Report for Spatial Database Project

Sushmita Bhattacharya(133050022),
Sharma Supriya(133050020),
Priyanka Chikhale(13305R016)

April 15, 2014

1 Geometry generalization

Geometry generalization is a well known concept in the field of cartography.It is a method by which information is selected and represented on a map such that it is supported by the scale of the medium of the map.It is not required that the geographical details are preserved.Only the most important features are preserved and lesser ones can be removed. In map generalization one of the difficult problems is to maintain the topological consistency between adjacent polygonal regions during the simplification process.

2 Problem Definition

- INPUT: A set of linear geometries that bound polygonal regions and a set of constraining points.
- OBJECTIVE: Simplify the linear geometries such that the relationship between the constraining points and linear geometries before and after the simplification does not change. In addition, the topological relationships between the original set of input linear geometries does not change after the simplification.
- Assumptions
  1. Data is in cartesian Space
  2. Linear geometries do not have self-intersections.
  3. Linear geometries do not intersect with other linear geometries except at end points.
  4. The constraining points do not intersect any of the linear geometries.

3 Our Work

1. We have done maximal simplification of each linestring. The results are correct.
2. We have simplified the linestring until no further changes in the set of the linestring is possible.
   In the Figure 1, If the line L1 is simplified first L2 will be simplified in the first pass itself. But if L2 is processed first then L2 will be simplified in the second pass after L1 is simplified (in the first pass).

3. We have used thread for speed up the concurrent simplification of the independent parts of a linestring.

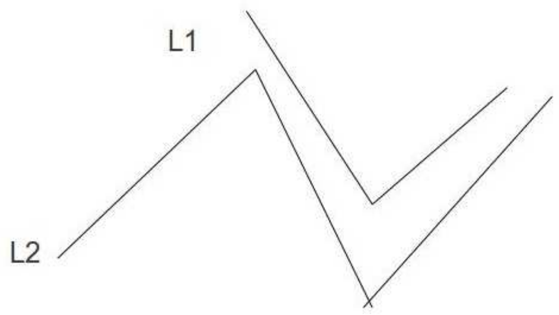


Figure 1: case 1

4 Algorithm

4.1 Input

1. The full path name of the linestrings gml
2. The full path name of the constraint points gml
3. The minimum number of points to be reduced from the set of linestrings

4.2 Output

The gml file and the insert script of the set of output LineStrings - the simplified linstrings.

4.3 Pseudo-Code

```
begin
    Read LineString and Points data
    Repeat until there is no changes in the
    for each linestring
```

```
        Simplify_Linestring() from start to end point
end

Simplify_Linestring()
    sort the points in the linestring order by the perpendicular distance between the points in the linestring and the line joining the end-points of the linestring
    L = linestring to be simplified in the step
    num_iteration = 1
    repeat
        cond = Number_points_or_linestrings_falling (area made from the points of L)
        if(cond==0) //some constraint point lies within the area formed by the line L
            index = the num_iteration th furthest point
            break
        L_new = L - set containing num_iteration times furthest points
        cond2 = Number_points_or_linestrings_falling(area made from the points of L_new)
        if(cond != cond2) // Some point or linestring falling in the area coverd by L but not by L_new
            break
        L = L_new
        num_iteration++

    if (cond!=0) //some points or linestrings falling within the area formed by L
    call Simplify_LineString() for the lines one starting from start to index and index to end in parallel thread.
else
    connect the start and end point

Number_points_or_linestrings_falling(area):
    count = 0
    mbb = the minimum bounding box for the geometrical object area.
    for each pl in point and linestring
        if(mbb contains pl)
            if(area contains pl)
                count++
    return count
```

5 Result

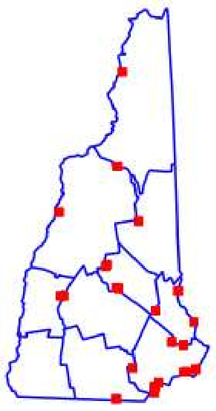


Figure 2: input

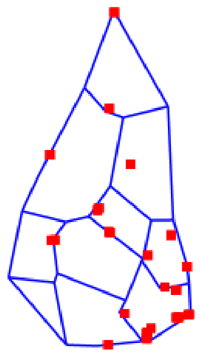
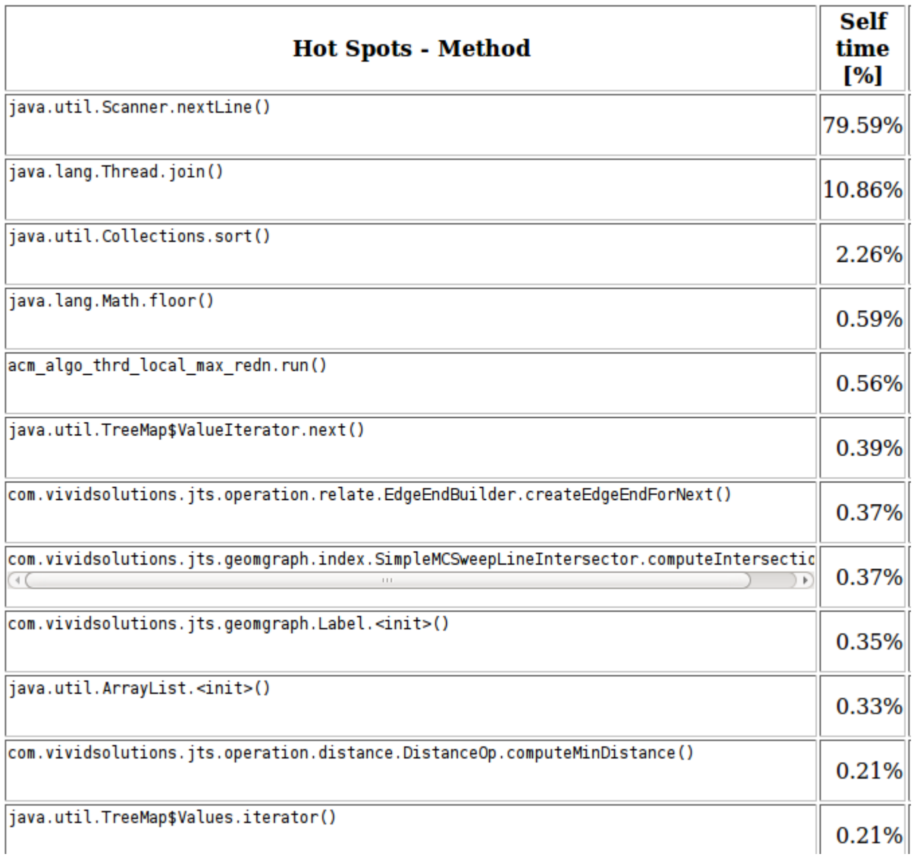


Figure 3: output

6 Performance

total runtime 1551ms
percentage of total time for parsing input and storing in internal data structure is 13%



Hot Spots - Method	Self time [%]
java.util.Scanner.nextLine()	79.59%
java.lang.Thread.join()	10.86%
java.util.Collections.sort()	2.26%
java.lang.Math.floor()	0.59%
acm_algo_thrd_local_max_redn.run()	0.56%
java.util.TreeMap$ValueIterator.next()	0.39%
com.vividsolutions.jts.operation.relate.EdgeEndBuilder.createEdgeEndForNext()	0.37%
com.vividsolutions.jts.geomgraph.index.SimpleMCSweepLineIntersector.computeIntersectio	0.37%
com.vividsolutions.jts.geomgraph.Label.<init>()	0.35%
java.util.ArrayList.<init>()	0.33%
com.vividsolutions.jts.operation.distance.DistanceOp.computeMinDistance()	0.21%
java.util.TreeMap$Values.iterator()	0.21%

Figure 4: performance evaluation

7 Future Work and Possible Improvements

1. We shall try to optimize the run time by using some special data structure.

2. We would use our own functions for the geometry operations to reduce runtime in case there are some overhead for calling the geometry operations provided by the jts library.