

Simultaneous Border Segmentation of Doughnut-Shaped Objects in Medical Images

Xiaodong Wu^{a,b} Michael Merickel^a

^aDepartment of Electrical and Computer Engineering

^bDepartment of Radiation Oncology

The University of Iowa

Iowa City, IA 52242, USA

<http://www.engineering.uiowa.edu>

{xiaodong-wu,michael-merickel}@uiowa.edu

Abstract

Image segmentation with specific constraints has found applications in several areas such as biomedical image analysis and data mining. In this paper, we study the problem of simultaneous detection of both borders of a doughnut-shaped and smooth objects in 2-D medical images. Image objects of that shape are often studied in medical applications. We present an $O(IJU(U-L)\log\frac{J}{U}\log(U-L))$ time algorithm, where the size of the input 2-D image is $I \times J$, M is the smoothness parameter with $1 \leq M \leq J$, and L and U are the thickness parameters specifying the thickness between two border contours of a doughnut-shaped object. Previous approaches for solving this segmentation problem are computationally expensive and/or need a lot of user interference. Our algorithm improves the straightforward dynamic programming algorithm by a factor of $O(\frac{J(U-L)M^2}{U\log\frac{J}{U}\log(U-L)})$. We explore some interesting observations, which make possible to apply the divide-and-conquer strategy combined with dynamic programming. Our algorithm is also based on computing optimal paths in an implicitly represented graph.

Article Type	Communicated by	Submitted	Revised
Regular paper	X. He	October 2005	May 2007

This research was supported in part by an NIH-NIBIB research grant R01-EB004640, in part by a faculty start-up fund from the University of Iowa, and in part by a seed grant award from the American Cancer Society through an Institutional Research Grant to the Holden Comprehensive Cancer Center, the University of Iowa, Iowa City, Iowa, USA. Part of this work was done at the Department of Computer Science, the University of Texas - Pan American, Edinburg, TX 78541.

1 Introduction

One of the biggest challenges in medical image analysis is accurate image segmentation, which is a key to solving problems in numerous applications such as medical diagnosis, surgical treatment planning, and brain mapping. Image segmentation aims to define accurate boundaries for the objects or regions of interest captured by the image data. This task is in practice quite often performed by human manual tracing. While manual tracing is robust, it is tedious, time-consuming, and can have a significant inter-observer and intra-observer variability [24]. Hence, efficient and effective automated segmentation methods are highly desirable for many applications. Most of the known image segmentation techniques used today are *region based* – examples include region growing [20], fuzzy connectivity [26, 14], and watershed techniques [27]. The second family segmentation techniques consists of *edge-based* (boundary-based) methods. Examples include active shape models [6, 8] and snakes [16, 28, 5], and level sets [19, 18]. Combinations of edge-based and region-based approaches are emerging, such as Active Appearance Models (AAM) [7]. All of these techniques are frequently iterative and their operations are based on a sequence of locally optimal steps, with no guarantee of achieving global optimality once they converge to a solution. As a result, segmentation is frequently locally incorrect and hence requires substantial human supervision and interaction. The region-based methods also often suffer from the problem of “leaking” into surrounding regions. In some applications, image segmentation needs to make use of additional shape information because the target objects are expected to have certain topological or geometric structures or satisfy specific constraints.

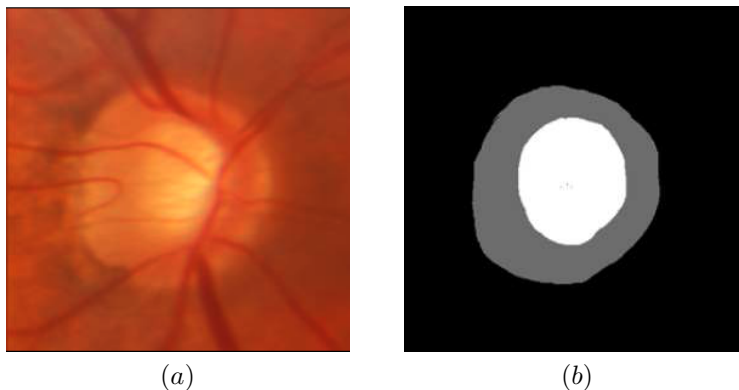


Figure 1: Illustrating a doughnut-shaped structure. (a) A retinal optic disc image consisting of the rim and the cup. (b) The manually traced result of the rim (gray) and the cup (white) by an expert.

In this paper, we study image segmentation for doughnut-shaped and smooth

objects in two dimensions. Doughnut-like shape and smoothness capture the properties of abundant objects in medical images, such as vessels, left ventricles, bones, ducts, and vertebrae. Figure 1 shows a photograph of a retinal optic disc, which consists of the rim and the cup whose boundaries are coupled with each other to form a doughnut-like shape. Conventional segmentation approaches treat such two boundaries independently and the contours are extracted separately, which ignore the relevant information of the coupled borders. Those methods sometimes fail to accurately identify the target contours, especially with the presence of poor contrast, noise, or adjacent structures near the target object [20, 23]. This paper considers the approach of simultaneous detection of coupled contours in 2-D medical images. This approach, intended to mimic the boundary detection strategy of a human observer who will use the position of one contour to create and/or confirm hypotheses about the position of the other contour, has attracted considerable research efforts [21, 23, 12, 29, 20]. There are two major methods for simultaneous detection of coupled contours: graph searching and variants of active contour models. Sonka *et al.* [21, 20] developed a method for simultaneous detection of both coronary borders in an $n \times n$ image. Their approach is based on searching an optimal path in a 3-D lattice graph with $O(n^3)$ vertices and edges. Unfortunately, it relies on users to define an approximate centerline between the coupled borders to construct the 3-D graph. Very recently, Spreuwers and Breeuwer [23] extended the active contour method by imposing the geometric properties of coupled boundaries and proposed a so-called *coupled active contour* model to detect the left ventricular epi- and endo-cardinal borders simultaneously. However, this approach suffers the same shortcoming as the active contour model. The major drawback is the lack of the capability of producing globally optimal solutions. The performance of the active contour model is in general sensitive to the initial contour, which has to be initialized very close to the true boundary of the target object. In this paper, we develop a new efficient algorithm based on graph searching for extracting globally optimal coupled-contours simultaneously with much less user interference.

In general, an original 2-D image can be described by a function $\mathcal{I}(\mathbf{x}, \mathbf{y})$ that defines the intensity of each pixel (x, y) in the image. As was done in [4, 20, 22, 24], we perform a polar coordinate transformation on $\mathcal{I}(\mathbf{x}, \mathbf{y})$ to obtain its corresponding image $\mathcal{P}(\mathbf{i}, \mathbf{j})$. Then, the doughnut-shaped object in $\mathcal{I}(\mathbf{x}, \mathbf{y})$ corresponds to a “strip” in $\mathcal{P}(\mathbf{i}, \mathbf{j})$ as shown in Figure 2. In this paper, we view $\mathcal{P}(\mathbf{i}, \mathbf{j})$ as the input. Let $\mathcal{P}(\mathbf{i}, \mathbf{j})$ be a 2-D image of size $I \times J$ (i.e., $\mathcal{P}(\mathbf{i}, \mathbf{j}) = \{(i, j) \mid i = 0, 1, \dots, I-1, j = 0, 1, \dots, J-1\}$). We focus on computing an optimal smooth strip in $\mathcal{P}(\mathbf{i}, \mathbf{j})$.

Formally, a 2-D object Q is said to be *stripped* with respect to a line l if for every line l' that is orthogonal to l , the intersection $Q \cap l'$ is a connected component (possibly an empty set). A 2-D object is *x-stripped* if the line l is the x -axis. We define the *thickness* of an x -stripped object Q at $x = x_0$ as the length of the intersection between Q and the line $l : x = x_0$. For an x -stripped object in medical images, we assume its thickness ranges from L to U with $0 < L < U$ (e.g., the wall thickness of vessels changes in a certain range). Roughly speaking,

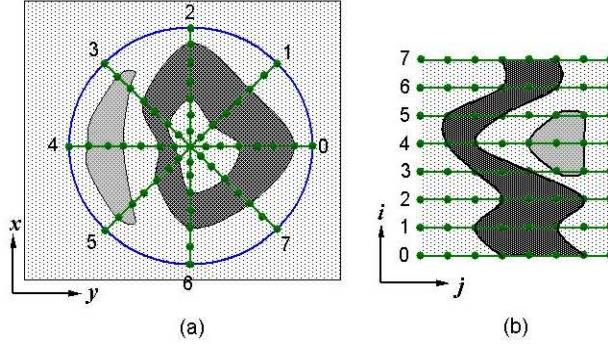


Figure 2: Illustrating the polar transformation on $\mathcal{I}(\mathbf{x}, \mathbf{y})$. (a) A schematic doughnut-shaped object. (b) Transforming the circular region indicated by the circle in (a), including the doughnut-shaped object.

the smoothness constraint means that two distinct pixels (i, j) and (i', j') of a 2-D image can be adjacent to each other on the boundary of a segmented object if the i -th and i' -th rows are neighboring to each other (i.e., $|i - i'| = 1$) and j is “close” enough to j' (i.e., $|j - j'| < M$, where M is an input parameter with $1 \leq M \leq J$). A 2-D image $\mathcal{P}(\mathbf{i}, \mathbf{j})$ can be viewed as representing a setting on a doughnut-shaped object, with the last row of $\mathcal{P}(\mathbf{i}, \mathbf{j})$ being treated as being adjacent to the first row (i.e., $\mathcal{P}(\mathbf{i}, \mathbf{j})$ is “bended” to form a 2-D torus). A smooth strip in $\mathcal{P}(\mathbf{i}, \mathbf{j})$ consists of two non-crossing *smooth contours* C_M ’s (i.e., coupled contours) in such a “torial” image, with each defined as follows:

1. C_M starts at a pixel $(0, j_0)$ in the first row of $\mathcal{P}(\mathbf{i}, \mathbf{j})$, for some $j_0 \in \{0, 1, \dots, J - 1\}$.
2. C_M consists of a sequence of I pixels $(0, j_0), (1, j_1), \dots, (I - 1, j_{I-1})$, one from each row of $\mathcal{P}(\mathbf{i}, \mathbf{j})$, such that for every $k = 0, 1, \dots, I - 1$, $|j_k - j_{(k+1) \bmod I}| < M$ (i.e., C_M satisfies the monotonicity and smoothness constraints).

Note that the contour C_M is really a closed path in the “torial” $\mathcal{P}(\mathbf{i}, \mathbf{j})$ that is monotone and smooth. The boundaries of some medical objects in 2-D images can be modeled as such coupled contours [22, 23, 20, 24, 15], and it is natural that one would like to find the “best” contours (i.e., ones with maximum total likelihood of pixels on the contours) to bound a sought object.

We present an $O(IJU(U - L) \log \frac{J}{U} \log(U - L))$ time algorithm for segmenting a smooth strip in image $\mathcal{P}(\mathbf{i}, \mathbf{j})$. Note that our time bound is independent of the smoothness parameter M , which could be as large as J . Our algorithm improves the straightforward dynamic programming algorithm by a factor of $O(\frac{J(U-L)M^2}{U \log \frac{J}{U} \log(U-L)})$. This segmentation problem is modeled as searching two optimal non-crossing paths in a graph of size $O(IJM)$ that is constructed from

the input image $\mathcal{P}(\mathbf{x}, \mathbf{y})$. Our algorithm is based on an interesting observation which enables us to apply divide-and-conquer strategy and to compute the optimal non-crossing paths in an implicitly represented graph by dynamic programming.

Image segmentation with specific shape constraints arises in various applications. Certain medical image analysis techniques (e.g., cardiac MRI and intravascular ultrasound imaging) are based on segmenting star-shaped and smooth objects [4, 9, 20, 22, 24, 3, 15]. Asano *et al.* [1] presented an $O(I^2 J^2)$ time algorithm for segmenting an x -monotone and connected object in a 2-D image based on optimizing the interclass variance criterion [13] and by using computational geometry techniques. Segmenting star-shaped/stripped/monotone and connected objects (which is seemingly quite restricted) can be used as an important step in image segmentation for more general settings [1, 15]. For instance, the primary difficulty with the active contour models is finding a good starting point for complicated objects; perhaps our algorithms could be used to get an approximation of the boundary and used to initialize the active contour model. In addition, segmentation of monotone and connected objects has been applied to extract optimized 2-D association rules from large databases for data mining and financial applications [10, 17, 25].

2 Detecting Smooth Strips in 2-D Images

This section presents our $O(IJU(U - L) \log \frac{J}{U} \log(U - L))$ time algorithm for segmenting a smooth stripped object in a 2-D medical image. We start with our modeling the segmentation problem as searching optimal two “non-crossing” paths in a graph, and then present our algorithms for the problem.

2.1 The Graph Model of the Problem

Let $G_M = (V, E)$ be a lattice graph, where $V = \{(i, j) \mid 0 \leq i < I, 0 \leq j < J\}$ and M is a given integer with $1 \leq M \leq J$. Each vertex (i, j) of G_M has a real valued weight w_{ij} . We define the M -neighborhood of an vertex $(i, j) \in V$, denoted by $\mathcal{N}_M(i, j)$, as a set of vertices on the same row with distance less than M away from (i, j) , i.e., $\mathcal{N}_M(i, j) = \{(i, k) \mid \max\{0, j - M + 1\} \leq k < \min\{j + M, J\}\}$. For each vertex $(i, j) \in V$, there is a directed edge going from (i, j) to every vertex in $\mathcal{N}_M((i + 1) \bmod I, j)$. Besides these edges, there is no other edge in the graph G_M . We call such a graph an M -smoothness lattice graph (e.g., see Figure 3(a)). Note that G_M is in fact a directed acyclic graph with vertex weights and has I rows and J columns. For a $j \in \{0, 1, \dots, J - 1\}$, let P_j be a path in G_M from the vertex $(0, j)$ to a vertex in $\mathcal{N}_M(I - 1, j)$. Such a path is called a *c-path*. We define the weight of a path P in G_M , $w(P)$, as the total weight of vertices on P , i.e., $\sum_{(i,j) \in P} w_{ij}$. Denote by $P[i]$ the column index of the vertex on path P at the i -th row. Given two integers $0 < L < U < J$, two *c-paths* P_j and $P_{j'}$ ($j < j'$) are called a *dual path* of G_M , denoted by $P(j, j')$, if for any i ($0 \leq i < I$), we have $L \leq P_{j'}[i] - P_j[i] \leq U$ (called *thickness constraint*).

For a dual path $P(j, j')$, if $j < j'$, we call c_path P_j (resp., $P_{j'}$) the *left path* (resp., *right path*) of $P(j, j')$ (e.g., see Figure 3(a)). The weight of the dual path $P(j, j')$ is the sum of the weights of P_j and $P_{j'}$. For any $j = 0, 1, \dots, J-1$, let $P(j, *)$ be a minimum-weight dual path in G_M that starts at the vertex $(0, j)$ (i.e., either the left path or the right path of $P(j, *)$ starts at the vertex $(0, j)$). Our goal is to compute a dual path P^* , whose weight is the minimum among all dual paths in G_M , i.e., $w(P^*) = \min\{w(P(0, *)), w(P(1, *)), \dots, w(P(J-1, *))\}$.

The problem of computing an optimal dual path P^* in G_M is well motivated by the need of detecting the coupled contours of smooth stripped objects in 2-D biomedical images $\mathcal{P}(\mathbf{i}, \mathbf{j})$ (i.e., smooth doughnut-shaped objects in $\mathcal{I}(\mathbf{x}, \mathbf{y})$). We model an input 2-D image $\mathcal{P}(\mathbf{i}, \mathbf{j})$ as a directed acyclic graph $G_M = (V, E)$ with vertex weights, such that each pixel of $\mathcal{P}(\mathbf{i}, \mathbf{j})$ corresponds to a vertex in V , and the edges of E represent the connections among the pixels to form feasible object borders, which, in fact, enforce the monotonicity and smoothness constraints. The weight of a vertex in V is inversely related to the likelihood that it may present on the desired border contour, which is usually determined by using simple low-level image features [24, 22, 20]. Thus, a dual path P^* with minimum total vertex weight in G_M corresponds to the desired coupled borders of a doughnut-shaped object in medical images. Such a path captures both the local and global structures in determining optimal contours in the image.

Chen *et al.* [4] developed an $O(IJ \log J)$ time algorithm for computing an optimal c_path in G_M . Actually, computing an optimal dual path in G_M is to seek two c_paths that satisfy the thickness constraint. One may consider the following greedy algorithm: Compute a minimum-weight c_path P^* in G_M by using Chen *et al.*'s algorithm; and then “remove” P^* from G_M and compute an optimal c_path P'^* in the resulting graph. Unfortunately, this heuristic does not work well since P^* and P'^* may violate the thickness constraint. Thus, we need to consider the left and right paths of a dual path simultaneously, which is the main difficulty in generalizing the algorithm in [4]. Another simple strategy is to consider all possible pairs of vertices $(0, j)$ and $(0, j')$ such that $L \leq |j - j'| \leq U$. For each pair $(0, j)$ and $(0, j')$, we compute a minimum-weight dual path $P^*(j, j')$ in $O(IJ(U - L)M^2)$ time using dynamic programming. Thus, the running time of this algorithm is $O(IJ^2(U - L)^2M^2)$. However, we can do much better. Our algorithm improves this solution by a factor of $O(\frac{JM^2(U-L)}{U \log \frac{J}{U} \log(U-L)})$ time by exploiting the intrinsic structures of dual paths.

2.2 The Structures of Dual Paths

In this section, we explore the structures of dual paths in G_M , which enables us to apply the divide-and-conquer paradigm. To simplify the discussion of dual paths, as in [4], we modify G_M in the following way: Duplicate the first row of G_M , append it after the last row of G_M , let the vertices of the appended row all have a weight zero, and add directed edges from the vertices of the last row of G_M to the vertices of the appended row based on the M -smoothness constraint. We denote the appended row as row I and the modified graph as

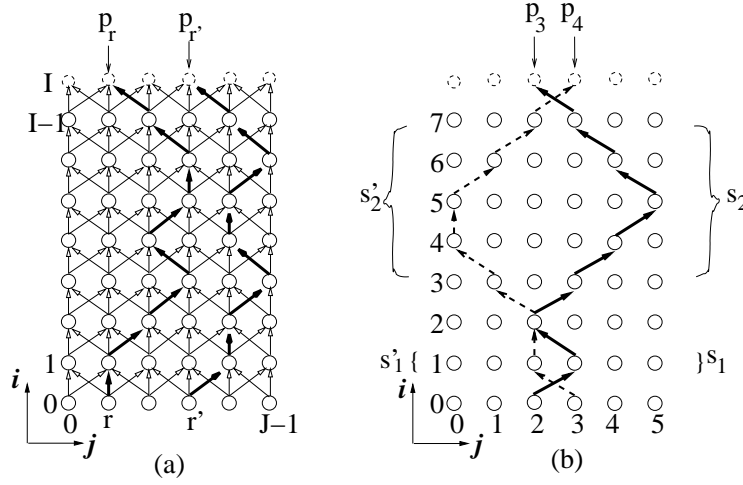


Figure 3: (a) A 2-smoothness lattice graph, in which dual path $P(r, r')$ consisting of two c_paths P_r and $P_{r'}$ is a dual path with $L = 1$ and $U = 3$. (b) Two c_paths crossing each other and their crossing pairs.

G_M^a . A 2-smoothness lattice graph G_M^a is shown in Figure 3(a), where the appended vertices are dashed circles. Note that any dual path $P(j, j')$ in G_M can be viewed as a dual path $P^a(j, j')$ in G_M^a that starts at the vertices $(0, j)$ and $(0, j')$ and ends at the vertices (I, j) and (I, j') , respectively. In Figure 3(a), the dual path $P(r, r')$ consists of two c_paths P_r (i.e., the left path) and $P_{r'}$ (i.e., the right path) indicated by solid thick edges. Henceforth, our focus will be on G_M^a and its dual paths, and we simply denote G_M^a by G_M and its dual paths by $P(j, j')$.

To exploit the intrinsic structures of dual paths, first let us see some useful observations of c_paths. Let P_j and $P_{j'}$ be two c_paths in G_M starting at vertices $(0, j)$ and $(0, j')$, respectively, with $0 \leq j < j' < J$. We say that each vertex $(i, P_j[i])$ on P_j has a *corresponding vertex* $(i, P_{j'}[i])$ on $P_{j'}$ at the i -th row. In a similar way, for each subpath $s = \{(i, P_j[i]), (i + 1, P_j[i + 1]), \dots, (i', P_j[i'])\}$ on P_j , denoted by $P_j[i \cdot i']$, with $0 < i \leq i' < I$, we define its *corresponding subpath* $s' = \{(i, P_{j'}[i]), (i + 1, P_{j'}[i + 1]), \dots, (i', P_{j'}[i'])\}$ on $P_{j'}$, denoted by $P_{j'}[i \cdot i']$. A vertex $(i, P_j[i])$ on P_j is said to be strictly to the *left* (resp., *right*) of $P_{j'}$ if its corresponding vertex $(i, P_{j'}[i])$ on $P_{j'}$ has a larger (resp., smaller) column index, i.e., $P_{j'}[i] > P_j[i]$ (resp., $P_{j'}[i] < P_j[i]$). Two c_paths P_j and $P_{j'}$ are said to *cross each other* if there exists a vertex on P_j being strictly to the right of $P_{j'}$. Given a subpath $s = P_j[i \cdot i']$ on P_j and its corresponding subpath $s' = P_{j'}[i \cdot i']$ on $P_{j'}$, with $0 < i \leq i' < I$, s and s' are said to form a *crossing pair* if $P_j[i - 1] \leq P_{j'}[i - 1], P_j[k] > P_{j'}[k]$ for $k = i, 1, \dots, i'$, and $P_j[i' + 1] \leq P_{j'}[i' + 1]$. If P_j and $P_{j'}$ cross each other, then there certainly exists at least one crossing pair between P_j and $P_{j'}$.

Observation 1 *Let two c-paths P_j and $P_{j'}$ start at vertices $(0, j)$ and $(0, j')$, respectively, with $j < j'$. If P_j and $P_{j'}$ cross each other, then there exists a crossing pair.*

Figure 3(b) illustrates two c-paths P_3 and P_4 crossing each other. For simplicity, we only show the edges on the paths. Therein, the vertex $(0, 2)$ on P_3 is strictly to the left of P_4 and the vertex $(4, 4)$ on P_3 is strictly to the right of P_4 . There are two crossing pairs, (s_1, s'_1) and (s_2, s'_2) , between P_3 and P_4 .

Now, let us consider a minimum-weight dual path $P(r, *)$. Recall that either the left path or the right path of $P(r, *)$ starts at the vertex $(0, r)$. WLOG, we assume that the left path of $P(r, *)$ starts at $(0, r)$ and the right path is $P_{r'}$ with $r < r'$. The next lemma is a key to our algorithm for computing the optimal dual path P^* .

Lemma 1 *Given a minimum-weight dual path $P(r, *)$ in G_M , for any $0 \leq j \leq r' - U$ (resp., $r + U \leq j < J$), there exists an optimal dual path $P(j, *)$ whose right path (resp., left path) does not cross $P_{r'}$ (resp., P_r), where $P_{r'}$ (resp., P_r) is the right path (resp., left path) of $P(r, *)$.*

Proof: We prove the part that for any $0 \leq j \leq r' - U$, there exists a minimum-weight dual path $P(j, *)$ whose right path does not cross $P_{r'}$. The symmetric part can be proved in a similar way.

Suppose that there exists a minimum-weight dual path $P(s, *)$ with $0 \leq s \leq r' - U$, whose right path does cross $P_{r'}$. WLOG, we assume that P_s is the left path and $P_{s'}$ is the right path of $P(s, *)$. Due to the thickness constraint, $s' - s \leq U$. Note that P_r (resp., $P_{r'}$) is the left path (resp., right path) of the optimal dual path $P(r, *)$. Now that $0 \leq s \leq r' - U$, we thus have $s' \leq r'$. Based on the assumption that $P_{s'}$ and $P_{r'}$ cross each other and Observation 1, $P_{s'}$ and $P_{r'}$ have crossing pairs (e.g., see Figure 4(a)). We denote the crossing pairs by $P_{s'}[i'_1 \cdot i'_2]$ and $P_{r'}[i'_1 \cdot i'_2], \dots, P_{s'}[i'_{2b-1} \cdot i'_{2b}]$ and $P_{r'}[i'_{2b-1} \cdot i'_{2b}]$, where $b > 0$. Then, consider the left paths P_s and P_r . Note that P_s and P_r may or may not cross each other. If P_s and P_r cross each other, then we denote the crossing pairs by $P_s[i_1 \cdot i_2]$ and $P_r[i_1 \cdot i_2], \dots, P_s[i_{2d-1} \cdot i_{2d}]$ and $P_r[i_{2d-1} \cdot i_{2d}]$. For simplicity, if $d = 0$, we mean that P_s and P_r do not cross each other.

The following observations are a key. Replacing $P_{s'}[i'_{2k-1} \cdot i'_{2k}]$ by $P_{r'}[i'_{2k-1} \cdot i'_{2k}]$ for all $1 \leq k \leq b$ gives a new c-path $P'_{s'}$; while substituting $P_s[i_{2k-1} \cdot i_{2k}]$ by $P_r[i_{2k-1} \cdot i_{2k}]$ for all $1 \leq k \leq d$ results in another new c-path P'_s in G_M . Further, both paths $P'_{s'}$ and $P_{r'}$ (resp., P'_s and P_r) do not cross each other. Such a replacement is called an *uncrossing operation* (see Figure 4). We need to prove that P'_s and $P'_{s'}$ form a feasible dual path $P'(s, s')$, i.e., P'_s and $P'_{s'}$ are c-paths and meet the thickness constraint.

Claim 1 *Both P'_s and $P'_{s'}$ are c-paths.*

We first show that P'_s is a c-path. It is sufficient to demonstrate that, for each crossing pair $P_s[i_{2k-1} \cdot i_{2k}]$ and $P_r[i_{2k-1} \cdot i_{2k}]$ ($1 \leq k \leq d$), both $(P_s[i_{2k-1} - 1], P_r[i_{2k-1}])$ and $(P_r[i_{2k}], P_s[i_{2k} + 1])$ are an edge in G_M after the uncrossing

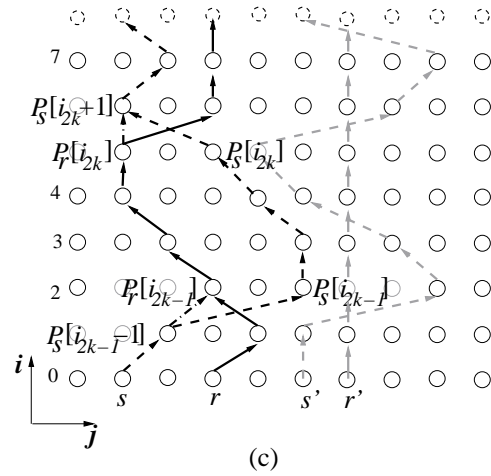
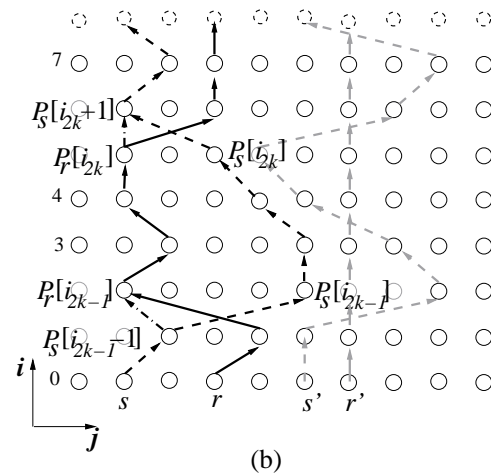
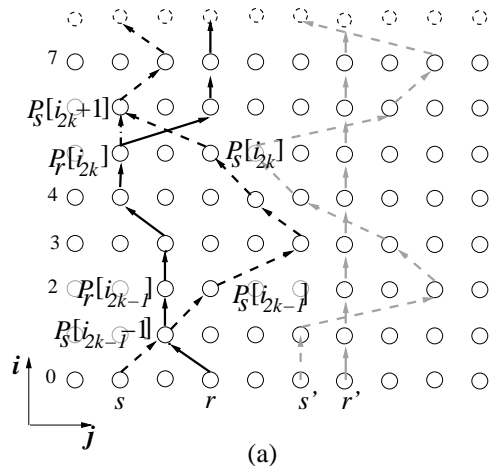


Figure 4: Illustrating the uncrossing operations. The dash-dotted edges are the result of the uncrossing operations.

operation (see Figure 4). Considering $(P_s[i_{2k-1} - 1], P_r[i_{2k-1}])$, we distinguish two cases.

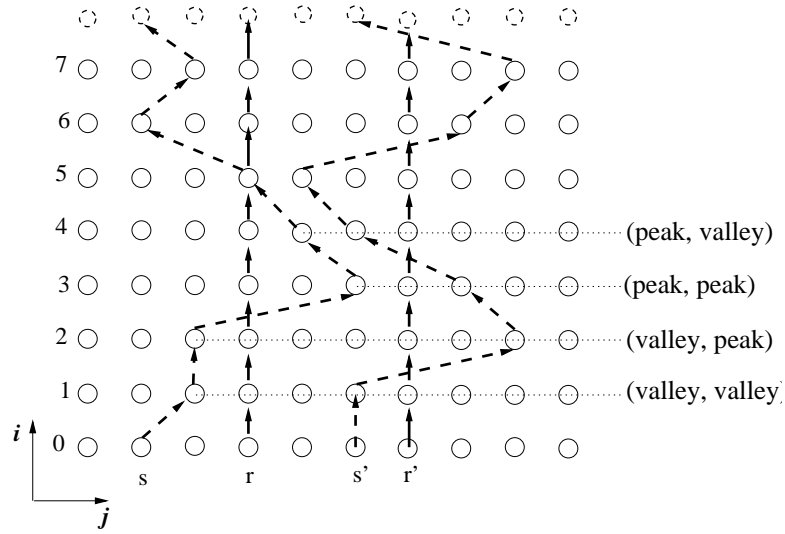
- Case 1: $P_s[i_{2k-1} - 1] = P_r[i_{2k-1} - 1]$ (see Figure 4(a)). Since $(P_r[i_{2k-1} - 1], P_r[i_{2k-1}])$ is an edge on P_r , $(P_s[i_{2k-1} - 1], P_r[i_{2k-1}])$ obviously is an edge in G_M .
- Case 2: $P_s[i_{2k-1} - 1] \neq P_r[i_{2k-1} - 1]$. Note that actually $P_s[i_{2k-1} - 1] < P_r[i_{2k-1} - 1]$. (1) If $P_s[i_{2k-1} - 1] \geq P_r[i_{2k-1}]$ (see Figure 4(b)), then $|P_s[i_{2k-1} - 1] - P_r[i_{2k-1}]| = P_s[i_{2k-1} - 1] - P_r[i_{2k-1}] < P_r[i_{2k-1} - 1] - P_r[i_{2k-1}] = |P_r[i_{2k-1} - 1] - P_r[i_{2k-1}]| < M$. Hence, $(P_s[i_{2k-1} - 1], P_r[i_{2k-1}])$ is an edge in G_M . (2) If $P_s[i_{2k-1} - 1] < P_r[i_{2k-1}]$ (see Figure 4(c)), then $|P_s[i_{2k-1} - 1] - P_r[i_{2k-1}]| = P_r[i_{2k-1}] - P_s[i_{2k-1} - 1] < P_s[i_{2k-1}] - P_s[i_{2k-1} - 1] = |P_s[i_{2k-1}] - P_s[i_{2k-1} - 1]| < M$. Thus, $(P_s[i_{2k-1} - 1], P_r[i_{2k-1}])$ is an edge in G_M .

Similarly, we can show that $(P_r[i_{2k}], P_s[i_{2k} + 1])$ is an edge of G_M . Hence, P'_s is a c-path in G_M . Using the same argument, $P'_{s'}$ can be shown to be a c-path. \square

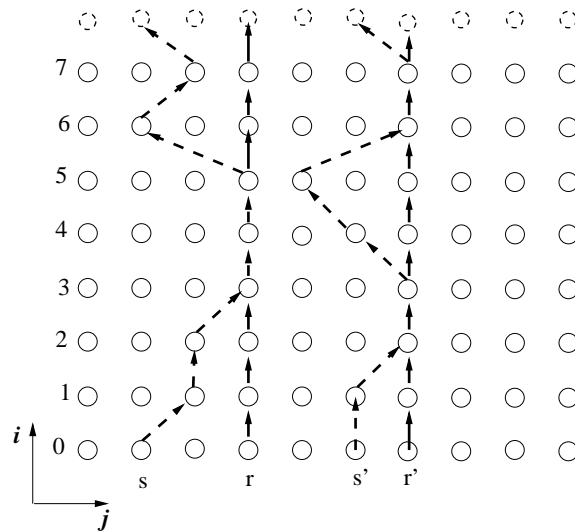
Claim 2 P'_s and $P'_{s'}$ satisfy the thickness constraint.

For any $1 \leq k \leq b$, we call $P_{s'}[i'_{2k-1} \cdot i'_{2k}]$ a *peak* while $P_{s'}[i'_{2k} + 1 \cdot i'_{2k+1} - 1]$ a *valley* of $P_{s'}$ with respect to $P_{r'}$, where $i'_{2b+1} = I$. Note that the column index of the vertex on $P_{s'}$ at row I (i.e., $P_{s'}(I)$) equals to $P_{s'}[0]$ (i.e., s') and $s' \leq r'$. We thus also say $P_{s'}[0 \cdot i'_1]$ is a valley of $P_{s'}$ with respect to $P_{r'}$. Similarly, if P_s and P_r cross each other (i.e., $d > 0$), $P_s[i_{2k-1} \cdot i_{2k}]$ ($1 \leq k \leq d$) are called peaks of P_s with respect to P_r , and $P_s[0 \cdot i_1]$ and $P_s[i_{2k} \cdot i_{2k+1}]$ ($1 \leq k \leq d$ and $i_{2d+1} = I$) are called valleys of P_s ; otherwise, we say the whole P_s is a valley with respect to P_r (note that $s \leq r$). In addition, if a vertex is on the peak (resp., valley) of $P_{s'}$ or P_s , we call it a *peak vertex* (resp., *valley vertex*) of the corresponding c-path. By performing the uncrossing operations, for each peak vertex of $P_{s'}$ (resp., P_s), its corresponding vertex on $P_{r'}$ (resp., P_r) is on the resulting c-path $P'_{s'}$ (resp., P'_s). Note that any vertex of $P_{s'}$ and P_s can be either a peak or a valley vertex. Hence, the vertex pair $((i, P_s[i]), (i, P_{s'}[i]))$ of $P(s, s')$ has four possible patterns: *(peak, peak)*, *(peak, valley)*, *(valley, peak)*, and *(valley, valley)* as illustrated in Figure 5(a). For each case, we can show that $L \leq P'_{s'}[i] - P'_s[i] \leq U$, as follows.

- Case 1: The vertex pair $((i, P_{s'}[i]), (i, P_s[i]))$ is of pattern *(peak, peak)*. $P'_{s'}[i] = P_{r'}[i]$ and $P'_s[i] = P_r[i]$. Since the dual path $P(r, *)$ satisfies the thickness constraint, we have $L \leq P'_{s'}[i] - P'_s[i] \leq U$.
- Case 2: The vertex pair $((i, P_{s'}[i]), (i, P_s[i]))$ is of pattern *(peak, valley)*. $P'_{s'}[i] = P_{r'}[i]$ and $P'_s[i] = P_s[i]$. Now that $(i, P_{s'}[i])$ is a peak vertex of $P_{s'}$ with respect to $P_{r'}$, we have $P_{s'}[i] > P_{r'}[i]$; while $(i, P_s[i])$ is a valley vertex of P_s with respect to P_r , hence, $P_s[i] \leq P_r[i]$. We thus have $P'_{s'}[i] - P'_s[i] = P_{r'}[i] - P_s[i] \geq P_{r'}[i] - P_r[i] \geq L$ and $P'_{s'}[i] - P'_s[i] = P_{r'}[i] - P_s[i] \leq P_{s'}[i] - P_s[i] \leq U$. Hence, $L \leq P'_{s'}[i] - P'_s[i] \leq U$.



(a)



(b)

Figure 5: Illustrating the proof of Lemma 1. The optimal dual path $P(r, *)$ is indicated by solid edges, while the optimal dual path $P(s, *)$ is indicated by dashed edges. (a) The left path P_s of $P(s, *)$ crosses the left path P_r of $P(r, *)$; the right path $P_{s'}$ of $P(s, *)$ crosses the right path $P_{r'}$ of $P(r, *)$. (b) Performing uncrossing operations on $P(s, *)$ in a) obtains another minimum-weight dual path starting at vertex $(0, s)$ such that its left and right paths do not cross the left and right paths of $P(r, *)$, respectively.

- Case 3: The vertex pair $((i, P_{s'}[i]), (i, P_s[i]))$ is of pattern (*valley, peak*).
 $P'_{s'}[i] = P_{s'}[i]$ and $P'_s[i] = P_r[i]$. Since $(i, P_{s'}[i])$ is a valley vertex of $P_{s'}$ with respect to $P_{r'}$, we have $P_{s'}[i] \leq P_{r'}[i]$. Similarly, considering vertex $(i, P_s[i])$, we have $P_s[i] > P_r[i]$. Thus, $P'_{s'}[i] - P'_s[i] = P_{s'}[i] - P_r[i] \geq P_{s'}[i] - P_s[i] \geq L$ and $P'_{s'}[i] - P'_s[i] = P_{s'}[i] - P_r[i] \leq P_{r'}[i] - P_r[i] \leq U$. Hence, $L \leq P'_{s'}[i] - P'_s[i] \leq U$.
- Case 4: The vertex pair $((i, P_{s'}[i]), (i, P_s[i]))$ is of pattern (*valley, valley*).
 $P'_{s'}[i] = P_{s'}[i]$ and $P'_s[i] = P_s[i]$. Since the dual path $P(s, *)$ satisfies the thickness constraint, we have $L \leq P'_{s'}[i] - P'_s[i] \leq U$.

Symmetrically, by replacing $P_{r'}[i'_{k-1} \dots i'_k]$ (resp., $P_r[i_{k-1} \dots i_k]$) by $P_{s'}[i'_{k-1} \dots i'_k]$ (resp., $P_s[i_{k-1} \dots i_k]$) for all $1 \leq k \leq b$ (resp., $1 \leq k \leq d$), we obtain a new c.path $P'_{r'}$ (resp., P'_r) such that $P'_{r'}$ and $P_{s'}$ (resp., P'_s and P_r) do not cross each other. In a similar way, we can show that the resulting c-paths P'_r and $P'_{r'}$ are a feasible dual path, denoted by $P'(r, r')$, which starts at vertices $(0, r)$ and $(0, r')$.

Claim 3 $w(P(s, s')) = w(P'(s, s'))$.

We next need to show that the total weight of all peaks of $P_{s'}$ and P_s equals to that of their corresponding subpaths on $P_{r'}$ and P_r , that is,

$$\begin{aligned} & \sum_{k=1}^b w(P_{s'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_s[i_{2k-1} \dots i_{2k}]) \\ &= \sum_{k=1}^b w(P_{r'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_r[i_{2k-1} \dots i_{2k}]). \end{aligned}$$

Note that unlike [4], the weight of an individual peak may not be equal to that of its corresponding subpath. We claim that

$$\begin{aligned} & \sum_{k=1}^b w(P_{s'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_s[i_{2k-1} \dots i_{2k}]) \\ &\leq \sum_{k=1}^b w(P_{r'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_r[i_{2k-1} \dots i_{2k}]). \end{aligned}$$

Otherwise, we perform uncrossing operations on $P(s, *)$ and $P(r, *)$ to obtain a feasible dual path $P'(s, s')$, as we have shown above. Notice that

$$\begin{aligned} & \sum_{k=1}^b w(P_{s'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_s[i_{2k-1} \dots i_{2k}]) \\ &> \sum_{k=1}^b w(P_{r'}[i'_{2k-1} \dots i'_{2k}]) + \sum_{k=1}^d w(P_r[i_{2k-1} \dots i_{2k}]). \end{aligned}$$

We thus have $w(P'(s, s')) < w(P(s, *))$, which is a contradiction to the optimality of $P(s, *)$. Hence,

$$\begin{aligned} & \sum_{k=1}^b w(P_{s'}[i'_{2k-1} \cdots i'_{2k}]) + \sum_{k=1}^d w(P_s[i_{2k-1} \cdots i_{2k}]) \\ \leq & \sum_{k=1}^b w(P_{r'}[i'_{2k-1} \cdots i'_{2k}]) + \sum_{k=1}^d w(P_r[i_{2k-1} \cdots i_{2k}]). \end{aligned} \quad (1)$$

In a similar way, by performing uncrossing operations on $P(r, *)$ and $P(s, *)$ to obtain a feasible dual path $P'(r, r')$, we can also show that

$$\begin{aligned} & \sum_{k=1}^b w(P_{s'}[i'_{2k-1} \cdots i'_{2k}]) + \sum_{k=1}^d w(P_s[i_{2k-1} \cdots i_{2k}]) \\ \geq & \sum_{k=1}^b w(P_{r'}[i'_{2k-1} \cdots i'_{2k}]) + \sum_{k=1}^d w(P_r[i_{2k-1} \cdots i_{2k}]). \end{aligned} \quad (2)$$

Hence, from equations (1) and (2), we have

$$\sum_{k=1}^b w(P_{s'}[i'_{k-1} \cdots i'_k]) + \sum_{k=1}^d w(P_s[i_{k-1} \cdots i_k]) = \sum_{k=1}^b w(P_{r'}[i'_{k-1} \cdots i'_k]) + \sum_{k=1}^d w(P_r[i_{k-1} \cdots i_k]),$$

and further, $w(P(s, s')) = w(P'(s, s'))$.

Thus, for any $0 \leq j \leq r' - U$, there exists a minimum-weight dual path $P(j, *)$ whose right path does not go across $P_{r'}$. The symmetric part that, for any $r + U \leq j < J$, there exists an optimal dual path $P(j, *)$ whose left path does not cross P_r , can be proved by using a similar argument. Therefore, the lemma holds. \square

Lemma 1 provides a basis for a divide-and-conquer solution for computing the optimal dual path in G_M . Given an optimal dual path $P(r, *)$ consisting of two c-paths P_r and $P_{r'}$ with $r < r'$, we can decompose G_M into two “smaller” subgraphs along $P(r, *)$, and then compute the optimal dual paths in such “smaller” graphs. Before going into details on the decomposition of G_M , we first present our algorithm for computing an optimal dual path $P(r, *)$ in the following section.

2.3 Computing Optimal Dual Path $P(r, *)$

This section shows how to efficiently compute a minimum-weight dual path in G_M , say, $P(r, *)$ that starts at the vertex $(0, r)$ for any $r \in \{0, 1, \dots, J-1\}$. Due to the thickness constraint, the possible vertices that the other c-path in $P(r, *)$ may start at are only a subset of vertices on row 0 whose column indices are in $S_1 = \{r + L \leq k \leq \min\{r + U, J - 1\}\} \cup S_2 = \{\min\{r - U, 0\} \leq k \leq r - L\}$. Of

course, the optimal dual path $P(r, *)$ can be obtained by computing minimum-weight dual paths $P^*(r, k)$ for all $k \in S_1$ and $P^*(k, r)$ for all $k \in S_2$. However, we can do better by judiciously explore the structures of $P(r, *)$.

Given two c-paths, P_j and $P_{j'}$, we say P_j is to the *left* (resp., *right*) of $P_{j'}$ if for any $0 \leq i \leq I$, $P_j[i] \leq P_{j'}[i]$ (resp., $P_j[i] \geq P_{j'}[i]$). The following lemma makes possible to apply the divide-and-conquer strategy to compute $P(r, *)$.

Lemma 2 (1) *Given an optimal dual path $P^*(r, u)$ ($u \in S_1$) whose left path is P_r and right path is P_u , for any $k \in S_1$ and $k > u$ (resp., $k < u$), there exists a minimum-weight dual path $P^*(r, k)$ such that its right path P'_k and left path P'_r are to the right (resp., left) of P_u and P_r , respectively.*

(2) *Given an optimal dual path $P^*(u, r)$ ($u \in S_2$) whose left path is P_u and right path is P_r , for any $k \in S_2$ and $k > u$ (resp., $k < u$), there exists a minimum-weight dual path $P^*(k, r)$ such that its left path P'_k and right path P'_r are to the right (resp., left) of P_u and P_r , respectively.*

Proof: The lemma follows by a similar argument for proving Lemma 1. \square

We compute the optimal dual paths $P^*(r, k)$ for every $k \in S_1$, as follows. First, the minimum-weight dual paths $P^*(r, r+L)$ and $P^*(r, \min\{r+U, J-1\})$ are computed (see Section 2.4). Denote by LL and LR the left and right paths of $P^*(r, r+L)$, respectively; while the left and right paths of $P^*(r, \min\{r+U, J-1\})$ are respectively denoted by RL and RR . For any $k \in S_1$, based on Lemma 2, the left path P of $P^*(r, k)$ is bounded by LL and RL (i.e., $LL[i] \leq P[i] \leq RL[i]$ for each row i) and the right path of $P^*(r, k)$ is bounded by LR and RR .

Let u be the median of S_1 (i.e., $u = \left\lceil \frac{(r+L) + \min\{r+U, J-1\}}{2} \right\rceil$). The minimum-weight dual path $P^*(r, u)$ consisting of c-paths P_r and P_u , is then computed. Using $P^*(r, u)$, we define four sets $J_i^L = \{LL[i], LL[i] + 1, \dots, P_r[i]\}$, $J_i^R = \{P_r[i], P_r[i] + 1, \dots, RL[i]\}$, $J_i^L = \{LR[i], RL[i] + 1, \dots, P_u[i]\}$, and $J_i^R = \{P_u[i], P_u[i] + 1, \dots, RR[i]\}$, for every $i = 0, 1, \dots, I$. Then, along each c-path of the dual path $P^*(r, u)$, we decompose the graph G_M into two subgraphs. $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are obtained by decomposing G_M along P_r , where $V_1 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^L\}$, $E_1 = \{e \in E \mid \text{both vertices of } e \text{ are in } V_1\}$, $V_2 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^R\}$, and $E_2 = \{e \in E \mid \text{both vertices of } e \text{ are in } V_2\}$; $G'_1 = (V'_1, E'_1)$ and $G'_2 = (V'_2, E'_2)$ are obtained by decomposing G_M along P_u , where $V'_1 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^L\}$, $E'_1 = \{e \in E \mid \text{both vertices of } e \text{ are in } V'_1\}$, $V'_2 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^R\}$, and $E'_2 = \{e \in E \mid \text{both vertices of } e \text{ are in } V'_2\}$. Based on Lemma 2, for any $k \in S_1$ and $k < u$ (resp., $k > u$), there exists a minimum-weight dual path $P^*(r, k)$ in G_M such that its right path lies in G'_1 (resp., G'_2) and its left path lies in G_1 (resp., G_2). Therefore, we recursively compute optimal dual paths $P^*(r, k)$ for $k \in S_1$ and $k < u$ (resp., $k > u$) in G_1 and G'_1 (resp., G_2 and G'_2). Clearly, the recursion tree of our above divide-and-conquer algorithm has $O(\log(U-L))$ levels; at each level, a subset of dual paths $P^*(r, k)$ is computed (in certain subgraphs of G_M).

Similarly, we can compute the minimum-weight dual path $P^*(k, r)$ for every $k \in S_2$. Thus, the following lemma holds.

Lemma 3 *For any given r ($r \in \{0, 1, \dots, J - 1\}$), the minimum-weight dual path $P(r, *)$ can be computed in $O(T \log(U - L))$ time, where T is the time for computing an optimal dual path $P^*(j, j')$ in G_M whose c -paths start at vertices $(0, j)$ and $(0, j')$.*

2.4 Computing Minimum-Weight Dual Path $P^*(r, r')$

In this section, we present our efficient algorithm for computing an optimal dual path $P^*(r, r')$ whose left and right paths start at vertices $(0, r)$ and $(0, r')$, respectively.

We begin with a less efficient dynamic programming algorithm for computing $P^*(r, r')$ in G_M . First, note that the edges of G_M can be represented *implicitly*. That is, without explicitly storing its edges, we can determine for every vertex of G_M the set of its incoming and outgoing neighbors in $O(1)$ time. Our algorithm uses this implicit representation of G_M . To help our presentation, we say two paths in G_M to be a *twin path* if they start at two vertices of row 0 and satisfy the thickness constraint. The weight of a twin path is the total weight of vertices on both paths. We denote by $m_i[j, k]$ the weight of the optimal twin path in G_M starting from the vertices $(0, r)$ and $(0, r')$ to vertices (i, j) and (i, k) , respectively. Due to the smoothness constraint, vertex (i, j) can be reached from any vertex of row $i - 1$ in $\{(i - 1, j') \mid \max\{0, j - M + 1\} \leq j' \leq \min\{J - 1, j + M - 1\}\}$; while vertex (i, k) can be reached from any vertex of row $i - 1$ in $\{(i - 1, k') \mid \max\{0, k - M + 1\} \leq k' \leq \min\{J - 1, k + M - 1\}\}$. But, the thickness constraint restricts our choices of the pair of vertices on row $i - 1$. Actually, for any j' such that $\max\{0, j - M + 1\} \leq j' \leq \min\{J - 1, j + M - 1\}$, we have $\max\{j' + L, k - M + 1\} \leq k' \leq \min\{j' + U, k + M - 1\}$. Hence,

$$m_i[j, k] = \min_{j'=\max\{0, j-M+1\}}^{\min\{J-1, j+M-1\}} \min_{k'=\max\{j'+L, k-M+1\}}^{\min\{j'+U, k+M-1\}} m_{i-1}[j', k'] + w(i, j) + w(i, k), \quad (*)$$

when $i > 0$ and $L \leq k - j \leq U$. Initially, $m_0[r, r'] = w(0, r) + w(0, r')$ and $m_0[j, k] = \infty$ if $j \neq r$ or $k \neq r'$. In addition, we use table $c_i[j, k]$ to keep track of the optimal twin paths, i.e., if the optimal twin path from $(0, r)$ and $(0, r')$ to (i, j) and (i, k) is via $(i - 1, j')$ and $(i - 1, k')$ on row $i - 1$, then $c_i[j, k] = (j', k')$. One can certainly apply a dynamic programming technique to compute the minimum-weight path $P^*(r, r')$. In fact, $m_I[r, r']$ is the weight of $P^*(r, r')$. Then, the real dual path $P^*(r, r')$ can be reconstructed by using table $c_i[j, k]$. For each possible pair of j and k , we need to compute the minimum of $O(M^2)$ values; and there are $O(J(U - L))$ such pairs on each row i . Hence, a straightforward dynamic programming algorithm takes $O(IJ(U - L)M^2)$ time to compute $P^*(r, r')$.

Interestingly, we are able to extend the technique developed in [4] to eliminate the M^2 factor for the time complexity.

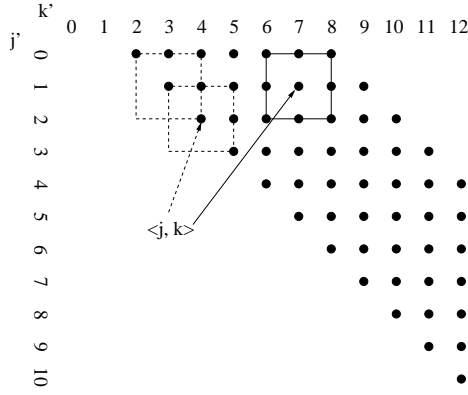


Figure 6: Incrementally computing the minimum of elements in m_{i-1} that are covered by a rectangle of size of $(2M - 1) \times (2M - 1)$. Herein, $M = 2$, $L = 2$, and $U = 8$

Suppose that all the optimal twin paths to vertices on row $i - 1$ have been computed and the weights of these paths, $m_{i-1}[j', k']$ for $0 \leq j' < J$ and $j + L \leq k' \leq \min\{J - 1, j + U\}$, are stored (e.g., see Figure 6). Based on equation (*), in order to compute $m_i[j, k]$, we need to know the minimum of $m_{i-1}[j', k']$'s for $j - M + 1 \leq j' \leq j + M - 1$ and $k - M + 1 \leq k' \leq k + M - 1$, which defines a rectangular region of size $(2M - 1) \times (2M - 1)$ in m_{i-1} . The center of the rectangle corresponds to the column index pair $\langle j, k \rangle$. Note that some pairs of $\langle j', k' \rangle$ may not correspond to a twin path. We may view $m_{i-1}(j', k')$ as ∞ for those index pairs. In Figure 6, the dots indicate the index pairs $\langle j', k' \rangle$ that correspond to a twin path in G_M . Thus, for each pair $\langle j, k \rangle$, we only need to compute the minimum of elements in m_{i-1} that are covered by the rectangle R centered at $\langle j, k \rangle$ with size of $(2M - 1) \times (2M - 1)$. Note that while moving the center of R from $\langle j, k \rangle$ to $\langle j, k + 1 \rangle$ or to $\langle j + 1, k \rangle$, only $O(M)$ elements in R are changed. Thus, one may maintain a priority queue to compute the minimum of elements in R . In this way, computing the minimum for an index pair $\langle j, k \rangle$ takes $O(M \log M)$ time. However, we can compute the minima for all $(J(U - L + 1)) \langle j, k \rangle$ pairs in $O(J(U - L))$ time.

Given an array A of n real numbers and an integer M with $1 \leq M \leq n$, the *min- M -neighbor* of $A[i]$ is defined as $\min\{A[k] \mid \max\{0, i - M + 1\} \leq k \leq \min\{n - 1, i + M - 1\}\}$. Chen *et al.* [4] developed a simple linear time algorithm for computing the min- M -neighbors for all elements in A . We next apply their technique to compute the minima for all $\langle j, k \rangle$ pairs, as follows. For each row $m_{i-1}[j']$ of m_{i-1} , compute the min- M -neighbor for every element in $m_{i-1}[j']$ in $O(U - L)$ time, since each row has at most $(U - L + 1)$ elements. The resulting min- M -neighbors are kept in another 2-D array m'_{i-1} . Then, for each column $m'_{i-1}[k']$ of m'_{i-1} , compute the min- M -neighbor for every element in $m'_{i-1}[k']$ in $O(U - L)$ time, since each column has at most $(U - L + 1)$ elements. Note that the min- M -neighbor of the j' -th element in $m'_{i-1}[k']$ equals to the minimum of

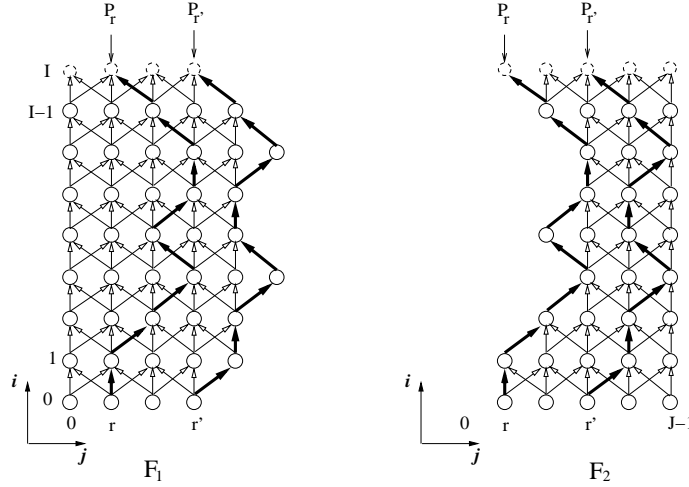


Figure 7: Illustrating the divide-and-conquer algorithm for computing the optimal dual paths $P(j, *)$.

elements in m_{i-1} that are covered by the rectangle R of size $(2M - 1) \times (2M - 1)$ when centered at $\langle j', k' \rangle$. Therefore, the minima for all $\langle j, k \rangle$ pairs can be computed in $O(J(U - L))$ time. Thus, Lemma 4 follows.

Lemma 4 *The minimum-weight dual path $P^*(r, r')$ can be computed in $O(IJ(U - L))$ time.*

Together with Lemma 3, we have the following lemma.

Lemma 5 *For any given r ($r \in \{0, 1, \dots, J - 1\}$), the minimum-weight dual path $P(r, *)$ can be computed in $O(IJ(U - L) \log(U - L))$ time.*

2.5 Our Algorithm

Now, we are ready to present our $O(IJU(U - L) \log \frac{J}{U} \log(U - L))$ -time algorithm for computing an optimal dual path P^* in G_M .

Note that the optimal dual path P^* of G_M can be obtained from $P(0, *)$, $P(1, *)$, \dots , $P(J - 1, *)$. To compute all dual paths $P(0, *)$, $P(1, *)$, \dots , $P(J - 1, *)$ in G_M , we first compute the minimum-weight dual path $P(\lceil \frac{J-1}{2} \rceil, *)$ using our algorithm in Sections 2.3 and 2.4. Assume the left path of $P(\lceil \frac{J-1}{2} \rceil, *)$ is P_r and the right path is $P_{r'}$ (note that either r or r' equals to $\lceil \frac{J-1}{2} \rceil$). Using $P(\lceil \frac{J-1}{2} \rceil, *)$, we define two sets $J_i^L = \{0, 1, \dots, P_r[i]\}$ and $J_i^R = \{P_r[i], P_r[i] + 1, \dots, J - 1\}$, for every $i = 0, 1, \dots, I$. Then along the dual path $P(r, *)$, we decompose the graph G_M into two subgraphs $F_1 = (V_1, E_1)$ and $F_2 = (V_2, E_2)$, where $V_1 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^L\}$, $E_1 = \{e \in E \mid \text{both vertices of } e \text{ are in } V_1\}$, $V_2 = \{(i, j) \mid i \in \{0, 1, \dots, I\}, j \in J_i^R\}$, and $E_2 = \{e \in E \mid \text{both}$

vertices of e are in V_2 }. Figure 7 illustrates the decomposition of the graph G_M into two subgraphs F_1 and F_2 along the dual path $P(r, *)$. Based on Lemma 1, for any $0 \leq j \leq r' - U$, there exists a minimum-weight dual path $P(j, *)$ of G_M in F_1 , and for any $r + U \leq j < J$, there exists a minimum-weight dual path $P(j, *)$ of G_M in F_2 . Hence, we recursively compute $P(j, *)$ for $0 \leq j \leq r' - U$ and for $r + U \leq j < J$ in F_1 and F_2 , respectively. However, for every j such that $r' - U < j < r + U$, the optimal dual path $P(j, *)$ may be neither in F_1 nor F_2 (simply performing uncrossing operations does not work well; the resulting two c_paths may violate the thickness constraint). Thus, we compute every minimum-weight dual path $P(j, *)$ for $r' - U < j < r + U$ in G_M using our algorithm in Sections 2.3 and 2.4. Since there are $O(U)$ such j 's, based on Lemma 5, the running time is $O(IJU(U - L)\log(U - L))$. Clearly, the recursion tree of our above divide-and-conquer algorithm has $O(\log \frac{J}{U})$ levels. At each recursion level k , the total size of the vertex sets of all the subgraphs is bounded by $O(IJ + 2^k IU)$. Thus, the total running time of the recursion level k is $O(I(J + 2^k U)U(U - L)\log(U - L))$. Hence, the total time of the overall divide-and-conquer algorithm is $O(IJU(U - L)\log \frac{J}{U} \log(U - L))$.

Theorem 1 *Given an implicitly represented M -smoothness lattice graph G_M , a minimum-weight dual path P^* in G_M can be computed in $O(IJU(U - L)\log \frac{J}{U} \log(U - L))$ time.*

3 Implementation and Experiments

To further study the behavior and performance of our dual path algorithm, we have implemented it using C++. Our implementation is based on the algorithm described in Section 2. The acceleration technique for the dynamic programming algorithm described in Section 2.4 has not been implemented in the current software. The algorithm has been implemented from scratch only utilizing the Boost C++ libraries¹ and the Blitz++² matrix library.

After the implementation, our algorithm/program was tested on a Dell XPS/Dimension 9150 with 2GB memory and 2.80GHz Intel Pentium-D CPU. We conducted preliminary tests on 82 manual tracings of stereo photographs of the optic nerve head. The segmented regions represented the rim and cup of the optic disc. The thickness constraints, L and U , and smoothness parameter, M , were selected manually based on empirical evidence. Some example results are demonstrated in Figure 8. Our experiments showed that the execution times of our dual path algorithm with realistic parameters are very fast, all under a few minutes for a typical 256×256 image. This is significantly faster than the traditional dynamic programming algorithm without the divide-and-conquer improvements (see Tables 1 and 2).

¹Boost C++ is a growing set of libraries that emphasizes compliance with the C++ Standard Library and can be found at <http://www.boost.org>.

²Blitz++ is a fast matrix library for C++ and can be found at <http://www.oonumerics.org/blitz>.

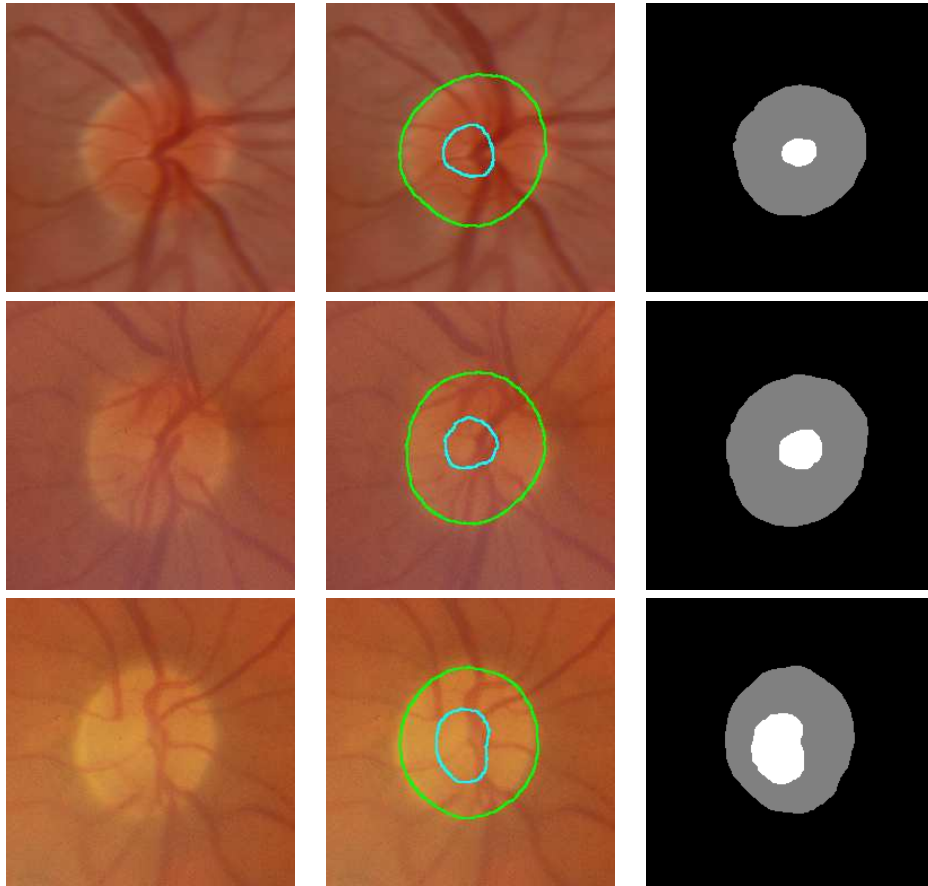


Figure 8: Example results for three different datasets. The left column shows the original images (the left images of the stereo pairs), the middle column is our segmentation results, and the right column shows the results manually traced by human experts.

U	Brute Force (s)	Dual Path (s)
2	5.915	2.684
4	28.235	10.190
8	127.417	35.951
16	497.983	91.359
32	1753.872	250.476
64		341.838
128		345.120

Table 1: Algorithm performance of varying thickness constraints. 128×128 vertex graph, $M = 3$, $L = 1$.

Graph Size	Brute Force (s)	Dual Path (s)
32×32	5.126	1.839
64×64	53.612	14.707
128×128	497.983	91.359
256×256	4241.395	599.940
32×64	10.962	3.298
32×128	22.623	6.121
32×256	45.680	11.836
32×512	92.765	23.643
64×32	21.624	6.837
128×32	59.593	17.615
256×32	130.465	56.021
512×32	272.332	98.308

Table 2: Algorithm performance of varying image sizes. $M = 3$, $L = 1$, $U = 16$.

References

- [1] T. Asano, D.Z. Chen, N. Katoh, and T. Tokuyama, Efficient algorithms for optimization-based image segmentation, accepted to *International Journal of Computational Geometry and Applications*.
- [2] P.J. Besl and R.C. Jain, Segmentation through variable-order surface fitting, *IEEE Trans. Pattern Anal. and Machine Intell.*, 10 (1988), pp. 167–192.
- [3] J. F. Brinkley, A flexible, generic model for anatomic shape: Application to interactive two-dimensional medical image segmentation and matching, *Computers and Biomedical Research*, 26 (1993), pp. 121–142.
- [4] D.Z. Chen, J. Wang, and X. Wu, Image Segmentation with Asteroidality/Tubularity and Smoothness Constraints, *International Journal of Computational Geometry & Applications*, 12(5)(2002), pp. 413-428.
- [5] L.D. Cohen, On Active Contour Models and Balloons, *CVGIP – Image Understanding*, 53 (2) (1991), pp. 211-218.
- [6] T.F. Cootes, D.H. Cooper, C.J. Taylor, and J. Graham, Trainable Method of Parametric Shape Description, *Image Vision Computing*, 10 (5) (1992), pp. 289-294.
- [7] T.F. Cootes, G.J. Edwards, and C. Taylor, Active Appearance Models, *IEEE Trans. Pattern Anal. and Machine Intell.*, 23 (2001), pp. 681-685.
- [8] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, Active Shape Models – Their Training and Application, *Computer Vision and Image Understanding*, 61 (1995), pp. 38-59.
- [9] R.J. Frank, D.D. McPherson, K.B. Chandran, and E.L. Dove, Optimal surface detection in intravascular ultrasound using multi-dimensional graph search, *Computers in Cardiology*, IEEE, Los Alamitos, CA, 1996, pp. 45–48.
- [10] T. Fukuda, S. Morishita, Y. Morimoto, and T. Tokuyama, Data mining using two-dimensional optimized association rules – Scheme, algorithms, and visualization, *Proc. SIGMOD Int. Conf. on Management of Data*, 1996, pp. 13-23.
- [11] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY, 1979.
- [12] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, Cortex Segmentation: A Fast Variational Geometric Approach, *IEEE Trans. on Med. Imag.*, 21(2002), pp. 1544-1551.
- [13] D.J. Hand, *Discrimination and Classification*, John Wiley & Sons, 1981.

- X. Wu, *Doughnut-Shaped Object Segmentation*, JGAA, 11(1) 215–237 (2007)236
- [14] G.T. Herman and B.M. Carvalho, Multiseeded Segmentation Using Fuzzy Connectedness, *IEEE Trans. Pattern Anal. and Machine Intell.*, 23 (5) (2001), 460–474.
- [15] K. P. Hinshaw and J. F. Brinkley, Shape-based interactive three-dimensional medical image segmentation, *SPIE Medical Imaging: Image Processing*, Volume 3034 K. M. Hanson, Ed. Newport Beach, CA, 1996, pp. 236–242.
- [16] M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active Contour Models, *Int. J. Comput. Vision*, 1(4)(1988), pp. 321-331.
- [17] Y. Morimoto, T. Fukuda, H. Matsuzawa, T. Tokuyama, and K. Yoda, Multivariate rules in data mining: A key to handling correlations in financial data, *Proc. KDD Workshop in Finance*, AAAI, 1998, pp. 54-59.
- [18] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, Cambridge, UK 2001.
- [19] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, Cambridge, UK, 2002.
- [20] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd edition, Brooks/Cole Publishing Company, Pacific Grove, CA, 1999, pp. 199–205.
- [21] M. Sonka, M.D. Winniford, and S.M. Collins, Robust Simultaneous Detection of Coronary Borders in Complex Images, *IEEE Trans. on Medical Imaging*, 14(1)(1995), pp. 151-161.
- [22] M. Sonka, X. Zhang, M. Siebs, M. S. Bissing, S. DeJong, S. M. Collins, and C. R. McKay, Segmentation of Intravascular Ultrasound Images: A Knowledge-Based Approach, *IEEE Trans. on Medical Imaging*, 14(4)(1995), pp. 719-732.
- [23] L. Spreeuwens and M. Breeuwer, Detection of Left Ventricular Epi- and Endocardial Borders Using Coupled Active Contours, *Computer Assisted Radiology and Surgery*, 2003, pp. 1147-1152.
- [24] D.R. Thedens, D.J. Skorton, and S.R. Fleagle, Methods of graph searching for border detection in image sequences with applications to cardiac magnetic resonance imaging, *IEEE Trans. on Medical Imaging*, 14 (1) (1995), pp. 42–55.
- [25] T. Tokuyama, Application of algorithm theory to data mining, *Proc. of Australasian Computer Science Conf.*, 1998, pp. 1-15. Also in *Proc. CATS98*.

X. Wu, *Doughnut-Shaped Object Segmentation*, *JGAA*, 11(1) 215–237 (2007)237

- [26] J.K. Udupa and S. Samarasekera, Fuzzy Connectedness and Object Definition: Theory, Algorithms, and Applications in Image Segmentation, *Graphics Models and Image Processing*, 58 (3) (1996), 246–261.
- [27] L. Vincent and P. Soille, Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations, *IEEE Trans. Pattern Anal. and Machine Intell.*, 13 (6) (1991), 583–598.
- [28] C. Xu and J.L. Prince, Snakes, Shapes, and Gradient Vector Flow, *IEEE Trans. Image Proc.*, 7 (1998), 359–369.
- [29] X. Zeng, L.H. Staib, R.T. Schultz, and J.S. Duncan, Segmentation and Measurement of the Cortex from 3-D MR Images Using Coupled Surfaces Propagation, *IEEE Trans. Med. Imag.*, 18(1999), pp. 927-937.