Computing Topological Adjacency Relations Between Iso-contours

Xingang Huang
Dept. of Computer and Information Science,
The Ohio State University,
2015 Neil Avenue, Columbus, OH 43210
huang@cis.ohio-state.edu

Feng Zhao
Xerox Palo Alto Research Center,
3333 Coyote Hill Road,
Palo Alto, CA 94304
zhao@parc.xerox.com

Abstract

Contoured charts are widely used to visualize 2D physical fields. Experts can identify global patterns and structures in a contoured chart by looking at the iso-curves and reasoning about their spatial relations. We develop an algorithm for computing the topological adjacency relations between iso-contours. The algorithm is novel in that it grounds the computation of spatial relations between aggregate spatial objects upon the computation of relations between the constituents. It is scale-independent and efficient. We present an application of the algorithm to weather data analysis for extracting patterns from numerical weather datasets.

Introduction

Contoured charts have been widely used to visualize 2D physical fields. They abstract out local fluctuations and retain global patterns, and are a concise and qualitative intermediary representation often suitable for studying the global behaviors of physical fields. For example, in weather analysis, contoured charts are a primary tool used by meteorologists to read weather conditions. From the charts, they detect different patterns such as pressure systems, troughs and fronts, and use them to forecast weather.

A contoured chart comprises a group of iso-curves. These curves are non-intersecting: they do not self-intersect or intersect each other. They are also separating: each curve divides the chart into two disconnected parts. Patterns in a contoured chart are formed as qualitatively distinct spatial configurations of the curves. To computationally identify a pattern, it is essential to quantify the spatial relations between the curves. One spatial relation of particular importance is the topological adjacency (TA) relation: two curves are topologically adjacent if they are not separated by any other curves.

Topological adjacency relations are useful in grouping iso-curves relevant to a pattern, and in serving as links between curves to form the structure of a pattern. The computation of the relations is a key component in our larger research effort in automating global spatial reasoning and pattern identification, the kinds of reasoning tasks routinely performed by meteorologists in analyzing weather data sets. It provides a set of basic spatial relations upon which more global, aggregate structural descriptions such as troughs, thermal packing, as well as features such as cold/warm fronts can be efficiently derived (Huang 2000; Huang & Zhao 2000b).

In this paper, we study the properties and the transitivity rules of the topological adjacency relations, develop an efficient algorithm to compute them, and present an application of the algorithm to weather analysis for extracting high/low pressure centers. The algorithm first determines an initial, partial set of topological adjacency relations for curves from more primitive relations on points, and then uses the partial adjacency information together with higher-level structural knowledge about adjacency graphs to recover the additional adjacency information.

Related Work

Topological spatial relations between regions have been studied mainly from two directions: the *Region Connection Calculus (RCC)* (Randell, Cui, & Cohn 1992; Bennett 1994; Cohn *et al.* 1997; Renz & Nebel 1999) in AI, and the *9-intersection* model (Egenhofer 1991; Engenhofer & Mark 1995) in GIS. RCC adopts a region topology in which regions are primary objects and the *connection* relation is the primary relation. Other relations between regions are defined upon the *connection* relation with a set of axioms and Boolean functions using first-order logic. RCC research (Bennett 1994; Renz & Nebel 1999) studies the composition rules of different spatial relations and uses these rules to uncover unknown relations from known ones.

The 9-intersection model adopts a point-set topology in which points are primary objects and regions are defined as sets of points. A topological relation between two regions is classified as one of the nine possible intersections between the interiors, exteriors and boundaries of the two regions (only empty and non-empty are distinguished). This classification of adjacency relations has been used in defining spatial query languages (Svensson & Zhexue 1991; Hadzilacos & Tryfona 1992).

RCC and the 9-intersection model work at different levels. The 9-intersection model works at the point

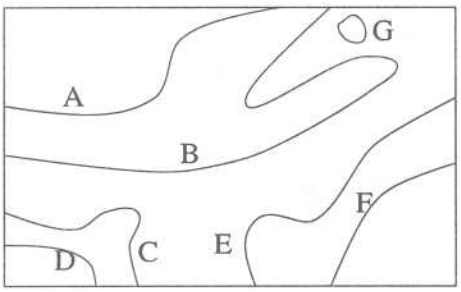


Figure 1: A group of separating, non-intersecting curves in a 2D space. Examples of the topological adjacency relation are: $TA(A,B)$, $TA(B,C)$, $TA(C,G)$, $\neg TA(A,C)$, $\neg TA(A,G)$. Examples of the same-side relation are: $SS(A,B,C)$, $SS(A,C,D)$, $SS(B,E,D)$, $\neg SS(B,F,E)$, $\neg SS(A,D,C)$, $\neg SS(D,F,E)$.

level and requires detailed descriptions of regions. RCC works at the region level and requires only qualitative descriptions of regions; thus, it avoids expensive point-level computations. A limitation of the 9-intersection model is that a relation between two regions has to be completely determined by the two regions involved; the model is not able to compute binary relations that are dependent on other regions, such as the topological adjacency relation studied in this paper. RCC exploits the transitivity of relations using a logic approach and is not restricted by this limitation. Though RCC reasons about regions, in real applications it has to rely on point-level computation to build the base relations that it can reason upon. A problem for both RCC and 9-intersection is that the relations they study are often too general to express rich spatial constraints found in many applications.

This paper studies topological relations between constrained aggregate spatial objects, i.e., separating and non-intersecting curves, on which more specific relations can be defined. It develops an algorithm which utilizes both point-level computations and curve-level reasoning. The point-level computations are different from the ones used in the 9-intersection model in that when computing point relations between two curves, the points of other curves are also considered. The constraints imposed upon the curves enable the definition of new spatial relations and the discovery of new transitivity rules which have not been previously studied using RCC.

Topological Adjacency Relation and Topological Adjacency Graph

In this section we examine the topological adjacency relation defined on a group of separating and non-intersecting curves in a 2D space. Common examples of such curve groups are the iso-contours in contoured 2D charts. Fig. 1 shows an example of such a group of curves, with relations to be defined shortly.

The separating and non-intersecting properties are only meaningful to a curve when it is a member of a group of curves in a 2D space. For conciseness, in this paper we often do not mention explicitly the curve group and the 2D space to which a curve belongs; and when we refer to a curve, it is assumed to be a member of a group of separating and non-intersecting curves in a 2D space. Due to the page limit, we will also omit some lengthy proofs and algorithms in the following sections. Interested readers should consult (Huang & Zhao 2000a).

We first define *topological adjacency* and *same-side* relations, and study their properties such as transitivity. We then study properties of a graph defined on a group of curves by their topological adjacency relations.

The Topological Adjacency Relation

Definition 1 (Topologically Adjacent) *Two different curves A and B are* topologically adjacent *(denoted as $TA(A,B)$) if they are not separated by any other curves. A curve is not* topologically adjacent *to itself.*

Definition 2 (Same-side) *Two curves A and B are on the same side of a curve C (denoted as $SS(A,B,C)$) if both A and B are in the same part of the space partitioned by C.*

Examples of these two kinds of relations are given in Fig. 1. The topological adjacency (TA) relation is a binary relation. It is symmetric but not transitive. Whether it is reflective depends on definition. We define it to be non-reflective so that it can induce a graph naturally. The same-side relation is a ternary relation. One of its basic properties is: $SS(A,B,C) \Leftrightarrow SS(B,A,C)$.

Lemma 1 *Let A,B be two curves. Then $TA(A,B) \Leftrightarrow \forall C \notin \{A,B\}, SS(A,B,C)$.*

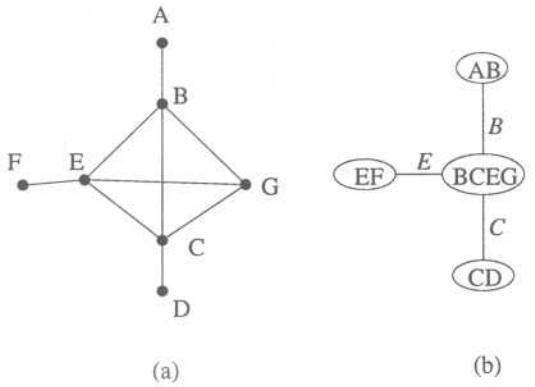
This lemma describes the connections between the two relations: if two curves are topologically adjacent, then they are on the same side of any other curves; if two curves are not topologically adjacent, then there exists another curve that separates them. The lemma is straight-forward from the definitions of the SS and TA relations.

Although the TA relation is not transitive, it becomes transitive if the three curves involved satisfy a same-side relation:

Lemma 2 *Let A,B and C be three different curves. Then $TA(A,B) \wedge TA(B,C) \wedge SS(A,C,B) \Rightarrow TA(A,C)$.*

Proof – by contradiction:

Assume $\neg TA(A,C)$. Then there exists a curve D, s.t. $\neg SS(A,C,D)$, i.e., A and C belong to different parts of the space D partitions. Since B can only be in one the two parts of the space partitioned by D, we have $\neg SS(A,B,D) \vee \neg SS(C,B,D)$. Therefore, we have $\neg TA(A,B) \vee \neg TA(B,C)$. Contradiction. ◁



(a) (b)

Figure 2: The T-graph (a) and the clique tree (b) of the curves in Fig. 1. The A-nodes of the T-graph are B, C and E, and the NA-nodes are A, D, F and G. This T-graph has four largest cliques. They are drawn as ellipses in (b) and are labeled by the curves they contain. An edge between two cliques is labeled by the A-curve the two cliques share, which is also the only curve the two cliques share.

Lemma 3 *Let $P_0P_1 \ldots P_n$ be a sequence of curves, in which any two consecutive curves P_i and P_{i+1} are topologically adjacent. Then $\forall i$, $0 < i < n$, $SS(P_{i-1}, P_{i+1}, P_i) \Rightarrow TA(P_0, P_n)$.*

Lemma 3 is a generalization of Lemma 2 and can be proven by induction. It can be used for reasoning about whether two curves are topologically adjacent through a chain of topologically adjacent curves.

Next we study the graph defined by the topological adjacency relations.

The Topological Adjacency Graph and Its Properties

Definition 3 (Topological Adjacency Graph) *The* Topological Adjacency graph (T-graph) *of a set of curves is a two tuple: (V, E), where V is the set of curves, and E is the set of all unordered pairs of curves A and B in V that satisfy $TA(A, B)$.*

The T-graph of the curves in Fig. 1 is shown in Fig. 2 (a). In a T-graph, each node is a curve and each edge is a topological adjacency between two curves. The nodes in a T-graph can be classified into two types:

Definition 4 (A-node (A-curve)) *An* A node (A-curve) *is an articulation node* [1] *in a T-graph.*

Definition 5 (NA-node (NA-curve)) *A* NA-node (NA-curve) *is a non-articulation node in a T-graph.*

A NA-curve can bound a region, together with the boundary of the space if necessary. This is because a NA-curve partitions a space into two parts such that all other curves are in one part and none are in the other. Therefore, the part contains no curves is a region bounded by the NA-curve alone. On the other hand, an A-curve cannot bound a region by itself because both the two parts it partitions contain other curves. Examples of NA-curves and A-curves are given in Fig. 2.

A graph can be represented by all its largest cliques. Next we show that the largest cliques of a T-graph have some interesting properties.

Lemma 4 *Let α and β be two different largest cliques in a T-graph. Then α and β share at most one node, and the node, if it exists, is an A-node of the T-graph.*

Proof: Omitted due to space limitation.

Theorem 1 *Define a graph $G = (V, E)$ using a T-graph T, where $V = \{\alpha : \alpha$ is a largest clique of $T\}$, and $E = \{(\alpha, \beta) : \alpha$ and β are in V and share a node$\}$. Then G is a tree.*

Proof:

Since a T-graph is a connected graph, its clique graph is also connected. Each edge in a clique graph corresponds to an A-node in a T-graph, whose removal will disconnect the T-graph. Hence, removing an edge of the clique graph will also disconnect the clique graph. Therefore, every edge of a clique graph is a bridge and the graph is a tree. ◁

Each of the largest cliques in the T-graph represents a connected region in the space which is bounded by all the curves in the clique. The tree structure of a clique graph can also be understood from the point of view of regions. N separating, non-intersecting curves divide a 2D space into $N + 1$ regions. On the other hand, suppose there are M A-curves and $N - M$ NA-curves. Each NA-curve can bound a region by itself. So $N - M$ NA-curves produce $N - M$ regions. M A-curves correspond to the M edges in the clique tree, so there are $M + 1$ largest cliques in the tree that bound $M + 1$ regions. The total number of regions counted from this way is also $(N - M) + (M + 1) = N + 1$. The clique tree of the curves in Fig. 1 is given in Fig. 2 (b).

Corollary 1 *A cycle in a T-graph is contained in one and only one of the largest cliques of the T-graph.*

Proof: Follows from Theorem 1.

Corollary 2 *Let A and B be two nodes of a T-graph G, $TA(A, B)$. Then a path between A and B is contained in the largest clique of G that contains both A and B.*

Proof: Follows from Corollary 1.

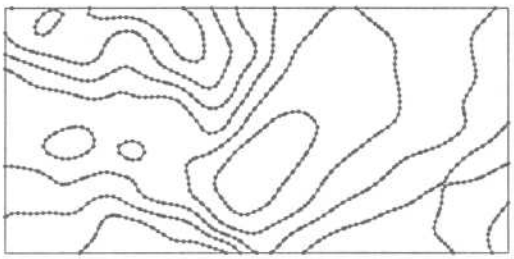
Computing the Topological-Adjacency Graph

In this section, we study how to compute the T-graph of a group of separating, non-intersecting curves in a convex 2D planar space. We first present an algorithm for computing a sub-graph of a T-graph. We then describe how to use this sub-graph to compute the T-graph. Finally, we present the entire algorithm and study its complexity. The algorithm requires each curve

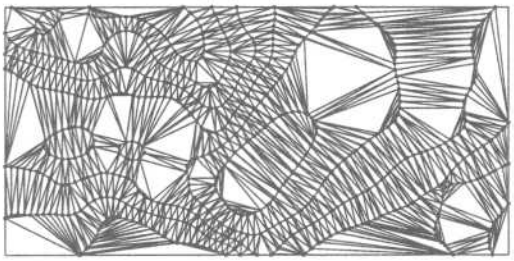
[1] An articulation node is a node of a connected graph whose removal will disconnect the graph.

be represented as a sequence of points. This is not a severe restriction since many contour charts in practice are generated from numerical grid data.

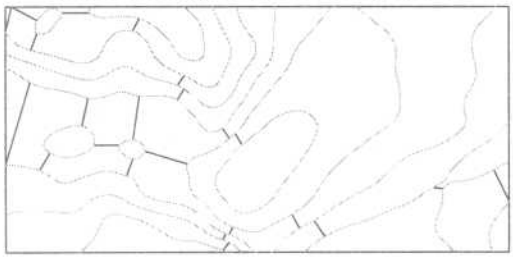
The D-graph



(a) Input



(b) Delaunay neighborhood graph



(c) D-graph

Figure 3: A sample run of Algorithm 1. Input (a) is a set of iso-curves contoured from a 2D pressure dataset. Each curve is represented as a sequence of points (dark dots). A Delaunay triangulation of all the points of the iso-curves is shown in (b). A D-graph (c) is then computed by Algorithm 1. In (c), gray lines represents iso-curves and dark lines curve adjacencies. A curve adjacency is determined by the shortest Delaunay edge between two curves.

A T-graph is a very useful neighborhood graph on a group of curves. A curve is an aggregate object whose constituent objects are points. We use a relation aggregation approach (Huang & Zhao 1999) to build a neighborhood graph of curves. In this approach, each edge in a neighborhood graph is treated as a neighborhood relation. The neighborhood relations between aggregate objects are built by aggregating the neighborhood relations between the constituent objects.

Algorithm 1 computes a neighborhood graph of curves. We call such a neighborhood graph a D-graph.

Algorithm 1 The D-graph computing algorithm

- Input: a group of separating, non-intersecting curves, each curve is represented as a sequence of points.
- Output: a graph whose nodes are all the curves.
- The algorithm:
  - Build a Delaunay triangulation neighborhood graph on all the points of the given curves.
  - Relation aggregation:
    - Examine every edge in the Delaunay neighborhood graph, if a edge connects two points that are on two different curves, establish an adjacency between the two curves if such an adjacency has not been established.

The algorithm first builds a Delaunay triangulation on all the points of the given curves. It then checks every Delaunay edge and builds an adjacency between two curves if the edge checked connects the two curves. The result is a D-graph. A run of the algorithm is given in Fig. 3. The algorithm has a time complexity of $O(Mlog(M))$, where M is the total number of points. A Delaunay triangulation takes $O(Mlog(M))$ time (Lee & Shachter 1980), and the relation aggregation step takes only $O(M)$ time since there are $O(M)$ edges in a Delaunay neighborhood graph, which is a planar graph.

A D-graph is connected and has the same node-set as its corresponding T-graph. Next we show that when the points in each curve are dense enough, a D-graph is a sub-graph of its corresponding T-graph.

Definition 6 (The closeness condition) *Let e be the minimum distance between any two points on different curves and let d be the maximum distance between any two consecutive points on a same curve. The set of points satisfies the* closeness *condition if $\sqrt{2}e > d$.*

Theorem 2 *If the points of all given curves satisfies the* closeness *condition, then the D-graph computed by Algorithm 1 is a subgraph of the T-graph.*

To prove Theorem 2, we only need to prove that two points on two non-topologically-adjacent curves will not be connected by a Delaunay edge. Since such an edge will have to cross between two consecutive points on a curve, when consecutive points are closer to each other, the edge will be too close to both the two points it crosses, and it will be excluded by Delaunay triangulation. The formal proof of Theorem 2 is omitted.

Computing a T-graph using a D-graph

Since a D-graph is connected, any two nodes of the graph can be connected by a path. According to Corollary 2, when a D-graph is a sub-graph of a T-graph, all the paths of the D-graph have the following properties:

1. If the two end nodes of a path are topologically adjacent, then all the nodes in the path are topologically adjacent to each other because they are in the

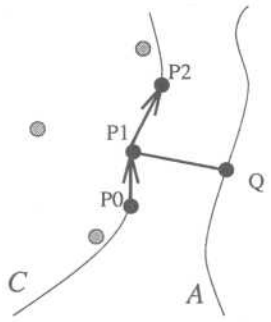


Figure 4: Determining which side of a curve C another curve A is on. Curves A and C are adjacent in the D-graph. P_1Q is a Delaunay edge between them. P_2 and P_0 are the next and previous points of P_1 in curve C, following the default traversing direction of C. This figures shows the configuration when curve C is turning right at point P_1. Q as drawn is on the right side of curve C. The three gray dots are the other three possible positions of point Q with respect to lines P_0P_1 and P_1P_2, in which Q would be on the left side of curve C.

same largest clique. Further more, for every three consecutive nodes P_i, P_{i+1} and P_{i+2} in the path, $SS(P_i, P_{i+2}, P_{i+1})$.

2. If the two end nodes of a path are not topologically adjacent, then there exist three consecutive nodes in the path P_i, P_{i+1} and P_{i+2}, s.t. $\neg SS(P_i, P_{i+2}, P_{i+1})$.

Therefore, whether two curves are topological adjacent can be decided by finding a path that connects their corresponding nodes in the D-graph, and checking whether any three consecutive nodes P_i, P_{i+1} and P_{i+2} in the path have the same-side relation: $SS(P_i, P_{i+2}, P_{i+1})$. We have developed a clique-building algorithm to compute all largest cliques of a T-graph from a D-graph by building depth-first-search trees of a D-graph and examining the same-side relations between a node, its parent and its grandparent in the trees. The algorithm has a time complexity of $O(N^2)$, where N is the number of curves. The details of the algorithm are omitted here.

Determining the Same-Side Relation

The clique-building algorithm requires the computation of $SS(A, B, C)$ when A and B are both adjacent to C in the D-graph. This can be done by selecting a default traversing direction of curve C, and using this traversing direction to determine which side (left or right) of the curve C the curves A and B are on.

When two curves A and C are adjacent in the D-graph, there exists a Delaunay edge connecting them. This edge can be used to determine the side of curve C on which curve A is located, as illustrated in Fig. 4. The side of curve C on which point Q (and curve A) is located can be determined by examining the spatial configurations between the four points P_0, P_1, P_2 and Q.

Fig. 4 shows the configuration when curve C is turning right at point P_1. In this configuration, if Q is on the right side of both lines P_0P_1 and P_1P_2, then Q is on the right side of curve C; if Q is on the left side of either line P_0P_1 or line P_1P_2 (i.e., positioned as the three gray dots in the figure), Q is on the left side. Likewise, the configuration when Curve C is turning left at point P_1 can be similarly solved.

The above method of determining side requires that the Delaunay edge P_1Q between two curves does not intersect the two curves at points other than P_1 and Q. This means there should be no cross edges – Delaunay edges that cross between two consecutive iso-points on a same curve. To guarantee no cross edges, a higher sampling density than the one specified in the closeness condition is required. One property of Delaunay triangulation proved in (Amenta, Bern, & Eppstein 1998) is that when a set of sample points S r-samples [2] a group of smooth curves, $r \leq 1$, then the Delaunay triangulation of S contains the polygonal reconstruction of the curves. So if the iso-points 1-sample the iso-curves, there will be no cross edges and the above method of determining side can guarantee correct result.

Complexity Analysis

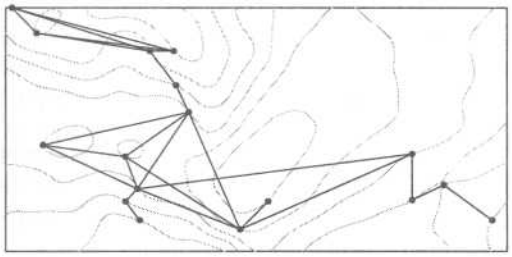
Combining the D-graph algorithm and the clique-building algorithm, we obtain an algorithm for constructing a T-graph. The algorithm has a time complexity of $O(Mlog(M) + N^2)$, where M is the number of points and N is the number of curves. In real applications such as computing the topological adjacency relations of iso-contours in a weather chart, N is usually a small number (typically less than 100) and the term N^2 can be ignored.

Results and Comparison

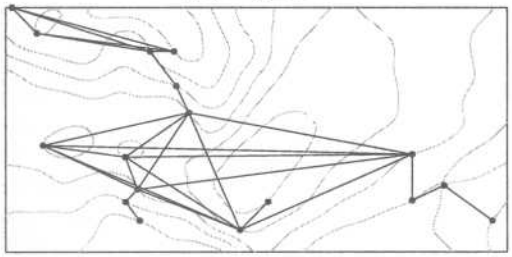
Fig. 5 and Fig. 6 give two T-graphs computed by our algorithm from the iso-contours of two pressure charts. The curves in Fig. 5 are the same as the curves in Fig. 3. The iso-contours are obtained by contouring pressure datasets using a 2D version of the marching-cube algorithm (Lorenson & Cline 1987).

The topological adjacency relation can also be computed using the coloring algorithm for computing the inside/outside relation described by Ullman (Ullman 1984). The curves are first mapped into a binary image, where curve pixels are set as black and other pixels white. Then all white pixels are activated to form connected regions. Two curves are topologically adjacent if they are the boundaries of a same region. The coloring algorithm is scale-dependent. Its time complexity is $O(H{\cdot}V)$, where H and V are the horizontal and vertical resolutions of the binary image used. Our algorithm

[2] A set of sample points r-sample a curve if for each point p on the curve, the distance from p to its nearest sample point is less than $r * LFS(p)$, where $LFS(p)$ is the local feature size of p, i.e., the distance from p to the medial axis of the curve.



(a) D-graph



(b) T-graph

Figure 5: The D-graph (a) and T-graph (b) of the curves in Fig. 3(a) computed by our algorithm. Nodes of a graph are drawn as dark dots located on the central points of their corresponding curves. Edges of a graph are drawn as lines between the dots. The D-graph in (a) is the same as the D-graph in Fig 3 (c) but drawn in a different format for better comparison with the T-graph.

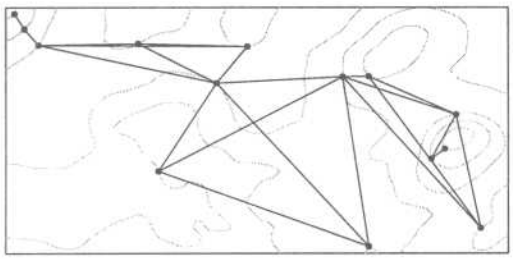
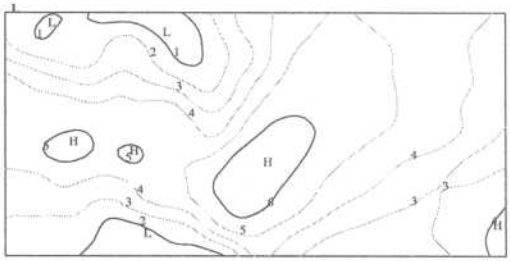


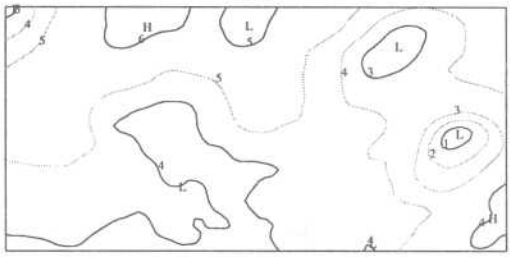
Figure 6: The T-graph computed by our algorithm using another group of iso-curves in a contoured chart.

is not scale-dependent and does not require curves being mapped to a binary image. Its time complexity is $O(Mlog(M))$, where M is the number of points on curves. It can be much more efficient than the coloring algorithm when M has the same magnitude as H and V.

Our T-graph algorithm does not check the *closeness* condition, neither does it check if the iso-points r-sample (r=1) the iso-curves. So the Delaunay triangulation computed may contain cross edges to corrupt the T-graph computation. This problem rarely occurs (in fact never occurred during our experimentations of the algorithm on tens of weather datasets). It also can be eliminated by modifying Algorithm 1 in the following way. After a Delaunay triangulation is built, check whether there is a Delaunay edge between every two consecutive points on every curve. If there is no such an edge for two consecutive points P and Q of curve C, subsample C by adding the midpoint of segment PQ into the curve, and build a new Delaunay triangulation incrementally. Repeat this step until every two consecutive points of every curves have a Delaunay edge. Now the Delaunay triangulation is guaranteed to have no cross edges and Algorithm 1 is guaranteed to output correct results.



(a)



(b)

Figure 7: Labeling high/low pressure centers. A high pressure center is labeled by a "H", and a low pressure center is labeled by a "L". The NA-curves are drawn in dark and A-curves in gray. The numbers besides curves are their contour levels. A high (low) pressure center is identified by a local maximum (minimum) NA-curve in the T-graph.

Weather Applications

T-graphs can be used to label high/low pressure centers in a pressure chart. A high (low) pressure center is a region of local maximum (minimum) pressure values compared with its neighbor regions in a pressure chart. It can be identified by a local maximum (minimum) NA-curve [3] in a T-graph . Fig. 7 shows the results of the labeling on the two pressure charts used in previous sections.

Conclusion

This paper describes a novel algorithm for building an important spatial relation on a group of curves. It uses

[3] Only an NA-curve can encompass a region by itself.

both local spatial relations (among points) and higher-level knowledge about graph structures to build more global spatial relations (among curves). The algorithm is scale-independent and efficient. We give a simple application of the topological adjacency relation to extracting high/low pressure centers for illustration. The topological relation, as well as the relation aggregation mechanism for building global relations using local ones, have been used in identifying more complicated weather patterns such as pressures troughs and weather fronts from spatial weather datasets (Huang & Zhao 2000b; Huang 2000).

Acknowledgment

The work is supported in part by FZ's NSF NYI grant CCR-9457802, ONR YI grant N00014-97-1-0599, and a Sloan Foundation Fellowship.

References

Amenta, N.; Bern, M.; and Eppstein, D. 1998. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing* 60:125–135.

Bennett, B. 1994. Spatial reasoning with propositional logic. In *Proc. 4th Int. Conf. on Knowledge Representation and Reasoning*, 51–62.

Cohn, A.; Bennett, B.; Gooday, J.; and Gotts, N. 1997. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica* 1:275–316.

Egenhofer, M. 1991. Reasoning about binary topological relations. In *Proc. 2nd Symposium on Large Spatial Databases, SSD-91, Lecture Notes in Computer Science 525*, 143–160. Berlin: Springer.

Engenhofer, M., and Mark, D. 1995. Modeling conceptual neighborhoods of topological line-region relations. *International Journal of Geographical Information Systems* 9:555–565.

Hadzilacos, T., and Tryfona, N. 1992. A model for expressing topological integrity constraints in geographic databases. In *Theories and Methods of Spatial-Temporal Reasoning in Geographic Space, Lecture Notes in Computer Science 639*, 252–268. New York:Springer-Verlag.

Huang, X., and Zhao, F. 1999. Seeing "objects" in spatially distributed datasets. In *Proceedings of the Third International Symposium on Intelligent Data Analysis, Lecture Notes in Computer Science (1642)*, 111–122. Springer.

Huang, X., and Zhao, F. 2000a. Computing topological adjacency relations between iso-contours. Technical Report OSU-CISRC-1/00-TR04, CIS Department, The Ohio State University.

Huang, X., and Zhao, F. 2000b. Relation-based aggregation: Finding objects in large spatial datasets. *Inter. J. of Intelligent Data Analysis* ,To appear.

Huang, X. 2000. *RelSA: Automatic Analysis of Spatial Data Sets Using Visual Reasoning Techniques with an Application to Weather Data Analysis.* Ph.D thesis, CIS dept., The Ohio State University.

Lee, and Shachter. 1980. Two algorithms for constructing delaunay triangulations. *Int'l J. Comput. and Info. Sci.* 18.

Lorenson, W., and Cline, H. 1987. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics* 21:163–169.

Randell, D.; Cui, Z.; and Cohn, A. 1992. A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, 165–176.

Renz, J., and Nebel, B. 1999. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence* 108:69–123.

Svensson, P., and Zhexue, H. 1991. Geo-sal: A query language for spatial data analysis. In *Advances in Spatial Databases - Second Symposium, SSD'91, Lecture Notes in Computer Science 525*, 119–140. New York:Springer-Verlag.

Ullman, S. 1984. Visual routines. *Cognition* 18:97–159.