1.73
16	2	8849	8908	0.67	1297	1300	0.23
	3	11931	13015	9.09	1960	1975	0.77
18	2	13854	13970	0.843	2161	2134	-1.25
	3	4608	5257	14.08	836	854	2.15

The maximum field widths ϖ considered were 14, 16, and 18 (first column). The clinical datasets were grouped based on the number of the resulting sub-IMs after splitting (second column). The total minimum beam-on time of the resulting sub-IMs for each group by using Chen and Wang's FSMP algorithm and our OFSB algorithms are shown in columns 3 and 4, respectively. The average increment in the total minimum beam-on time obtained using our OFSB algorithm over Chen and Wang's FSMP algorithm are shown in column 5. The comparison results in terms of the number of MLC apertures of the resulting sub-IMs are shown in columns 6, 7 and 8.

Table 4.3: The comparison of execution times.

d	ϖ	$T_{FSMP}(s)$	$T_{OFSB}(s)$	T_{FSMP}/T_{OFSB}
4	14	6.81	0.85	8.01
	16	11.11	0.94	11.82
	18	17.42	1.11	15.69
5	14	32.72	0.94	34.81
	16	59.86	1.19	50.30
	18	103.84	1.33	78.08
6	14	138.19	1.13	122.29
	16	286.19	1.30	220.15
	18	572.05	1.50	381.37

d is the number of resulting sub-IMs from the splittings, and ϖ is the maximum field width. The intensity maps we used were of width $d * \varpi + \xi_\varpi$ for $\xi_\varpi = 0, 1, \ldots, \varpi - 1$. For each configuration (d, ϖ, ξ_ϖ), we randomly generated 100 IMs and ran each of the FSMP and OFSB programs on every IMs to calculate the average execution time. Notice that the running time of the FSMP algorithm largely depends on ξ_ϖ. To make a fair comparison, in the table we show for each (d, ϖ) pair the accumulation of the average execution times for the configurations (d, ϖ, ξ_ϖ) with ξ_ϖ ranging from 0 to $\varpi - 1$.

maps were 25, 35, 45, 60, and 75, to get different number of sub-IMs. The maximum intensity level was set to 100. The average increment in the total beam-on time of the resulting sub-IMs obtained using our OFSB algorithm over Chen and Wang's FSMP algorithm was only 6.09%. Note that the FSMP algorithm can produce optimal splits with respect to the total minimum beam-on time. In terms of the number of MLC apertures, our algorithm slightly outperformed the FSMP algorithm. Table 4.1(a) shows the comparison results in term of the beam-on time and the number of MLC apertures for the FSMP and OFSB algorithms. Our experiments also demonstrated the increase of the total minimum beam-on time and the number of MLC-apertures were pretty small (as shown in Table 4.1 (b)) after splitting comparing to those obtained without splitting. For example, when the IMs needed to be split into 7 sub-IMs, the total minimum beam-on time only increased by less than 20% and the number of MLC-apertures by about 30%.

For clinical data, the widths of the tested intensity maps ranged from 15 to 47. The maximum intensity level of each IM was normalized to 100. The maximum field widths used were 14, 16, and 18. For the tested IMs, the total minimum beam-on time of the resulting sub-IMs obtained using our OFSB algorithm was only slightly larger than that obtained by Chen and Wang's FSMP algorithm, with an average increase of 6.2%; while the number of MLC apertures output by our algorithm was comparable to that by Chen's algorithm, with an average increase of 0.9%. Table 4.2 shows the comparison results of the two algorithms for those 105 clinical IMs.

Our algorithm runs very fast comparing to Chen and Wang's FSMP algorithm.

For most of the intensity maps, the execution time was within a second, while the FSMP algorithm took hundreds of seconds. Table 4.3 shows the comparison results of the execution times.

4.5.2 Experiments for the GOFSB algorithm

The performance of the GOFSB algorithm was tested using 54 intensity maps that were generated with a commercial inverse treatment planning system (Pinnacle v8.0m). The size of the intensity maps was of 101×101 and the maximum intensity levels ranged from 42 to 147. 23 of those tested IMs were needed to be split into 3 sub-IMs and the sizes of the tumor contour were between $27cm$ and $30.5cm$. The remaining 31 IMs were needed to be split into 2 sub-IMs and the sizes of the tumor contours ranged from $15.5cm$ to $19.5cm$.

We assume to do the planning for the Varian LINAC System. The maximum leaf spread was set to $14.5cm$ and the minimum feathering width was set to $3cm$. Our splitting method was compared with that used in Pinnacle v8.0m. The total number of MUs and the total number of MLC-apertures required to deliver the split generated by Pinnacle and that generated by our method were computed using the leaf sequencing algorithm in Pinnacle. Note that our method allowed varying feathering widths ($\geq 3cm$) while Pinnacle used a fixed width of the feathering regions.

For all the 54 datasets, the average decreases in MUs and in the number of MLC-apertures obtained using our algorithm over Pinnacle v8.0m were, respectively, 18.95% and 12.22%. The maximum decreases were 63.68% and 50.00% for the number of MUs and MLC-apertures, respectively. 39 out of the 54 (72.22%) intensity maps got decreased in MUs and the average decrease for those intensity maps was 24.83%; while 34 intensity maps out of the 54 (62.96%) datasets got decreased in the number of MLC-apertures and the average decrease for those intensity maps was 20.86%.

For those 23 tested IMs that were needed to be split into 3 sub-IMs, the number of MUs and the number of MLC-apertures were reduced by, respectively, 25.70% and 14.70% on average. The maximum decreases in MUs and in the number of MLC-apertures were 63.68% and 50.00%, respectively. In terms of number of MUs, 19 out of the 23 (82.61%) IMs got reduced MUs and the average decrease for those IMs was 30.65%. For the number of MLC-apertures, 17 IMs out of the 23 (73.91%) datasets got reduced measure with an average decrease of 20.53%. The two methods produced splitting results of the same total number of MLC-apertures for 2 IMs. The results are tabulated in Table 4.4.

Table 4.4: Comparison between Pinnacle and our field splitting method on IMs to be split into 3 sub-IMs

IMs	# of MUs			# of MLC-apertures		
	Pinnacle	GOFSB	% Decrease	Pinnacle	GOFSB	% Decrease
1	353	257	27.2%	29	22	24.1%
2	485	369	23.9%	31	28	9.7%
3	309	273	11.7%	47	41	12.8%
4	224	219	2.2%	35	37	-5.7%
5	273	297	-8.8%	49	49	0.0%
6	294	309	-5.1%	27	29	-7.4%
7	271	263	3.0%	28	27	3.6%
8	229	240	-4.8%	22	22	0.0%
9	505	235	53.5%	43	29	32.6%
10	368	241	34.5%	33	30	9.1%
11	702	255	63.7%	41	23	43.9%
12	427	292	31.6%	41	40	2.4%
13	464	264	43.1%	39	37	5.1%
14	298	233	21.8%	40	38	5.0%
15	247	225	8.9%	33	28	15.2%
16	357	246	31.1%	36	30	16.7%

Table 4.4 – Continued

IMs	# of MUs			# of MLC-apertures		
	Pinnacle	GOFSB	% Decrease	Pinnacle	GOFSB	% Decrease
17	365	345	5.5%	29	31	-6.9%
18	355	207	41.7%	40	20	50.0%
19	270	205	24.1%	38	26	31.6%
20	176	217	-23.3%	23	27	-17.4%
21	331	186	43.8%	45	33	26.7%
22	249	215	13.7%	30	25	16.7%
23	300	241	19.7%	44	30	31.8%
Total	7852	5834	25.7%	823	702	14.7%

For those 31 tested IMs to be split into 2 sub-IMs, the performance of our method was worse than that for those to be split into 3 sub-IMs. The average decreases in MUs and in the number of MLC-apertures obtained using our method over Pinnacle v8.0m were 7.87% and 9.55%, respectively, with maximum decreases of 27.43% and 38.24%. Considering the number of MUs, 20 out of the 31 (64.52%) tested IMs got reduced MUs and the average reduction for those IMs was 13.37%. In term of the number of MLC-apertures, 17 out of the 31 (54.84%) IMs got reduced measure and the average decrease for those IMs was 21.32%. The two methods produced splitting results of the same total number of MLC-apertures for 6 IMs. The results are tabulated in Table 4.5.

Table 4.5: Comparison between Pinnacle and our field splitting method on IMs to be split into 2 sub-IMs

IMs	# of MUs			# of MLC-apertures		
	Pinnacle	GOFSB	% Decrease	Pinnacle	GOFSB	% Decrease
1	241	200	17.0%	22	19	13.6%
2	94	88	6.4%	22	22	0.0%
3	78	81	-3.9%	14	15	-7.1%
4	99	91	8.1%	24	22	8.3%
5	91	94	-3.3%	22	22	0.0%
6	186	173	7.0%	20	20	0.0%
7	277	219	20.9%	19	20	-5.3%
8	163	162	0.6%	34	39	-14.7%
9	202	178	11.9%	22	18	18.2%
10	261	200	23.4%	23	17	26.1%
11	186	186	0.0%	22	22	0.0%
12	211	224	-6.2%	25	25	0.0%
13	318	258	18.9%	34	23	32.4%
14	166	163	1.8%	15	22	-46.7%
15	136	154	-13.2%	24	20	16.7%
16	94	105	-11.7%	25	21	16.0%

Table 4.5 – Continued

IMs	# of MUs			# of MLC-apertures		
	Pinnacle	GOFSB	% Decrease	Pinnacle	GOFSB	% Decrease
17	152	123	19.1%	30	20	33.3%
18	149	146	2.0%	28	23	17.9%
19	101	93	7.9%	22	23	-4.6%
20	89	98	-10.1%	34	21	38.2%
21	112	115	-2.7%	28	21	25.0%
22	204	183	10.3%	19	24	-26.3%
23	206	182	11.7%	20	17	15.0%
24	87	100	-14.9%	27	20	25.9%
25	64	75	-17.2%	29	23	20.7%
26	112	107	4.5%	21	22	-4.8%
27	137	144	-5.1%	30	33	-10.0%
28	126	104	17.5%	32	28	12.5%
29	84	75	10.7%	30	26	13.3%
30	116	110	5.2%	24	24	0.0%
31	237	172	27.4%	23	19	17.4%
Total	4779	4403	7.9%	764	691	9.6%

4.6 Discussion

Since not all MLC systems require field splitting, e.g., Siemens MLCs and Tomotherapy do not require field splitting in clinical practice, our algorithm mainly benefit when we use small MLC systems (e.g., Varian system) to treat large tumor sites. In clinical practice, large tumor sites include many pelvic cases, e.g., cervix and prostate (with lymph node metastases), and some head & cases, e.g., larynx.

It is desirable to maximize the delivery efficiency while splitting a large IM. Two measures, the number of MUs and the number of segments, are associated with the delivery efficiency of an IM. We certainly can use either one to measure the complexity of an IM and to minimize the total complexity of the sub-IMs of a split. There are two issues with that method. First, the resulting optimization problem is computational intractable. It may take a prohibitively long time to compute the splitting result for a relatively large intensity field. Secondly, minimizing the total number of either MUs or segments may result in a split of compromised quality with respect to the other measure. Previous work [47, 45, 80, 18] mainly focused on improving the total MU efficiency of a split while ignoring the total number of segments. In this paper, we use the *sum of positive gradients* to measure the complexity of an IM, which is closely related to the number of MUs. Experiments on our GOFSB algorithm also revealed that the total MLC-aperture efficiency got improved, comparing to the field splitting method in Pinnacle v8.0m. On average, the decreases in MUs and in the number of MLC-apertures obtained using our method over Pinnacle v8.0m were, respectively, 18.95% and 12.22% for all 54 tested datasets. To the best of our knowledge, this is the first report to address both the total MU and the total MLC-aperture efficiencies of a field splitting method.

The performance of our GOFSB algorithm on the tested IMs to be split into 2 sub-IMs was not as good as that on the IMs to be split into 3 sub-IMs. It seemed that our adopted complexity measure (i.e, the sum of positive gradients) might not work quite well for less “complex” IMs. Consider those tested IMs to be split into 2 sub-IMs. We observed that for those IMs whose total number of MUs of the resulting split by Pinnacle v8.0m was less than 150, the performance of our method was almost the same as that of Pinnacle in terms of the total MU efficiency. In addition, for those IMs whose total number of segments of the resulting split by Pinnacle was less than 22, the splitting method in Pinnacle outperformed our algorithm considering the total segment efficiency, and the average *increase* in the number of segments obtained using our algorithm over Pinnacle was 9.38%. We also observed that, interestingly, for those IMs on which the Pinnacle splitting method outperformed our algorithm in terms of the total MU efficiency, our algorithm outperformed the Pinnacle method regarding to the total segment efficiency and the average decrease was 14.34%. This may indicate that the Pinnacle splitting method focuses on optimizing the total MU efficiency, especially for less “complex” IMs.

CHAPTER 5
CONCLUSION AND FUTURE WORK

In this chapter, we conclude the thesis and discuss some possible future problems.

5.1 Summary of Results

In this thesis, we study some problems arise in the field of intensity-modulated radiation therapy IMRT) – the auto-contouring using region properties, the field decomposition problem, and the field splitting problem. Experiments on clinical data with the algorithms reveal good performance.

5.1.1 Auto-Contouring Using Regional Properties

An algorithm to find globally optimal solution to segmentation by minimizing the intraclass variance is reported. Our approach detects an optimal region bounded by two coupled terrain-like surfaces in a volumetric image in a low-order polynomial time. We employ the techniques of parametric search, shape probing in computational geometry, and 3-D graph-search. The developed approximation algorithm exhibits a significantly improved running times when compared with our optimality guaranteeing algorithm while still producing highly close-to-optimal solutions. A fast algorithm for the simplified version of this problem - the single surface detection problem - is also developed.

Experiments on computer phantoms demonstrate the correctness of our algorithm. Accurate and consistent performance on CT and MRI images of airway wall, cardiac ventricle, and liver lesion show good applicability of our algorithm to clinical cases. Sub-pixel results are achieved for most of the clinical data.

5.1.2 Field Decomposition

We solve the matrix decomposition problem that seeks to decompose a "complicated" matrix into two "simpler" matrices.

We propose two complexity definitions and develop algorithm for the two complexity definitions. Our algorithm is based on a non-trivial graph construction scheme inspired by Wu *et al.*'s VCE method [80], which enables us to formulate the decomposition problem as computing a minimum s-t cut in a 3-D geometric multi-pillar graph.

Experiments on clinical intensity map matrices demonstrates the performance of our algorithm. Using the positive gradient sum cost, our algorithm produces as much as 27.3% less MLC-apertures with an average of 13.1% comparing with the SLS method using a single direction for delivery. Our method with total variation cost performs better in terms of number of MU. Using our decomposition algorithm, the average improvement percentage of number of MU is 45.05%.

5.1.3 Field Splitting

We develop a near linear time algorithm for solving the field splitting problem while minimizing the total complexity of the resulting sub-IMs by formulating it to a shortest path problem in a directed acyclic graph (DAG). A less efficient algorithm for a more general case is also developed based on dynamic programming approach. In both of the two approaches, the minimum increase in complexity when splitting a feathering region is used and formulated as a shortest path problem. This problem is of its own interest.

To minimize the maximum beam-on time of the resulting sub-IMs, we consider an interesting min-max slope path problem in a monotone polygon which is solvable in linear time.

Experiments shows that our algorithm is very fast with good performance. A comprehensive comparison study is conducted against a commercial treatment planning system in radiation therapy for the general case algorithm. Our field splitting method outperforms Pinnacle in terms of both the total MU efficiency and the total MLC-aperture efficiency. The average decreases in the number of MUs and in the number of MLC-apertures obtained using our method over a conventional method are, respectively, 19.0% and 12.2%. To the best of our knowledge, this is the first study of the field splitting method considering both the total MU and the total number of MLC-aperture efficiencies.

5.2 Future Work

Here we point out some directions of our future research for each of the problems we studied. We also proposed some other problems we can study in the future in the area of IMRT.

5.2.1 Auto-Contouring Problem

In the development of our auto-contouring algorithm, we focused on the optimality of the algorithm while ignoring some applicability issues. In the future works, we may put more efforts on the following aspects to make our algorithm perform better in clinical applications.

1. One limitation of our auto-contouring algorithm is that the method can only detect those surfaces that can be unfolded to be terrain-like, including cylindrical or tubular surfaces (or a closed surface, if we build the graph from a mesh). However, one may experience difficulty to unfold more complex objects into terrain-like surfaces (e.g. branches). Thus, one direction of our future work is to make the method topologically adaptive.

2. Another problem with our contouring algorithm is the smoothness definition. With our smoothness constraint, a zigzagging boundary may be regarded as "smooth". To solve this problem, we may use second-order smoothness instead of the current first-order smoothness when setting the smoothness constraint. This may greatly improve the smoothness of our segmentation result.

5.2.2 Field Decomposition Problem

In the future we may focus on improving the complexity definition and running time efficiency of our algorithm.

1. About the complexity definition, although the current definitions achieves good improvement, using the number of MUs, or if possible, number of MLC-apertures

(which may not be captured by a formula) directly may potentially improve more on the performance of the algorithm. But under current framework of our algorithm, using these complexity definitions will make the problem computationally intractable. We thus need to seek other approaches or develop approximation algorithms under current structure.

2. Recall that our field decomposition algorithm has a pseudo-polynomial time complexity which may not seem good. Using local search method may be able to improve the efficiency but to achieve a true polynomial time algorithm there is still a lot of work to do.

5.2.3 Field Splitting Problem

Our future work on field splitting problem will focus on improving the performance, and its applicability to clinical cases. In this context, there are two primary areas needing attention.

1. Similar to the field decomposition problem, we need to improve the complexity definition to achieve a better performance.

2. More experiments should be done in order to figure out when our algorithm can make an improvement and should be applied. Experiments are also needed to demonstrate its applicability to clinical cases by studying how field splitting algorithms affect the difference between delivered intensity map and the prescribed dose.

5.2.4 Tumor Tracking Problem

Recent study shows that respiratory movement can cause considerable changes in the area and displacement of the tumor [68]. It is more desirable to irradiate the tumor by gating the timing according to the movement of the tumor. Thus real-time tumor tracking can help improve the accuracy of radiation delivery. Real-time tracking requires the capability to automatically adjust the relative position between irradiation and the moving tumor.

All approaches to real-time tracking requires the measurement of tumor position on a time scale faster than the motion itself. Most commonly, the measurement is made via radiographic imaging. Although our algorithm in Chapter 2 can be extended to higher dimensional cases, the time complexity makes it difficult to be utilized on tumor tracking problems. More efficient approaches, e.g. matching of a deformable template, need to developed to solve this problem.

5.2.5 Intensity Map Smoothing Problem

It has been verified that smoother intensity maps can be expected to produce shorter delivery time. Süss and Küfer's work [72] built a theoretical foundation for the consideration of intensity map smoothness. They suggested to use the smoothness of an intensity map as part of the objective function when doing optimization (step 2 of the workflow).

We consider doing the smoothing operation between optimization and leaf sequencing. Of course this may compromise the accuracy of the plan comparing with the prescribed dose and thus balance needs to be kept between effectiveness and accuracy.

REFERENCES

[1] A. Aggarwal, M.M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.

[2] T. Asano, D.Z. Chen, N. Katoh, and T. Tokuyama. Efficient algorithms for optimization-based image segmentation. *International Journal of Computational Geometry and Applications*, 11(2):145–166, 2001.

[3] N. Boland, H.W. Hamacher, and F. Lenzen. Minimizing beam-on time in cancer radiation treatment using multileaf collimators. *Networks*, 43(4):226–240, 2004.

[4] T. Bortfeld, J. Bürkelbach, R. Boesecke, and W. Schlegel. Methods of image reconstruction from projections applied to conformation radiotherapy. *Physics in Medicine and Biology*, 35(10):1423–1434, 1990.

[5] T. Bortfeld, D. Craft, A. Trofimov, T. Halabi, and K.-H. Kufer. Qantifying the tradeoff between complexity and conformality. *Medical Physics*, 32:2085, 2005.

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

[7] A.L. Boyer, G.E. Desobry, and N.H. Wells. Potential and limitations of invariant kernel conformal therapy. *Medical Physics*, 18:703–712, 1991.

[8] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[9] Y. Boykov and V. Kolmogorov. An experimental comparison of mincut/ max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1124–1137, 2004.

[10] A. Brahme. Optimization of stationary and moving beam system for dynamic treatment techniques. *Medical Physics*, 27:2198–2208, 1988.

[11] T. Chan and L. Vese. An active contour model without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[12] D.Z. Chen, O. Daescu, X.S. Hu, X. Wu, and J. Xu. Determining an optimal penetration among weighted regions in two and three dimensions. In *Proceedings*

of the 5th Annual Symposium on Computational Geometry (SoCG), pages 322–331, 1999.

[13] D.Z. Chen, M.A. Healy, C. Wang, and X. Wu. A new field splitting algorithm for intensity-modulated radiation therapy. In *Proceedings of the 13th Annual International Computing and Combinatorics Conference (COCOON)*, pages 4–15, 2007.

[14] D.Z. Chen, X.S. Hu, S. Luan, S.A. Naqvi, C. Wang, and C. Yu. Generalized geometric approaches for leaf sequencing problems in radiation therapy. *International Journal of Computational Geometry and Applications*, 16(2-3), 2006.

[15] D.Z. Chen, X.S. Hu, S. Luan, C. Wang, and X. Wu. Geometric algorithms for static leaf sequencing problems in radiation therapy. *International Journal of Computational Geometry and Applications*, 14(5):311–339, 2004.

[16] D.Z. Chen, X.S. Hu, S. Luan, C. Wang, and X. Wu. Mountain reduction, block matching, and medical applications. In *Proceedings of the 21st Annual ACM Symposium on Computational Geometry (SoCG)*, pages 35–44, 2005.

[17] D.Z. Chen, S. Luan, and J. Xu. Topological peeling and implementation. In *Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC)*, pages 454–466, 2001.

[18] D.Z. Chen and C. Wang. Field splitting problems in intensity-modulated radiation therapy. In *Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC)*, pages 701–711, 2006.

[19] D.Z. Chen, J. Wang, and X. Wu. Image segmentation with monotonicity and smoothness constraints. In *12th International Symposium on Algorithms and Computation*, pages 467–479, 2001.

[20] Y. Chen, Q. Hou, and J.M. Galvin. A graph-searching method for MLC leaf sequencing under constraints. *Medical Physics*, 31:1504–1511, 2004.

[21] R. Cole and C.K. Yap. Shape from probing. *Journal of Algorithms*, 8(1):19–38, 1987.

[22] D.J. Convery and M.E. Rosenbloom. The generation of intensity-modulated fields for conformal radiotherapy by dynamic collimation. *Physics in Medicine and Biology*, 37:1359–1374, 1992.

[23] A.M. Cormack. A problem in rotation therapy with X-rays. *International Journal of Radiation Oncology, Biology, Physics*, 13:623–630, 1987.

[24] J. Dai and Y. Zhu. Minimizing the number of segments in a delivery sequence for intensity-modulated ratiation therapy with multileaf collimator. *Medical Physics*, 28(10):2113–2120, 2001.

[25] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Edition.* Springer-Verlag, 2000.

[26] J. Deng, T. Pawlicki, Y. Chen, J. Li, S.B. Jiang, and C.-M. Ma. The MLC tongue-and-groove effect on imrt dose distribution. *Physics in Medicine and Biology*, 46:1039–1060, 2001.

[27] D. Dobkin, H. Edelsbrunner, and C.K. Yap. Probing convex polytopes. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–392, 1986.

[28] N. Dogan, L.B. Leybovich, A. Sethi, and B. Emami. Automatic feathering of split fields for step-and-shoot intensity modulated radiation therapy. *Physics in Medicine and Biology*, 48:1133–1140, 2003.

[29] X. Dou, X. Wu, J.E. Bayouth, and J.M. Buatti. The matrix orthogonal decomposition problem in intensity-modulated radiation therapy. In *Proceedings of the 12th International Computing and Combinatorics Conference (COCOON)*, pages 156–165, 2006.

[30] K. Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Applied Mathematics*, 152:35–51, 2005.

[31] P.M. Evans and M. Partridge. A method of improving the spatial resolution of treatments that involve a multileaf collimator. *Physics in Medicine and Biology*, 45:609–622, 2000.

[32] A.X. Falcao, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live wire on the fly. *IEEE Transactions on Image Processing*, 19:55–62, 2000.

[33] W. Fu, J. Dai, Y. Hu, D. Han, and Y. Song. Delivery time comparison for intensity-modulated radiation therapy with/without flattening filter: A planning study. *Physics in Medicine and Biology*, 49:1535–1547.

[34] G. Gallo, M. Grigoriadis, and R. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18:30–55, 1989.

[35] M. Goitein. *Radiation Oncology: A Physicist's-Eye View*. Springer, 2007.

[36] A.V. Goldberg and R.E. Tarjan. A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery*, 35:921–940, 1988.

[37] L. Grady. Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3d with application to segmentation. *Proceedings of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:69–78, 2006.

[38] D. Gusfield and E. Tados. A faster parametric minimum-cut algorithm. *Algorithmica*, 11(3):278–290, 1994.

[39] D.J. Hand. *Discrimination and Clasification*. John Wiley & Sons, 1981.

[40] B. Hardemark, H. Rehbinder, and J. Lof. Rotating the MLC between segments improves performance in step-and-shoot IMRT delivery. *Medical Physics*, 30, 2003.

[41] T. Heimann, S. Munzing, H. Meinzer, and I. Wolf. A shape-guided deformable model with evolutionary algorithm initialization for 3d soft tissue segmentation. In *Proceedings of the 20th International Conference on Information Processing in Medical Imaging (IPMI)*, pages 1–12, 2007.

[42] L. Hong, A. Kaled, C. Chui, T. LoSasso, M. Hunt, S. Spirou, J. Yang, H. Amols, C. Ling, Z. Fuks, and S. Leibel. IMRT of large fields: Whole-abdomen irradiation. *International Journal of Radiation Oncology and Biology Physics*, 54:278–289, 2002.

[43] D. Kainmueller, H. Lamecker, S. Zachow, M. Heller, and H. Hege. Multi-object segmentation with coupled deformable models. In *Proceedings of the 12th Annual Conference of Medical Image Understanding and Analysis (MIUA)*, pages 34–38, 2008.

[44] T. Kalinowski. Reducing the number of monitor units in multileaf collimator field segmentation. *Physics in Medicine and Boilogy*, 50:1147–1161, 2005.

[45] S. Kamath and S. Sahni. A generalized field splitting algorithm for optimal IMRT delivery efficiency. *Physics in Medicine and Biology*, 52(18):5483–5496, 2007.

[46] S. Kamath, S. Sahni, J. Li, J. Palta, and S. Ranka. A generalized field splitting algorithm for optimal IMRT delivery efficiency. *Medical Physics*, 32(6):1890, 2005.

[47] S. Kamath, S. Sahni, S. Ranka, J. Li, and J. Palta. Optimal field splitting for large intensity-modulated fields. *Medical Physics*, 31(12):3314–3323, 2004.

[48] N. Katoh and T. Ibaraki. A parametric characterization and an ϵ-approximation scheme for the minimization of a quasiconcave program. *Discrete Applied Mathematics*, 17:39–66, 1987.

[49] F.M. Khan. *The Physics of Radiation Therapy.* Lippincott Williams & Wilkins, 2003.

[50] M. Klincewicz. Tumor autocontouring - efficiently maximizing dose and minimizing damage to healthy tissue. *Radiology Today*, 10(11):22, 2009.

[51] H.G. Kuterdem, P.S. Cho, R.J. Marks II, M.H. Phillips, and H. Parsaei. Comparison of leaf sequencing techniques: Dynamic vs. multiple static segments. In *Proceedings of the 8th International Conference on the Use of Computer in Radiation Therapy (ICCR)*, pages 213–215, 2000.

[52] K. Li, S. Millington, X. Wu, D.Z. Chen, and M. Sonka. Simultaneous segmentation of multiple closed surfaces using optimal graph searching. In *Proceedings of the 19th International Conference on Information Processing in Medical Imaging (IPMI)*, pages 406–417, 2005.

[53] K. Li, X. Wu, D.Z. Chen, and M. Sonka. Globally optimal segmentation of interacting surfaces with geometric constraints. *Proceeding of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:394–399, 2004.

[54] H.K. Liu. Two- and three-dimensional boundary detection. *Computer Graphics and Image Processing*, 6:123–134, 1977.

[55] X. Liu, D.Z. Chen, X. Wu, and M. Sonka. Optimal graph-based segmentation of 3d pulmonary airway and vascular trees across bifurcations. In *Proceedings of the 1st International Workshop on Pulmonary Image Analysis*, pages 103–111, 2008.

[56] R. Lu, P. Marziliano, and C.H. Thng. Liver tumor volume estimation by semi-automatic segmentation method. In *Proceedings of IEEE Engineering in Medicine and Biology Society*, pages 3296–3299, 2005.

[57] S. Luan, C. Wang, D.Z. Chen, X.S. Hu, S.A. Naqvi, X. Wu, and C.X. Yu. An improved MLC segmentation algorithm and software for step-and-shoot IMRT delivery without tongue-and-groove error. *Medical Physics*, 33(5):1199–1212, 2006.

[58] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1:169–182, 1972.

[59] U. Montanari. On the optimal detection of curves in noisy pictures. *Communications of the ACM*, 14:335–345, 1971.

[60] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42:577–685, 1989.

[61] M. Oldham and C. Rowbottom. Radiosurgery with a motorized micro-multileaf-collimator: initial experiences in planning and commissioning. In *Proceedings of the 43rd Annual Meeting of the American Association of Physicists in Medicine (AAPM)*, 2001.

[62] K. Otto and B.G. Clark. Enhancement of IMRT delivery through MLC rotation. *Physics in Medicine and Biology*, 47:3997–4017, 2002.

[63] D. Pope, D. Parker, P. Clayton, and D. Gustafson. Left ventricular border detection using a dynamic search. *Radiology*, 155:513–518, 1985.

[64] L.D. Potter, S.X. Chang, T.J. Cullip, and R.A.C. Siochi. A quality and efficiency analysis of the $imfast^{TM}$ segmentation algorithm in head and neck "step & shoot" IMRT treatments. *Medical Physics*, 29(3):275–283, 2002.

[65] W. Que. Comparison of algorithms for multileaf collimator field segmentation. *Medical Physics*, 26:2390–2396, 1999.

[66] K. Roedersheimer, D.Z. Chen, S. Luan, and L. Xing. The impact of multileaf collimator rotation in IMRT planning. *Medical Physics*, 32(6):1973–1974, 2005.

[67] T. Schwarz, T. Heimann, I. Wolf, and H. Meinzer. 3D heart segmentation and volumetry using deformable shape. *Computers in Cardiology*, 34:741–744, 2007.

[68] S. Shimizu, H. Shirato, and K. Kagei. Impact of respiratory movement on the computed tomographic images of small lung tumors in three-dimensional (3D) radiotherapy. *International Journal of Radiation Oncology, Biology, Physics*, 46:1127–1133, 2000.

[69] R.A.C. Siochi. Minimizing static intensity modulation delivery time using an intensity solid paradigm. *International Journal of Radiation Oncology and Biology Physics*, 43:671–680, 1999.

[70] R.A.C. Siochi. Virtual micro-intensity modulated radiation therapy. *Medical Physics*, 27:2480–2493, 2000.

[71] M. Sonka, X. Zhang, S. DeJong, S. Collins, and C. McKay. Segmentation of intravascular ultrasound images: A knowledge-based approach. *IEEE Transactions on Imaage Processing*, 14:719–732, 1995.

[72] P. Süss and K.H. Küfer. Smooth intensity maps and the bortfeld-boyer sequencer. *Berichte des Fraunhofer-Instituts für Techno- und Wirtschaftsmathematik Report*, 109, 2007.

[73] R. Svensson, P. Kallman, and A. Brahme. An analytical solution for the dynamic control of multileaf collimation. *Physics in Medicine and Biology*, 39:37–61, 1994.

[74] S.Webb. *Intensity-Modulated Radiation Therapy.* Taylor & Francis, 2001.

[75] S. Webb. Optimization of conformal radiotherapy dose distributions by simulated annealing. *Physics in Medicine and Biology*, 34:1349–1370, 1989.

[76] Q. Wu, M. Arnfield, S. Tong, Y. Wu, and R. Mohan. Dynamic splitting of large intensity-modulated fields. *Physics in Medicine and Biology*, 45:1731–1740, 2000.

[77] X. Wu. Efficient algorithms for intensity map splitting problems in radiation therapy. In *Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON)*, pages 504–513, 2005.

[78] X. Wu. Efficient algorithms for the optimal-ratio region detection problems in discrete geometry with applications. *International Journal of Computational Geometry and Applications*, 19(2):141–159, 2009.

[79] X. Wu and D.Z. Chen. Optimal net surface problems with applications. In *Proceedings of the 29th International Colloquium in Automata, Languages and Programming(ICALP)*, pages 1029–1042, 2002.

[80] X. Wu, D.Z. Chen, K. Li, and M. Sonka. The layered net surface problems in discrete geometry an medical image segmentation. *International Journal of Computational Geometry and Applications*, 17(3):261–296, 2007.

[81] X. Wu and X. Dou. Optimal field splitting with feathering in intensity-modulated radiation therapy. In *Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management (AAIM)*, pages 327–336, 2007.

[82] P. Xia and L.J. Verhey. MLC leaf sequencing algorithm for intensity modulated beams with multiple static segments. *Medical Physics*, 25:1424–1434, 1998.

[83] C.X. Yu. Design considerations of the sides of the multileaf collimator. *Physics in Medicine and Biology*, 43(5):1335–1342, 1998.