Circular Shortest Paths by Branch and Bound

Ben Appleton [a] * Changming Sun [b]

[a] *Intelligent Real-Time Imaging and Sensing Group, School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia*

[b] *CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 1670, Australia*

Abstract

Shortest path algorithms are used for a large variety of optimisation problems in network and transportation analysis. They are also used in image analysis for object segmentation, disparity estimation, path finding and crack detection. Sometimes the topology of the problem demands that the path be circular. Such circular path constraints occur in polar object segmentation, disparity estimation for panoramic stereo images and in shortest paths around a cylinder. In this paper we present a new efficient algorithm for circular shortest path determination on a u-by-v trellis in $O\left(u^{1.6}v\right)$ average time. We impose a binary search tree on the set of path endpoints and use a best-first Branch and Bound search technique to efficiently obtain the global minimum circular path. The typical running time of our circular shortest path algorithm on a 256×256 image is in the order of 0.1 seconds on a 1GHz Dell P3 workstation under the Linux operating system. Applications to crack detection and object segmentation are presented.

Key words: Circular shortest path; Dynamic programming; Branch and Bound

1 Introduction

The computation of shortest paths is a fundamental problem from graph theory with a range of efficient solutions. Many optimisation problems may be phrased in graph theoretic terms as a shortest path problem, leading to fast global solutions. Such problems occur in network analysis, transportation and image processing.

* Corresponding author. Tel.: +61 7 3365 4510; Fax.: +61 7 3365 4999

‡‡ *E-mail addresses:* appleton@itee.uq.edu.au, changming.sun@csiro.au

A shortest path is defined as a path of least cost between two nodes, where the cost is the sum of edge and vertex weights along the path. By suitable choice of graph and associated costs many problems may be transformed into a shortest path computation. Shortest path algorithms typically rely upon a labelling method [1].

Shortest path algorithms have been applied to finding curvilinear features such as cracks and roads in images. Buckley and Yang [2] developed a regularised shortest-path extraction method based on time-delayed dynamic programming and demonstrated its application to road detection in satellite images and fracture detection in borehole core images from geophysics. In [3], Barzohar and Cooper present a geometric-stochastic road detection technique based on a MAP (Maximum A Posteriori probability) formulation realised by dynamic programming. Merlet and Zerubia [4] extended a variant of the A* algorithm to cliques for detecting roads and valleys in satellite (SPOT) images.

Stereo matching was first formulated as a shortest path problem in the mid-80's by Ohta and Kanade [5] and Lloyd [6]. The shortest path approach remains much the same in current research [7], where the path through the correlation matrix of maximum sum is obtained.

In some applications the topology of the problem demands a circular path. In object segmentation the boundary of the object is typically a closed contour and hence the path endpoints must be connected. In [8], Bamford and Lovell use a Viterbi-based shortest path algorithm to segment cell nucleii using a radial trellis centred on the cell. A two-pass scheme is proposed to find an approximate segmentation. The circularity problem was also recognised by Jermyn and Ishikawa [9] who proposed using a minimum mean-weight cyclic path algorithm for object segmentation. However their algorithm is computationally expensive running in $O\left(u^2v^2\right)$ time on a u-by-v image. The restriction of the goal function to the mean weight of a path is also a significant limitation.

Panoramic stereo is another example of the need for circular paths, as paths through the cross-correlation matrix should have both endpoints connected to ensure a smooth surface [10]. In [11,12], Sun and Pallottino propose a number of Circular Shortest Path (CSP) algorithms to solve these and other problems. Three of these, the Image Patching Algorithm (IPA), the Multiple Backtracking Algorithm (MBTA) and their hybrid give solutions in $O\left(uv\right)$ time complexity but are not guaranteed to obtain the global CSP. Sun and Pallottino proposed a fourth approach, the Multiple Search Algorithm (MSA) which is guaranteed to find the global CSP in $O\left(u^2v\right)$ time by solving u independent shortest path problems. They demonstrated the application of their algorithms to object segmentation, disparity estimation for panoramic stereo and crack detection on borehole cores.

In this paper we present a new efficient algorithm for CSP determination on a u-by-v trellis in $O\left(u^{1.6}v\right)$ average time. Our algorithm guarantees to find the globally optimal CSP. Section 2 covers the preliminary work in shortest path algorithms and formalises the problem at hand. In Section 3 we propose a Branch and Bound solution to the determination of CSPs. Section 4 analyses the computational complexity of our algorithm. Section 5 gives results verifying the complexity analysis of Section 4 and shows a number of applications of the proposed CSP algorithm. Section 6 concludes.

2 Preliminaries

2.1 Problem Specification

In this subsection we specify the problem that we are attempting to solve, that of the circular shortest path. Consider a discrete image $c : \mathbb{Z}_u \times \mathbb{Z}_v \to \mathbb{R}$, where $c\left(i,j\right)$ is the cost of traversing the pixel at row i and column j in the image. For example, such an image could be the negative log of the probability for each point lying on a crack, or it could be some decreasing function of the cross-correlations between a stereo image pair.

Let G be the directed, vertex-weighted graph with vertex set $V = \mathbb{Z}_u \times \mathbb{Z}_v$. With the notation $a = (a_1, a_2) \in V$ we define the edge set

$$E = \left\{(a,b)\,|a,b \in V,\, a_2 = b_2 - 1,\, |a_1 - b_1| \leq 1\right\}$$

Such a graph G is known as a *stable acyclic sequential layered graph* [12], or simply as a *trellis* [8]. See Figure 1 for an example.

A shortest path across such a trellis is defined as a sequence of connected vertices

$$P = \left(a^j \,\middle|\, a^j \in V,\, j \in \mathbb{Z}_v,\, \left(a^j, a^{j+1}\right) \in E\right)$$

which minimises the path length

$$L\left(P\right) = \sum_{j=1}^{v} c(a^j)$$

over all admissable paths P. In contrast to standard shortest path problems we enforce the additional constraint that $|a_1^1 - a_1^v| \leq 1$, such that the two endpoints a^1, a^v of the path are connected when periodically extending the trellis.

Here we consider the single-source shortest path problem: for a given vertex set $S \subseteq V$ (the source), find the shortest path from S to a for all connected $a \in V$. Perhaps the most well known solution to the single-source shortest path problem is Dijkstra's algorithm [13], which operates in $O(|V| \log |V|)$ time for any graph with $c \geq 0$ on $|V|$ vertices. However, due to the trellis structure of our graph a simpler algorithm is available. Following Sun & Pallottino [11] we use a dynamic programming technique to assign the optimal distances one column or layer at a time.

As the graph is vertex-weighted, a simple approach is to assign the distance to vertex $b \in V$ based on the distances in the previous layer as

$$distance(b) = c(b) + \min\{distance(a) \,|(a,b) \in E\}$$

The distances of the vertices in the first layer are simply their costs:

$$\forall a \in V \,|a_2 = 1 \;,\; distance(a) = c(a)$$

This recursive formulation of the distances admits a solution by dynamic programming. Note that unlike Dijkstra's algorithm this technique allows for negative vertex weights. The resulting algorithm has complexity $O(uv)$, regardless of the cost function c. See [14,1] for a review of shortest path algorithms. The following gives the algorithm for finding an ordinary shortest path by dynamic programming:

Algorithm 1 (Shortest Path by Dynamic Programming)

Initialisation:
for *each* $a^i = (i, 1)$ *in the first layer*
  $distance(a^i) = c(a^i)$

Propagation:
for *each column* $j \in [2, v]$
  for *each vertex* $b = (i,\ j), \quad i \in [1,\ u]$
    $distance(b) = c(b) + \min\{distance(a)|(a,\ b) \in E)\}$
    $parent(b) = \arg\min\{distance(a)|(a,\ b) \in E)\}$

Backtracking:
$p^v = \arg\min\{distance(a)|a_2 = v\}$
for j *from* $v - 1$ *to* 1
  $p^j = parent(p^{j+1})$

3 A Branch and Bound CSP Formulation

Algorithm 1 will not generally give a circular path across the trellis. Such a global constraint is difficult to fit into the framework of a shortest path algorithm which is efficient primarily because it admits a local solution for each vertex. In [12,11], Sun and Pallottino suggest five solutions. Three of these, the Image Patching Algorithm, the Multiple Backtracking Algorithm and their hybrid give solutions in $O(uv)$ time complexity but are not guaranteed to obtain the global CSP. A fourth approach known as the Multiple Search Algorithm (MSA) was proposed to solve the CSP problem in $O(u^2v)$ time by solving u independent shortest path problems, one for each source vertex in the first layer. By construction each such path is circular, and hence the CSP can be determined by taking the shortest of these circular paths.

Sun & Pallottino observed that the shortest path is a lower bound for the CSP. As a simple proof by contradiction, if this were not true then the CSP would be shorter than the shortest path. By corollary if a shortest path is circular then it is the CSP. Our new algorithm makes use of a generalisation of this property.

Consider solving the shortest path problem starting from source set $S \subseteq L_1$, where $L_1 = \{a \in V \,|a_2 = 1\}$ is the first layer of the trellis G. We define the set of admissable paths (with respect to S) as all $P = (p^i)$ such that $\exists s \in S : |s_2 - p_2^v| \leq 1$, i.e. whose end points connect back to an element of the source set S.

Lemma 1 *Let $S, S' \subseteq L_1$ be two source sets with $S' \subseteq S$. The length $L(P)$ of the shortest path $P = (p^i)$ admissible with respect to S is less than or equal to the length $L(P')$ of the shortest path $P' = (p'^i)$ admissible with respect to S'.*

PROOF. Let $\Phi(S)$ be the set of admissable paths with respect to S, and $\Phi(S')$ defined likewise for S'. Then $\Phi(S') \subseteq \Phi(S)$, so $\min(L(\Phi(S))) \leq \min(L(\Phi'(S)))$. Therefore the length of the shortest admissable path through S is a lower bound for the length of all admissable paths through S'. $\square$

Corollary 2 *Observe that if $|S'| = 1$ then $\Phi(S')$ consists of all circular paths through S'. Hence by Lemma 1 the length of the shortest admissable path through S is a lower bound for all circular paths through S.*

Figure 2 gives an example of an admissable path on a typical source set S, depicted in thick lines.

We now impose a binary search tree on the set of potential source vertices in the first layer. The root node of the search tree corresponds to the source

set $S = L_1$, the entire first layer. Each node's children partition its source set into two contiguous halves. The leaves of the tree are all u single vertices of L_1. This tree allows us to apply the divide and conquer paradigm to the problem of determining the CSP. We apply a best-first Branch and Bound [15] algorithm to search the binary tree, evaluating only those subtrees which may contain the CSP. Figure 3 exemplifies the binary tree for $u = 8$.

We now observe a number of properties of the tree. By Lemma 1 the shortest path for any node in the search tree forms a lower bound for the entire subtree rooted at that node. The act of partitioning a search node's source set S into its children χ_1 and χ_2 will never break up a circular path. However, any non-circular path will eventually split as its starting and ending points are not adjacent. So the potential circular paths become more prominent as we search further down the tree, until only circular paths are considered at the leaves.

We perform a priority-first-search of the binary tree, with the negative lower bound of a node being assigned as its priority. The priority queue is initialised with the root node of the search tree, whose source set corresponds to L_1. At each step in our algorithm we take the node in the search tree with the lowest bound and split it into its two subtrees. The simple shortest path algorithm (Algorithm 1) is used to evaluate each subtree's lower bounds given their source sets before inserting them into the priority queue. We also backtrack along their shortest paths to determine if the shortest path for each search node is circular. We give our Branch and Bound CSP algorithm as follows:

Algorithm 2 (Branch and Bound Circular Shortest Path)

Initialisation:
Initialise the Root search node as a non-circular path of arbitrary length
enqueue*(Root)*

Priority First Search:
while*(true)*
  $n =$ **dequeue**(), *where n is a node of the binary tree*
  if *n is circular*
    return n
    halt
  else for *each child χ of n*
    Compute the shortest path across the trellis for the source set χ
    Store in χ whether the shortest path was circular
    enqueue(χ) *with priority equal to the negative shortest path length*

Lemma 3 *Algorithm 2 gives the globally shortest circular path.*

PROOF. Observe that the source sets in the queue form a partitioning of

L_1. By the corollary of Lemma 1 then the minimum lower bound of the nodes in the priority queue is a global lower bound on the length of the CSP. Thus if the node with highest priority has as its shortest path a circular path we are guaranteed that this path is the globally shortest circular path across the trellis. □

4 Complexity Analysis

The running time of our algorithm is dependent upon the structure of the cost array. In general Branch and Bound techniques have poor worst-case behaviour [16]. For example, consider Figure 4.

By suitably extending this pattern we may construct a trellis of arbitrarily large size for which the binary search tree is inefficient. Observe that any contiguous source set of 4 vertices will generate a shortest path of length 0. We are required to split each search node down to 3 or fewer source vertices, which is equivalent to evaluating up to one half of the nodes in the binary search tree. This class of cost functions will run in $O(u^2v)$ time, the same order as the Multiple Search Algorithm [11].

However, such pathological instances are extremely improbable and should not be a cause for concern. A classic parallel is Hoare's quicksort algorithm [17], which despite an $O(N^2)$ worst-case bound for sorting N elements is on average one of the fastest algorithms for sorting general data. Here we present a simple analysis of the average running time of our algorithm.

4.1 Properties of the Branch and Bound CSP Algorithm

The halting condition for our algorithm is that the best circular path obtained is shorter than or equal to all other shortest paths for each search node. Hence our algorithm must split each path which is shorter than the circular shortest path. We call a path which is shorter than the CSP a *short path.*

The search tree must be evaluated along all branches from the root to the nodes which split the short paths. The priority-first search ensures that only the nodes which split the short paths are evaluated, minimising the number of shortest path computations required. Paths whose endpoints are far from meeting will be split near the root of the search tree, and may well be split along the way to other short paths. Paths whose endpoints are nearly connected are near-circular and quite possibly form part of the circular shortest path, so are worth investigating.

4.2 Analysis of the Branch and Bound CSP Algorithm

Let N_n be the number of search nodes to be evaluated in the subtree rooted at node n. Let R denote the root of the search tree, with χ_1 and χ_2 its children. Due to the data dependant complexity of Branch and Bound it is difficult to analyse the average case, so following Zhang and Korf [16] we take a simple model of the search dynamics by assuming that each node has a constant probability β of being split given that its parent has been split. Then the expected number of shortest path computations to perform is

$$E(N_R) = 2 + \beta E(N_{\chi_1}) + \beta E(N_{\chi_2})$$

Given that χ_1 and χ_2 symmetrically partition the first layer, we obtain $E(N_{\chi_1}) = E(N_{\chi_2})$ so we have

$$E(N_R) = 2 + 2\beta E(N_{\chi_1})$$

The tree has $\lceil \lg u \rceil + 1$ levels with the root always split, where $\lceil \lg u \rceil$ denotes the smallest integer greater than or equal to $\lg u$. Recursing we obtain

$$E(N_R) = 2 + 2\beta(2 + 2\beta(\cdots)) = 2\left(1 + 2\beta + (2\beta)^2 + \cdots + (2\beta)^{\lceil \lg u \rceil}\right)$$

$$\text{So } E(N_R) = \begin{cases} 2 \cdot \frac{(2\beta)^{\lceil \lg u \rceil + 1} - 1}{2\beta - 1} & \beta \neq \frac{1}{2} \\ 2(1 + \lceil \lg u \rceil) & \beta = \frac{1}{2} \end{cases}$$

The cost of splitting a search node is $O(uv)$, the time taken to perform a shortest path computation for each child node.

For $\beta > \frac{1}{2}$, $E(N_R) = 2 \cdot \frac{(2\beta)^{\lceil \lg u \rceil + 1}}{2\beta - 1} \leq \frac{8\beta^2}{2\beta - 1} \cdot u^{\lg(2\beta)}$ for large u, giving an overall time complexity of $O\left(u^{1+\lg(2\beta)} v\right)$.

For $\beta = \frac{1}{2}$, we obtain an overall time complexity of $O(uv \lg u)$ directly.

For $\beta < \frac{1}{2}$, $E(N_R) = \frac{2}{1 - 2\beta}$ for large u, giving an overall time complexity of $O(uv)$.

5 Results

5.1 Computational Speed

In order to estimate the average running time and to verify the simple complexity analysis of Section 4.2 we tested our algorithm on a large number of

random images. Image sizes ranging from 200×200 to 4000×4000 were considered, with pixel values drawn independently from a uniform random distribution. The running times on each image size were averaged over 100 samples. The tests were performed on a 1GHz Dell P3 workstation under the Linux operating system. The results (Figure 5 and Table 1) agree with our simple complexity analysis over the wide range of sizes tested. Linear regression of Figure 5 gives a slope of $2 + \lg(2\beta) = 2.6$ or $\beta = 0.76$ for uniformly randomly distributed trellises. This equates to an $O(u^{1.6}v)$ average running time on random data. The algorithm is likely to perform significantly better for real applications, as problems for which a CSP is desired have innately periodic and structured cost functions.

5.2 Object Segmentation

To demonstrate the application of CSPs to object segmentation two cell images are presented. Figure 6(a) depicts *Cyclostephanos Dubius*, a diatom. Segmentation is performed by taking a polar unwrapping of the cell about an interior point, then finding the CSP across the resulting trellis as shown in Figure 6(b). This segmentation is then projected back onto the original image as shown in Figure 6(c).

Figure 7(a) shows three overlapping cell nuclei where the central cell is to be separated. Segmentation is performed on the polar unwrapping of the gradient image, by shortest path (b) and by circular shortest path (e). The resulting segmentations (c, f) clearly demonstrate the necessity of circular paths in object segmentation.

5.3 Borehole Crack Detection

Figure 8(a, c) depicts two borehole core surface images. These planar images are an unwrapping of the cylindrical surface of the borehole core (b, d), so it is natural to require that the endpoints of our shortest path connect. The CSP algorithm presented in this paper was applied to both images to detect dark curvilinear features such as cracks and flaws.

5.4 Comparison to Existing Methods

Here we compare the existing methods to Algorithm 2 on a trellis (Figure 9(a)) similar to Figure 4. Figure 9(b) shows the shortest path across the trellis without the circularity constraint. Figure 9(c) shows the path obtained by the

MBTA/IPA algorithm of Sun and Pallottino [11,12]. Figure 9(d) shows the correct CSP obtained by our Branch and Bound CSP algorithm. The path obtained by our new algorithm is the global optimum.

6 Conclusions

We have developed a new algorithm for finding the circular shortest path across a trellis. This algorithm finds the shortest path under the constraint that the starting and ending positions are connected by using a Branch and Bound search of the endpoints with a dynamic programming evaluation of corresponding path lengths. This constraint is an important consideration in many applications including polar object segmentation, disparity estimation for panoramic stereo images and shortest paths around a cylinder. The algorithm that we developed runs in $O(u^{1.6}v)$ average time on uniformly random images, with a worst case $O(u^2v)$ no worse than the Multiple Search Algorithm. It is interesting to note that for some problems with low branching probability the order of the algorithm may drop to $O(uv)$, the same order as the shortest path computation. The algorithm was applied to polar object segmentation and crack detection on borehole cores. It was shown (Lemma 3) to obtain the globally shortest circular path.

Acknowledgements

We thank Mark Berman for suggesting the worst case trellis of Figure 4. We also thank Hugues Talbot for his comments and suggestions. We are very grateful to Houyuan Jian for his comments and suggestions for the paper. Figure 6 was taken from the ADIAC public data web page: http://www.ualg.pt/adiac/pubdat/pubdat.html (CEC contract MAS3-CT97-0122). We are grateful to Axon Imaging for supplying the cell image used in Figure 7.

References

[1] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73(2):129–174, May 1996.

[2] M. Buckley and J. Yang. Regularised shortest-path extraction. *Pattern Recognition Letters*, 18(7):621–629, 1997.

[3] M. Barzohar and D. B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):707–721, 1996.

[4] N. Merlet and J. Zerubia. New prospects in line detection by dynamic programming. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):426–431, April 1996.

[5] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Anal. Mach. Intell.*, 7(2):139–154, March 1985.

[6] S.A. Lloyd. Stereo matching using intra- and inter-row dynamic programming. *Pattern Recognition Letters*, 4:273–277, September 1986.

[7] Changming Sun. Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques. *International Journal of Computer Vision*, 47(1/2/3):99–117, May 2002.

[8] Pascal Bamford and Brian Lovell. Unsupervised cell nucleus segmentation with active contours. *Signal Processing (Special Issue: Deformable models and techniques for image and signal processing)*, 71(2):203–213, 1988.

[9] Ian H. Jermyn and Hiroshi Ishikawa. Globally optimal regions and boundaries. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 904–910, Kerkyra, Greece, September 1999.

[10] Changming Sun and Shmuel Peleg. Fast panoramic stereo matching using cylindrical maximum surfaces. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 2003. Accepted.

[11] Changming Sun and Stefano Pallottino. Circular shortest path in images. *Pattern Recognition*, 36(3):709–719, March 2003.

[12] Changming Sun and Stefano Pallottino. Circular shortest path on regular grids. In *Asian Conference on Computer Vision*, pages 852–857, Melbourne, Australia, January 22–25 2002.

[13] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[14] G. Gallo and S. Pallottino. Shortest path algorithms. *Annals of Operations Research*, 13:3–79, 1988.

[15] P. H. Winston. *Artificial Intelligence*. Addison-Wesley Publishing Company, Inc, 1984.

[16] W. Zhang and R. Korf. An average-case analysis of branch-and-bound with applications: Summary of results. In *Proc. 10th National Conf. on Artificial Intelligence, AAAI-92*, pages 545–550, San Jose, CA, July 1992.

[17] C. A. R. Hoare. Quicksort. *Computer Journal*, 5(1):10–15, 1962.

Biographical Sketch

BEN APPLETON is currently undertaking a PhD in the area of Image Analysis at the University of Queensland, Brisbane, Australia. He received his bachelor degrees in Science and Electrical Engineering from the University of Queensland in 2001. His research interests include active contours and energy minimisation algorithms.

CHANGMING SUN received his PhD in the area of Computer Vision from Imperial College of Science, Technology and Medicine, London in 1992. Then he joined CSIRO Mathematical and Information Sciences, Australia where he is currently a Principal Research Scientist carrying out research and working on applied projects. His research interests include computer vision and photogrammetry, image analysis, pattern recognition, and bioinformatics. Dr Sun is a member of the Australian Computer Society and the Australian Pattern Recognition Society.

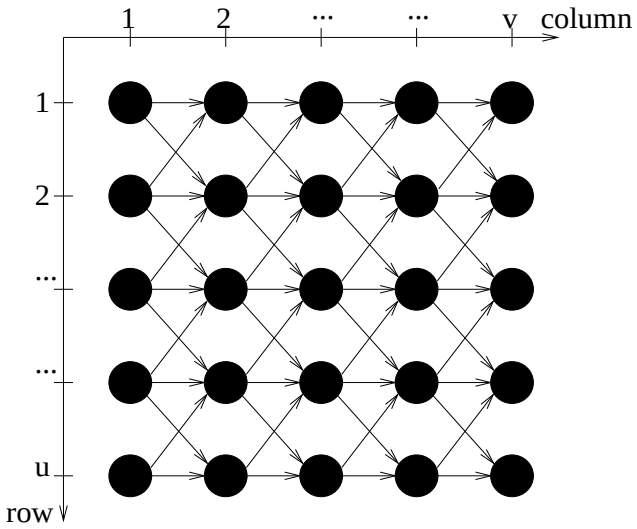


Figure 1.

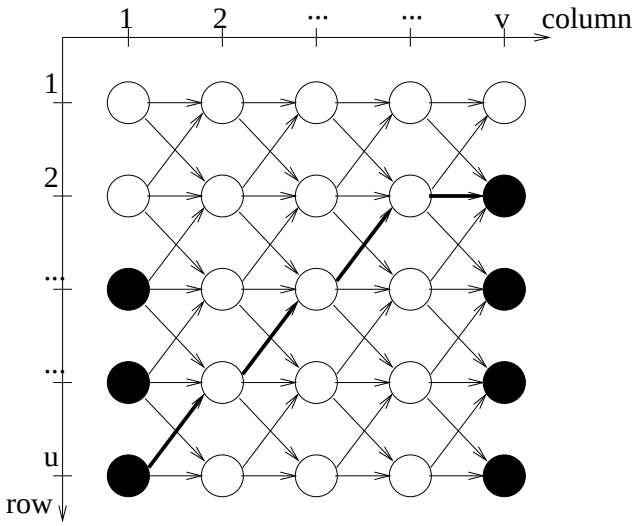


Figure 2.

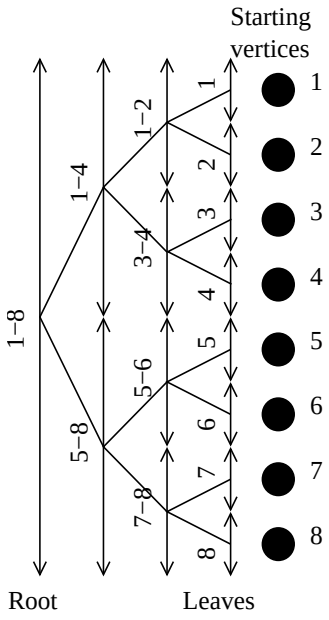


Figure 3.

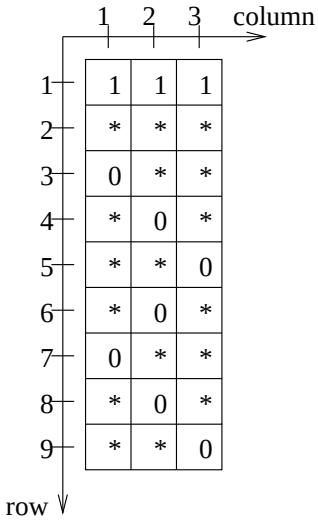


Figure 4.

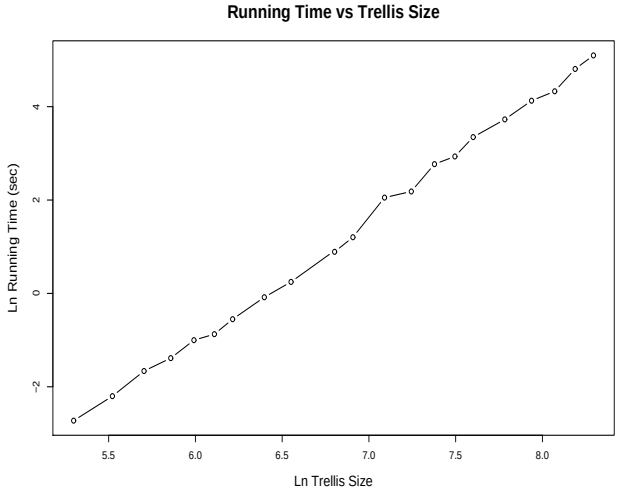
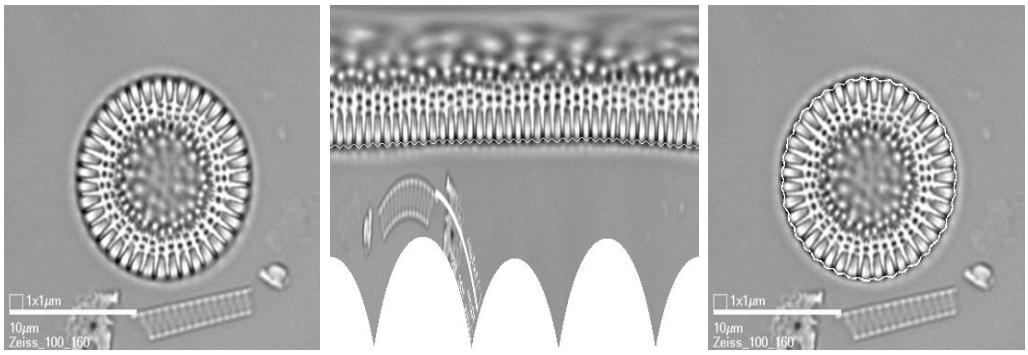
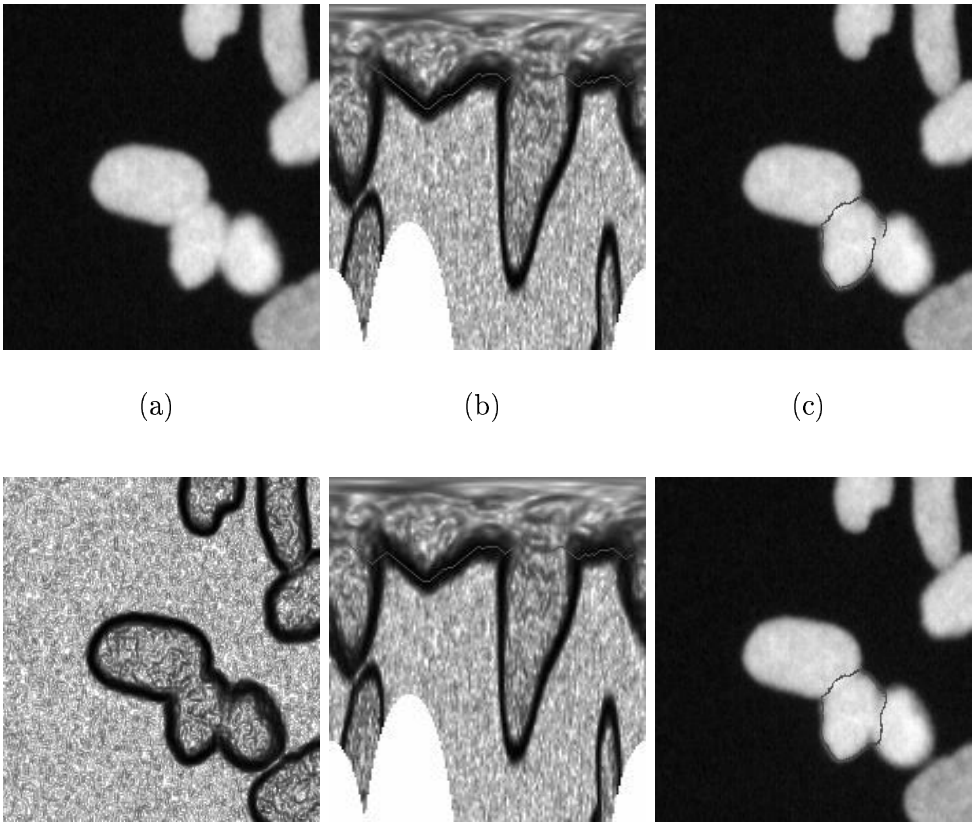

Running Time vs Trellis Size
Ln Running Time (sec)
-2
0
2
4
5.5
6.0
6.5
7.0
7.5
8.0
Ln Trellis Size

Figure 5.



(a) (b) (c)

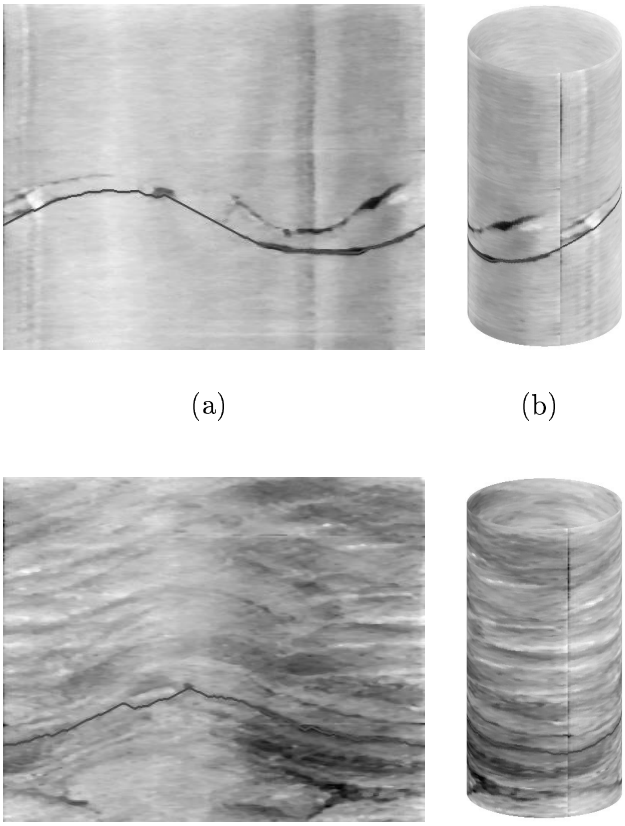
Figure 6.



(a) (b) (c)

(d) (e) (f)

Figure 7.



(a) (b)

(c) (d)

Figure 8.

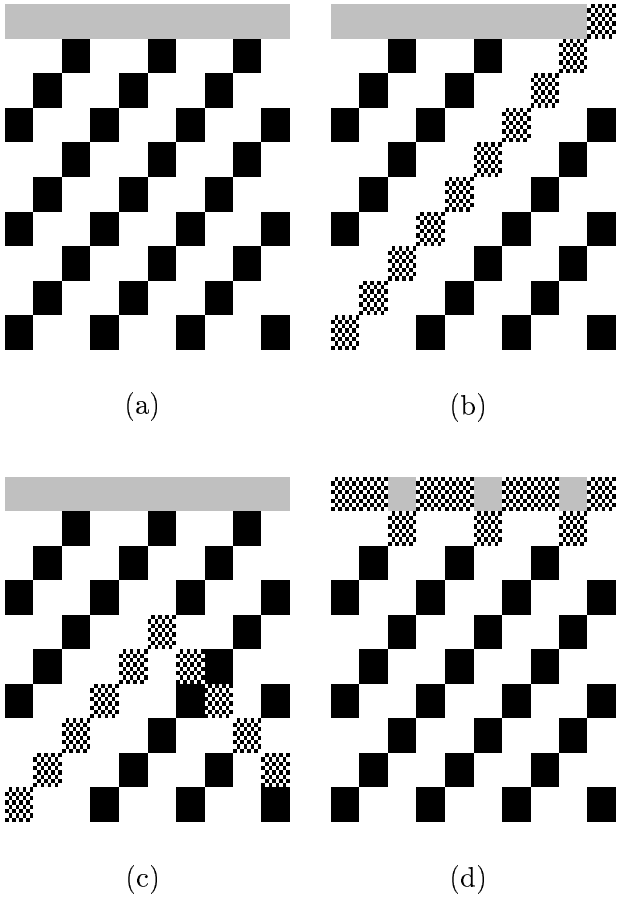


Figure 9.

Table 1

Trellis Size	Average Running Time (sec)	MSA Running Time (sec)
200×200	0.065	1.62
500×500	0.575	26.4
1000×1000	3.33	186
2000×2000	28.5	1443
4000×4000	164	12060

Figure 1. An example trellis.

Figure 2. An example of a shortest path with restricted endpoints. Valid end points are shown in black, with the corresponding shortest path in bold.

Figure 3. The binary search tree for $u = 8$. Only L_1, the first layer of the trellis containing the potential source vertices is depicted.

Figure 4. A pathological trellis. Asterisks represent vertices of infinite cost. This pattern may be extended to any width by replication of the central column, and to any height by extending the row pattern.

Figure 5. Average running time of our new algorithm for uniformly random images (shown in logarithmic scale).

Figure 6. Segmentation of *Cyclostephanos Dubius*. (a) The original microscope image of the diatom. (b) The polar unwrapping with circular shortest path overlayed. (c) The corresponding segmentation contour.

Figure 7. Cell nuclei segmented by shortest path and circular shortest path. (a) The original cell image. (d) The inverse gradient image. (b, c) The shortest path in the polar domain, with the corresponding open contour. (e, f) The circular shortest path in the polar domain, with the corresponding closed contour.

Figure 8. Two borehole core images: (a, c) are images shown flat; (b, d) are images projected onto a cylinder. The circular shortest paths are overlayed.

Figure 9. Comparison of various CSP methods. (a) A pathological cost function. Black is 0, grey is 1 and white is 11. (b) The shortest path, weight 1. (c) The path obtained by MBTA/IPA, weight 44. (d) The correct path obtained by our Branch and Bound CSP algorithm, weight 7.

Table 1
Average running time for a few trellis sizes. Averages were obtained over 100 trials on uniformly random cost functions. MSA running time is listed for comparison.