The Study on Dynamic Topology Based On Land Use Management

Hou Yanping Fan Hong

Abstract The thesis puts forward data structure and algorithm of dynamical topology that are based on the studies of traditional static topology. Then an appropriate model and a practical algorithm are designed to solve all sorts of problems that are confronted with in Land Use Management.

Key words Land Use Management，static topology，dynamical topology，topological arcs reference arrays，topological alteration-processing model

Abstract The advent of topology in land use management has revolutionized the way that work efficiency is increased in traditional work by hand. Traditional topology has such function that it can create and handle spatial relationships between points, lines and polygons whole at a time no matter changes how small. It is called as static topology.

With the improvements in applications, traditional mode hasn’t met the needs of current development better. In many practical applications, such as rural land sift, urban cadastral management, country investigate in utilization, figures and attributes of land subdivision have changed more frequently, however, involved range is not wide enough to adopt static topology because of bad efficiency. At the same time, it is difficult to sustain old and new data of figures and attributes. It’s a good idea that after edit and modify of some local geographic objects, corresponding objects’ topological relationships have been modified merely. It is called as dynamical topology.

Dynamical topology has such function that after static topology, if some figures or attributes of some local geographic objects are edited and modified, it can create and handle spatial relationships between points, lines and polygons only between ones that have been modified or that influenced by modified ones. Relationships wouldn’t be sustained or modified entirely any more. In this mode, one hand is spatial point of view that the objects which are handled by dynamical topology are local geographic objects which are really need to change the topology, and the other hand is temporal point of view that the process which the spatial relationships have been established dynamically is a dynamically commutative one between users and systemic data.

Consider the data process models and interrelated data structures and algorithms from before.

I Data Process Model of Dynamical Topology

Lets talk about spatial and temporal point of view first.

Spatial point of view is that: If changes have happened on figures or attributes in some graphics, the relationships between geographic objects don't need establishing entirely any more. Only partial ones that have been modified or that influenced by modified ones need reestablishing again. The data don't need submission whole any more. Only corresponding objects' ones need submission. In order to meet the expectation, efficient index structure should be established to realize searching partial objects within the whole region quickly and smoothly. Among R-tree index, grid index, quad-tree index, octagon-tree index, the algorithm of grid index is the easiest, and the relationships between points and lines are emphasized mainly by it. Besides, Minimum Bounding Rectangle of arcs and polygons can well serve the searches and calculates of the corresponding objects.

Temporal point of view is that: Process temporal describing and managing of partial topological relationships. Dynamical topology is that it can create and handle spatial relationships between partial basic topological features. Such as a series of the adding, deleting, moving, uniting or splitting ends, vertices, nodes of arcs or polygons, etc. They are stored orderly, and so, complex topological changes can be divided into a series of simple operations on topological features, which can be described in accordance with the sequence of operations.

After partial topological features have been changed, the operations that result in topological relationships changes can be divided into several simple ones and finished by steps and by stages. The operations, such as adding, deleting, moving, uniting or splitting ends, vertices, nodes of arcs or polygons, etc. can't result in intersections. Otherwise the changes have to be finished immediately and automatically. It is called as systemic submission, which can be compared with the user submission that users finish the changes initiatively. During the course of operations, the operated objects and operations manners are recorded in accordance with the sequence of operations schedule. At the same time, partial topological relationships (mainly node-arc relations) are sustained or modified successively. And after hours, the whole process of topological changes are sustained or modified in sequence of recorded ordinal operations schedule.

It is obvious that the models involved in establishing topological relationships are the topological alteration-processing model besides node-arc-polygon model of traditional static topology and grid model of spatial index and relational model of database etc. The topological alteration-processing model has high demands in data structure and algorithm, such as the fleetly searching of ends, vertices, nodes, arcs and polygons, etc. and the recording the operated objects and operations manners during the course of topological changes in order to sustain or modify the relationships between the topological features.

II Data Structure of Dynamical Topology

The data structure and algorithms of dynamical topology are based on the studies of traditional static topology, of cause, they need improve on, such as in order to meet the expectation of dynamical topology, information changes on relevant points, arcs and polygons should be better recorded in traditional static topology. For example, topological points reference, topological arcs reference arrays, minimum-bounding rectangle of arcs and polygons, directional arc-lists of polygons etc. For example, the reference of each end on an independent and unclosed arc is 1, and information of the arc is stored in topological arcs reference arrays. To each vertex on arcs, the reference is 2 and no information is stored in topological arcs reference arrays in order to simplify the data structure and avoid redundant data. To the ends on a closed arc, the each reference is 2 too because they are same one and the information of this closed arc is stored in topological arcs reference arrays. To each node on arcs, the reference is no less than 3 and the information of all arcs whose head or tail is the node is stored in topological arcs reference arrays.

In addition, during the dynamical topological alteration processing the operations on ends, vertices, nodes, arcs or polygons will stand a good chance to result in adding, deleting, splitting, uniting or modifying polygons, and even probably have an effect on parent-child relationships between polygons, that is, inclusive relationships or insular relationships between polygons. Record the sequence of operations during the dynamical topological alteration processing, and modify or sustain the parent-child relationships after the systemic submission.

III Algorithms of Dynamical Topology

Because all polygons are composed of points and arcs, the operations on polygons can also be composed of the sequence of the ones on points and arcs that are composed of them. Great concern on data structure and algorithms of dynamical topology should be made just for points (including independent points, ends, vertices and nodes) and arcs in adding, deleting, moving, splitting and union.

■ ***Modify Arcs***

That is operations on arcs adding, deleting and moving wholly etc.

● ***Adding***

◆ ***Arcs adding can't result in intersections with other topological features.***

In this condition, it is perhaps adding an isolated line that can add the arc to the systemic arc set. Also perhaps adding a closed line that not only need to add the arc to the systemic arc set, but add the new polygon which is composed of by it to the systemic polygon set as well. At the same time, the operations should be recorded in order to sustain or modify parent-child relationships between polygons after the systemic submission.

◆ ***Arcs adding result in intersections with other topological features.***

There is a restrictive qualification with this understanding, that is, it is allowed to make it closed but not to make it intersect by itself. This line can be cut into several parts, which is called as arcs. Then add these arcs to the systemic arc set one by one, and sustain or modify relationships correspondingly.

As can be seen from Figure 1: Add line **ACE** to arc set, and it makes intersect with polygon FGHLMNF. Line **ACE** is cut into arcs **AB, BCD** and **DE**. And add them to the systemic arc set one by one, then split the polygon FGHLMNF into polygon **BCDGFB** and **BCDHLMNB.**

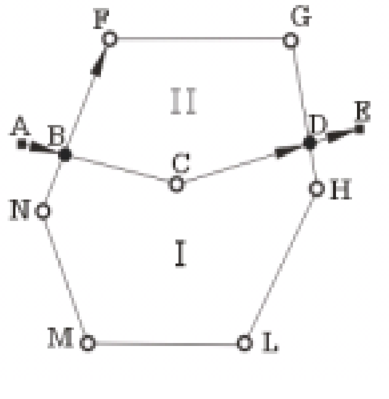


FIGURE 1 Adding Lines

Consider the pseudo code example from before: The procedure can be separated into two parts.

PROCEDURE 1: Add arcs that are cut into parts to the minimum arc arrays.

INPUT: New Arc List added listPt
The Systemic Arc Set
CArray<TArc*,TArc*> m_ArcArray

OUTPUT: the point sets that make of minimum arcs
CArray<TPoint_tp*,TPoint_tp*> NewPtArray
The Minimum Arc Arrays
CArray<TArc*,TArc*> AddArcArray

STEP 1: Find the first point (i.e. head) of the new added line.
TPoint_tp* pPtPre = listPt.GetHead()
Keep point pPtPre stored into NewPtArray.
Traverse the list (listPt) sequentially from the second point, and constitute the line segment pLine with the previous one.

STEP 2: Determine whether pLine intersects the each arc that is taken out of the systemic arc set (m_ArcArray) sequentially. First, determine whether the minimum-bounding rectangle of pLine intersects the one of the each arc that comes from the systemic arc set sequentially. If doesn't intersect, go continuously to the next one. If does, insert the intersections to the arc that is used to be compared with, and store them to the NewPtArray. After hours, the arc is cut into several new segments in term of intersections. Then add them to the m_ArcArray. The Left-Polygon and Right-Polygon will succeed the ones of the old arc.

STEP 3: **S**ort by size of distance from the points both on NewPtArray and pLine to the head of the list pLine, then put them to the NewPtArray over again.

STEP 4: Go to the step 1, and constitute a segment with the previous one and the next one from listPt. Then go to the step 2 and 3 until traverse listPt sequentially from the very beginning to the end. Finally, add the final point to the NewPtArray.

STEP 5: Generate the minimum arc arrays (AddArcArray) according to the points of NewPtArray. Traverse NewPtArray sequentially, and constitute a new arc with the points that every couple of them whose reference isn't 2 and ones between the couple whose reference is 2 from NewPtArray. Then store these new arcs into AddArcArray.

PROCEDURE 2: Add arcs that belongs to the minimum arc arrays to the systemic arc set.

INPUT: **T**he Minimum Arc Arrays
CArray<TArc*,TArc*> AddArcArray
The Systemic Arc Set
CArray<TArc*,TArc*> m_ArcArray
The Systemic Polygon Set
CArray<TPolygon*, TPolygon*> m_PolyArray

OUTPUT: The Systemic Arc Set modified m_ArcArray
The Systemic Polygon Set modified m_PolyArray

STEP 1: Traverse the minimum arc arrays (AddArcArray) sequentially one by one (TArc* pArc). If the reference of each end of pArc isn't 1, its Left-Polygon and Right-Polygon must be searched for from m_ArcArray.

STEP 2: Search for pArc's Left-Polygon and Right-Polygon.

First, get the topological arcs reference array (arrayArcRef) of the pArc's tail (m_pPt_End), and take them out except pArc and dangle arcs and closed arcs. If there aren't any, go to the step 1. If there is only one, go to the step 3 and search for the Right-Polygon. If there is more than one, calculate the counter-clockwise angles between the direction of the arc advancing and the segment that starts at m_pPt_End and ends of the adjacent point that belongs to the respective arc of them, then sort by size of the angles.

STEP 3: **S**earch for pArc's Right-Polygon.

Take the arc out that has been found in step 2 whose value of the angle is the minimal along the direction of its advancing, then treat it as the operations on pArc in step 2, and find the arc out whose value of the angle is the minimal by counter-clockwise along the direction of its advancing until reach to the arc whose end is the head of pArc, which can find the pArc's new Right-Polygon. Then store the arcs that have been searched for sequentially into the arc arrays of an object PolyRight. If the direction of the points that consist the arc is as same as the direction of the arcs that surround PolyRight, the direction of the arc is TURE, otherwise, FALSE. Then store the directions of the arcs into directional arc-lists of PolyRight. If there aren't any arcs that can accord with the items, it is demonstrated that pArc hasn't Right-Polygon, and set its PolyRight null.

STEP 4: **S**earch for pArc's Left-Polygon.

Be similar to the step 3, but the difference is the arcs which have been found out whose value of the angle is the maximum by counter-clockwise along the direction of its advancing until reach to the arc whose end is the head of pArc. Then store the arcs that have been searched for sequentially into the arc arrays of another object PolyLeft. Accordingly, the directions of the arcs should be stored into directional arc-lists of PolyLeft. If there aren't any, it is also demonstrated that pArc hasn't Left-Polygon, and set its PolyLeft null.

STEP 5: First pArc is added into m_ArcArray, then its Right-Polygon (PolyRight) and Left-Polygon (PolyLeft) are to be analyzed if has any.

To PolyRight, traverse the all arcs except pArc. At the same time, determine their directions bArcDir and conduct the following process:

```
if( bArcDir == TRUE && pArc->m_pPolyRight != NULL)
     pArc->m_pPolyRight = PolyRight;
else if( bArcDir == FALSE && pArc->m_pPolyLeft != NULL)
     pArc->m_pPolyLeft = PolyRight;
```

To PolyLeft:

```
if( bArcDir == TRUE && pArc->m_pPolyLeft != NULL)
     pArc->m_pPolyLeft = PolyLeft;
else if( bArcDir == FALSE && pArc->m_pPolyRight != NULL)
```

pArc->m_pPolyRight = PolyLeft;

We can determine its rotating direction by calculating the area of the polygon. If direction of PolyLeft is clockwise and of PolyRight is counter-clockwise, they are the outer bounding polygons. Add PolyLeft and PolyRight into m_PolyArray.

STEP 6: Go to the step 1, and search for Right-Polygon and Left-Polygon of the next arc until traverse the minimum arc arrays sequentially from very beginning to the end.

● ***Deleting***

Several complexions are classified in order that we can discuss each of their algorithms when an arc is deleted according as whether the ends of the arc are closed or not, how much is the reference of the each end of the arc and whether Right-Polygon and Left-Polygon of the arc are in existence or not.

◆ ***Deleting an independent unclosed arc.***

It is called as an independent unclosed arc if the references of the ends of an arc are both 1. And so, deleting it can’t result in any changes on topological relationships between topological features. The arc is deleted from systemic arc set directly.

◆ ***Deleting a Dangle Arc.***

It is called as a dangle arc if the reference of one of the ends of an arc is 1 only, and the other is no less than 3. If its reference of head (or tail) is 1, its reference and topological arcs reference array of tail (or head) need modifying when the dangle arc is deleted — Delete the arc from topological arcs reference array of tail (or head), and its value of reference is subtracted 1. After being modified, if its reference turns into 2, and moreover, there are two different arcs in its topological arcs reference array, they need uniting together to one at here, and the point (tail or head) becomes a vertex on a new arc.

◆ ***Deleting a closed arc.***

It is called as a closed arc when the ends of an arc are the same one. The algorithm of deleting a closed arc is similar to the one of deleting a dangle arc. Of cause, they have difference, that is, its value of reference of the tail (or head) is subtracted 2 not 1 from reference. Accordingly, the polygon surrounded by this arc should be deleted from the systemic polygon set. Don't need to modify the reference of the tail (or head) if the closed arc is independent. At the same time, the operation on the closed arc should be recorded in order to sustain or modify the parent-child relationships between polygons after systemic submission.

◆ ***Deleting an arc whose ends are different nodes.***

It is called as a generic arc whose ends are different nodes. When deleting it, first the respective reference of its tail and its head is subtracted 1 severally, and delete it from respective topological arcs reference array of the tail and the head. Then sustain or modify the relationships between its Right-Polygon and Left-Polygon. Finally, delete it from the systemic arc set directly.

● ***Moving wholly***

Moving an arc wholly is that the all points that consist of it make the same coordinate excursion synchronously. If both of its ends aren't nodes, the operation must be limited severely, that is, it is an independent arc that can be moved wholly. The algorithm of the operation is similar to adding a new arc, and here don't give unnecessary details any longer.

■ ***Modify Points***

Points can be classified into several categories according to their reference — independent points, ends and vertices and nodes of the arcs, etc. Each category will be discussed from before:

● ***Independent Points***

The structure of independent points is very simple, and the relationships between arcs and points, polygons and points can be calculated synchronously. So the access to the topological relationships between independent points and other topological features isn't provided for the structure. Adding, deleting or modifying an independent point can be calculated successively after operations.

● ***Ends of Arcs***

It is called as ends of an arc whose reference is 1, and they are regarded as a special kind of nodes. There are three operations on an end, moving, deleting or uniting. Particular discussions will be made latterly because the operation of uniting may result from vertices or nodes.

◆ ***Moving***

If the operation of moving the end of an arc can't result in intersections with other topological features, only coordinates and indices of the end need modifying accordingly. Otherwise, the operation will result in the systemic submission, which causes the partial changes of topological relationships. The algorithm of the operation is similar to adding a new arc, moving an arc wholly, etc. And here say more than is needed.

◆ ***Deleting***

The operation of deleting an end of an arc means deleting the arc indeed if there are only two points (the head and the tail) in its point-list, which the algorithm of the operation can consult the one of deleting a dangle arc. If there are no less than three points in its point-list, the operation of deleting an end of an arc is that the end is removed from its point-list and the adjacent point is regarded as a new end whose reference is changed into 1.

● ***Vertices of Arcs***

It is called as vertices of an arc whose reference is 2. And the ends of a closed arc are regarded as a vertex too, but a difference should be made between them. There are five operations on a vertex, adding, deleting, moving, splitting or uniting. The algorithm of uniting will be discussed latterly.

◆ ***Adding***

Adding a vertex has no effect on other topological features and relationships need no change. Add the vertex to the point-list, and add its information to the grid index.

◆ ***Deleting***

After deleting a vertex, its next point will link the previous one directly. But such an operation must be restricted in the condition that there are no less than four points in point-list of the closed arc, otherwise the closed arc will turn into a to-and-fro segment which makes no practical meaning.

Select a vertex that need deleting and delete it. If the line by which its next point and the previous one will be linked directly will intersect with no other topological features, the vertex will be deleted from its point-list. Otherwise, the operation will cause the systemic submission, and its algorithm is similar to adding a new arc, moving an arc wholly, etc. And here say more than is needed.

- ***Moving***

After moving a vertex, if the two segments that pass through it intersect with no other topological features, the relationships between them will sustain changelessness. Modify the coordinates and the index of the vertex merely. Otherwise, the operation will cause the systemic submission, and its algorithm is similar to deleting a vertex above, which can consult the algorithm adding a new arc.

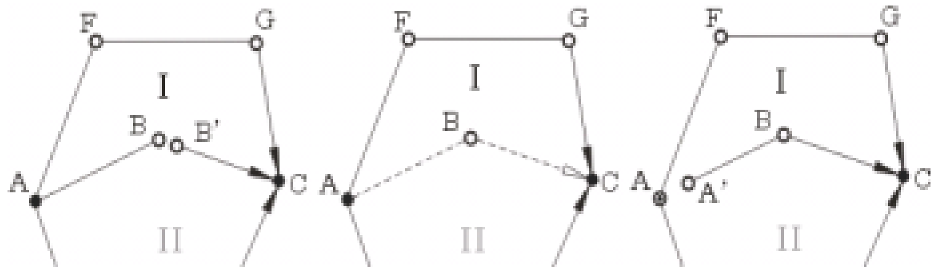
- ***Splitting***

To a closed arc whose end reference is 2, splitting a vertex will result in an independent arc from it. To a closed arc whose end reference is no less than 3, splitting a vertex will result in two dangle arcs from it. At the same time, delete the Right-Polygon and the Left-Polygon from the systemic polygon set.

To an independent unclosed arc or a dangle arc or a generic arc, splitting a vertex will result in two arcs from it. If the Right-Polygon and the Left-Polygon of the old arc are not null, they even should be united into one.

The operations, such as deleting a generic arc (which has been discussed above), splitting a vertex, splitting a node (which will be discussed soon) and so on, maybe induce union between the Right-Polygon and the Left-Polygon of the old arc that is deleted or the vertex or the node is on.

As can be seen from Figure 2: Polygon I and Polygon II are united into one because of splitting the vertex B into B and B’ (e.g. **(1)**), deleting the generic arc ABC (e.g. **(2)**) or splitting the node A into A and A’ (e.g. **(3)**).



（1）　　　　　　　　　　（2）　　　　　　　　　　（3）

FIGURE 2 Splitting a Vertex, Deleting a Generic Arc or Splitting a Node Induce Union.

Consider the following pseudo code example for union:

INPUT: TArc* pArc
The Systemic Polygon Set
CArray<TPolygon*, TPolygon*> m_PolyArray
OUTPUT: The Systemic Polygon Set modified m_PolyArray

STEP 1: Because the Right-Polygon and the Left-Polygon of an independent unclosed arc or a dangle arc are both null, and ones of a closed arc are the same one but they have opposite rotary directions, three complexions can be classified according as whether the pointers of pPolyRight and pPolyLeft are null or not.

- If its Right-Polygon and Left-Polygon are both null, they do not need uniting.
- If neither of its Right-Polygon and Left-Polygon are null, and there is an outer bounding polygon in them, they need uniting.
- If neither of its Right-Polygon and Left-Polygon are null, and neither of them is an outer bounding polygon, they need uniting.

STEP 2: Process the second complexion:

Suppose the outer bounding polygon is marked with pPoly1, and the other is marked with pPoly2, and the new polygon after union is marked with pPoly0. Put the arcs that are in arc arrays of pPoly1 but except pArc and that are in arc arrays of pPoly2 into the arc arrays of pPoly0 sequentially. PArc is replaced sequentially by the arcs that are in arc arrays of pPoly2 but except pArc and that are in arc arrays of pPoly1. At last, pPoly1 and pPoly2 that act as the

Right-Polygon or Left-Polygon of all arcs are all replaced by pPoly0, and add it to m_PolyArray as a new outer bounding polygon.

Process the third complexion:

Suppose pPolyLeft is marked with pPoly1, and pPolyRight is marked with pPoly2, and the new polygon after union is marked with pPoly0. The algorithm is similar to above, but the difference is that pPoly0 shouldn't be marked with an outer bounding polygon any more.

STEP 3: Delete pPoly1 and pPoly2 from m_PolyArray, and set the Left-Polygon and Right-Polygon of pArc null.

Finish processing.

● ***Nodes of Arcs***

It is called as nodes of an arc whose reference is no less than 3, which means that all operations on a node will have an effect on arcs on which the node is acted as the head or tail. And so, adding or deleting a node is restricted severely, even there is a limit to move a node within compass in case the operation would cause the new intersections. The operations on a node, such as splitting or uniting, can well be executed. The algorithm of uniting will be discussed latterly.

◆ ***Moving***

The algorithm of moving a node becomes very simple owing to the restriction above. The relationships will sustain changelessness except that the coordinates and the index of the node should be modified merely.

◆ ***Splitting***

The operation on a node of splitting is that select the node that need splitting first, then the direction of splitting, that is, which arc will be split from the node.

The corresponding algorithm would look like this:

Traverse the topological arcs reference array of the node (arrayArcRef) sequentially, and find out the arc that will be split from the node, then remove it from arrayArcRef. At the same time, generate a new point as its new head (or new tail) within some cluster tolerance to substitute for the node. If the Right-Polygon and the Left-Polygon of the old arc aren't null, they even should be united into one. The algorithm can consult the one of splitting a vertex. Of cause, the reference of the node (NumRef) should be modified accordingly. The node should be changed into a vertex on condition that NumRef turns into 2 after the arc has been split from the node.

● ***Uniting Points***

Particular and concentrative discussions are made from before because the operation of uniting may result from ends, vertices or nodes. There are two restrictions to unite two points: One is that the distance of the two points should be within a limited range, and the other is that there aren't any features between them, i.e. the segment that connects the two points can't intersect with other features.

◆ ***Uniting Two Ends***

The head can be united with the tail on the same arc to shape a new polygon on the condition that the number of the points that consist of the arc is no less than 4. Delete the end of the arc first, and add the head to its point-list at the end. Then modify information of its reference and topological arcs reference array.

As can be seen from the example **(1)** in Figure 3: The operation of uniting A and B can result in a new polygon AFEDCBA.

The operation of uniting two ends on the different arcs can't cause the changes of topological relationships if another point is an end too. As can be seen from the example **(4)** in Figure 3. There are four complexions to unite two ends, that is, head and head, head and tail, tail and head, tail and tail, etc.

As can be seen from the example **(2)** in Figure 3: Be similar to example **(4)**, the operation of uniting two ends on the different arcs can't cause the changes of topological relationships.

As can be seen from the example **(3)** in Figure 3: The operation of uniting two ends on the different arcs can cause the changes of topological relationships, which a new polygon shapes. It can be seen that a new arc after being united is added to the systemic arc set whose algorithm of the operation is similar to adding a new arc and the PROCEDURE 2 above can well be the same with it.

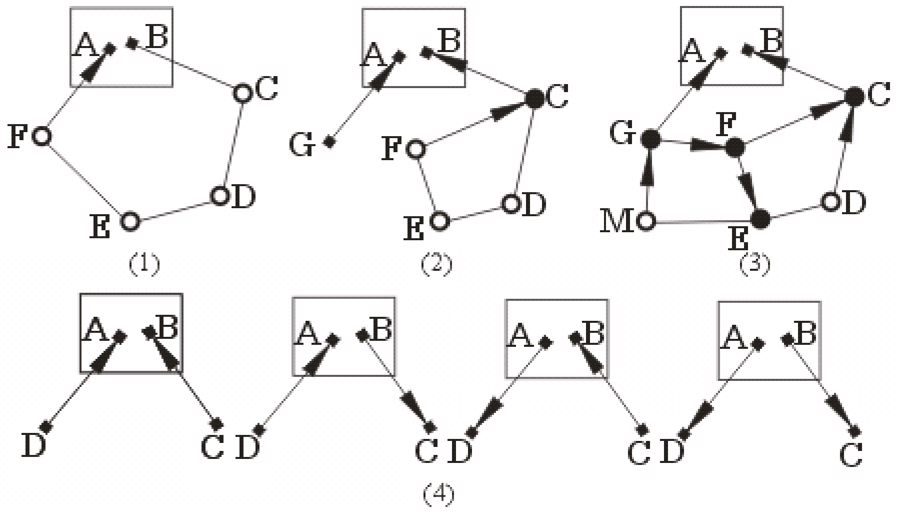


FIGURE 3 Uniting Two Ends

◆ ***Uniting an End with a Vertex***

As can be seen from the example **(1)** in Figure 4: The head (or tail) can be united with a vertex on the same arc to shape a new polygon on the condition that the number of the points from the head (or tail) to the vertex is no less than 4.

The corresponding algorithm would look like this:

INPUT: TArc* pArc
The End of the arc TPoint_tp* pPt1,
The Vertex of the arc TPoint_tp* pPt2,

OUTPUT: TArc* pArc
The New arc TArc* pArcNew，
The New Polygon TPolygon* pPoly

STEP 1: If pPt1 is the head of pArc, traverse TPoint_tp* pPt in the point-list of pArc sequentially from the tail one by one. Otherwise, traverse pPt from the head instead. Here suppose the latter.

STEP 2: When traversing pPt in the point-list of pArc, if it is equal to pPt2, pArc should be truncated from pPt2, and set it as the tail of pArc. At the same time, a new arc is generated named pArcNew, and pPt2 is added to its point-list as its head. Of cause, its topological arcs reference array should be modified in time. If pPt isn't equal to pPt2, determine whether pArcNew has been generated or not. If it doesn't exist, that means pPt2 hasn't been found out yet, and go on traversing continuously. If it does exist, pPt should be added to the point-list of pArcNew as a temporary tail. Then remove it from the point-list of pArc.

STEP 3: Finish traversing. Generate the new independent polygon by arcs of pArcNew , and the new arcs and new polygon should be added to the systemic arc set and systemic polygon set.

Seen from the example **(1)** in Figure 4: The end A is united with the vertex C, and the polygon CDEFC is generated, which has been cut off the old arc BCDEFA.

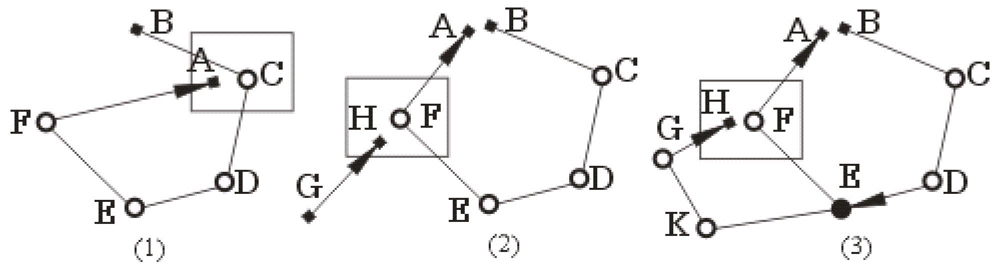


FIGURE 4 Uniting an End with a Vertex

As can be seen from the example **(2)** and **(3)** in Figure 4 above: To unite the head (or tail) with a vertex on another different arc, two complexions are offered by the reference of the tail (or head) of the arc which the head (or tail) is on — an independent arc and a dangle arc. Seen from the former: Arc BCDEFA is truncated from F (To a closed arc BCDEFAB, it needn't truncating from F, but it need sorting again from the vertex F as the new end). Remove the point H from the point-list of GH, and add the point F to it. At this time, the vertex has turned into a node, and its reference and topological arcs reference array should be modified accordingly. Seen from the latter: Except the process above, probably a new polygon has been generated because of union, which can be regarded that the dangle arc that has been united to the vertex is added to the systemic arc set, and the algorithm can consult the procedure 2 of adding an arc.

◆ ***Uniting an End with a Node***

The two complexions are divided into by the reference of the other end of the arc that the head (or tail) is on — an independent arc (the first example in Figure 5) and a dangle arc (the second and the third examples in Figure 5).

To the former, remove the end from the point-list of the independent arc, and add the node to it. At the same time, its reference and topological arcs reference array should be modified accordingly.

To the latter, two complexions are also offered according as whether the node is the other end of the arc that the united end is on. If the node is, the operation of uniting the end with it will result in a new dangle polygon when there are no less than 4 points in the point-list of the dangle arc. Except the process of the former, the new dangle polygon will be added to the systemic polygon set, which will be recorded as an operation recorder in order to create the parent-child relationships between polygons after the systemic submission. If the node isn't the other end of the arc, the operation of uniting the end with it will result in a new adjacent polygon when there are no less than 3 points in the point-list of the dangle arc. The algorithm can also consult the procedure 2 of adding an arc.

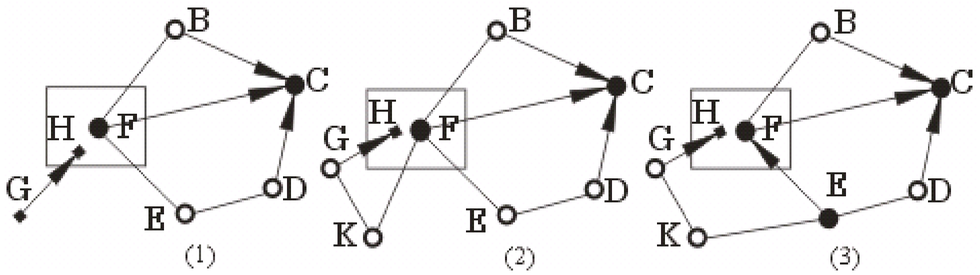


FIGURE 5 Uniting an End with a Node

◆ ***Uniting Two Vertices***

There are two complexions according as whether the vertices are on the same arc. One is that the two vertices are on the same arc; the other is that the two vertices are on the different arcs.

To the former, there are at least two points between them when the two vertices can be united. If the arc is an independent one, seen from the first example in Figure 6, there is gong to be a new dangle polygon to be generated between the two vertices after their union. And the remainder will be truncated into two new dangle arcs. If the arc is a closed one, seen from the second example in Figure 6, except the process of the former, another new polygon will be generated because the ends of the two new dangle arcs are the same one.

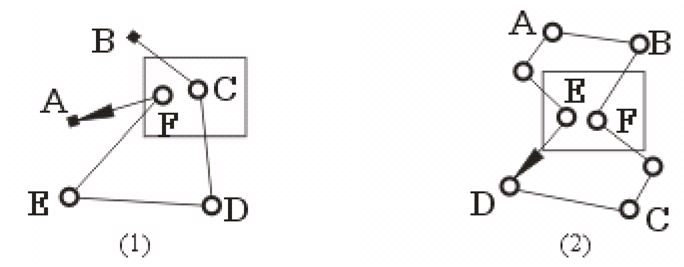


FIGURE 6 Uniting Two Vertices on the Same Arc

Consider the following pseudo code example:

```
INPUT:    TArc*                     pArc
            the point-list of pArc    listPt， nSize
            TPoint_tp*    pPt1, pPt2 (the vertices that need uniting)
OUTPUT:   TArc* pArc
          The New arc1          TArc* pArcNew1
          The New arc2          TArc* pArcNew2
          The New polygon1      TPolygon* pPoly1
```

The New Polygon2   TPolygon* pPoly2

STEP 1: int nPos1 = 0;   int nPos2 = 0;

nPos1 and nPos2 are the orders (i) of the position of pPt1 and pPt2 in the point-list of pArc.

STEP 2: Traverse pPt in listPt from its head.

```
for ( i = 0; i < nSize; i++ )
 pPt = listPt.GetAt(i);
 if ( pPt == pPt1 )
     nPos1 = i;
else if( pPt == pPt2 )
     nPos2 = i;
```

Finish traversing.

STEP 3: Judge the absolute value of the difference between nPos1 and nPos2.

```
if ( ABS(nPos1 – nPos2) < 2 )
         return;
else
{
         int nPosMin = min(nPos1,nPos2);
         int nPosMax = max(nPos1,nPos2);
}
```

Traverse pPt in listPt from its head again. Keep the points that range the orders from 0 to nPosMin in listPt of pArc. And add the points that range the orders from nPosMin to nPosMax into pArcNew1. Then unite its head with its tail, so pPoly1 will be generated. Add the points that range the orders from nPosMax to (nSize – 1) into pArcNew2. At last, add pArcNew1 to the systemic arc set, and add pPoly1 to systemic polygon set.

STEP 4: If the head of pArc is the tail of pArcNew2, which means that the old pArc is a closed arc. So pPoly2 is generated with pArc and pArcNew2 surrounding. Then add pPoly2 to systemic polygon set too.

STEP 5: Finish processing.

To the latter, the categories of the arcs should be identified first according to the references of their ends.

① An independent unclosed arc whose reference of ends are both 1.

② A closed arc whose ends are the same one and the reference is 2.

③ A dangle arc whose reference of one of its ends is 1, and the other is no less than 3.

④ A generic arc whose reference of its both ends are no less than 3.

Then, the following diagram shows how these complexions work together:

No.	The arc which pPt1 is on		The arc which pPt2 is on	
	Reference of its head	Reference of its tail	Reference of its head	Reference of its tail
1	1	1	1	1
2	1	1	2	2
3	1	1	1	>= 3
4	1	1	>= 3	>= 3
5	2	2	2	2
6	2	2	1	>= 3
7	2	2	>= 3	>= 3
8	1	>= 3	1	>= 3
9	1	>= 3	>= 3	>= 3
10	>= 3	>= 3	>= 3	>= 3

TABLE 1 Uniting Two Vertices on Different Arcs

The vertex on an independent unclosed arc will turn into a node when united with a vertex on another arc. The independent unclosed arc will be split into two new arcs from it, and its reference and topological arcs reference array should be modified accordingly, such as the complexions of number 1, 2, 3, 4.

After united with a vertex on another arc, the vertex on the closed arc will act as the new end (Because the head and the tail are the same one.) of this closed arc, and its reference and topological arcs reference array should be modified accordingly. Then sort all points in the point-list of the old closed arc by sequence again from the new end, such as the complexions of number 2, 5, 6, 7.

The vertex on a dangle arc will also turn into a node when united with a vertex on an independent unclosed arc or a closed arc. The dangle arc will be split into two new arcs from it, and its reference and topological arcs reference array should be modified accordingly, such as the complexions of number 3 and 6. Especially, a new polygon will be probably generated when united with a vertex on a generic arc or another dangle arc, such as the complexions of number 8 and 9.

As can be seen from the example **(1)** and **(2)** in Figure 7: After vertex E has been united with vertex F, the new arc GE and FB need adding to the systemic arc set in the first example, and so does GE in the second example. The algorithm can consult the procedure 2 of adding an arc.

The vertex on a generic arc will turn into a node too when united with a vertex on an independent unclosed arc or a closed arc or a dangle arc. The generic arc will be split into two new arcs from it, and its reference and topological arcs reference array should be modified accordingly, such as the complexions of number 4, 7, 9. When united with a vertex on another generic arc, a new polygon will be perhaps generated, such as the complexions of number 10.

Seen from the example **(3)** in Figure 7: After vertex F has been united with vertex E, the arc BFC is split into two arcs, the new arc BF and FC. They are regarded as new added arcs, and search their Right-Polygons and Left-Polygons according to the algorithm of the procedure 2 of adding an arc showed above.

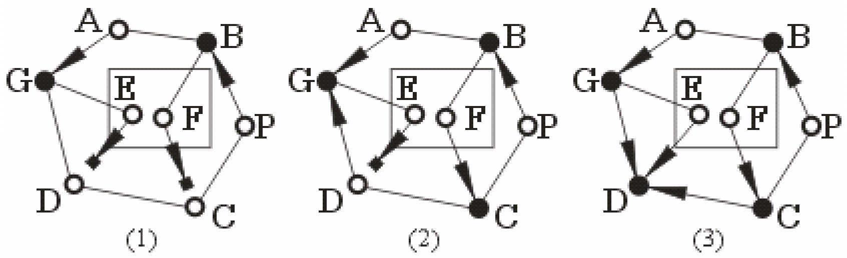


FIGURE 7 Uniting Two Vertices on Different Arcs

◆ ***Uniting a Vertex with a Node***

There are two complexions according as whether the vertex and the node are on the same arc. One is that they are on the same arc; the other is that they are on the different arcs.

To the former, the algorithm is similar to uniting an end with a vertex on the same arc. And to the latter, the algorithm is similar to uniting an end with a vertex on the different arcs.

◆ ***Uniting Two Nodes***

Two complexions can be classified also according as whether the two nodes are on the same arc. One is that they are on the same arc, that is, the head is united with the tail on the same arc, whose algorithm is similar to uniting two ends on the same arc. The other is that they are on the different arcs whose algorithm is similar to uniting two vertices on the different arcs.

■ ***Modify Parent-child Relationships between Polygons Because of Topological Changes***

A tree-structure will be established during the process of static topology in order to describe the parent-child relationships between polygons, that is, inclusive relationships between polygons. All operations that perhaps cause alteration in parent-child relationships between polygons have been recorded in a special structure during every topological change, such as operations that cause polygon adding, deleting and moving, boundary changing, etc. After topological alterations, parent-child relationships between polygons should be modified accordingly; whose algorithm is that add or delete nodes in the tree-structure. Here don’t give unnecessary details.

The algorithms and the processes of dynamical topological changes described above have been discussed particularly in terms of respective operation mentioned in the topological alteration-processing model. And so relevant dynamical topological relationships have been established already. The algorithm of moving an arc wholly, moving an end or a vertex, deleting a vertex, uniting points, etc. is similar to the one of adding a new arc. And the algorithm of splitting a vertex or a node is the same as the one of deleting an arc.

Research by Experiments

An independent experimental system has been designed in order to validate the data structure and algorithm of establishing dynamical topological relationships. It provides the functions of **INPUT** or **IMPORT**, **TOPOLOGICAL FEATURES CHECK-UP**, **TOPOLOGICAL COHERENCE CHECK-UP**, **EDIT GEOGRAPHIC OBJECTS** (including points, lines and polygons **ADDING**, **DELETING** and **MOVING** etc.), **ESTABLISH STATIC TOPOLOGICAL RELATIONSHIPS**, **EDIT TOPOLOGICAL FEATURES** (including ends, vertices, nodes, arcs and polygons **ADDING**, **DELETING**, **MOVING**, **SPLITTING** and **UNITING** etc.), **ESTABLISH DYNAMICAL TOPOLOGICAL RELATIONSHIPS** and **OUTPUT** or **EXPORT**. A sound conclusion can be drawn that the topological alteration-processing model is valid and preponderant compared with the models of static topology by the same data.

Conclusions

The data structure and algorithms of establishing dynamical topological relationships and even the topological alteration-processing model have been discussed in the thesis. Because land subdivisions have been split or united more frequently in land use management, the algorithms of adding or deleting an arc, splitting or uniting points have been introduced emphatically and detailedly. Finally the model is proved to be valid, practical and useful by typical examples of Land Use Management System of SHANXI province.

Of cause, if the model can realize the temporal management and modification of figures and attributes well and truly, the needs will be met further in statistics, query, comparison and analysis, etc. in land use management. The research needs a further advancement.

References

1.Langran.G, Time in GISs Taylar & Franeis Ltd, 1992,

2.Egenhofer M. J. and Franzosa, D. R., Point-Set Topological Spatial Relations[J], International Journal of Geographical Information System, 1991, 5(2): 161-174

3. Robert.G.Cromley, Digital Cartography, Prentice Hall, Englewood Cliffs, New Jersey 07632

4. Wu Hehai, Cartographic Database System, Beijing, Serving and Mapping Press, 1991

5.Wu Xincai,, the Basic Technology and Developments of GIS, The Science of the Globe, Journal of China University of Geosciences, 1998,7, 23-4

6.Xiao Lebing, Zhong Ershun, Liu Jiyuan, Song Guanfu, Oriented Objects Design and Implementation of the Data Model of Holistic GIS, Beijing, http://www.gischina.com